



FILE: MAIN2

Compliments of FRESS

A File Retrieval and Editing SyStem

Release 9.1     2 MAY 79

FRESS Resource Manual

Release 9.1  
2 MAY 79

[Including Multi-Window Specifications]

Last Updated: October 1, 1979  
Printed: October 1, 1979

Jonathan M. Prusky '79

Department of Computer Science  
Brown University  
Providence, Rhode Island 02912



FRESS. It ain't just another pretty face.

Hypertext. It ain't like dusting crops, boy.

HYPertext. IT JUST GOES TO SHOW THAT IT'S ALWAYS SOMETHING.

AVAILABLE FRESS DOCUMENTATION

This supersedes all previous FRESS documentation. It replaces the following:

FRESS NEWSLETTER, Vol. 2, No. 1, March 3, 1976, "FRESS Command Macros"

"Command Macro Documentation Update"

FRESS Reference Manual -- Structure and Commands (2nd printing, January 15, 1976)  
All sections have been replaced by this document, except for Sections 2, 3, and 4 which can now be found in the FRESS User's Guide -- Updated

The following additional FRESS documentation is available (or will be soon):

FRESS User's Guide -- Updated (in preparation)

This manual serves as an overview to FRESS. Concepts, facilities, and possible applications of FRESS are discussed. It is a combination of the old FRESS Concepts and Facilities, Sections 2, 3, and 4 of the old FRESS Reference Manual, and Sections 1, 2, 3.1-3.11, 6, and 7 of the old FRESS User's Guide.

A FRESS Guide to Large Document and Thesis Preparation  
(third revision in preparation)

This document is available online, by typing ".tman" in the FRESS environment.

All of these documents are (or will be) available in each Terminal Center document rack and for purchase in the Brown Bookstore (except for the Thesis Manual). For information on available documentation, type ".frsnews" while in FRESS.

PREFACE

All previous documentation has insufficiently and at times incorrectly explained many sophisticated FRESS features. This necessitated not simply reworking the old manuals, but a research project that explained and tested each facility fully. While I have not produced a user's manual, I have made considerable efforts to provide this reference manual with a clear, easy-to-reference layout, complete with many examples and extensive cross-references.

A special effort was made to address users who do not have prior computer experience. The comments I received from circulating a late draft were valuable. I would like to thank all those who have assisted with this project in one way or another. First, I thank Andy van Dam for sponsoring me and understanding that the manual would take more than a year for a student to complete. In conjunction with him, I also would like to thank the Department of Computer Science for sponsoring this project as my undergraduate Honors thesis. Ken Fairchild '75, Eric Albert '80, and Phil Wisoff '78 (former FRESS project director) were helpful in the early stages of development. Steve Feiner '73 has offered many suggestions. Steve DeRose '81 made several useful comments. Kevin Connolly '79 helped comment from the perspective of a new user. Norman Meyrowitz '81 was invaluable not only from a journalistic point of view, but also because his careful reading pointed out many inconsistencies and boundary conditions not initially covered. Most importantly, I would like to thank Alan Hecht '80 (former FRESS project director) for everything. Alan served as editor, researcher and supporter in our joint effort to make FRESS look respectable. He has read every word in the manual at least fifteen times. I gratefully thank him for the dozens of hours he spent looking through five-year old listings, making minor modifications to the system so the documentation could as closely as possible approximate actuality. Both of our perfectionist attitudes have been so finely tuned it is difficult to resist making improvements wherever we see them. This product would not exist in its present form had Alan not been so committed. Alison Lehr '79 provided support and understanding.

Finally, I want to thank the Operations staff of the Brown Computer Center. Every member on duty while I was working was extremely cooperative and went above and beyond the call of duty in helping me to get each iteration printed in the proper format, often at unscheduled times on very "special" forms.

JMP  
10/01/79

HOW TO READ THIS MANUAL+: -- HOW TO READ\*\*+

## D I S C L A I M E R

Although this manual has been very carefully edited by the author, no warranty, expressed or implied, is made by Brown University as to the accuracy of the material, nor shall the fact of distribution constitute any such warranty, and no responsibility is assumed by the author or Brown University in connection therewith.

This is a reference manual for FRESS ("File Retrieval and Eding SyStem"). Anyone interested in learning what FRESS is and its unique approach to text-processing should find this manual useful. However, it should be noted that specific references are made to the installation at Brown University where FRESS runs under VM/370.

ORGANIZATION

Section 1 serves as an introduction to FRESS, including possible applications, support available, and a brief overview of FRESS -- which is in no way intended to stand alone. Section 2 explains how to invoke FRESS and the characteristics of each version (for different terminals). Because of the sophisticated editing features that enhance the capability of older hardware, FRESS has necessarily reserved many characters for special control operations. A discussion of these is supplied in Section 3. All codes necessary to format a document are described in Section 4. Standard FRESS commands are explained in Section 5. "House Functions" are a special type of FRESS command that perform "house-keeping chores"; they are described in Section 6. Section 7 is a technical, though thorough discussion of the Command Macro Facility, which includes a list of the system-supplied Command Macros. Appendices cover the justification algorithms, different types of Structure, keyword specification, boolean requests, lists of commands, what to do if problems occur, and a long list of system messages. The Table of Contents serves as an index.



## NOTATION

- Words which normally would not be capitalized, but are in this manual, have a special meaning in FRESS. They are introduced either where they are first described or in Section 1.3.
- Any completely capitalized words refer to FRESS Commands, or if preceded by an ampersand, "&", they refer to House Functions. The appropriate descriptions can be found in Section 5.8 or in Section 6, respectively.
- The symbol "ø" indicates a literal blank.

## Specifications

Generally, the notations used throughout this manual are entirely consistent with TM37, "Notational Conventions," which should be referenced for detailed examples. The rules are summarized below.

Specifications (e.g., commands, format codes) are always found in a box, as are many figures. Various sections may supplement these rules with more specific guidelines:

- Any alphabetic characters in a specification box may be specified in upper and/or lower case.
- Symbolic parameters in specifications are enclosed by syntactic brackets ("<" and ">"). They are not to be substituted for literally, but indicate rather a parameter of that general form.
- When there is a limited selection of parameters, each choice will be listed under the symbolic parameter, or listed on a single line below the parameter, separated by or-bars (e.g., A|B|C).
- Optional parameters are enclosed by brackets ("[" and "]").
- Default parameters, if any, will be underlined when they can be generalized.
- Any punctuation characters (e.g., "+", ",", ".", "-") found in a specification are to be taken literally.

## 1 INTRODUCTION

### 1.1 HISTORY

FRESS is a research project in sophisticated tools for authors and online readers which has developed over the course of more than a decade under the direction of Computer Science Professor Andries van Dam.

FRESS' predecessor, the Hypertext Editing System (HES), was developed at Brown University from 1968 through the Spring of 1969 as a joint project with Theodor Nelson under an IBM research contract. It was designed to run on the IBM 2250 display console. In the Spring of 1969 the design of FRESS began, and during January 1971, the basic FRESS system was installed on the Brown and National CSS Service Bureau 360/67 based time-sharing systems using typewriter terminals. In May of that year a version of the system allowing simple alphanumeric display terminals to be used was installed. By June of 1972, various versions of FRESS were used as production text-editing systems at Brown<sup>1</sup> the Catholic University of Nijmegen, Holland<sup>1</sup> and a government agency.

Under a \$79,000 grant from the Exxon Education Foundation (June, 1973 - June, 1974), previously designed, but undeveloped Hypertext and graphics features were added to the basic text-editing system. This grant underwrote Computer Assisted Instruction (CAI) at Brown University during the Spring semester of the 1973-74 year. The interdisciplinary university-level course, UC126, "Man, Energy, and the Environment" was taught by Physics Professor George Seidel. Twelve students participated in the experiment by doing all of their course reading using a computer-linked display console, and a database specially prepared with various forms of Hypertext.

The results of the experiment were inconclusive. It was determined that the FRESS Hypertext system was feasible, effective, and enjoyable in an educational environment encompassing students with diverse backgrounds. However, the evidence from this single experiment was not sufficient to prove whether or not the Hypertext database helped the students to learn more or better. Also, while the students explored most of the instructor-created Hypertext, there was insufficient incentive for them to enlarge it.

A grant from the National Endowment for the Humanities<sup>3</sup> funded a second experiment using Hypertext in education. In that project, FRESS was used to help teach poetry in the Brown

---

<sup>1</sup>Initially under CP/CMS and now under VM/370.

<sup>1</sup>Under OS/MVT

<sup>3</sup>NEH-22258-75-95

University English 16 course "Critical Analysis of British and American Literature: Poetry" taught by Professor Robert Scholes and James Catano. Since the Hypertext facilities that were new for the first experiment were already installed, attention was focused on the development of the Hypertext database itself. All reading and homework was done online by the twelve students participating in the special section of the class. Thus, in addition to their critiques of the poems, they were able to leave comments and questions in the text as well as create their own interest-based trails, enriching the database for subsequent readers. Instructor's comments on student work were also added to the database.

The results of this experiment, while statistically not definitive because of the small number of students involved, were extremely encouraging with respect to quality of work performed and the quantity produced. Among the observations and benefits obtained, were 1) The Hypertext system was relatively easy to learn, whether or not the user had a technical background; 2) The Hypertext users produced prodigious amounts of written work, despite the fact that English 16 was not primarily a writing course; 3) Independent evaluation showed students in the Hypertext section to have improved more than students in the other sections. The instructors all felt that FRESS enabled the students to make significant progress, especially in the areas of increased self-confidence and improved critical abilities; 4) Active interaction with the material was extensive and a great improvement over the earlier experiment; 5) The benefits from classroom and Hypertext experiences were complementary. Both were needed in the course format and neither should be emphasized too strongly at the expense of the other.

In general, the results of the experiment indicated that the use of a Hypertext system provided substantial educational benefits to students and instructors. In particular, the Hypertext system provides a convenient and powerful medium for stimulating the communication of ideas among students and their instructors. The product of this environment is a significant new approach to the development of critical and writing skills. The following year another section of English 16 was led by James Catano, using the same database; similar encouraging results were obtained.

The system as it stands today is used primarily for text editing of large documents with online Hypertext Structure for easy access and/or selective printing. Keywords, decimal section outlines, labels, etc., are used for maintaining bibliographies, reading lists, multiple versions of a manuscript with many common parts, and large manuscripts such as books and theses. Development has largely stopped and the limited available manpower is used for maintenance and bug fixing.

1.2 SUPPORT

## N O T I C E

FRESS was developed at Brown, and is currently supported only by a part-time student programmer. There are still bugs in FRESS, particularly in the editor, which sometimes causes it to act unpredictably. It is possible for bugs in the editor to directly damage a FRESS file to a point that necessitates assistance from the FRESS staff. The lack of FRESS support may sometimes delay a user, in that there is not enough manpower to service all system-related problems as soon as they are reported. Any potential FRESS user should be aware that because of this, and related problems, he is more "on his own" than if he uses other software. There is also no commitment to support FRESS in the event that hardware substantially different from the current IBM line is acquired.

There are two ways in which users can contact the FRESS staff:

- (1) In writing or by telephone, through the Department of Computer Science [(401) 863-3300], Box 1910, Brown University, Providence, Rhode Island 02912; and
- (2) Online, through the "FRESS-communication" facility. If in the FRESS environment, it can be invoked simply by typing ".frscomm" or ".help". While in CMS, the user can get access to FRESS by issuing the CMS command "LIBRARY FRESS" (this is unnecessary if the user has already been in the FRESS environment during the current terminal session).

The "FRESS-communications" facility allows the user to send a message to the FRESS staff and/or a copy of a FRESS file that does not appear to be working the way the user intended it to. If a file is sent, the user should not use it again until contacted by the FRESS staff.

There is also a standard way for the FRESS staff to communicate with all FRESS users. When FRESS is invoked, the first line typed has two dates. The first refers to the date when the current FRESS version (e.g., "version 9.1") was created, and the second (in parenthesis) indicates the last time that the ".info" command macro was updated. By typing ".info," the FRESS users can find out any current news.

### 1.3 OVERVIEW

It is assumed that the user already has an understanding of FRESS concepts and facilities (see DOCUMENTATION), although some features are briefly described in this section.

FRESS is best used for applications in which online display is important, or in preparing long documents in which all sections are to be kept together in the same file and are to be accessed frequently.

The FRESS editor was specifically designed to edit flowing text as opposed to more rigidly formatted computer programs. It is stream-oriented as opposed to line-oriented (see Stream Editor below).

Some commands such as SURROUND and MOVE, apply specifically to text processing (see Appendix D for lists of commands by type). A user can easily move a portion of text with one command. In one command line, the text can be moved anywhere within the 2100 character Editing Buffer. There is also a facility that allows the user to defer telling FRESS where the text is to be moved, so he can travel and search for the new location anywhere in that or another file, and then easily specify it as the destination of the move (see Deferrals below).

The FRESS editor can pinpoint desired locations and jump through a file in a speedy, logical manner (see Hypertext below). Many different files can easily be accessed and edited during the same editing session, without having to repeatedly leave the editing environment (see GFILE).

#### • The FRESS Environment

FRESS is a self-contained environment designed specifically for text manipulation. It is used for both creating and viewing files, as well as for formatting text. Because of this, the FRESS environment can be either in Command Mode or Input Mode.

Command Mode is the initial state when FRESS is invoked. From Command Mode, files can be manipulated (e.g., created (MFILE), opened (GFILE), erased (SFILE)). A vast array of other commands allow the user to enter Input Mode, edit a file, travel through a file, or perform system tasks. All commands are described in Section 5.

Input Mode can be entered through various commands. It is typically used to enter into the file text strings too long to enter using one of the editing commands. As text is entered, any formatting codes (see Section 4) are checked, and error messages are typed as necessary. A modified version of Input Mode, called

Swift Input Mode does not do this verification, but it is faster and less expensive to use -- it should be used anytime Input Mode would be used. Any formatting errors can always be found later using FERROR.

FRESS has defined special meanings for several characters. These may seem confusing at first, but if Section 3 is read carefully they should not pose much of a problem.

- Stream Editor

The FRESS editor is a stream editor, which means the file can be thought of as consisting of a single indefinitely long "superline" or "stream" of text, rather than many discrete lines (as in a conventional program editor). Text appears at the terminal on consecutive "logical lines," the length of which the user may select (see SDISPLAY).

- Editing Buffer/Display Window

As a stream editor, the scope of FRESS editing commands is not limited to just one line. Instead, an Editing Buffer of 2100 characters is searched for the first appearance of the desired "context pattern."<sup>4</sup> After each edit, the system displays a portion of the Display Window, which is simply the top part of the Editing Buffer. FRESS will optionally type any portion of the Display Window after each editing command (see SDISPLAY). It is also possible to PRINT the entire Display Window (display it at the terminal), without having to relocate the Editing Buffer (which requires some overhead). The Editing Mode, which determines when the Editing Buffer is relocated, and the Display Mode, which determines whether the context string around each edit location will be typed, are described in Section 2 and are set by SMODE. Both modes can also be temporarily overridden (see Section 2.3.1).

- Getting Output

The user can have a file fully formatted and sent to the high-speed printer or the terminal using FULLPRINT.

---

<sup>4</sup>The string being edited. It can be specified by just the characters at its left and right boundaries (see the description of ellipses in Section 3).

OTYPE should be used to get a printout of the file as it appears online, reflecting the current Viewing Specifications (Viewspecs) (see Online Display below).

- Hypertext

A FRESS file can be given capabilities for non-linear travel,<sup>5</sup> information retrieval, and sophisticated online display.

These facilities are all derived from Hypertext, special text created by various editing commands, not entered literally. Hypertext (Structure) gives a file the framework necessary for elaborate manipulation. Rather than scanning for literal text, particular types of Hypertext can be retrieved, enabling non-linear travel through the file. Hypertext is also used to selectively retrieve Blocks of text (see below). The different types of Hypertext do not appear in the formatted printout (except for Decimal Labels) and can optionally be suppressed online:

- Labels: A Label is simply a location in the file to which the user has chosen to give a name of up to sixteen characters (see MLABEL). GLABEL is used to move display to exactly the Labeled point in the file. These Labels should be used to access standard or important places in the file, regardless of how much editing is done.
- Blocks: Arbitrarily long chunks of text can be grouped into logical units called Blocks. When MBLOCK is issued, the text specified is surrounded with special Hypertext characters. This Block can then be manipulated as a single unit, greatly simplifying an operation such as moving (see the Block qualifier in Section 5.5). It is also possible to travel from one end of the Block to the other using JUMP. A Block may also contain a Decimal Label (section number) that appears in formatted output (see IDBLOCK, MDBLOCK). The Decimal Label is dynamically updated when Decimal Blocks are rearranged. Furthermore, Hypertext references to these Blocks called Tags, are dynamically updated if the number of the Decimal Block changes. Decimal Blocks may be imbedded several levels deep, but are easily distinguished online when the proper viewing specifications are set (see SVIEW). Blocks may also be coded for boolean retrieval and/or for selective online viewing (e.g., some Blocks can be made "invisible" -- see SKDISPLAY).
- Jumps/Splices: It is possible to piece together separate parts of the same file or separate files with conditional Jumps and Splices (see MJUMP, MSPLICE). Two points are logically connected (both for editing or for formatting) when

---

<sup>5</sup>Not sequentially (line-by-line).

a special Keyword string is properly set (see SKJUMP). The presence of this type of Hypertext allows the user to travel from one end of the jump to the other via JUMP, even when the Keywords are not set.

- Areas: An Area is a segment of the Text or Work Space (see File Structure below) through which linear travel commands may not pass. Areas are useful for keeping text and Hypertext that is related to the main corpus, associated with a file, but out of the main stream where it will not be FULLPRINTed (unless there is a Jump or Splice into the Area). See NAREA and SPLAREA for more information.

- Online Display

FRESS has an extensive interactive online display program which allows the user to vary the way a file appears online. Variable Viewing Specifications determine which part of the file is to appear online, and in what form (see SVIEW). The text can be right justified and the various types of Hypertext can be optionally viewed. Several formatting codes, such as those for a new paragraph, indentation, new line, etc., can take effect at the terminal. This helps to give a better idea of how the document will appear when it is fully formatted. In addition, portions of text (Blocks with Display Keywords) can be made to selectively "disappear" so that the user does not see them when the file is edited or printed.

- Key Delimiter

Most commands take one or more parameters and many have parameters which themselves have several options. The character separating the command from the first parameter is called the Key delimiter; it must also be used to separate each parameter. Options within a parameter must use a delimiter other than the Key delimiter. For clarity, a blank may separate the command and the Key delimiter. It is also possible to use a blank as the Key delimiter (but then the blank fill character, "\_", must be used to indicate blanks within parameters -- see Section 3).

- Deferrals

Editing commands do not restrict the location or the scope of the edit to the Editing Buffer (see <lp> and <scope> in Section 5.7). Either can be deferred to another place in the same or



different file by using the defer character and defer separator (see Section 3).

- File Structure

A FRESS file is organized into several system-defined regions called Spaces (accessible using DSPACE). The Text Space is where all the text for a document resides (with any Hypertext the file may contain). The Keyword, Label, and Structure Spaces are effectively Hypertext directories. They exist for the convenience of the user in manipulating various forms of Hypertext and allow examination of the file on a higher level than a scan through the text itself would yield. Hypertext in these Spaces may be edited directly. This "inverse editing" is useful for making easy or global Hypertext changes. There is also an Annotation Space where comments on the main text can be put (e.g., footnotes). When this facility is used, the user can easily examine these annotations as a group and optionally decide to print them as footnotes, or not to print them at all (see SKANNOTATION). The Annotation Space can also be used for making comments (annotations) while browsing online. These comments will be referenced in the Text Space with a special Tag.

Similar to the Text Space is the permanent Work Space where the user can put sections, or bits of text separate from the Text Space. With this facility, chunks of text that appear in multiple places in the file can easily be copied from one location. The Work Space can also serve as a collection point, as the user roams through the file moving or copying bits of information to group together elsewhere or to be temporarily stored in the Work Space.

The Text, Annotation, and Work Spaces are called Main Spaces because literal text and/or Hypertext can be located there.

Figure 1.1 illustrates how a file is organized. Hypertext is shown as it would appear, while periods (".") indicate regular text. The arrows indicate how JUMP can be used to travel between Spaces. It should also be noted that the "END OF..." lines delimit each Space. They cannot be deleted, or traveled over. However, the "\*\*\*AREA\*\*" lines in the Structure Space can be crossed using TYPE. The arrows between Spaces illustrate how JUMP can be used for non-linear travel from one Space to another.

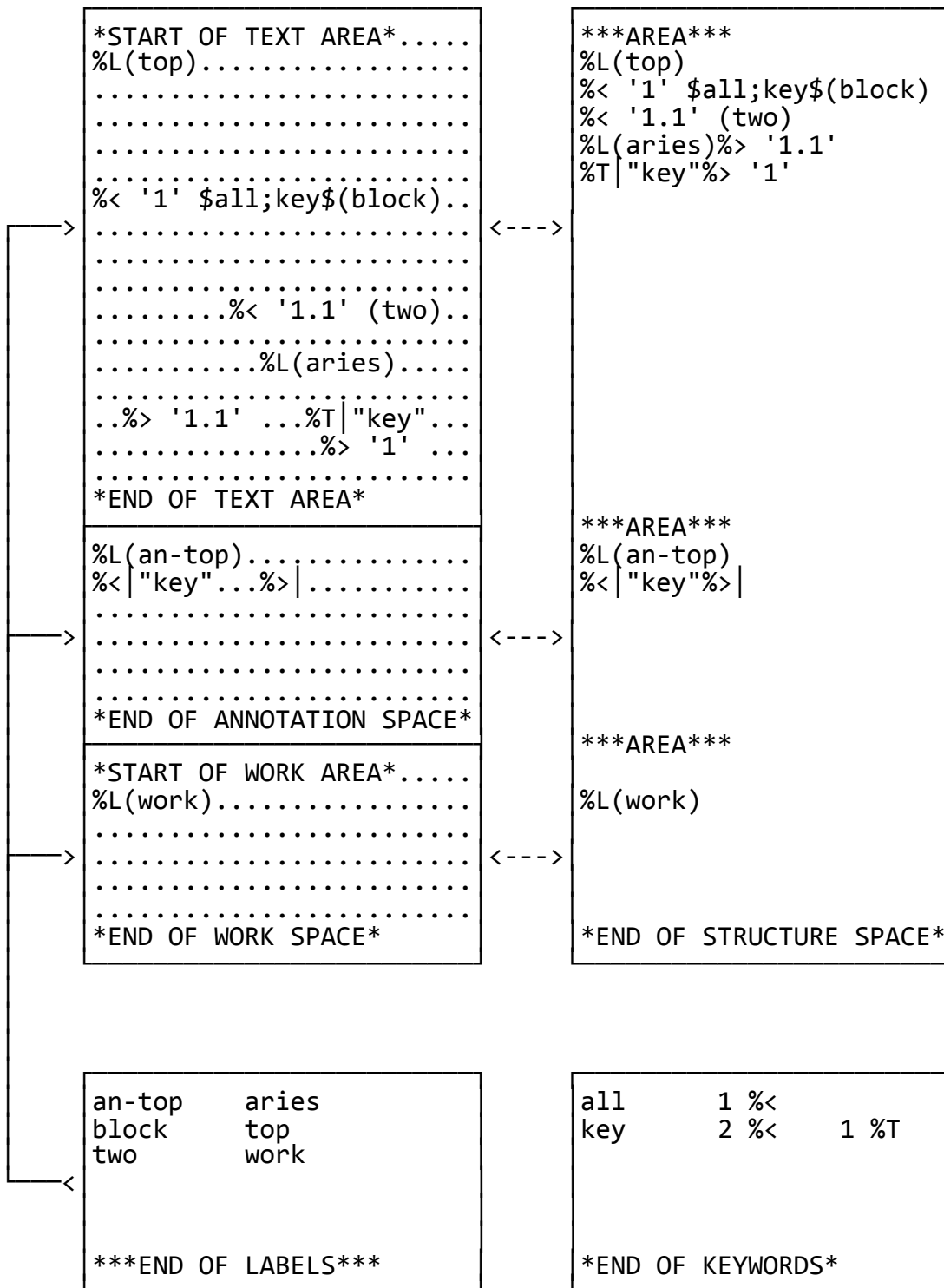


Figure 1.1 -- System-Defined Spaces

- Implied Insert Point

The Implied Insert Point is a convenient, but sometimes confusing, facility available to the user who wants the location of the editing command to default to the word before the text string which was last edited. Except for the Block creation commands (IBLOCK, IDBLOCK, MBLOCK, and MDBLOCK), any of the commands listed in Appendix D.2 set an Implied Insert Pointer as described above (&POP describes how the Block commands set two Pointers).

If the <iip> parameter is omitted from any command, the <lp> defaults to the last Implied Insert Pointer set (see Section 5.7 for an explanation of <lp> and <iip>). A list of up to eight pointers are maintained in an Implied Insert Stack on a last-in, first-out (LIFO) basis. &POP gives several examples using the Implied Insert Pointer.

- As-Is Mode

It is often convenient to enter text as it will appear in the formatted printout, without having to type the necessary format codes indicating a new line and the tab key (e.g., constructing tables).

As-is Mode translates a carriage return and depression of the physical tab key to the appropriate format codes. As-is Mode can be entered either by using the House Function &ASIS, or by indicating the appropriate parameter on many commands which put the user into Swift Input Mode.

## 2 USING FRESS

FRESS is available to the general public on any standard terminal; it may also be used in a "multi-window" format on an Imlac PDS-1D mini-computer (not currently available to the general public).

A user must have a virtual machine of at least 512k to use FRESS; this can be specified either at login-time or later in the terminal session. If "0512K" is appended to an account number when logging-in (e.g., "L <account> 512K"), the user will initially invoke a 512k machine. The CMS command "DEFINE STORAGE 512K" can be used to specify the proper machine size if the user is already logged-on to a smaller machine. The system will type "STORAGE=00512K," "CP ENTERED..." The user should respond with "IPL CMS" to return to CMS. The procedure described below can then be used to invoke FRESS.

### 2.1 INVOKING FRESS

There are two general types of terminals used: those with upper/lower case characters and those with only upper-case characters. To accommodate the many differences among terminals, several different FRESS versions are available. FRESS is invoked from CMS by typing:

```
FRESS [NOS] <version> [<maclib>...]
      A
      B
      D
      H
      I
      R
      T
```

NOS.....(NO SPOOL) If specified, the spool class of the virtual printer remains unchanged, otherwise it is set to CLASS F (upper and lower case print chain).

<version>...The FRESS version to be invoked (see Section 2.2).

<maclib>....The command macro libraries (up to nine, separated by blanks) to be included in the FRESS session (see Section 7).

NOTES

- All descriptions in this manual refer to upper/lower case versions except for Section 2.2.1 which describes the upper-case only ("T") version.
- The FRESS version selected affects only how the file appears at the terminal. The internal format of a file is independent of the version of FRESS being used.

2.2 FRESS CHARACTERISTICS

Figure 2.1 summarizes the characteristics of each FRESS version:

<u>Suggested</u> <u>Terminal</u>	<u>Ver-</u> <u>sion</u>	<u>Display</u> <u>Wind. Line</u>	<u>Mode</u> <u>Dsp Edit</u>	<u>Char</u> <u>Del</u>	<u>Line</u> <u>Del</u>	<u>Line</u> <u>End</u>	<u>Esc.</u> <u>Char</u>	<u>"~"</u> <u>---</u>
Mag-card	A	700 50	D T					!
	B	910 70	D T					
Datel	D	700 50	D T					
Hazeltine	H	910 70	D T	RUB	LF	ESC		\
Imlac PDS	I		D S					"5"
Teleray	R	910 70	D T	BSP	LF	{	ESC	\
Teletype	T	910 70	D T					\

BSP...backspace key  
 ESC...escape key  
 LF....line-feed key  
 RUB...rub-out key

Figure 2.1 -- Different Version Characteristics

NOTES

- Any entry above is unconditionally set by FRESS (except for the Display-Line size which is a default -- see below). Any null entries default to the value in effect before FRESS was invoked.
- Terminal refers to a commonly used terminal-type for that version.
- Version: because FRESS is a stream-oriented editor, special buffer sizes need to be defined to limit the scope of display

and editing commands. The Editing Buffer for all versions is 2100 characters; the size of the Display Window is version dependent. The display line refers to the default line width (which can be changed using the SDISPLAY command).

- Mode: Display and Edit Modes are discussed in Section 2.3).
- Logical character delete, logical line delete, logical line end, and logical escape characters are discussed in Section 3.
- Special keys are assigned to represent a not sign in several versions because some terminals do not have an explicit "-" key. The special meaning for this character is discussed in Appendix A.

### 2.2.1 Explanation of the "T"-Version

The "T" version is for use on terminals which display only upper-case characters.

- On upper/lower case terminals, Labels and format codes (see Section 4) are displayed in the case in which they are entered. On upper-case-only terminals, however, there is no distinction made when these two kinds of special purpose text are displayed. That is, no percent or double percent signs are displayed to indicate upper case characters. Since it is impossible to differentiate between upper and lower case, Labels and macros should always be specified exclusively in lower case when an upper-case terminal is likely to be used. In addition, certain commands (FLIP, CAPITALIZE, and UNCAPITALIZE) will force all macro names included in their <scope>'s to lower case, making it desirable to always specify macro names in lower case.

#### 2.2.1.1 Representing Upper-Case Characters (in the "T" version)

It is possible to represent capital letters by preceding them by the special capitalize character ("%").

- Since the percent signs indicating upper-case are not actually part of the file, it is not possible to capitalize characters by inserting percent or double percent signs, or by surrounding a text string with double percent signs using the SURROUND command. The percent signs must be entered with the characters either in Command or Input Mode.
- A string of capital letters can be represented using the multi-cap delimiters ("%") at each end of the string.

- When a file is displayed, capital letters are represented using the same percent sign procedures as above.
- Numbers and punctuation (except exclamation points -- "!") may be considered upper-case characters when preceded by two or more upper-case letters (see the third example below).

#### EXAMPLES

%THE QUICK BROWN FOX  
The quick brown fox

%THIS TEXT EDITOR IS %%FRESS.%%  
This text editor is FRESS.

%P.%T. %BARNUM  
P.T. Barnum (the periods in this string are not considered upper-case characters)

#### 2.2.1.2 Pattern Matching

- Care must be taken when pattern scanning on an upper-case terminal using LOCATE or when specifying any context string. Unlike in the normal (upper/lower case) mode, the pattern specified either as a context string or LOCATE pattern must exactly match the way the text appears when displayed, including the percent signs indicating upper-case. This is because all pattern scanning is done in the Display Window (which is the text displayed on the terminal), not in the file itself. All characters in the Display Window appear in upper-case, even though some of the characters are actually lower case in the file.
- The scan for pattern matching in LOCATE commands in the forward direction starts at the second character of the Display Window. Thus, on an upper/lower case terminal, consecutive LOCATEs of the same pattern will always find different instances of the pattern. On an upper-case terminal, consecutive LOCATEs of a pattern beginning with a capital letter, but not specified as such, will find the same instance each time, since the word matching the pattern also starts at the second character of the Display Window. Therefore, leading percent or double percent signs should be included in the pattern to ensure the LOCATEs will find separate instances.

#### EXAMPLES

- The string "%SENATOR %MC%GOVERN" would not be found if the user specified "L/MCGOVERN" because of the percent sign appearing in the Display Buffer. It would be found by specifying "L/MC%G".

- Consider the string "%THIS TEXT EDITOR IS %%FRESS.%%". Because the period at the end is considered part of the multi-cap string, the last word could not be found by typing "L/%%FRESS%" since it does not appear that way in the display. It would be found by typing "L/%%FRESS.%%" or by "L/%%FRESS". In fact, it would even be found by typing "L/FRESS".

### 2.2.1.3 Literal Text

When specifying <text> parameters in commands such as INSERT, SUBSTITUTE, IBEFORE, and USUBSTITUTE, percent and double percent signs must be used to indicate upper-case characters exactly as they are used in Input Mode. Thus to correct a typographical error which caused the string "%FRXSS%" to be inserted in the file, the user might type

SU/FRXSS/%%FRESS%

Actually, the second pair of percent signs is redundant, since the multi-cap string is automatically ended by the carriage return.

### 2.2.1.4 Correcting Character Case Errors

- Often a section of text is entered in the wrong case, either by neglecting to use the proper version of FRESS or by forgetting the percent signs to indicate upper-case. In these instances, CAPITALIZE and UNCAPITALIZE may be used to correct the error. UNCAPITALIZE can be used to convert an entire file which was accidentally inputted in upper-case. If the entire file is specified as <scope>, all characters will flip to lower case except those appearing after "sentence-ending" punctuation - periods and (literal) exclamation points.

### 2.2.1.5 Summary of Rules

- 1) When pattern matching, all percent signs included inside the desired string to indicate capitalization must be specified in the pattern. If percent signs are leading or trailing, they may be omitted.
- 2) If consecutive LOCATEs for the same pattern are to be done, the leading percent or double percent signs should be included in the pattern to ensure the LOCATEs will find separate instances.



- 3) To specify a literal <text> parameter (e.g., the second parameter of SUBSTITUTE), percent and double percent signs must be included to indicate upper-case characters. The carriage return will end the last multicap string in the <text> parameter if it is not explicitly ended.
- 4) In Input Mode or when using OREAD, the end of a line will end the last multicap string on that line if it was not explicitly ended. However, this is not true in Swift Input Mode. An "unmatched" double percent sign will cause all characters to be capitalized until the user returns to Command Mode.

## 2.3 DISPLAY AND EDIT MODES

Optional Display (viewing) Modes have been incorporated into FRESS to provide concise, meaningful feedback and to allow suppression of redundant Display and Editing Modes which the user may select for verification of an editing operation (see SMODE); the defaults for each version are given in Figure 2.1. The choices are Display or Brief Mode, and Transcription or Static Mode.

### DISPLAY MODES

- Display Mode, the default setting, displays a portion of the Display Window (see SDISPLAY) after every editing and traveling operation. If commands are stacked on a single line, multiple displays will be generated.
- Brief Mode suppresses all typing at the terminal.

### EDIT MODES

- Transcription Mode facilitates transcribing changes from marked-up hard copy where an editor normally proceeds linearly in a forward direction. FRESS automatically moves the start of the user's display after each edit. The display will start at the first word preceding the point in the file at which the last edit occurred. When used in conjunction with Display Mode and a short line length, Transcription Mode is a convenient means of verifying most edits.
- Static Mode prevents the Editing Buffer from moving with each edit. This is particularly useful on a display console where many lines of text are visible at once, thus obviating the need for moving the buffer very often. To relocate the buffer in Static Mode, the user must explicitly travel.

### 2.3.1 Overriding Display and Edit Modes

- The Display and/or Edit Mode in effect may be temporarily overridden for a single command line by appending to the command line two blanks and a period ("␣␣."), and one or two optional mode indicators.

#### Mode indicators

(none of the indicators) Brief Mode: suppress all printing for all commands on the physical command line. The same effect can be achieved by immediately following the first command with a period ("."). For example, "l./dog" is equivalent to "l/dog .".

\* Suppress printing until the last command on the line.

D Use Display Mode for all commands on the command line.

S Use Static Mode for all commands on the command line.

T Use Transcription Mode for all commands on the command line.

- Several commands can be specified on one line using the Command Separator (">") (see Section 5.4).
- Individual commands in a multi-command line preceded with a "<" will be put into Brief Mode.
- A "(" preceding a command will suppress all display until a command preceded by a ")".

#### EXAMPLES

```
glabel/hello>print 3␣␣.*
```

```
will print:
%L(hello)
This is line1
This is line2
```

```
glabel/hello>print 3␣␣.D
```

```
will print:
%L(hello)
%L(hello)
This is line1
This is line2
```

[Note: the first "%L(hello)" is printed because of the "glabel/hello"]

### 3 CONTROL CHARACTERS

Since standard keyboards on terminals have only a limited number of control characters, FRESS reserves some characters to indicate special command language functions and format codes. These characters must be used as specified below.

#### 3.1 SUMMARY TABLE

Figure 3.1 summarizes the definitions and conventions for representing control characters literally in files, specifying them in command lines, or in lines while in Input Mode.

- Some characters always have special meanings (e.g., "%"), while others have special meanings in Command Mode (e.g., "=") and still others have special meanings only sometimes in Command Mode (e.g., "&").
- Any character which has a listing under "Literal Input" must be entered into the file using either the representation shown or the special character code listed under "Sp Char Code" (other characters may optionally use their special character codes).
- If the Literal Command explanation uses "<n>" (e.g., "<n>+1"), it refers to <n> occurrences of the symbol.
- If there is no entry under "Literal Command" or "Literal Input," the control character can be specified as any other character would be.

**Figure 3.1 -- Control Character Summary Table**

3.2 DESCRIPTIONS

Each character in Figure 3.1 is described in this Section. In addition to its definition, the literal representation is explained. The Special Character Code (see Section 4.6) is indicated in brackets "[", "]".

<u>CHAR</u>	<u>NAME AND EXPLANATION</u>
BSP	<p>backspace character (default -- see Section 2.2)</p> <p>DEFINITION: Entered via the backspace key on an upper/lower case terminal, or control-h on an upper-case only terminal.</p> <p>LITERAL: The backspace character is always used as a literal character, though it may not be entered as the only text in an INSERT command or input line. Leading backspaces are ignored in an input line. Backspaces are usually used to achieve overstruck characters during FULLPRINT, and should normally be entered in this context (see Section 4.6 for a more complete discussion of overstruck characters). [!22]</p>
_	<p>blank fill (underscore)</p> <p>DEFINITION: If the Key delimiter in a FRESS command is a blank, all underscores are interpreted as literal blanks.</p> <p>LITERAL: A non-blank Key delimiter must be used. [!109]</p>
%	<p>capitalize character ("percent" sign)</p> <p>DEFINITION: In a &lt;text&gt; parameter (SUBSTITUTE, INSERT, MLABEL second parameters) or in lines of Input Mode, it causes the next character to be capitalized; it is ignored if next character is non-alphabetic.</p> <p>LITERAL: !/ [!63]</p>

> command separator ("greater than" sign)

DEFINITION: The character that separates multiple commands on one line (see Section 5.4).

LITERAL: In Command Mode, use two symbols for every symbol to be represented. [!110]

? defer character (question mark)

DEFINITION: Either <scope> or <lp> can be deferred in a command specification. If either is to be completely deferred, a "?" should be specified; if one is to be partially deferred (one end of <scope>), the procedures described below under "defer separator" should be followed.

LITERAL: For pattern parameters of command lines containing only "?"s, use one more symbol than the number of symbols to be represented. [!111]

= defer separator ("equals" sign)

DEFINITION: It is possible to defer the beginning and/or the endpoints of <scope> by replacing it with "?=" or "=?", respectively. Once the deferred location has been found, it can be resolved to the system by specifying it after a "?" and a delimiter.

LITERAL: In Command Mode, use two symbols for every symbol to be represented. [!126]

... ellipsis (three periods)

DEFINITION: It is possible to identify a string by specifying the beginning and end points with ellipses ("...") in between. The endpoints must be within the editing buffer of 2100 characters. If they are not, one end can be deferred (see "defer separator" above). Note, if ellipses are specified as one end of <scope> (e.g., "c/start..."), FRESS will take a single extra character from the end of <scope> which the ellipses represent. For example, given the string "123xx456": if "c/...xx/\*" is

specified, the resulting string is  
"12\*456."

LITERAL: In Command Mode, when more than two periods must be represented, specify one more period than is needed (e.g., "...." means three periods). [!75!75!75]

! format code delimiter (exclamation point)

DEFINITION: Format codes are delimited by "!" (see Section 4).

LITERAL: !! [!61]

any Key delimiter

DEFINITION: The Key delimiter is the first character after the command and the command qualifier (if specified). If this character is a blank, the Key delimiter is the character following the blank. The Key delimiter delimits parameters of the command.

LITERAL: Within a command parameter, the Key delimiter can be used literally only in the last parameter; hence a different Key delimiter should be used if this character is needed in earlier parameters.

@ logical character delete ("at" sign, CMS default -- see Section 2.2)

DEFINITION: In all user typed lines, <n> @ signs "erase" the previous <n> non-@ sign characters.

LITERAL: [!124]

ESC logical escape ("double quote" sign (")), CMS default -- see Section 2.2)

DEFINITION: It is possible to represent a CMS line editing character (logical character delete, logical escape, logical line delete, logical linend) literally by preceding it with the logical escape character. For example, if the escape character is "\*" and the logical linend

character is "#", the string "\*#" would be interpreted as the literal character "#".

% logical hyphen ("percent" sign)

DEFINITION: If an Input line ends with this character, the usual blank character (to separate words) will not be appended to the line.

LITERAL:   !/  [!63]

¢ logical line delete ("cent" sign, CMS default -- see Section 2.2)

DEFINITION: In all user typed lines, it "erases" all characters preceding it on the current line, up to and including the last logical linend character.

LITERAL:   [!74]

# logical linend ("pound" sign, CMS default -- see Section 2.2)

DEFINITION: The logical linend character is equivalent to a carriage return for stacking multiple commands on a user-typed line (see Section 5.4). This character can also be used to get into the CP environment, or simply to execute a CP command. If "#CP" is issued, the user is put into the CP environment, while if "#CP <cp command>" is issued, the CP command <cp command> is executed (and the user remains in the FRESS environment). This is true in both Command and Input Mode.

LITERAL:   [!123]



## &amp; LP character (ampersand)

DEFINITION: A different character other than the last character of a string can be specified as the LP character when an ampersand is used in <scope> or <lp> (see Section 5.3).

LITERAL: Use <n>\*3 symbols in <scope> or <lp> parameters, <n> symbols all other times. [!80]

## | LP separator ("or bar")

DEFINITION: Line and character displacements into the editing buffer can be specified before a given command is executed in conjunction with this character (see Section 5.3).

LITERAL: In pattern parameters of command lines only if it follows one or two signed numbers at the beginning of the parameter, <n>+1 symbols represent <n> symbols. [!79]

## %% multi-cap delimiter (double "percent" signs)

DEFINITION: In an input string or a single line of Input Mode, all alphabetic characters are capitalized between a beginning and ending %%; if the second %% is left off, the remainder of the string or line is capitalized; if the second %% is left off in OREAD or Swift Input Mode, then the input is capitalized until another %% is encountered on another line or until Input Mode is exited (see Section 2). In Command Mode, if %% is specified, a carriage return implies a closing %% if it wasn't specified.

LITERAL: !/!/ [!63!63]

¬ special blank ("not" sign)

DEFINITION: During FULLPRINT, a "not" sign ("¬") is translated to a blank but is regarded as a text character by the justification algorithm (see Appendix A).

LITERAL:    !¬   [!62]

% Structure code delimiter ("percent" sign)

DEFINITION: All Structure (Hypertext) is delimited by "%", which means that no "%" can appear literally in a file.

LITERAL:    !/   [!63]

- qualifier separator (hyphen)

DEFINITION: There are five qualifiers that can modify the <scope> of a command when the command is followed directly (no intervening blanks) by a hyphen ("-") and one of the modifiers: "b"-Block, "c"-character, "l"-line, "o"-Structure order, or "w"-word (see Section 5.5 for a complete description).

LITERAL:    no restrictions [!96]

## 4 FORMATTING

### 4.1 OVERVIEW

FRESS has five general types of formatting codes. The format of each is given in the box to the left of each description below. The general format is:

`!<delimiter><code><data><delimiter>`

#### Alter codes

`!+<code>[<data>]+`

This type of code is used to specify long-lasting formatting specifications, such as page margins and line width. Alter codes can also be used to print the current date. An alter code takes effect on the first physical page of output in which it is imbedded, thus, if there are two of the same type of alter codes which conflict on the same page, the last will override. Example: `"!+settab1=15;2=25+"` (set tab stops at

#### Edit codes

`!-<code>[<data>]-`

This type of code is used for local text formatting such as starting new paragraphs, new lines or pages, tabbing, indenting, and centering. Edit codes affect only their unique place in the text. Example: `"!-s3`

Macro codes

!.<macroname>.

These codes are user-defined. They can be any mixture of format codes and/or text strings; a common usage is as an abbreviation for a long text string  
Example: "!.usa." (to abbreviate United States)

Special Character codes

!<nnn>

These codes allow the user to enter into the file, system delimiters and characters not available on a standard keyboard. Example: "!175" specifies

Underscore codes

!(0<data>!)

When these codes surround a text string, it is underlined during FULLPRINT.  
Example: "!(0underlined!)" prints as  
"underlined"

## 4.2 NOTATION FOR FORMATTING CODE SPECIFICATIONS

Only some of the formatting codes take effect online, but all are effective when FULLPRINT is issued, even when routed to the terminal (2741 option).

The edit and alter codes are shown in upper case for clarity, but all literal codes and parameters may be specified in upper and/or lower case.

- The following conventions are followed for <data> fields:

n	numeric parameter (arbitrary number of digits)
x	indexed parameter - used as an index into a table
[;<parm>...]	an optional parameter which may be repeated as many times as desired (including the indicated delimiter). For example, given the specification: "THE INVITEES ARE <guest>[AND <guest>...]," the user could specify "THE INVITEES ARE PAUL" or "THE INVITEES ARE JOHN AND GEORGE AND PAUL" or "THE INVITEES ARE PAUL AND JOHN AND GEORGE AND RINGO," etc.

- Each format code description includes examples in which the literal code is usually shown underlined, and a fully formatted version or an explanation follows right beneath it. For example,

!+settab1=15;2=25+

"+" is the delimiter, "settab" is the code, and "1=15;2=25" is the data

### 4.3 ALTER CODES

Figure 4.1 summarizes several properties and the default values of the codes explained in this section. A "■" marked under "TERM" indicates that the code takes effect online (at least starts a new line), while a "■" marked under "FU" means that code's effect begins on the next line in the formatted (FULLPRINTed) output. A "□" means that the code does not take effect either online, or when the file is FULLPRINTed (depending on the column).

CODE	PURPOSE	DEFAULT	NEW LINE?	
			FU	TERM
!+B+	blank underscore mode	H	□	□
!+COLUMN+	set column mode	1	■	□
!+DATE+	print the date	-	□	□
!+DEPTH+	set page depth	66	■	■
!+DI+	set decimal label indent	3	□	□
!+GRID+	draw a grid	-	□	□
!+GUTTER+	set column separator	3	■	■
!+HEAD+	select heading table	B	□	■
!+JUST+	set justification	C0;L0	□	□
!+MARGIN+	set margins	0,0,9,6	■	■
!+OFFSET+	set page offset	0	■	■
!+PARA+	set paragraph	5,1	□	□
!+SETTAB+	set tabular columns	10*index	□	□
!+SPACE+	set line spacing	1	□	■
!+TABLE+	set table mode	10*index by 10	■	□
!+TITLE+	set running titles	-	□	□
!+TOFC+	set Table of Contents	all headings	□	□
!+WIDOW+	set widow depth	2	□	□
!+WIDTH+	set page width	65	■	■

Figure 4.1 -- Alter Code Properties

### NOTES

- Alter codes must appear by themselves -- there is no keystroke saving method of combining them.
- "Multi-column mode" refers to text formatted into more than one column (see !+COLUMN+), while "table mode" refers to text under control of the !+TABLE+ alter code.

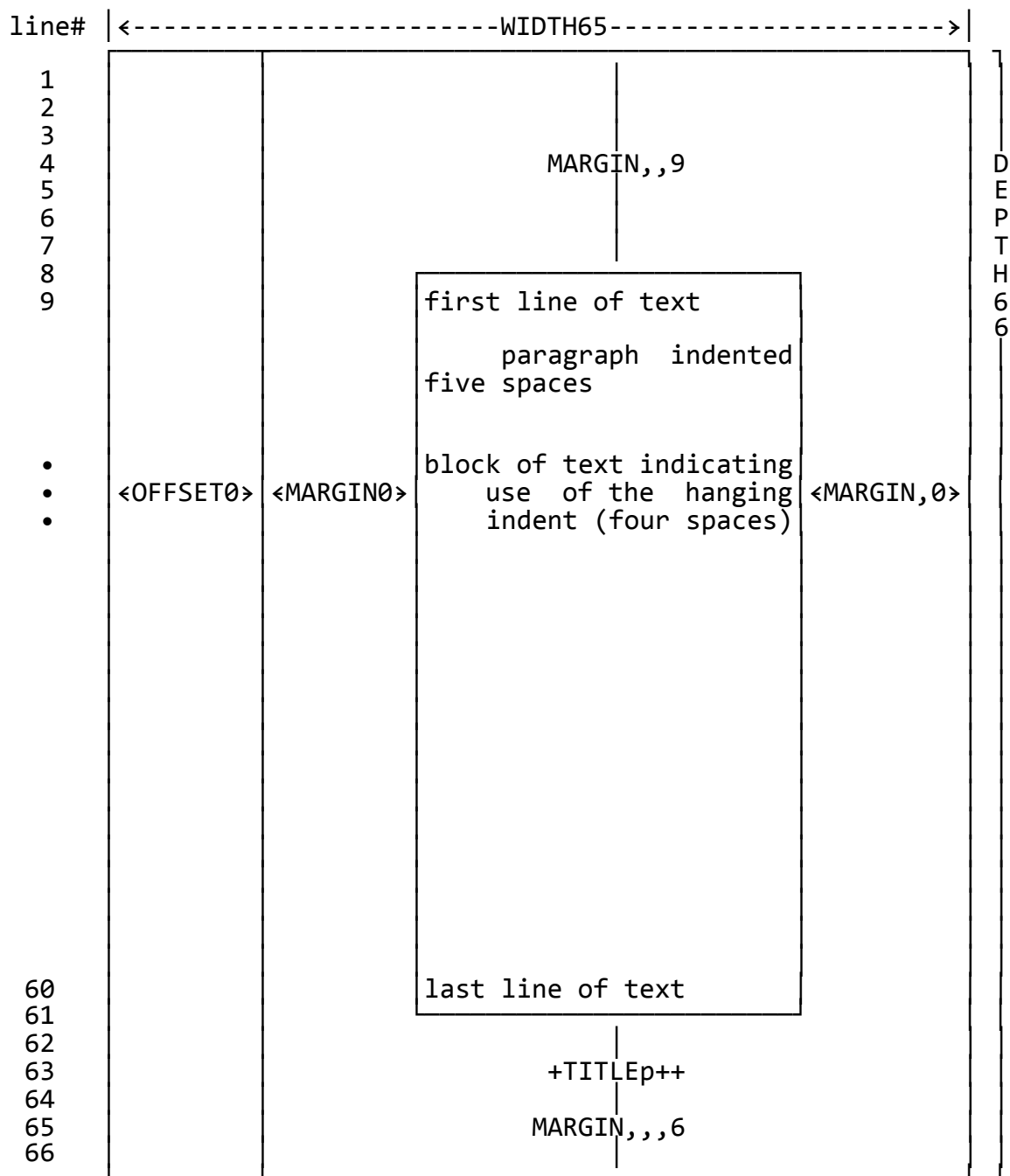


Figure 4.2 -- Page Layout Showing Alter Code Defaults

!+B+

!+B+      set the blank underscoring mode

This code determines which blanks will be underscored within an underscored text string.

!+B<mode>+  
           H  
           L  
           U

<mode>

a letter indicating which blanks within a string to be underscored will also be underscored

H...only those blanks in headings will be underscored

L...no blanks will be underscored

U...all blanks will be underscored (including those in headings)

## NOTES

- Underscores can either be inserted manually into the text (see Section 4.7) or by surrounding the text to be underscored with "!(0" and "!) " on the left and right respectively.

## EXAMPLES

[given]=> an underscored text string

!+BL+  
     an underscored text string

!+BU+  
     an underscored text string



!+COLUMN+

!+COLUMN+ set the number of text columns

The text will henceforth be formatted into <n> columns.

```

!+COLUMN<n>+
      1
      2
      3
      4
      5
    
```

<n>...the number of columns into which the text is formatted (one to five)

### NOTES

- The manner in which multiple columns are formatted on a page is determined by the <col> parameter of the !+JUST+ alter code.
- In Multi-Column Mode, !+WIDTH+ is segmented into several sections: a binding offset which is taken from alternate sides of the page (see the !+OFFSET+ alter code), text columns, and space(s) of equal size between columns. Each space is called a "gutter."
- The width of each column can be calculated using the following formula:

$$(!+WIDTH+ - !+OFFSET+ - !+GUTTER+ *(n-1))/n$$

- The following diagram shows how one or two columns would be laid-out in relation to other alter codes (defaults shown):

```

|<-----!+width65+----->|
|<--!+offset0+-->|<-----!+column1+----->|
|                  |<--!+column2+-->|<--!+gutter3+-->|<--!+column2+-->|
  
```

WARNINGS

- Multi-Column Mode has bugs. It should only be used if the following points are kept in mind:
  - If multi-column mode is to be used, some text in single-column mode (the default) must precede the text to be formatted in multiple columns.
  - It is best to have a left-margin of zero (!+margin0+).
  - Footnotes should not be used if <n> is greater than two. In !+column2+ mode, footnotes will be placed at the foot of the column in which they appear. The footnote margins will be within those of the column.
  - The column mode should not be changed on the last line (or near the bottom) of the page.
  - The !-T- and !-U- edit codes may cause FULLPRINT to format incorrectly in Multi-Column Mode.

EXAMPLES

- The default values of !+WIDTH+, !+GUTTER+ and !+OFFSET+ (65, 3 and 0 respectively) in combination with !+COLUMN2+, would form two 31 character columns.

(formatted text)

Text-processing is the system by which document preparation is assisted by machine. A	computer provides several advantages when an author goes "online", the biggest of which is that once the first draft is	entered into the computer, one need never retype an entire document; only errors have to be corrected.
--	---	---

(unformatted text)

```
!+column3+!+gutter6+
  !-p-Text-processing is the system...corrected.!+column1+
```

!+DATE+

!+DATE+    print the current date

The current date will be printed where this code appears in the text.

!+DATE<format>+  
    1  
    2

<format>

The format in which the date is to be printed

- 1...as "mm/dd/yy"
- 2...as "month dd, 19yy"

### NOTES

- This code behaves more like an edit code than an alter code in that the date appears only once (where the code is specified).
- The current date can also be printed as part of a title by imbedding "&1" or "&2" into a title string (see !+TITLE+).

### EXAMPLES

Today is !+DATE1+  
Today is 01/05/19

Today is !+DATE2+  
Today is January 5, 1919

!+DEPTH+

!+DEPTH+    set the number of text lines

This code determines the number of text lines per printed page.

!+DEPTH<lines>+  
66

<lines>...the number of lines per page desired

#### NOTES

- The actual number of lines available for text in the current column or page is equal to the !+DEPTH+ code minus the top and bottom margins.
- The size of <lines> is governed by the !+PAGE+ alter code.

#### EXAMPLES

- !+DEPTH66+ is the correct value for paper 11" long, printed 6 lines per inch (unless it is desired to print over page boundaries). [!+DEPTH66+ was used for this manual.]

!DI+

!DI+      set the decimal block indentation level

This code sets the left margin indentation for decimal blocks.

!DI<indent>+  
    3

<indent>...the margin increment between decimal levels

#### NOTES

- An additional (n-1) \* <indent> spaces is added to the left margin of every Decimal Block of level <n>.
- Decimal block margins (and heading levels) can be set independently of the !DI code, by using the "Macros" option of FULLPRINT (and the appropriate system-defined format macros -- see Section 4.7).

#### EXAMPLES

##### !DI3+ (the default)

The left margin of decimal block '1.1' will be 3 greater than that of block '1', the left margin of block '1.1.1' will be 6 greater than that of block '1', etc.

##### !DI0+

The left margins of all decimal blocks will be the same (unless altered by !MARGIN alter codes). This enables more text to fit on a page.

!+GRID+

!+GRID+     draw a grid

The !+GRID+ alter code is a facility which allows the FRESS user to represent graphs, tables, charts, etc. graphically.

!+GRID<grid>+

<grid>...the coded representation of the table to be drawn

#### NOTES

- A complete discussion of how to create !+GRID+ codes is given below under "GRID CONSTRUCTION," "HORIZONTAL EXPANSION," and "VERTICAL EXPANSION." Reference tables for each grid character and its expansion code are supplied under "REFERENCE TABLES." Examples are found under "'PIECES' of GRIDS," and "BOXES AND GRIDS". "STEP-BY-STEP EXAMPLE" demonstrates how a grid code is created.
- As with the box format code (!-B-), text following the !+GRID+ code begins in the upper left-hand corner of the grid. Therefore, the user will usually skip a line before any text is entered.
- Text typed inside a grid will replace any grid characters in the same position. This is also true of special blanks ("-"), but not regular blanks.
- A !+GRID+ code acts as a conditional page. Consequently, if a new page is forced, the grid will be left justified. The user should take care to allow for this, perhaps by explicitly adding a conditional page code (!-K-) before the !+GRID+ code.
- Besides generating the various tables shown below, !+GRID+ is also useful for constructing diagrams such as flowcharts and decision trees.

Reference Tables

- A grid is a combination of special characters (left corner, right corner, etc.) known collectively as grid characters. Since most of these grid characters are not available on typewriter keyboards, they must be represented by grid character symbols. The tables which follow are designed to aid the user by summarizing much of the information given below. Figure 4.3 illustrates the grid symbols and the characters they represent; Figure 4.4 shows all the expansion characters. (Recall that the symbol "ø" denotes a blank.)

┌	┐	└
├	┤	┴
┬	┴	┴
ø	-	

<=>

A	B	C
D	E	F
G	H	I
ø	-	

Figure 4.3 -- Grid Characters and Character Symbols

S	C	H	V
A	┌	-	
B	┐	-	
C	└	ø	
D	├	-	
E	┤	-	
F	┴	ø	
G	┬	-	ø
H	┴	-	ø
I	┴	ø	ø
ø	ø	ø	ø
-	-	-	ø
		ø	

Meaning of column headings:

S -- symbol  
C -- grid character  
H -- horizontal expansion character  
V -- vertical expansion character

Figure 4.4 -- Vertical and Horizontal Expansions of Grid Characters

Grid Construction

```

GRID ALTER CODE   = !+GRID<grid>+
<grid>            = <subgrid>
                  = <subgrid,grid>

<subgrid>         = <vert iteration> (<row>) <vert expansion>
<row>             = <string>
                  = <string,row>

<string>          = <hor iteration> <symbol> <hor expansion>
<vert iteration>  = 1|2|3|4|5|6|7|8|9|0|null
<vert expansion>  = 1|2|3|4|5|6|7|8|9|0|null
<hor iteration>   = 1|2|3|4|5|6|7|8|9|0|null
<hor expansion>   = 1|2|3|4|5|6|7|8|9|0|null

<symbol>          = A|B|C|D|E|F|G|H|I|-| | or |

```

Figure 4.5 -- Grid Code Formal DefinitionNOTES

- A grid is a combination of special characters (left corner, right corner, etc.) known collectively as grid characters. Since most of these grid characters are not available on typewriter keyboards, they must be represented by grid character symbols.
- An important concept employed by the !+GRID+ code is one of horizontal expansion. FRESS expands a grid character by placing one or more expansion characters after the grid character when the grid is printed. Figure 4.4 lists the characters used to expand each grid character under the "H" column.
- It is also possible to vertically repeat (iterate) a whole line (row) of characters at one time.
- After complete expansion of a grid, the line pointer is at the start of the !+GRID+ code (while after a sub-grid, it is after the sub-grid just drawn).



## Horizontal Expansion

Horizontal expansion is specified in the `!+GRID+` code by placing a number after the symbol to be expanded. Thus, the symbol "A" is interpreted as the grid character "┐" by FRESS (see Figure 4.3). The string "A3" is interpreted to mean, "using the appropriate horizontal expansion character, extend the grid character represented by the symbol "A" to a length of 3 characters." By specifying "A3," therefore, the user causes "┐" to be printed. Similarly, "E5" results in "┌───" and "┌2" results in "┌" (horizontal expansion characters can be found in Figure 4.4).

## Iteration

The pattern described by a grid character symbol and its expansion may be repeated by preceding the symbol with an iteration number. Thus, "3E2" means that the pattern represented by the symbol and expansion number "E2" is to be printed three times, or

The row

A pattern specification such as "A," "4D3," and so on, is called a string. Strings may be combined, if they are separated by commas, to specify horizontal patterns more complex than the ones we have seen so far. For example, "A2,3B2,C" results in

--	--	--	--	--

The string or strings used in a `!+GRID+` code are always enclosed in parentheses; the combination of strings inside parentheses is called a row. A `!+GRID+` code must consist of at least one row, and that row must consist of at least one string.

Vertical Expansion

Whereas grid characters may be expanded individually in the horizontal direction, vertical expansion applies to an entire row. Vertical expansion is specified much the same way as horizontal expansion is: the row to be expanded is followed by a vertical expansion specification. (Vertical expansion characters are shown in Figure 4.4 under the "V" column.)

Thus, while the row "(A2,3B2,C)" results in

```

┌───┐
│   │
└───┘

```

"(A2,3B2,C)3" results in

```

┌───┐
│   │
│   │
│   │
└───┘

```

The row has been expanded to a length of three in the vertical direction.

Vertical iteration

A row and its expansion can be repeated in the vertical direction by means of a vertical iteration number placed before the row; this is analogous to the method used to extend a symbol horizontally. Thus, specifying "3(A2,3B2,C)2" causes the following to be printed:

```

┌───┐
│   │
└───┘
┌───┐
│   │
└───┘
┌───┐
│   │
└───┘

```

- It is important to keep in mind that expansion takes place before iteration, so that it is the expanded grid character which is repeated.
- A row and its modifiers (i.e., vertical iteration and expansion specifications) are called a subgrid. A grid is made of one or more subgrids, which are separated by commas. The first box of Figures 4.3 and 4.4 were created with the following !+GRID+ codes respectively:

```

!+grid(a4,2b4,c)2,3(d4,2e4,f)2,(g4,2h4,i)+
!+grid(a4,3b4,c)2,(d4,3e4,f)13,(g4,3h4,i)+

```

## RESTRICTIONS

- `!+GRID+` codes cannot be preceded or followed by normal characters (use tab codes).
- No more than 106 characters are allowed in a `!+GRID+` code, nor may a `!+GRID+` code draw a grid wider than 106 characters.

### HINTS

- Use iteration characters in `!+GRID+` codes wherever possible. For example, the following two codes are equivalent, but the second form is preferred:

```
!+grid(a,-3,f 6,d,-4,c)+
!+grid(a4,f6,d5,c)+
```

Both produce:

- If an "INVALID PARM" message appears for a !+GRID+ code which appears to be correct, start INSERTing 1's where they are implied in the code as iteration or replication factors, until the error message goes away. This problem generally occurs only with long, complex codes. As an example of "implied 1's", the following two codes are equivalent:

```
!+grid(a3,b,c)+
!+grid1(1a3,1b1,1c1)1+
```


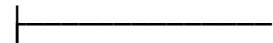
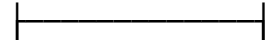
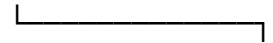



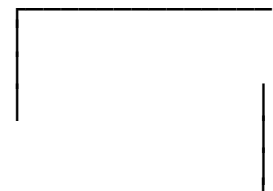
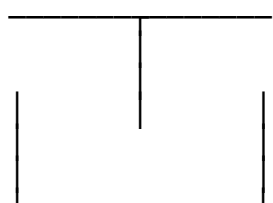
Both produce:

## EXAMPLES

- A 32-character horizontal line could be created with either of the two codes: `"!+grid(32-)+"` or `"!+grid(-32)."`
- A vertical line of 32 characters could be created with the code: `"!+grid(|)32+."`
- More examples are given below under "'PIECES' OF GRIDS", and "BOXES AND GRIDS."

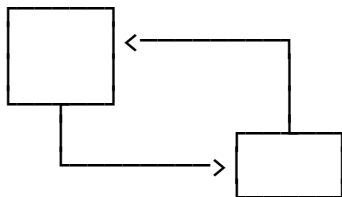
"PIECES" OF GRIDS

The following example show various "pieces" of grids, along with the !+GRID+ codes used to generate them.

	!+grid(15-)+
	!+grid(d15)+
	!+grid(d14,f)+
	!+grid(g14,c)+
	!+grid( )5+
	!+grid(b)5+
	!+grid(b)4,(g2)+
	!+grid(a15)4+
	!+grid( 14, )3,(14-,i)+
	!+grid(7-,b,7-)4+
	!+grid( 14, )3,(g14,i)+

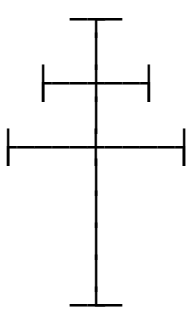
BOXES AND GRIDS

The following examples show how box (!-B-) and grid (!+GRID+) codes can be intermixed. Assume that the leftmost line is in column 0. In the explanations below, grid characters are indicated by superscripts (illustrated at the bottom of the page).



```
[1] !-s2;b7,4-
[2] !-s0;t*7-<!3+grid(8-,c)3+
[3] !-s1;t*3-!1+grid(b)2+
[4] !-s0;t*13;b7,3-!8-t*16-!3+grid(h)+
[5] !-s0;t*3-!7+grid(g9)+!9-t*12->
```

- [1] skip 2 lines; draw a box 7 characters wide, 4 characters deep.
- [2] skip to a new line; start at column 8; draw a left arrow-head ("<sup>3</sup><"); horizontally expand a dash ("-") 8 places; vertically expand upper-right-box character("<sup>3</sup>") 3 lines.
- [3] skip 1 line; start at column 3; vertically expand a middle-box-down character("<sup>1</sup>") 2 characters.
- [4] skip to a new line, start at column 13; draw a box 7 characters wide and 4 characters deep; go to the 17th column and add a middle-box-up character("<sup>8</sup>").
- [5] skip to a new line; start at column 4; horizontally expand a lower-left-box("<sup>7</sup>") 9 characters, go to column 13, draw a right arrow-head ("<sup>9</sup>>").



```
[6] !+grid( 4,-,b2)2,( 2,d3,e3,f),( 5,|)+
[7] !-s3-!5+grid(d5,e5,f),( 5,|)4,( 4,-,h2)+
```

- [6] vertically expand for 2 lines [4 blanks, a dash ("-"), a middle-box-down("<sup>1</sup>") horizontally expanded 2 characters]; 2 blanks, horizontally expand a right-cross("<sup>4</sup>") 3, horizontally expand a cross("<sup>5</sup>") 3, left-cross("<sup>6</sup>"); 5 blanks, vertical marker ("|").

- [7] skip 3 lines; horizontally expand a right-cross("<sup>5</sup>") 5, horizontally expand a cross("<sup>6</sup>") 5, left-cross("<sup>6</sup>"); vertically expand for 4 lines [5 blanks, vertical-marker ("|")]; expand horizontally [4 blanks, dash ("-"), middle-box-up("<sup>8</sup>")] 2 characters.

(1)...	(1)...	(3)...
(4)...	(5)...	(6)...
(7)...	(8)...	(9)...

STEP-BY-STEP EXAMPLE

The following is a step-by-step example of how the square grid used in Figure 4-3 was produced. In each step, additions to the GRID code are underlined.

1. The grid was to start in column 24, after skipping two blank lines on the page. Text within the grid was to start in column 26. (discussion is limited to the grid on the right in Figure 4-3.) To position the grid correctly, the following edit and alter codes were used:

```
!+settab1=24;3=30;4=34+
!-s2;t1-!+grid...
```

2. To "draw" the top line, the grid symbols A, B, and C were needed. A and B were expanded to a length of 4 characters in the horizontal direction, and the pattern generated by the string B4 was needed twice (an iteration factor of two).

GRID code: !+grid(a4,2b4,c)+

Generates:

```
┌───┬───┬───┐
```

3. The top line of the grid had to be expanded to a length of two in the vertical direction; this was done by placing the vertical expansion number "2" after the row.

GRID code: !+grid(a4,2b4,c)2+

Generates:

```
┌───┬───┬───┐
└───┘
```

4. Using the grid symbols D, E, and F, a horizontal subgrid was added to the grid. As in step 1, symbols D and E were expanded and an iteration number was used to generate the pattern represented by "E4" twice.

GRID code: !+grid(a4,2b4,c)2,(d4,2e4,f)+

Generates:

```
┌───┬───┬───┐
├───┤───┤───┤
└───┘
```

5. The horizontal subgrid was then expanded to a length of 2 in the vertical direction.

GRID code: `!+grid(a4,2b4,c)2,(d4,2e4,f)2+`

Generates:


6. A vertical iteration factor of three was used to indicate that the expanded horizontal subgrid was to appear three times.

GRID code: `!+grid(a4,2b4,c)2,3(d4,2e4,f)2+`

Generates:


7. Finally, the grid was "closed" by using the symbols G, H, and I, which were expanded and iterated as above.

GRID code: `!+grid(a4,2b4,c)2,3(d4,2e4,f)2,(g4,2h4,i)+`

Generates:


8. Text was entered into the grid by entering the following lines immediately after the `!+GRID+` code:

```
!-s0;t2-A!-t-B!-t-C
!-s1;t2-D!-t-E!-t-F
```

and so on. Note that Skip edit codes (`!-S-`) were used in such a way as to avoid overlaying the lines of the grid.

`!+GUTTER+`

`!+GUTTER+` set gutter size between columns

This code determines the amount of space (the "gutter") between columns.

`!+GUTTER<separation>+  
    3`

<separation>...the spacing between columns

#### EXAMPLES

`!+GUTTER4+!+COLUMN2+`

Starting at the point where these codes are located, the text will be formatted into two columns of equal width: the first column from position 0 to 30, the second column from position 34 to 64 (the "gutter" is four spaces).



!+HEAD+

!+HEAD+    define heading

This code will redefine one or more entries of heading Table B (see !+HEAD+ select heading table).

```
!+HEAD<headdef>[;<headdef>...]+
<headdef>: <level>=<sum>,<before>,<after>
```

<level>....the heading  
<sum>.....the characteristics of the heading, as described  
by the sum of any of the following:

```
0  <nothing>
1  preceded by a new page
4  CAPITALIZED
32 underscoring
```

<before>...the number of lines to skip before the heading  
<after>...the number of lines to skip after the heading  
level to be redefined

### WARNINGS

- Great care should be taken when using this code. Minimal error checking is done and invalid formats may cause FULLPRINT to fail.

### EXAMPLES

- An expansion of each heading table can be found in the "!+HEAD+ select heading table" description.
- The two heading codes: !+HEAD1=37,0,5+ and !+HEAD2=36,2,1+ can be combined into one with the same effect: !+HEAD1=37,0,5;2=36,2,1+.

!+HEAD+

!+HEAD+      select heading table

This code selects the table into which the !-H- edit codes are indexed.

!+HEAD<table>+  
           A  
           B

<table>

A...index into Table A

B...index into Table B

	HEAD LEVEL	ALL CAPS	UNDER- SCORE	LINES BEFORE	SKIPPED AFTER
<u>Table A:</u> (system)	1	X	X	New Page	5
	2	X	X	2	1
	3	X		2	1
	4		X	1	1
	5	X	X	1	0
	6		X	1	0
<u>Table B:</u> (user)	1	X	X	New Page	5
	2	X	X	3	2
	3	X		3	2
	4		X	3	2
	5	X	X	1	0
	6		X	1	0

Figure 4.6 -- Heading Table Characteristics

NOTES

- The six heading levels (!-H- edit codes as well as Decimal Block headings) are characterized by either Table A or Table B as described in Figure 4.6.
- Table A (the system defined table) will always be the same, but the user can redefine Table B through the use of !+HEAD+ define heading codes.

EXAMPLES

- If !+HEADB+ is specified, each subsequent !-H2- code would force the heading text to upper case, underscore it and skip three lines before and two lines after the heading text.
- Table A can be thought of as being defined by the following codes: !+HEAD1=37,0,5+, !+HEAD2=36,2,1+, !+HEAD3=4,2,1+, !+HEAD4=32,1,1+, !+HEAD5=36,1,1+, !+HEAD6=32,1,0+.
- Table B defaults to by the following codes: !+HEAD1=37,0,5+, !+HEAD2=36,3,2+, !+HEAD3=4,3,2+, !+HEAD4=32,3,2+, !+HEAD5=36,1,0+, !+HEAD6=32,1,0+.

!+JUST+

!+JUST+    set justification mode

This code sets the column and line justification modes.

```

!+JUST<line>+

        or

!+JUST<col>[;<line>]+
      C0      L0
      C1      L1
      C2      L2
      C3      L3
    
```

```

<col> -- column justification:
      C0 -- full widow elimination, padding and balancing
      C1 -- no column padding
      C2 -- no widow elimination (implies C1)
      C3 -- no column balancing (implies C1 and C2)

<line> -- line justification
      L0 -- fully justified lines
      L1 -- bell justification (lines centered, no blanks added)
      L2 -- text set flush left with ragged right margin
      L3 -- text set flush right with ragged left margin
    
```

### NOTES

- For a more complete discussion of the justification algorithms, see Appendix A.
- When changing justification on the same physical page, skip codes (or possibly other edit codes which start new lines) must precede the text with the new justification. In other words, the new justification [2] is not invoked until a new line is started. The order of the required format codes must be:

```

(!+JUST+ code [2]) !-S<n>-(the lines to be justified under
                        !+JUST+ [2])
(!+JUST+ code [1]) !-S<n>-(the rest of the text)
    
```

WARNINGS

- <col> may not work properly, see !+COLUMN+ for other warnings.

EXAMPLES!+JUSTL0+

This is an illustration of full justification, the default. As can easily be seen, the left edges are all aligned, as are all of the right edges.  
This is the start of a new line.

!+JUSTL1+

This is an illustration of bell justification.  
Lines are centered without any blanks added. As can be seen, the left edges are ragged, as are the right.  
This is the start of a new line.

!+JUSTL2+

This is an illustration of left justification only. No extra blanks are added to pad out the right margin.  
As can be seen, the left edges are justified, but the right edges are ragged.  
This is the start of a new line.

!+JUSTL3+

This is an illustration of right justification only. No extra blanks are added to pad out the left margin. As can be seen, the right edges are justified, but the left edges are ragged.  
This is the start of a new line.

!+MARGIN+

!+MARGIN+ set text margins

This code can be used to reset the number of blank spaces left between the text and column boundaries.

!+MARGIN[<lm>][, [<rm>][, [<tm>][, [<bm>]]]]+  
                   0          0          9          6

<lm>...left margin  
 <rm>...right margin  
 <tm>...top margin  
 <bm>...bottom margin

NOTES

- The specification for this code can be thought of as: "!+MARGIN<lm>,<rm>,<tm>,<bm>+" except that at least one operand must appear, and only missing operands preceding an operand that does appear must be represented by commas (","). For example, "!+MARGIN6+" simply sets the left margin to be 6 without disturbing any other settings.
- In Multi-Column Mode, the four margins apply to each column.

EXAMPLES

!+MARGIN0,0,5,5+  
 reset the left and right margins to zero and the top and bottom margins to five

!+MARGIN,,5,5+  
 reset only the top and bottom margins [note the inclusion of a comma for <rm>]

!+MARGIN0,0,7+  
 appropriate margins for thesis paper [note, it was not necessary to include a comma for <bm> to default]

!+OFFSET+

!+OFFSET+ set binding offset

This code subtracts an additional margin from the line width (see !+WIDTH+ alter code).

!+OFFSET<margin>+  
0

<margin>...the number of spaces to be subtracted

NOTES

- The !+OFFSET+ is taken alternately from the left and right sides of the pages (odd and even pages respectively).
- !+OFFSET+ is generally used to leave room for binding the printed material.
- The "Offset" option of FULLPRINT should be used to indent (or center) an entire document uniformly from only the left side of the page. The !+MARGIN<lm>+ code could also be used.

WARNINGS

- The !+OFFSET+ code does not affect title codes. The user must force the title string to be offset the correct number of blanks by including special blanks ("-") in the title string. For example, to impose the effect of an !+OFFSET5+ on the title code "!+title0+Introduction\*\*+," the title code should be changed to "!+title0+-----Introduction\*\*+." The title is shifted so it appears in the margin aligned with the text.

!+PAGE+

!+PAGE+      define output buffer page size

This code sets the output buffer page size.

!+PAGE<width>,<depth>+  
          110      66

#### NOTES

- When the "Large" FULLPRINT option is specified, the !+PAGE+ alter code parameters may be increased up to !+PAGE132,136+.
- Except for the case described above, the default value should be sufficient for all applications and should not be changed.

#### RESTRICTIONS

- The !+PAGE+ parameters are constrained by the following formula:

$8192 > (\text{<width>} + 8 \text{ rounded up to be divisible by } 4) * (\text{<depth>} + 1)$

#### EXAMPLES

!+PAGE110,66+  
    this is the default



!+PARA+

!+PARA+      define paragraph

This code sets the number of spaces indented, and the number of lines skipped by the !-P- code.

!+PARA[<indent>][,<lines>]+  
                  5                  1

<indent>...number of spaces to indent  
<lines>....number of lines to skip before the paragraph

#### NOTES

- If !+PARA+ is specified without either <indent> or <lines>, that parameter will default to the value in the last specification (or the initial default given above).

#### EXAMPLES

!+PARA3,0+

Each new paragraph will be indented 3 spaces, but will not be preceded by any blank lines.

!+PARA0,1+

Each new paragraph will be preceded by one blank line, but will not be indented. This code is useful for formatting text into blocks.

!+SETTAB+

### !+SETTAB+ set tab stops

This code resets the table of tab stops indexed into by the  
!-T<x>- edit code.

```
!+SETTAB<tabstops>;<tabstops>...]+
<tabstops>: <index>=<col>[<mod>]
                C
                L
                R
```

<index>...the tab number

<col>.....the column in which the tab stops

<mod>.....a positional modifier:

C...the text is centered at the tab stop

L...the leftmost part of the text is at the tab stop

R...the rightmost part of the text is at the tab stop

### NOTES

- If no !+SETTAB+ appears in the text, the tab stop table defaults to:

"!+SETTAB1=10;2=20;3=30;4=40;5=50;6=60;7=70;8=80;9=90;10=100+"

EXAMPLES!+SETTAB3=10;4=20+

resets the third tab stop to column 10 and the fourth tab  
stop to column 20

[given]=> !+SETTAB5=30<mod>+

!-T5-This when <mod> equals R

This when <mod> equals R

!-T5-This when <mod> equals C

This when <mod> equals C

!-T5-This when <mod> equals L

This when <mod> equals L

Note: The above text segments could also have been positioned by a  
!-t5r-, !-t5c-, and !-t5l- code respectively, instead of the  
modified !+SETTAB+ and "-T5-" code combination. The !+SETTAB+ is  
for a more permanent alteration.

!+SPACE+

!+SPACE+ set line spacing

This code sets the line spacing henceforth from where it is encountered.

!+SPACE<spacing>+  
1

<spacing>...a number from 1 to 14. <spacing> - 1 blank lines are inserted between formatted lines of text.

#### NOTES

- !+SPACE+ sets the number of allowed widow lines to twice <spacing> (See !+WIDOW+ code).
- When line spacing is changed, the !+WIDOW+ value is reset to the default value of twice the current spacing, overriding any explicit !+WIDOW+ code already in effect (if a different value is desired, a new !+WIDOW+ code must be inserted).

#### EXAMPLES

!+SPACE1+  
single spacing

!+SPACE2+  
double spacing

!+TABLE+

!+TABLE+ define table of tab column boundaries

This code puts the user into Table Mode: text is formatted into one or more columns bounded as specified, which are directly accessible using a tab code (!-T<x>- or !-U<x>-).

```
!+TABLE<bounds>[;<bounds>...]+
<bounds>: <index>=<left>,<right>[<mod>]
                                     C
                                     L
                                     R
```

<index>...the tab number

<left>...the column in which the text column begins

<right>...the column in which the text column ends

<mod>.....a positional modifier:

C...the text is centered within the column

L...the text is left justified within the column

R...the text is right justified within the column

## NOTES

- This code is similar (but unrelated) to !+SETTAB+, except that rather than specifying a single position, it defines two column positions, the left and right boundaries of the table entry (slot), between which the text following the tab code is placed.
- Each new line code (!-S<n>-) specifies the beginning of a new table row (as well as spacing "n" lines), whereas each tab code (!-T<x>- or !-U<x>-) specifies the beginning of a new table slot (but not a new line).
- Any text following a tab code too large for a column is automatically continued on the next line (at the proper column position).

WARNINGS

- All entries of a table must be preceded by a tab code. If not, the text in the rightmost part of the table will lose one or more characters on that line when FULLPRINTed.
- Table codes should always appear in the file in pairs; a null table code ("!+TABLE+") should follow the table to end Table Mode so that subsequent tabs, tables and justification will work correctly.
- If a table slot is defined that exceeds the current line width (!+WIDTH+), the table will not be formatted properly, though no error message will be printed.

EXAMPLES

```
!-s2-leading text!+table1=11,25;2=31,45+
    !-s1;i8-!+grid(a19,b19,c)9,(d19,e19,f)8,(g19,h19,i)+
!-s1;u1-I wonder what heaven will be like. And will they be
able to change a twenty dollar bill?!-u2-Marshmallows will melt
on a hot day in Providence where the rain always shines.
!-s3;u1-It's difficult to be objective when a tree falls on
your head.!-u2-Melnick wants all government officials to dress
like hens.
!+table+!-s3-trailing text
```

leading text

I wonder what heaven will be like. And will they be able to change a twenty dollar bill?	Marshmallows will melt on a hot day in Providence where the rain always shines.
It's difficult to be objective when a tree falls on your head.	Melnick wants all government officials to dress like hens.

trailing text

!+TITLE+

!+TITLE+ define running title strings

This code specifies a "title string" to be printed on every page starting with the current one.

```
!+TITLE<type>[=<line#>[<page>]]+<string>+
      D                      C
      E                      L
      F                      R
      O
      P
      S
      T
```

PARAMETERS

<type>.....any of the seven alphabetic letters representing title string "D," "E," "F," "O," "P," "S," or "T."

<line#>....the line (counted from the top of the page) on which the title is to be printed (a numeric parameter)

<page>.....a modifier to indicate which pages the title is to appear on:

C...all pages

L...even numbered pages (left)

R...odd numbered pages (right)

<string>...the title text

NOTES

- The seven title types ("D," "E," etc.) stand for "Date," "Evenfoot," "Foot," "Oddfoot," "Pagenumber," "Subtitle" and "Title." They each default in a manner conventional for the keyword (see the table below). However, each can be redefined to appear on any line or page-type, using the parameters above.
- Titles will continue to appear as specified, until canceled with the appropriate null title code (e.g., !+titleD++). This takes effect for the current page, and each subsequent page, unless that particular title code is later redefined.
- The current date can appear in a title by including as part of <string>, either "&1" (which yields the form "mm/dd/yy"), or "&2" (which yields the form "Month dd, 19yy").
- The current page number will replace a colon (":") included in <string>. If a colon is not included in any title string, the page number will overlay the middle segment of !+titleP+, enclosed by two dashes (e.g., "-1-").
- If a line number is explicitly included in a title code (e.g., !+titleO=63+string+, as opposed to !+titleO+string+), the title appears on all pages without regard to the default. This can be overridden by using the <page> modifier (e.g., !+titleO=63L+string+).

Keyword	Line Number	Margin Even page	Alignment Odd page	Page
<u>D</u> ate	7	right	left	C
<u>E</u> venfoot <sup>1</sup>	63	left	-	L
<u>F</u> oot <sup>1</sup>	63	left	right	C
<u>O</u> ddfoot <sup>1</sup>	63	-	right	R
<u>P</u> ageno	63	center	center	C
<u>S</u> ubtitle	7	left	right	C
<u>T</u> itle	4	left	right	C

Figure 4.7 -- Title String Default Settings

---

<sup>1</sup>Defaults to 3 lines from the bottom of the page, which would only be 63 if the current value of !+DEPTH+ is 66 (the default value).



MARGIN ALIGNMENT

- To override the system margin alignment defaults, <string> must be segmented into three parts, delimited by an asterisk ("\*"); e.g., instead of +<string>+ in a title code, +<seg1>\*<seg2>\*<seg3>+ is specified.
  - <seg1> and/or <seg2> and/or <seg3> may be null.
  - <seg1> is left justified with the left margin (overriding any !+OFFSET+ code), <seg2> is centered on the page, and <seg3> is right justified with the right margin (overriding any !+OFFSET+ code).
- When the default margin alignment is overridden, the respective text segment is formatted explicitly, regardless of the default which is designed for two-sided, flip-flop printing.

<-----WIDTH----->		line#	<-----WIDTH----->	
titleT		1		titleT
		2		
		3		
		4		
		5		
		6		
		7		
		8		
		9		
titles	titled	D	<OFF>	titled
		E		
		P		
		T		
		H		
		•		
		•		
		•		
		•		
titleE	titleP	62		titleP
		63		
		64		
		65		
		66		
(Even Page)			(Odd Page)	

Figure 4.8 -- Default Title String Locations

RESTRICTIONS

- No format codes (including underscore or macro codes), or hypertext (decimal references) may appear in <string>.
- It is not possible to have two variations of the same title string in effect simultaneously (e.g., !+titleT=4L+LEFT-SIDE\*\*+ and !+titleT=4R+\*\*RIGHT-SIDE+).

EXAMPLES

!+titleT+\*\*Everypage Title+

This title will appear right-justified, on line 4 of every page.

!+titleS=5+-:-+

Each page will be numbered in a thesis format: five lines from the top, and centered.

!+titleE=63C+FRESS Resource Manual\*Release 9.1+

On line 63 of every page, "FRESS Resource Manual" will be left-aligned and "Release 9.1" will be centered.

!+titleF+\*\*Section 4 -- :+

"Section 4 -- <page#>" will be printed right-aligned three lines from the bottom of the page (which defaults to line 63).

!+TOFC+

!+TOFC+    define Table of Contents

This code determines which headings will appear in the Table of Contents, and how much they will be indented.

```
!+TOFC<entry>[;<entry>...]+  
<entry>: <head>[=<indent>]
```

<head>.....the heading level to appear in the Table of Contents  
<indent>...the number of spaces the entry is to be indented in the  
Table of Contents (this defaults to one less than the  
heading level; it must not exceed 13)

#### NOTES

- All headings will appear in the Table of Contents unless a !+TOFC+ code is present, in which case only those heading levels specified will appear. Thus, to suppress a Table of Contents, specify "!+TOFC<n>," such that <n> is a heading level that does not appear in the file.
- The !+TOFC+ code takes effect only for headings which appear after it in the file.
- The line spacing in effect at the end of the document is the same for the Table of Contents. Thus, for a double-spaced table, include a !+SPACE2+ code at the bottom of the file.
- The Table of Contents is numbered with roman numeral(s), starting with "-i-". It is possible to start numbering with a different roman numeral, by issuing the command "CMS TOFC <n>," just prior to FULLPRINT. <n> is a number between 1 and 9 specifying roman numeral at which numbering is to start (e.g., "CM TOFC 6" would cause the first page to be numbered "-vi-"). Any <n> specified remains in effect for the entire FRESS session (unless the "CMS TOFC <n>" command is re-issued).

WARNINGS

- The right margin for the Table of Contents is one character wider than the right margin in effect at the end of the document. To make it appear to be the same, reset `!+WIDTH+` at the bottom of the file.

EXAMPLES`!+TOFC1;2;3+`

The text following a `!-H1-`, `!-H2-`, or `!-H3-` code is entered into the Table of Contents. The indentation will default to zero, one, and two respectively.

`!+TOFC1=0;2=3;3=5+`

The same headings as above will be entered, but they will be indented zero, three, and five spaces, respectively.

`!+TOFC3;5+`

Only headings of levels three and five will appear in the Table of Contents.

!+WIDOW+

!+WIDOW+ set widow level (stand-alone lines)

This code sets the number of lines allowed to stand alone at the top and bottom of a page (or column, when column justification is in effect -- see Appendix A).

!+WIDOW<lines>+

<lines>...the number of lines that can stand alone

#### NOTES

- The default widow value is twice the current spacing value as specified by the space code (!+SPACE+).
- This code is useful in helping to avoid stray lines at the bottom or top of a page.
- The last <lines> lines of a paragraph will not be allowed to sit alone at the top of a page, nor will a new paragraph be started if it will only have <lines> in it at the bottom of a page, because of this code.

!+WIDTH+

!+WIDTH+    set width of text line

This code sets the maximum number of characters in a line.

!+WIDTH<spaces>+  
    65

#### NOTES

- If the number of blank character positions left on a line is smaller than the size of the next word because !+WIDTH+ is smaller than the word or because of a tab, the width of a line will be extended by one character. The word in question will be split over two lines and a hyphen ("-") will be inserted in the right margin after the split.
- When this code is issued, any default column boundaries are also reset.
- The maximum !+WIDTH+ is 110 spaces (unless the "Large" option of FULLPRINT is specified, which allows 132).

#### EXAMPLES

!+WIDTH63+

The correct width for thesis paper.

!+WIDTH132+

The 132 character width allowed when the "Large" option of FULLPRINT is specified.

!+WIDTH65+

A width of 65 characters (the default).

#### 4.4 EDIT CODES

Edit codes are used for text formatting such as starting new paragraphs, new lines or pages, tabbing, indenting, and centering. They have only a local effect on the document, at their unique place in the text. Edit codes are delimited by a "-".

#### NOTES

- Several edit codes may be concatenated (combined) within one set of delimiters, using ";" to replace the "-!-" string which normally would separate edit codes. For example, "!-s4-!-i10-!-j13-" could be replaced by "!-s4;i10;j13-", to skip four lines, indent 10 spaces and impose a hanging indentation of 13 spaces.
  - When a number of edit codes that perform the same function are concatenated (strung together), the combined effect of the code is determined by the following rules:
    - 1) A new page takes effect only once per edit code, and it clears any current skip lines (!-s4;n;n-reduces to !-n- while !-n;s4-is executed as specified).
    - 2) The code specifying the maximum number of skipped lines is used (!-s4;s2- reduces to !-s4-).
    - 3) The last horizontal displacement code specified is used (!-p;i7- reduces to !-s1;i7-).
    - 4) The order of concatenation is irrelevant for codes which do not perform the same function (e.g., !-i7;j5- is the same as !-j5;i7-).

- Figure 4.9 summarizes several properties and the default values of the codes explained in this section. A "■" marked under "TERM" indicates that the code takes effect online (at least starts a new line), while a "■" marked under "FU" means that code's effect begins on the next line in the formatted (FULLPRINTed) output. A "□" means that the code does not take effect either online, or when the file is FULLPRINTed (depending on the column).

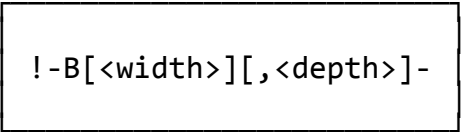
<u>EDIT CODE</u>	<u>PURPOSE</u>	<u>DEFAULT</u>	<u>NEW LINE?</u>	
			<u>FU</u>	<u>TERM</u>
!-B-	draw box	previous code	□	□
!-C-	center line	-	■	□
!-E-	end heading	-	□	□
!-F-	begin/end footnote	flip-flop	□	□
!-H-	heading entry	-	■	■
!-I-	indent	-	■	■
!-J-	hanging indent	-	■	■
!-K-	conditional column	previous code	□	□
!-N-	new page/column	-	■	□
!-P-	paragraph	see !+PARA+	■	■
!-R-	revision bar	flip-flop	□	□
!-S-	skip lines	0	■	■
!-T-	tab	next index	□	■
!-U-	justified tab	next index	□	□
!-X-	expand line	-	□	□

Figure 4.9 -- Edit Code Properties



!-B-!-B-        draw box

This code will draw a box with the upper-left corner positioned at the current column position, on the current line.

!-B[<width>][,<depth>]-

#### PARAMETERS

- Both <width> and <depth> must be greater than one.
- If either parameter is omitted, its value will be the same as that of the previous box. If no box has previously been specified, an error will occur (represented by a long string of dashes).

#### NOTES

- No format code may be concatenated after a box code (using the semi-colon, ";"). However, if a full specification is given (including the exclamation point, "!"), a format code is allowed.
- This code acts like a conditional page code (!-K-) in that a box may not be split over a page.
- A tab code preceeding a box code which forces a new page will be ignored (i.e., the box will be left justified). To allow for this, the user should precede the box and tab codes with a separate conditional page code.
- A box starts drawing at the last tab position (therefore multiple boxes on a line must be drawn with a !+GRID+ code).
- Complicated sketches within boxes and other diagrams can be drawn using the !+GRID+ alter code.

EXAMPLES

!-t\*20;b7,3-  
!-s1-put a box around the fox

put a box around the 

fox
-----

!-b25,5-!-s1;t\*12c-Centered

Centered
----------

!-C-

!-C-            center line

This code centers the string of text between it and the next edit code which begins a new line.

!-C-

### NOTES

- The text is centered around the position `!+WIDTH+/2`.
- If the original string occupies more than one line, each line until the next edit code that starts a new line, is centered.
- `!-C-` will implicitly center the text on a new line, distinguishing it from previous text.
- See `!-T-` about how to center a string at a particular column.

### EXAMPLES

!-c-This line is centered.  
                                This line is centered.

!-c-Name!-c-Rank!-c-Serial Number

Name  
Rank  
Serial Number

This is what will happen if you try to center a line which is  
                                larger than the current `!+WIDTH+`

!-E-

!-E-           end heading

This code ends a heading without beginning a new line.

!-E-

### NOTES

- When this code is issued, the number of lines to be skipped after the heading implied by the heading code is ignored.
- Blanks that appear between a heading and the periods leading to its page number in the Table of Contents can be eliminated by ending the heading with the !-E- code. [The offending blanks can also be identified and then removed using SVIEW and DELETE.]

### RESTRICTIONS

- This code should only be used as intended (to delimit headings).

### EXAMPLES

- Sometimes footnotes are attached to headings, but the user does not want the footnote reference (the superscript) to appear in the Table of Contents. Using the !-E- code to separate the heading and the footnote code (!-F-) will avoid this. In the example below, only the string "Section Title" would appear in the Table of Contents:

!-h1-Section Title!-e;f-footnote!-f-

Section Title<sup>1</sup>

---

<sup>1</sup>footnote

!-F-

!-F-        start/end footnote

Text between two of these codes will become a footnote and will be set at the bottom of the current column (page) when the file is FULLPRINTed.

!-F[<n>]-

<n>...the footnote number. If specified, the automatic footnote counter will be reset to <n>. If not, it is incremented by one (it starts at 1).

#### NOTES

- Space is set aside at the bottom of the current page (column) in which the footnotes appear. A blank line and a horizontal line separate the main text from the footnote(s).
- The value of the footnote counter appears as a superscript at the location of the starting !-F- code and at the left margin of the line in the footnote area in which the footnote appears.
- A !-F- must be used to close the footnote, regardless of whether <n> was specified in the opening code.
- The !-E- edit code is useful for attaching footnotes to headings, while keeping them from appearing in the Table of Contents.

#### RESTRICTIONS

- Footnotes must not be longer than one page.

WARNINGS

- Footnotes sometimes cause FULLPRINT to fail. If FULLPRINT stops in the vicinity of a footnote, try adjusting the file by inserting a few extra lines, rearranging some of the text, or skipping to a new page before the footnote.
- Footnotes should not be used if the text is being formatted into more than two columns (!+column2+).
- All footnotes are delimited at both ends by footnote codes. Failure to close a footnote will probably result in an "oversized footnote", causing FULLPRINT to fail.
- The footnote starts in column one, disregarding any margins set by !+MARGIN+. If in !+COLUMN2+ Mode, the footnote will stay within column boundaries, otherwise, it uses the full line width.

EXAMPLES

I had a great time in Muir Forest.!-f-located in Northern California!-f- It was a wonderful vacation.

I had a great time in Muir Forest.<sup>3</sup> It was a wonderful vacation.

!-h1-Heading!-e;f-footnote!-f-

Heading<sup>4</sup>

---

<sup>3</sup>located in Northern California

<sup>4</sup>footnote

!-H-

!-H-        start heading

This code defines a new heading as the string of text between it and the next edit code.

```
!-H<level>-
  1
  2
  3
  4
  5
  6
```

## NOTES

- Headings are text strings which are specially formatted, according to the heading table currently in effect (see !+HEAD+ - select alter code), their level, and the appropriate heading definition (see !+HEAD+ - define alter code). Headings cannot be longer than 249 characters.
- Heading formats for each level are defined by the !+HEAD+ (define) alter code. This determines the number of lines to skip before and after the heading, as well as capitalizing or underscoring the heading.
- Each heading may also appear in the Table of Contents, as it appears in the text (see the !+TOFC+ alter code).

## EXAMPLES

- The title of this section appears literally in the file as "!-H3-!!-H-~~~~~start !(0heading!)" (the whole thing is underlined because of the heading code).

!-I-

!-I-          indent line

This code begins a new line, indented <n> spaces.

!-I[<n>]-  
           0

### NOTES

- To both skip blank line(s) and indent use the !-p- code.

### EXAMPLES

!-i7-Indent seven spaces from the margin so it is clear how the "indent line" code works.

Indent seven spaces from the margin so it is clear how the "indent line" code works.

!-s0;i5-This simulates the paragraph edit code ("!-P-") when its parameters default

This simulates the paragraph edit code ("!-P-") when its parameters default

!-i3-line1!-i3-line2!-i3-line3

line1  
line2  
line3



!-J-

!-J-            hanging indent

The first line following this code is started at the current left-margin, while the succeeding lines are indented <n> spaces.

!-J[<n>]-  
    0

#### NOTES

- No indentation occurs at the new line started by the !-J<n>- code unless it is concatenated to a !-I<n>- code.
- A hanging indentation is ended by the next format code which begins a new line.
- !-J- will implicitly begin a new line even if not accompanied by another format code which explicitly begins a new line.

#### EXAMPLES

- Each "note" in this manual (such as this), is preceded by the formatting sequence: "!-s1;i3;j6-~!175~." The not signs ("-") are logical blanks (see Appendix A); the "!175" is a "•".

This is a very long string that clearly illustrates a hanging indent of three (!-J3-); it is continued and continued and continued and continued and continued and continued and continued and continued...

!-K-

!-K- conditional page (column)

This code will begin a new page (column) if less than <n> lines are left in the current page (column); otherwise it is ignored.

!-K[<n>]-  
0

### NOTES

- This code should precede tables to be sure they will not be split over a page boundary.
- If the user plans to paste in pictures or diagrams that require a set amount of contiguous space, this code should be specified separately and previous to the skip codes (!-S-) that reserve the space to keep it all on the same page.

### EXAMPLES

!-K6-!-S6-

If there are not six lines left in the page, a new page will be started and six lines will be left at the top.

!-K6;S6-

If there are not six lines left in the page, a new page will be started with text at the top of the page (six lines will not be skipped).

!-N-

!-N-      new page (column)

This code forces the following text to start a new page, or possibly a new column if multi-column format has previously been specified.

!-N[<page>][<mode>]-  
      0          C  
      n          R

<page>...the number of the new page, if not specified, is one greater than that of the last page (page numbering is suppressed if "0" is specified)

<mode>...this defaults to the current numeral type (initially arabic, roman numerals if "R" is specified). If in Multi-Column Mode and "C" is specified, a new column is started (otherwise the "C" is ignored).

#### NOTES

- If "!-N0-" is not specified, page numbering begins with page "1".
- To return to arabic numerals when the current numeral mode is roman numerals, <n> must be explicitly specified.
- To skip multiple pages, a special blank ("-") must be specified between separate format codes because of format code combination rules. A page without text will not be skipped; however, "-" is considered text.

EXAMPLES!-n-!-n-!-n-

This sequence of codes will leave two blank pages in the document (note the use of the special blank "-").

!-n5-

This code will cause a new page to be started; it will be numbered page "5".

!-nr2-

A new page will be started and numbered "ii"; subsequent pages are numbered with roman numerals.

!-n0-

A new page will be started and page numbering will be suppressed (starting with the new page).

!-nc-

This code will start a new column (if in multi-column format -- see the !+COLUMN+ alter code).

!-P-

!-P-        start paragraph

This code starts a new paragraph.

!-P-

### NOTES

- The number of lines skipped and characters indented may be reset with the `!+PARA+` alter code (the default is to skip one line and indent five spaces).

### EXAMPLES

(unformatted text)

Assume this is the last line of a paragraph.

!-p-The effect of the "!"-P-" code before "The effect" is now evident: it skipped one line, and then indented five spaces for the new paragraph.

(formatted text)

Assume this is the last line of a paragraph.

The effect of the "!"-P-" code before "The effect" is now evident: it skipped one line, and then indented five spaces for the new paragraph.

!-R-

!-R-        start/end revision bars

This code will cause a revision bar ("|") to be inserted immediately to the left of the left margin (on every line), until the next !-R- code is encountered.

!-R-

### NOTES

- To remove the revision bars, all !-R- codes must be deleted.
- Revision bars can be entered as format macros (see Section 4.5), which can then be removed merely by redefining the macro as null.

### EXAMPLES

|     !-r-revised text!-r-  
     revised text

|        This paragraph is surrounded by revision bars. A common application for this code is to denote those sections of a document that have changed between drafts.

### In format macros:

!.r1=!-r-.

     This macro might be used to define revisions from the first draft.

!.r1=.

     This macro would nullify the revisions indicated by !.r1. in the above example.

!-S-

!-S- skip line(s)

This code starts a "new line" and then skips <n> blank lines.

!-S[<n>]-  
0

#### NOTES

- The new line itself may cause some skipped lines if the file is double or triple spaced (e.g., !-s2- will cause 3 lines to be skipped if the file is double spaced)

#### EXAMPLES

!-s2-  
this code will leave two blank lines in the text

!-s-  
this code is the same as !-s0-

!-T-

!-T-        tab

The text that follows this code and is delimited by the next edit code, is positioned at the column position specified.

```

!-T[<col>][<mod>][<fill>]-
      *n      C      ,<char>
      x      L      ,<dec>
              R
    
```

<col>...this parameter controls the column (modified by <mod>) which the printer tabs to before printing the text. It defaults to the next index (x).

\*n.....move to absolute column n+1 (i.e., !-t\*0- moves to column 1)

x.....the value of the xth entry in the table defined by the !+SETTAB+ alter code (1-10). These values default to every ten spaces (1=10, 2=20,...10=100). See !+SETTAB+ alter code.

<mod>...this parameter indicates whether the left-most (L), right-most (R), or center (C) position of the text string will be aligned with <pos>. Explicitly specifying <mod> will override any modifier specified with the !+SETTAB+ alter code.

<fill>...this parameter will place the "fill character" into any spaces skipped over as a result of the tab.

<char>...a literal, non-control character (see Section 3 -- CONTROL CHARACTERS).

<dec>...the decimal representation of the character (see Section 4.6)



NOTES

- All the parameters are optional; the simplest form of the tab code is `!-T-` which positions the text at the next table position (see `!+SETTAB+`).
- Some text (at least a special blank) must follow the tab, for the `<fill>` character to work properly.
- All online tabs are currently treated as left tabs. The default tabular index is reset to one at every new line code.

WARNINGS

- The tab code will not work properly if `<col>` plus the left-margin is not less than `!+WIDTH+` minus one. If it is equal to one less, FULLPRINT will insert a dash ("-") and put the following text on the next line. If it is equal to `!+WIDTH+`, the following text will be put on the next line. If it is greater than `!+WIDTH+`, the tab will be ignored, but a problem may develop later in the FULLPRINT.
- None of the tab codes start new lines. Users should be careful or else the text will overlay.
- If a literal character is specified as a "fill character", it will automatically be capitalized. To "fill" with a lower case character, the decimal representation should be specified.
- If an `!-E-` edit code is imbedded within a string of text following a tab code, it is treated as a backspace.

EXAMPLES!-t1,96-

indicates that "-" (96 is the decimal representation of "-")  
is to be used as the fill character

!-t\*24-

positions text to column 25

Pencils!-T\*65r,.-\$1.20

Pencils.....\$1.20

!-T\*66r,.-~

This is a right-justified tab with "." as the fill  
character. It generates the following line [note, the use  
of the special blank "~" causes the line to "fill" up to  
absolute column 66 and then print a blank on the same line  
(in column 67)]:

.....

!-b25,5-!-s1;t\*12c-Centered

Centered
----------

!-U-

!-U- justified tab

The text between this code and the next edit code that starts a new line, is right and left justified between the tab position specified by <col> and the right-margin.

!-U<col>-  
\*n  
x

<col>....this parameter controls the column which the printer tabs to before printing the text. It defaults to the next index (x).

\*n.....move to absolute column n+1 (i.e., !-t\*0- moves to column 1)

x.....the value of the xth entry in the table defined by the !+SETTAB+ alter code (1-10). These values default to every ten spaces (1=10, 2=20,...10=100).

## NOTES

- This code establishes a temporary left-margin at the tab position, or if Table Mode is in effect, a temporary left-margin at the tab position and a temporary right-margin at the right table column boundary.
- This tab is typically used in tables where text must be left and right justified (see the !+TABLE+ alter code and the !-T- edit code for more information).

## RESTRICTIONS

- This code should not be used in Multi-Column Mode.

!-X-

!-X-            expand line

This code forces the line to be expanded (right justified) when FULLPRINTed.

!-X-

### NOTES

- This code would generally be used to force normal justification on the last line before a !-N- or !-S- code (rather than leaving a sentence fragment that is justified only on the left).

### RESTRICTIONS

- The effect of any !-X- code does not exceed the physical line of the printout in which it appears.

### EXAMPLES

This is a normal sentence.  
This is a normal sentence.

!-X-This is an example of an !(0expanded!) sentence.  
This is an example of an expanded sentence.

#### 4.5 FORMAT MACRO CODES

Format Macros are user defined codes which are abbreviations for arbitrary combinations of text and/or format codes. A macro definition must be of the form:

```
!.<macroname>=<anystring>.
```

<macroname>...the name of the format macro. This will be replaced during FULLPRINT with <anystring>.

<anystring>...the string (often containing format codes) for which <macroname> is an abbreviation. It replaces <macroname> during FULLPRINT.

#### NOTES

- Unless the macro definition is SCROLLED over during the FRESS session, any occurrence of the macro code will not be defined. Online this will cause the "UNDEFINED MACRO" message to be typed. If FULLPRINT does not process the appropriate macro definition, any undefined macro will be translated to the null string and will be flagged on the far right with "\*\*\*\*."
- Macros may be redefined at any time. Each definition is in effect until another definition is encountered. The effect of a macro can be eliminated by defining it to be the null string as follows: "!.<macroname>=."
- Macros may contain other macros. Simply imbed a macro code within the macro definition, e.g., "!.mac2=<text>!.mac1.". If a macro definition contains an undefined macro code, any occurrence of the higher level macro will be flagged as undefined.
- Macros have the same effect online as any format codes imbedded in them. For example, the macro "!.in3=!-i3-!", causes an indentation of three spaces wherever the code "!.in3." appears in the text.
- Unlike format codes, character case is important in identifying macro names, "!.in3." is not equivalent to "!.IN3.". To avoid confusion and to make text input easier, it is wise to use only lower case characters when defining macro code names.

SPECIAL CONTROL MACROS

- When the "Macros" option of FULLPRINT is specified, the heading appearance and margins for each Decimal Block encountered are controlled by a series of macro definition pairs, which must be inserted into the file by the user (see FULLPRINT).
  - The headings of decimal level <n> are determined by the macros: "!h<n>.". If the heading definition for level one is "!h1=-h1-.", its appearance will be the same as if the Macros option was not used.
  - The margins for blocks of decimal level <n> are defined by the macros: ".dm<m>.", where <m>=<n>-1. To reproduce the margin definition for level one under a non-Macros FULLPRINT, "!dm0=!!+margin0,0+" should be included in the file.
  - Heading appearances (whether a particular heading level is capitalized, etc.) can be redefined using the !+HEAD+ -- define Heading alter code.

RESTRICTIONS

- The macro definition must be less than 100 characters long. The length of a macro definition is determined by expanding imbedded macros, as well as considering the delimiters ("!" and ".") and the macroname.
- The special character code for the period (!75) must be used if a literal period (".") is to be included in a macro definition.
- Any format codes within a macro must be specified in full (i.e., it is not possible to include part of the format code inside the macro, and then concatenate the rest as normal text. Macros may not be imbedded within edit or alter codes. This is illustrated in the following examples of illegal codes:

```
!.s=!-s1;.
!.s.i5-This text would be indented five.
!.s.i10-and this indented ten if these codes worked.

!.t=!+titleT=5c+.
!.t.would be the title's text if this code worked+
```

These macros are invalid because format codes are indivisible.

## SUGGESTIONS

- Keep all macro code definitions at the top of the file in alphabetical order for easy reference. Define a null macro, "!.end of macros=." at the end of all the macro definitions, and SCROLL to that point at the beginning of every editing session.
- Alternatively, create a file of commonly used macros, and IMBED it at the top of every file in which the macros are to be accessed. To suppress the "UNDEFINED MACRO" message, issue "FMSG OFF."

## EXAMPLES

- Macros can be used not only to save keystrokes, but also to make global formatting changes. For example, if a document is to contain several hundred formulae, it is recommended that each be formatted by a macro code, rather than an explicit format code. In this way, just the macro definition would have to be changed to reformat each formula.
- An even simpler example is using a macro to abbreviate a long text string. For example, if the macro definition "!.usa=United States of America." was imbedded in a file, the user would only have to insert "!.usa." to have "United States of America" appear in the FULLPRINT.
- A macro, "!.skip." could be defined to replace the following string:

```
!-s4;i10;j13-  
!.skip=!-s4;i10;j13-.
```

This could further reduce the number of keystrokes required to issue this code and would make redefinition of the code everywhere it occurs easy with one change to the macro.

## RELATED COMMANDS

FMSG, IMBED

#### 4.6 SPECIAL CHARACTER CODES

Special Character codes allow the user to enter into the file system delimiters and characters not available on the keyboard. These codes will be translated to the desired character when the file is FULLPRINTed, and are of the form:

`!<nnn>[*]`

<nnn>...A two to three digit number (41-255) representing a character from one of the tables below [the decimal value of the machine code for the character].

\*.....This optional character makes possible unambiguous strings of special characters and numbers (it is ignored by the FULLPRINT program -- see notes below).

#### SPECIAL CASES

Three commonly used characters can either be represented in the file using the format above, or through an alternate representation as follows:

```
!! literal exclamation point ("!")
!~ literal "not" character ("~")
!/ literal percent ("%")
```

#### NOTES

- The user may create overstruck special characters by making use of the backspace key on the terminal, i.e., the character "¸" was "created" by typing "b"-"backspace key"-"/" (the special character code for a backspace is !22).
- To overprint two characters one of which is a special character code, the user must specify the special character code first, followed by the literal backspace, followed by the ordinary character.
- To avoid the ambiguity of having a literal number in the text following the special character format code, an asterisk ("\*") may be used as a delimiter for any special character, e.g., "!175\*5", rather than "!1755". The asterisk will not appear in the text when FULLPRINTed (i.e., "!175\*5" prints as "•5").



RESTRICTIONS

- Overprinting is restricted to a single special character.
- No character may overprint itself (thus no boldface).
- A character represented by a format code cannot be overprinted with another character represented by a format code. Thus, the special character representation of a backspace (!22) cannot be used in conjunction with another special character code (i.e., a literal backspace must be used).

EXAMPLES

- In the examples below, "\$" represents a literal backspace.

input: "b!22/"  
yields: "b\$"

input: !193 - !194  
yields: A - B

input: !173!139(a+b) \* c!155 + d!189 = f  
yields: [{(a+b) \* c} + d] = f


input: !173\*24!189  
yields: [24]

#### 4.6.1 ALX Print Train

The special character codes below represent the ALX print train (class E).

4.6.2 TN Print Chain

The special character codes below represent the TN print chain (class F).

!41.....	!84.....`	!127....."	!170.....i	!213.....N
!42.....	!85.....i	!128.....Ø	!171.....L	!214.....O
!43.....	!86.....î	!129.....a	!172.....r	!215.....P
!44.....	!87.....ï	!130.....b	!173.....[	!216.....Q
!45.....	!88.....ì	!131.....c	!174.....≥	!217.....R
!46.....	!89.....ß	!132.....d	!175.....•	!218..... <sup>1</sup>
!47.....	!90.....!	!133.....e	!176..... <sup>0</sup>	!219.....£
!48.....	!91.....\$	!134.....f	!177..... <sup>1</sup>	!220.....\$
!49.....1	!92.....*	!135.....g	!178..... <sup>1</sup>	!221.....g
!50.....2	!93.....)	!136.....h	!179..... <sup>3</sup>	!222.....ú
!51.....	!94.....;	!137.....i	!180..... <sup>4</sup>	!223.....ÿ
!52.....	!95.....	!138.....↑	!181..... <sup>5</sup>	!224.....\
!53.....	!96.....-	!139.....{	!182..... <sup>6</sup>	!225.....÷
!54.....	!97...../	!140.....≤	!183..... <sup>7</sup>	!226.....S
!55.....	!98.....Å	!141.....ç	!184..... <sup>8</sup>	!227.....T
!56.....	!99.....Ä	!142.....+	!185..... <sup>9</sup>	!228.....U
!57.....	!100.....À	!143.....†	!186.....[	!229.....V
!58.....	!101.....Á	!144.....°	!187.....]	!230.....W
!59.....1	!102.....Ä	!145.....j	!188.....]	!231.....X
!60.....2	!103.....Å	!146.....k	!189.....]	!232.....Y
!61.....!	!104.....Ç	!147.....l	!190.....≠	!233.....Z
!62.....¬	!105.....Ñ	!148.....m	!191.....-	!234..... <sup>2</sup>
!63.....%	!106.....!	!149.....n	!192.....{	!235.....Ô
!64.....	!107.....,	!150.....o	!193.....A	!236.....Ö
!65.....A	!108.....%	!151.....p	!194.....B	!237.....Ø
!66.....â	!109....._	!152.....q	!195.....C	!238.....'
!67.....ä	!110.....>	!153.....r	!196.....D	!239.....Õ
!68.....à	!111.....?	!154.....t	!197.....E	!240.....0
!69.....á	!112.....°	!155.....}	!198.....F	!241.....1
!70.....ã	!113.....^	!156..... 	!199.....G	!242.....2
!71.....å	!114....."	!157.....>	!200.....H	!243.....3
!72.....ç	!115...../	!158.....±	!201.....I	!244.....4
!73.....ñ	!116.....´	!159.....■	!202.....	!245.....5
!74.....¢	!117.....·	!160.....-	!203.....ô	!246.....6
!75.....	!118.....¸	!161.....°	!204.....ö	!247.....7
!76.....<	!119.....¸	!162.....s	!205.....‘	!248.....8
!77.....(	!120.....¸	!163.....t	!206.....ó	!249.....9
!78.....+	!121.....`	!164.....u	!207.....õ	!250..... <sup>3</sup>
!79.....	!122.....:	!165.....v	!208.....}	!251.....Û
!80.....&	!123.....#	!166.....w	!209.....J	!252.....Ü
!81.....é	!124.....@	!167.....x	!210.....K	!253.....Ù
!82.....ê	!125.....‘	!168.....y	!211.....L	!254.....Ú
!83.....ë	!126.....=	!169.....z	!212.....M	!255.....‘

#### 4.7 UNDERSCORE CODES

Text is underscored (when FULLPRINTed) when it is surrounded with a special two-part code: "!(0" must precede the text, and "!) " must follow the text. The general form is:

`!(0<text>!)`

<text>...the text to be underscored

#### NOTES

- The underscoring of blanks appearing within <text> is controlled by the !+B+ alter code.
- Text can be inserted already surrounded by underscore codes, or these can be added afterward; either by using the UNDERSCORE comand or "by hand."
- Text can also be underscored during input with the use of backspaces and literal underscore characters ("\_"). FRESS will convert these to the proper format codes.
  - Characters may be underscored one at a time, or all together.

#### WARNINGS

- If the right delimiter of an underscore code ("!)") is accidentally ommitted, all text up to the next right delimiter code will be underlined except the first line of text after any new line format code (e.g., !-S-, !-I-).
- If the closing delimiter ("!)") is unmatched, the entire file will be underlined after the next opening delimiter ("!(0").

## RESTRICTIONS

- The total number of characters entered on a line when in input may not exceed 130 characters because of hardware limitations. All literal underscore and backspace characters are included in this count.
- When literal underscoring is used, no format codes (including special character codes) may be included in the underscored text.

## EXAMPLES

!(0text!)  
text

## RELATED COMMANDS

UNDERSCORE

5

6

7



8

## 9 FRESS COMMANDS

### 9.1 DESCRIPTION

All commands and parameters may be typed in upper or lower case. Abbreviations for commands are allowed, provided they begin with the minimum abbreviation contained in the box at the top of the page and highlighted by capital letters in the command specification model.

#### NOTES

- A cross-reference of similar or contrasting commands is provided under the heading "RELATED COMMANDS."
- The command-type (e.g., Editing, Display) is specified at the top of each description page. A complete list of commands, listed by type, is given in Appendix D.1.
- A list of possible system responses is given under the heading "MESSAGES," and descriptions of those messages are listed in Appendix E.
- Some commands are tagged with a special symbol, which denotes the following:
  - <sup>1</sup> A command valid only with the single window version.
  - † A picture manipulation command used only with the multiple window version.

9.2 LINE AND CHARACTER DISPLACEMENT

To facilitate the proper identification of an <lp> or a <scope>, line and/or character displacement numbers may be specified in any command which requires either of these parameters [see Section 5.7 for an explanation of <lp>, <scope>, and other parameters]. The format of the command specification when using these displacement values is:

```
<command><del><line-disp>[<char-disp>]|<scope><parms>
                        0      <lp>
```

<command>.....The editing command to be executed.

<del>.....The key delimiter.

<line-disp>...An optionally signed ("+" or "-") number which indicates the number of lines to be displaced from the top of the Editing Buffer. This is effectively a SCROLL of <line-disp> lines before context-scanning begins.

<char-disp>...A signed ("+" or "-") number (default zero) indicating the number of characters to be displaced into or from the line before context-scanning begins, after a SCROLL of <line-disp>.

<parms>.....Any other parameters and delimiters the command requires.

## NOTES

- `<line-disp>` and `<char-disp>` must be contiguous (no intervening blanks), and must be followed by the LP-Separator ("|"), whether or not a context pattern follows.
- The first displayed line of the Editing Buffer is numbered zero.
- The first character of each line is numbered zero.
- The LP Separator is used to indicate to FRESS that line and character displacement numbers are being used.
- This facility is useful for modifying the file above the top of the current Editing Buffer, as well as modifying a text pattern which occurs at least once before in the Editing Buffer.

## WARNINGS

- This facility should not be specified for any command which does not have an `<lp>` or `<scope>` parameter.

## EXAMPLES

D/2|amount

"amount," which occurs after the second line (i.e., starting on the third line) of the Editing Buffer, is DELETED. (Perhaps "amount" also occurs in the first or second lines, but this occurrence is not the one the user wishes to delete.)

S/-1+10|amount/text

"text" is SUBSTITUTED for "amount" which begins somewhere after the tenth character in the line above the top of the display.

D/2+2|

The user wishes to DELETE the third character in the third line of the buffer. Omitting the context pattern causes the first character after the indicated SCROLL and move to be considered the `<lp>` character. In this case, the user specified a SCROLL of 2 lines and a move of 2 characters into the line. A more useful application might be to delete the word which contains the third character in the third line of the buffer. In this case the user could type "D-w/2+2|" (see Section 5.5).

### 9.3 THE SPECIAL "&" LP CHARACTER

The last character of a context pattern normally determines the `<lp>` or `<scope>`. This rule changes when one or two ampersands ("&") are included in the specification, a useful facility for the user who wishes to use the full Editing Buffer without SCROLLing before each command.

#### <lp>

- When an ampersand is included as part of an `<lp>`, the character immediately preceding the ampersand is taken as the `<lp>` character, no matter how long the context string is.
- A useful application of this facility is changing the second occurrence of a character in a line, when both occurrences are preceded by identical contexts, but each is followed by a different context. The desired character can be LP'ed by specifying it followed by an ampersand and enough trailing context to uniquely identify it.
- Example:

i/Note: &here is another/ButØ

[before]=> Note: here is one Note: here is another  
[after]=> Note: here is one Note: But here is another

- Example: "f&ather" LP's the "f," rather than the "r."

#### <scope>

- When two ampersands are specified within a `<scope>`, the `<scope>` is defined to be the string beginning with the letter before the first ampersand and ending with the letter before the second ampersand.

c/Note: h&ere is& another/but I like

[before]=> Note: here is one Note; here is another  
[after]=> Note: here is one Note; but I like another

- Example: "m&ast&her" produces "mast" as the `<scope>`.
- Example: "ma&st&" produces "ast" as the `<scope>`.
- Example: "ma&&st" produces "a" as the `<scope>`.

## 9.4 SPECIFYING MULTIPLE COMMANDS ON ONE LINE

More than one command can be put on a command line by separating each with either:

- 1) The FRESS Command Separator (">"), or
- 2) The CMS Linend character (defaults to the pound sign("#") but can be changed using the "CMS TERMINAL LINEND <char>" command.

If a command is preceded by "<" or "(" certain Display Mode characteristics will be overridden (see Section 2.3.1).

### 9.4.1 The FRESS Command Separator

When the FRESS Command Separator is used, the entire command line uses the current Editing Buffer for each edit, regardless of the Editing Mode in effect (see SMODE). Thus the order of the commands on the command line does not matter.

#### NOTES

- Each command separated by the Command Separator will act only on <scope>s or <lp>s already in the Editing Buffer.
- The ">" can be thought of as a carriage return character at the end of a line which contains only a single command.
- After a command line is processed, the top of the new Editing Buffer is located at the point of the last edit, unless a travel command follows the last editing command. In that case, the top of the new Editing Buffer is at the point specified by the last traveling command.
- Some commands such as FULLPRINT, REVERT, and all the House Functions (see Section 6) cannot be followed on a command line by any stacked commands.
- A pair of ">" characters is required to represent a single literal occurrence of ">" anywhere in Command Mode.

- Using the FRESS Command Separator is less expensive than using the CMS Linend character. In addition, the Display Mode can be temporarily overridden (see Section 2.3.1) by appending two blanks and one of the two Mode indicators below to the command line:
  - "." all printing will be suppressed
  - ".\*" all printing is suppressed except for the last command.

#### WARNINGS

- The entire command line will not be executed if any command causes one of the following "Question Mark" errors (a question mark is typed -- to see the message, type back a "?"):

CONTEXT PATTERN NOT FOUND  
ERROR IN COMPOUND SCOPE  
FUNCTION TAKES NO PARAMETER  
INCORRECT VALUE IN PARM POSITION  
INVALID FUNCTION SEPARATOR  
PROCESSING NEEDED TOO MUCH SPACE  
SEMANTICS CHECK  
TOO FEW VALUES SPECIFIED

#### EXAMPLES

S/amount1/text>D,amount2>-5

This input line will SUBSTITUTE "text" for "amount1," DELETE "amount2," and then SCROLL back 5 lines.

S /old pattern/new pattern>S amount text

This input line will look for "amount" in the current Editing Buffer (not in the Editing Buffer starting at "new pattern").

I/place/IF A>>B THEN

This example will INSERT the text "IF A>B THEN" at the location specified by "place."

#### 9.4.2 The CMS Linend Character

When the CMS Linend character is used, the command line is executed, logically treating each instance of the Linend character as a carriage return. This facility is useful when a static Editing Buffer is not desired (e.g., when traveling between commands is necessary).

#### NOTES

- The Linend character is also the means by which traveling commands can be interspersed between editing commands. For example, if "#" is the CMS Linend character:

"gl/lab1#c/first/second#ds#i/A\*/top line"

- Example: Three edits more than 2100 characters apart could be specified on the same line if separated by the Linend character. The search for the context pattern of the second edit would be in an Editing Buffer 2100 characters from the first edit, and the third edit could be within 2100 characters of the second edit (see Section 1.3, Editing Buffer/Display Window, and Section 2 for default Buffer sizes).



## 9.5 COMMAND QUALIFIERS

A command "qualifier" enables the user to specify a command <scope> with minimal effort. Commands are specified, using qualifiers, as follows:

```
<command>-<qualifier> <parameters>
```

```
B  
C  
L  
O  
W
```

B...Block  
C...character  
L...line  
O...order  
W...word

### NOTES

- A hyphen ("-") must be typed immediately after the command name and before the qualifier, with no intervening blanks or delimiters.
- A qualifier may be specified with any command which has a <scope> parameter.

### RESTRICTIONS

- Qualifiers cannot be used with the LOCATE command.

TYPESBLOCK (B) .....

The Block qualifier expands <scope> to the first Block containing <scope> (everything between the Block delimiters "%<" and "%>").

Example: To DELETE the Decimal Block 2.4, position the display before the '2.4' label and specify: "d-b/2.4". The entire Block numbered 2.4 is deleted.

The Block qualifier can also be used to reorder Decimal Blocks, but the ".mob" system Command Macro (see Section 7.3) is easier to use.

CHARACTER (C) .....

The character qualifier causes the command to act only upon the last character of <scope>. For example:

d-c/anks

[before]=> special blanks  
[after]=> special blank

If the "C" qualifier hadn't been used, the "anks," rather than just the "s," would have been DELETED.

LINE (L) .....

The line qualifier replaces the <scope> parameter with the first line of the Display Window (see SDISPLAY). This qualifier is especially useful when text lines are cluttered with formatting codes and/or Hypertext.

ORDER (O) .....

The order qualifier restricts <scope> to Hypertext. The entire piece of structure containing <scope> is affected by the edit.

If a command is immediately preceded by a percent sign ("%"), it is executed as though the order qualifier was specified.

Example: To DELETE the label "%L(section)", the user need only specify: "d-o/%"

WORD (W) .....

The word qualifier expands <scope> to the whole word in which <scope> is found (defined as a blank followed by a string of characters until another blank or punctuation).

The blank before the word is considered part of the word when using the word qualifier, except with the CAPITALIZE, CHANGE, SUBSTITUTE, and UNCAPITALIZE commands.

Example: "d-w/ual" will delete the first word containing the string "ual" (including the preceeding blank).

Example: To change the word "sufficient" to "adequate," a user need only specify: "c-w/su/adequate."

If this qualifier is used on a format code, or a string of characters immediately preceded by a format code, it behaves differently. In the examples given below, the underlined portion indicates the deleted portion of the code (the "!" and anything else that is not underlined is not deleted):

!+margin0,0,9,6+

!-p-

!.fill.

!175

!(0text!)

## 9.6 PARAMETER SPECIFICATION

Most commands take one or more parameters which are enclosed by syntactic brackets (" $<$ " and " $>$ ") in the command description.

### NOTES

- Parameters must be specified in the order listed, separated by a Key delimiter, that is, the first character after the command name (blank, "/", etc.).
- Parameters are assigned left to right, unless explicitly left null (two consecutive delimiters). All required parameters are assigned values before any optional parameters.
- If too few input values are specified, an error message is typed.
- It is impossible to specify too many input values. After all required and optional parameters have been assigned, the remaining text on the command line is taken as part of the last parameter.
- Optional parameters are those followed in the command specification by the symbol " $\circ$ ". A default, if specified (by underscores), will be used if an optional parameter is not specified.

## 9.7 PARAMETER LISTING

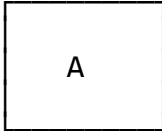
The following is a list of standard parameters listed in many command descriptions. Parameters that do not appear here are described in full where specified.

- `<bool>`.....A boolean request string composed of keywords, Attribute-Value pairs, and weights combined with the logical operators "`~`" (not), "`&`" (and), and "`|`" (or) [the descending order of precedence]. The precedence may be changed by the use of parentheses. Two examples of valid boolean request strings are: "`key1&key2|key3`" and "`~(key1&key2)&key3`" (see Appendix C, "Keyword Strings").
- `<dkeys>`.....A display Keyword string. Each Keyword cannot be more than 16 characters; the entire string cannot be more than 255 characters.
- `<file>`.....A filename, which is a literal character string of no more than 8 characters. If there is no `<file>` parameter in a command, or if the `<file>` parameter is optional and not specified, the command is executed on the current file. If "`<current>`" is specified as the default, it refers to the current file.
- `<keys>`.....A Keyword string made up of Keywords, Attribute-Value pairs, and Weights. Each Keyword must be no more than 16 characters long; the entire Keyword string must be no more than 255 characters long.
- `<iip>`.....A `<lp>` that defaults to the Implied Insert Point: the last position where text was added (deleted) or was supposed to have been added (deleted). See Appendix D for all commands which use the `<iip>`. See also `&POP` for an example of how to utilize `<iip>` when inserting Blocks ("`%<...>%`").
- `<label>`.....A literal character string of no more than 16 characters which acts as a Label in a file.
- `<lit>`.....A literal character string whose use and format is determined by the particular command.

- <lp>.....A Location Pointer which designates a position in the file, often to indicate where some text or Structure should be placed. It is specified as a context string of one or more characters which may contain embedded ellipses ("..."); the last character matched by the string is considered the <lp> except in the case of IBEFORE, when the special LP character("&") is used (see Section 5.3), or when the Character qualifier("C") is used (see Section 5.5). A default specified as "<beginning>" refers to the beginning of the file. <lp> may also be indicated with the light pen on the IMLAC.
- <n>.....An unsigned number, considered to be positive unless otherwise indicated (in the command description). Specific uses vary by command.
- <options>...A character string which may contain certain characters indicating certain options available. The <options> list is unique to and defined in any command which uses it.
- <pass>.....A literal character string of no more than 7 characters which acts as a password to a file. The password in effect is the one specified in GFILE when the file was opened, or in the most recent CPASSWORD command.
- < pict>.....A picture name. It must be no more than 8 characters in length.
- <scope>.....An amount of text to be affected by the command. It is specified either as a context string or as a pair of <lp>s identifying the beginning and ending points of the amount, one or both of which may be deferred.
- <space>.....A system-defined Space. The minimum abbreviation of each is one letter: Annotation (A), Keyword (K), Label (L), Picture (P), Structure (S), Text (T), Work (W). See Section 1.3, File Structure, for an overview of these regions.
- <text>.....A literal character string meant to be inserted in the file as a piece of text, as specified by the particular command. If in any command which uses this parameter, the user specifies a a delimiter instead of <text> (e.g., "i/old text/"), regular Input Mode will be entered (see INSERT for a more complete example).
- <vs>.....A viewing specification string. Viewspecs are used to specify how the file is to be displayed and

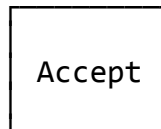
formatted online. A complete description of Viewspec strings is given with the SVIEW command.

<wind>.....A display window (multi-window version only). The choices are "1," "2," "3," or "4."

9.8 COMMAND DESCRIPTIONSA

- A      Accept

ACCEPT makes permanent the result of the previous editing operation.

NOTES

- ACCEPT neutralizes the ability to REVERT the last editing command.
- ACCEPT might be used to save the last edit, if the user must leave the terminal and is afraid of computer failure that would lose it.

MESSAGES

PROCEED

RELATED COMMANDS

REVERT



AP

- AP    Add Password

APASSWORD adds a new password to the file. Only a restricted set of commands is permitted to be executed when the file is subsequently accessed with the password.

```
APassword <pass> <systring> <list>
                        ALL
                        DISPLAY
                        NONE
                        *
```

<pass>.....The password to be added.

<systring>...One of the four system defined strings listed below, which indicate the commands allowed under <pass> (as modified by <list>):

ALL	all functions allowed
NONE	no functions allowed
DISPLAY	only display functions
*	same functions as present password

<list>.....The list of commands allowed under <pass> (modifying those defined by <systring>). <list> must be specified using the following form:

<sign> <command>[Ø<command>...] [<list>]

- Blanks must separate <commands>s not separated by <sign>s. Each <command> is indicated by any substring which would be considered legal when actually specifying the command.
- <sign> specifies whether the command is to be permitted ("+") or prohibited ("-"). A given sign has scope over all command mnemonics encountered until the next sign.

NOTES

- When a file is created (see MFILE), a password is assigned to it ("DEFAULT" if none is specified), as well as a <list> comprised of all FRESS commands.
- The password currently in effect is either the password specified when the file was last accessed with GFILE, or the password last specified with the CPASSWORD command, whichever came last.
- Any number of passwords may be added to a file to narrow the range of commands available to restricted users.

RESTRICTIONS

- New passwords must be no more than seven alphanumeric characters.

EXAMPLES

ap/default/<allowable commands>

The file may now be accessed in the future with the system password "DEFAULT," but only the execution of those commands specified in <allowable commands> will be allowed.

ap/pass/\*-i d

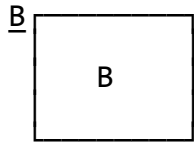
A GFILE issued with the password "pass," would not allow insertions (i) or deletions (d).

MESSAGES

INVALID FUNCTION  
PASSWORD ALREADY EXISTS

RELATED COMMANDS

CPASSWORD, DPASSWORD, GFILE, MFILE



- B Bottom of Space

BOTTOM moves the display pointer to the display line above the last Area line of the Space indicated, and displays it.

Bottom <space>° <wind>°

#### NOTES

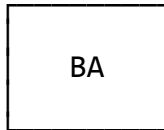
- <space> defaults to the Space currently being viewed.

#### MULTIWINDOW

- The bottom of the indicated space, of the file displayed in the window number specified (defaults to the current window), will be displayed.
- The display will appear in the current window if <window> is not specified.

#### RELATED COMMANDS

BAREA, DSPACE



- BA Bottom Area

BAREA moves the display pointer to the display line above the Area line of the current Area and displays it.



#### RESTRICTIONS

- BAREA works only in the Text and Work Spaces.

#### MESSAGES

FUNCTION NOT ALLOWED IN SPECIFIED SPACE

#### RELATED COMMANDS

BOTTOM, DSPACE

BARS

- BARS Bars

BARS surrounds <scope> with revision format codes (!-r-).

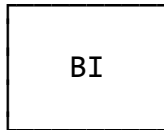
BARS <scope>

#### NOTES

- Revision bars ("|") will appear to the left of the left margin on the lines containing the text indicated in <scope>, when the file is FULLPRINTed.
- This facility offers a good way to show selective changes in large documents.
- See the !-R- edit code for additional comments on revision bars.

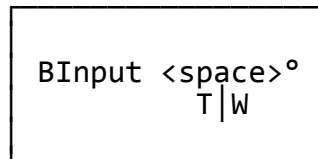
#### RELATED COMMANDS

SURROUND



- BI Bottom Input

BINPUT causes regular Input Mode to be entered at the bottom of the last Area of either the Text or Work Spaces.



T...Text Space  
W...Work Space

#### NOTES

- The current Space is assumed if <space> is not specified, and the user is put into Input Mode.
- SIBOTTOM is a faster and cheaper way of accomplishing the same task.

#### RESTRICTIONS

- The total number of characters entered on a line when in input may not exceed 130 characters because of hardware limitations. All literal underscore and backspace characters are included in this count.

#### MESSAGES

\*END OF TEXT AREA\*  
INPUT  
UNDEFINED SPACE

#### RELATED COMMANDS

INSERT, SIBOTTOM, SINPUT, SITOP, TINPUT

BT

- BT Block Trail Continuous

BTCONTINUOUS creates a logically continuous Trail of all Blocks whose Keywords satisfy the boolean request, for online perusal (FULLPRINT will not follow a Block Trail).

BTcontinuous <bool>° <wind>°

#### NOTES

- <bool> defaults to the previous specification of <bool>.
- Display is set at the top of the first Block.
- All Blocks are displayed contiguously, one after the other. Thus, SCROLLing over the end of a Block in a continuous Trail will automatically cause the display to move to the top of the next Block in the Trail. However, if Block boundaries are crossed to find a pattern (e.g., to LOCATE through a Trail), the Trail will be lost and must be explicitly reestablished. This can be done by simply issuing the "Block Trail" command without the <bool> parameter (it will default).
- TRAIL may be used to travel forward or backward along the Trail, as well as returning to the Trail if it is left.
- The Block Trail environment may be left by any non-linear travel function (e.g., DSPACE, GLABEL).
- Nested Blocks will only be retrieved if each higher level Block also satisfies the request.
- If Weights are specified in the boolean request, the Blocks are retrieved in order of Weight, from highest to lowest.

EXAMPLES

bt/Keyword1&key2|key3

All Blocks with both Keyword1 and key2 are retrieved, as well as all Blocks with key3.

bt/Keyword1&(key2|key3)

All Blocks with both Keyword1 and key2 are retrieved, as well as all Blocks with both Keyword1 and key3.

bt/~key1

This request is invalid since it requests all Blocks which do not have key1; it is not permitted to enumerate all Keyworded Blocks (possibly hundreds).

bt/key,6

All Blocks having keys with a Weight between 15 and 6, or no Weight (which satisfies all requests) are retrieved.

MULTIWINDOW

- The block will be displayed in the specified window, or some window other than the current one, if not specified.

RESTRICTIONS

- A Block Trail can consist of no more than 100 Blocks (an internal limit).

MESSAGES

- Error messages occur when 1) all Keyworded Blocks are requested, 2) a Keyword requested is not in the file, or 3) the request is syntactically incorrect (mismatched parentheses, two &'s in a row, etc.).

RELATED COMMANDS

BTDISCRETE, TRAIL



BTD

- BTD Block Trail Discrete

BTDISCRETE creates a discrete (non-continuous) Trail of all Blocks whose Keywords satisfy the boolean request. Except as noted below, BTDISCRETE has the same characteristics as BTCONTINUOUS.

BTDiscrete <bool>° <wind>°

#### NOTES

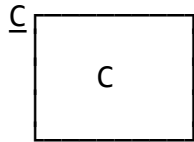
- A discrete Trail contains the same Blocks as a continuous Trail, but it displays them singly in their normal context in the file instead of in a chain.
- The user must specifically request to go the next (or previous Block using TRAIL.
- SCROLLing through the end of a Block is permitted, but TRAIL must be issued to return to the discrete Trail.
- See BTCONTINUOUS for more information.

#### MESSAGES

- Error messages occur when 1) all Keyworded Blocks are requested, 2) a Keyword requested is not in the file, or 3) the request is syntactically incorrect (mismatched parentheses, two &'s in a row, etc.).

#### RELATED COMMANDS

BTCONTINUOUS, TRAIL



- C    Change

CHANGE replaces a string of text in a file with a new string.

Change <scope> <text>

<scope>...the old text to be substituted for  
<text>....the literal string to be substituted

#### NOTES

- If <text> is omitted, Input Mode will be entered (an example of this is illustrated in the description of SUBSTITUTE, an identical command).

#### MESSAGES

BOTH LP'S MUST BE IN THE SAME DATA FIELD  
CANNOT DELETE START OR LAST END LINE OF SPACE  
DELETE INCLUDES TOO MANY BLOCKS  
EXPLAINERS ARE LIMITED TO 255 CHARACTERS  
INTERNAL LIMIT REACHED -- EDIT CANNOT BE DONE  
LABELS LIMITED TO 16 CHARACTERS  
NO REGULAR TEXT ALLOWED IN STRUCTURE SPACE  
PASSWORDED FILE NOT OPEN  
VIEWSPECS LIMITED TO 255 CHARACTERS  
WARNING: INTERFILE JUMP FOUND WITHOUT PMUJ  
WARNING: TEXT IN ANNOTATION BLOCK NOT DELETED

#### RELATED COMMANDS

INSERT, IBEFORE, SUBSTITUTE, USUBSTITUTE

CA

- CA Capitalize Text

CAPITALIZE forces the indicated text to upper-case.

CApitalize <scope>

<scope>...the string to be capitalized

#### NOTES

- All regular Text of Jump Explainers can be capitalized.

cap/reg...ners

[before]=> All regular text or Jump Explainers  
[after]=> All REGULAR TEXT OR JUMP EXPLAINERS

- If a format code and its beginning delimiter ("!") are encountered in <scope>, the code will be forced to lower case.

cap/!-P-watch...code.

[before]=> !-P-watch the "P" of the format code.  
[after]=> !-p-WATCH THE "P" OF THE FORMAT CODE.

#### MESSAGES

FUNCTION NOT ALLOWED IN STRUCTURE

#### RELATED COMMANDS

UNCAPITALIZE

CF

- CF Copy File

CFILE duplicates the entire specified file.

CFile <file1> <file2>

<file1>...the file to be copied  
<file2>...the new filename

#### NOTES

- CFILE is useful for making backup copies, saving intermediate drafts, or generating multiple versions of a document.

#### RESTRICTIONS

- <file1> must be on the A-disk.
- <file2> is created on the A-disk.

#### MESSAGES

COPY SUCCESSFUL  
FILE ALREADY EXISTS  
FILE TO COPY NOT FOUND  
INVALID FILENAME

CFW

- CFW Copy From Work Space

CFWORK copies all the text currently in the Work Space, into the Text Space, after the <lp> specified.

CFWork <iip>°

<iip>...the <lp> after which the copied text will be inserted

#### NOTES

- The text in the Work Space remains unaltered.
- The text from all Areas in the Work Space is copied - no selectivity is possible, however, the Area lines themselves cannot be copied, so a text-only edit should be indicated, if necessary.
- If less than the complete Work Space is desired, COPY should be used, with the <lp> deferred.

#### MESSAGES

NOTHING IN WORK SPACE TO BE MOVED

#### RELATED COMMANDS

COPY, CTWORK, MFWORK, MTWORK

CM

- CM    Execute CMS Subset Environment Command

CMS will execute the specified CMS subset command.

CMS <CMS subset command>

or

CMS

<CMS subset command>...the CMS subset command to be executed

#### NOTES

- If no <CMS subset command> is specified, the user is put into the CMS subset environment (see IBM VM/370: User's Guide). Once there, any number of CMS subset commands can be issued. To return to FRESS, the CMS subset command RETURN must be typed.
- Some useful CMS subset commands are CLOSE, COST, SPOOL, and QUERY (see the IBM VM/370: User's Guide for a complete list).

#### MESSAGES

CMS ERROR CODE <nnnnn>  
INVALID SUBSET COMMAND  
PROCEED  
UNKNOWN CP/CMS COMMAND

CO

- CO Copy Text

COPY duplicates the specified text.

COpY <scope> <lp>

<scope>...the text to be copied

<lp>.....the location after which the copied text will be  
inserted

#### NOTES

- The text remains unchanged in its original location.
- Format codes may be included in the text being copied.
- Text may be copied from one file or Space to another, by deferring the <lp> and then traveling to another file or Space using either GLABEL or GFILE, and resolving the <lp>.

#### RESTRICTIONS

- To preserve uniqueness, Hypertext in the original string may not be copied.
- Blocks may be copied interfile, but not when the "B" qualifier is used.

#### MESSAGES

HYPERTEXT ENCOUNTERED: ACCEPT (A), REJECT(R), OR TEXT ONLY(T)  
MOVETO LP BETWEEN SCOPE LP'S

EXAMPLES

co/ double...phrase/phrase

[before]=> We should double this phrase.

[after]=> We should double this phrase double this phrase.

- Interfile Copying:

[user] co/Multi-file sentence/?  
[system] WAITING

[user] g1 location file2  
[system] %L(location)

[user] ?/tion)  
[system] %L(location)  
Multi-file sentence

Here <lp> is deferred, the second file (file2) is retrieved and the display is moved to one of the Labels in file2, "%L(location)." The <lp> is then resolved.

RELATED COMMANDS

MOVE



CP

- CP    Change Password

CPASSWORD switches the current operational password on the file.

CPassword <pass>

#### NOTES

- CPASSWORD is useful for allowing someone else to view the file without any alteration capability (see APASSWORD), sparing the need to use FFILE and retrieve it under a different password.
- If the new password does not have a CPASSWORD capability, it will not be possible to subsequently change passwords again (unless FFILE is used to free the file and GFILE is issued with another password).

#### RESTRICTIONS

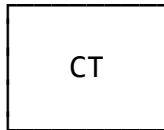
- <pass> must be a previously created password (see APASSWORD).

#### MESSAGES

INVALID PASSWORD

#### RELATED COMMANDS

APASSWORD, DPASSWORD, GFILE



- CT Copy to Label

CTLABEL copies the specified text to the point immediately after the specified Label.

`CTlabel <scope> <label>`

<scope>...the text to be copied

<label>...the Label after which the text will be copied to

#### NOTES

- The text remains in its original location.
- CTLABEL is useful for gathering miscellaneous notes on a single subject. After creating an appropriate Label, CTLABEL can be used to copy and collect relevant sections, while scanning through the file.

#### RESTRICTIONS

- This command cannot be used to copy between files.
- To preserve uniqueness, Hypertext in the original string may not be copied.

#### MESSAGES

HYPertext ENCOUNTERED: ACCEPT (A), REJECT(R), OR TEXT ONLY (T)  
LABEL SPECIFIED DOES NOT EXIST

#### RELATED COMMANDS

COPY, MLABEL, MTLABEL

CTW

- CTW Copy to Work Space

CTWORK copies the desired text to the bottom of the last Area in the Work Space.

CTWork <scope>

<scope>...the amount to be copied

#### NOTES

- The original text remains in the main text body.
- The descriptions of COPY and MOVE should be consulted for contrast.

#### RESTRICTIONS

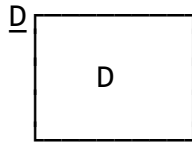
- To preserve uniqueness, Hypertext in the original string may not be copied.

#### MESSAGES

HYPERTEXT ENCOUNTERED: ACCEPT (A), REJECT(R), OR TEXT ONLY(T)

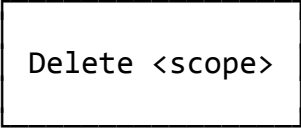
#### RELATED COMMANDS

COPY, MTWORK, MOVE



- D Delete Text

DELETE deletes the specified portion of the file.



Delete <scope>

<scope>...the string to be deleted

#### NOTES

- In Transcription Mode (see SMODE), if a line of the display buffer is deleted (using the "L" qualifier or otherwise), the line that was above it becomes the first line of the display buffer.

#### EXAMPLES

d/TWOØ

[before]=> SYSTEM SHUTDOWN AT 2200 FOR TWO HOURS.  
[after]=> SYSTEM SHUTDOWN AT 2200 FOR HOURS.

#### MESSAGES

BOTH LP'S MUST BE IN THE SAME DATA FIELD  
CANNOT DELETE START OR LAST END LINE OF SPACE  
DELETE INCLUDES TOO MANY BLOCKS  
DELETE SCOPE MUST BE WITHIN KEYWORD  
INTERNAL LIMIT REACHED -- EDIT CANNOT BE DONE  
PASSWORDED FILE NOT OPEN  
WARNING: INTERFILE JUMP FOUND WITHOUT PMUJ  
WARNING: TEXT IN ANNOTATION BLOCK NOT DELETED

#### RELATED COMMANDS

CHANGE, SUBSTITUTE, USUBSTITUTE

DP

- DP Delete Password

DPASSWORD deletes the specified password from the current file.

DPassword <pass>

#### NOTES

- The default password, "default," can be deleted using this command.

#### WARNINGS

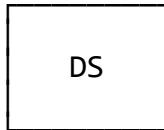
- Deleting all of the file's passwords will make it inaccessible.

#### MESSAGES

PASSWORD DOES NOT EXIST

#### RELATED COMMANDS

APASSWORD, CPASSWORD



- DS Display Space

DSPACE moves the display to the top of the specified system-defined Space.

```
DSpace <space>°<wind>°
      <current>
```

<space>...Display is moved to the first entry of the specified system Space, unless otherwise indicated below:

Annotation...The Annotation Space -- Annotation Blocks are listed in the order of creation.

Keyword.....The Keyword Space -- Keywords, Attributes, and Values are listed in alphabetical order. Values are also listed alphabetically with the Attribute (see Appendix C.3).

Label.....The Label Space -- all Labels are listed alphabetically.  
Picture Picture Space

Structure....The Structure Space -- all Structure (Hypertext) is listed geographically (the order in which it appears in the file).

Text.....The Text Space -- the "\*START OF TEXT AREA\*" line is pointed at and displayed.

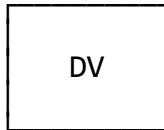
Work.....The Work Space.

#### MULTIWINDOW

- The space is chosen from the current space of the file displayed in the specified window, and replaces the contents of that window.
- The current window is assumed if no window is specified.

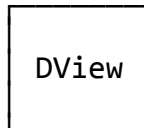
MESSAGES

SPECIFIED SPACE DOES NOT EXIST  
UNDEFINED SPACE  
UNKNOWN SPACE



- DV    Display Viewspecs

DVIEW displays the current viewing specifications (Viewspecs).



#### NOTES

- The Viewspecs determine which parts of the file will be seen online, and how (see SVIEW for description).
- DVIEW expresses the Viewspecs in terms of one of four standard Viewspec strings modified by optional additions and deletions which correspond to the system-defined strings described in SVIEW:
  - VS1    NORMAL: Online formatting, no justification, and format and Structure codes displayed.
  - VS2    PRINT: Online formatting, no format codes displayed, no Structure displayed, right justification. No editing is allowed.
  - VS3    EDIT:    Minimal online formatting, no right justification, format codes displayed.
  - VS4    <null string>: used when none of the other standard strings are close to what the viewspecs actually are.
- If a special blank character is set (B<k>), DVIEW will only return "B," not the special character.

#### WARNINGS

- DVIEW may not always reflect the correct viewspecs.
- DVIEW should not be used with the FRESS Command Separator (see Section 5.4).



EXAMPLES

VS3+B -SP C

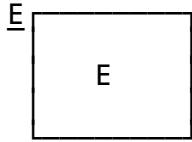
The usual EDIT characteristics, but blanks will appear as a previously set special character, Structure code delimiters will not be shown ("SP"), and Keywords will not be displayed ("C").

VS4+T FL

Only regular text ("T") and Area lines ("FL") will be displayed.

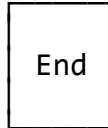
RELATED COMMANDS

SVIEW



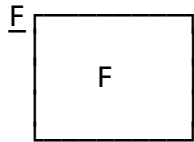
- E      End FRESS Session

END returns the user to the CMS environment.



#### NOTES

- All open files are closed and all editing changes are made permanent.
- The virtual printer is closed, but the spool class remains the same as while in the FRESS environment.
- The line-end and character-delete settings effective in FRESS remain in effect after returning to CMS.



- F      Free File

FFILE closes the specified file, releasing it from the current file list.

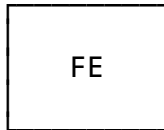
Ffile <file>

#### NOTES

- An ACCEPT is issued automatically by FRESS before the file is "freed", thus making permanent the last editing operation.
- Once freed, no interfile operations may be performed involving the file until a subsequent GFILE is issued, which opens a file.

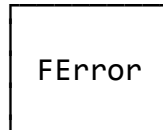
#### MESSAGES

FILE NOT IN USE  
PROCEED  
TOO FEW VALUES GIVEN



- FE Find Error

FERROR will move display to the first formatting error ahead of the current location.



### MODIFIERS

FERROR may be used with the modifiers "B" and/or "L" in any combination or order.

B...Backwards: The search will be backwards towards the top of the file.

L...Long: The entire file will be searched (rather than just 8000 characters).

### VARIANTS

FEB	FEBack
FEL	FELong (to bottom of area)
FEBL, FELB	FEBLong, FELBack (to top of the area)

### NOTES

- Errors in edit codes, alter codes, or format macros will be found.
- If no error is within range, the display will remain at the current location, and a message will be printed.
- FERROR will find errors within 8000 characters of the start of the buffer, unless the "L" modifier is used, in which case the command will search to the end of the file.

### MESSAGES

FERROR: ERROR NOT FOUND, END OF AREA  
FERROR: ERROR NOT FOUND, END OF COUNT

FL

- FL Flip

FLIP changes the case of each character within <scope>.

FLip <scope>

<scope>...the text to be "flipped"

#### NOTES

- Upper-case characters become lower-case, and vice versa, except as noted below.
- If the beginning delimiter ("!") of a format code is included in <scope>, it be forced to lower case [to avoid confusion on upper-case only terminals; (see CAPITALIZE)].
- The first non-blank character after any "sentence-ending" punctuation (period, question mark, or literal exclamation point) or after any format code will be capitalized. This allows multiple sentences to be FLIPped while ensuring the first letter of each sentence to be in upper-case. However, the "sentence-ending" punctuation or the format code must be included in <scope> for this to be true (see UNCAPITALIZE).

#### MESSAGES

FUNCTION NOT ALLOWED IN STRUCTURE

#### RELATED COMMANDS

CAPITALIZE, FOOTNOTE, SURROUND, UNCAPITALIZE, UNDERSCORE

EXAMPLES

f1/!...stand

[before]=> !-p-UNDERstand  
[after]=> !-p-UnderSTAND

Note that "-p-" was not capitalized, but the "U" remained capitalized.

f1/St...OKAY?

[before]=> !-s-Straighten THESE TWO out. OKAY?  
[after]=> !-s-STRAIGHTEN these two OUT. Okay?

FMSG

- FMSG FRESS Message Control

FMSG selectively turns on or off any formatting error messages as well as other messages encountered when editing.

FMSG <option>  
OFF|ON

#### OPTIONS

OFF...messages will be suppressed

ON....messages will appear as normal

#### EXAMPLES

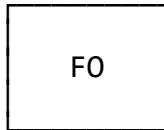
- Should the user keep macro definitions in a file that is IMBEDded, "FMSG OFF" might be issued, for example, to turn off "UNDEFINED MACRO" messages.

#### MESSAGES

PROCEED

#### RELATED COMMANDS

MCOMMENT



- FO Make Footnote

FOOTNOTE transforms the specified text into a footnote.

FOotnote <scope> <n>°

<scope>...the text to be made into a footnote  
 <n>.....the optional footnote number

#### NOTES

- The specified text becomes a footnote; it is surrounded with footnote format codes (!-f-):

fo/I...note.

[before]=> I want to become a footnote.  
 [after]=> !-f-I want to become a footnote. !-f-

- A footnote number can optionally be specified:

fo/Make...five./5

[before]=> Make me footnote five.  
 [after]=> !-f5-Make me footnote five. !-f-

- See the !-F- edit code description for a further discussion of footnotes.

#### MESSAGES

FUNCTION NOT ALLOWED IN STRUCTURE

#### RELATED COMMANDS

CAPITALIZE, FLIP, SURROUND, UNDERSCORE, UNCAPITALIZE



FU

- FU Full Printout

FULLPRINT produces fully formatted output.

```

Fullprint <options>° <file>° <pass>° <lp>°
          <current> <current> <beginning>
          APPENDIX<n>
          AT<1|n>
          DECLEV<6|n>
          FROM<1|n>
          HYPER
          LARGE
          MACROS
          OFFSET<0|n>
          PDISK
          STOP
          TO<32000|n>
          UPPER
          VIDEO
          1403|2741

```

<options>...Any number of the modifying parameters listed below.  
Multiple options must be delimited by commas:

<options> = <option>[,<option>...]

- Appendix<n> • The first level of each Decimal Block, beginning with Block <n>, is labelled with a letter instead of number -- starting with "A." For example,
- A Appendix
    - A.1 Sub-Appendix
  - Decimal Tags referencing these Blocks will be changed accordingly.
  - Example: If the option "AP4" is included in the FULLPRINT of a file with six decimal Blocks, the first three Blocks are numbered "1," "2," "3," but the fourth, fifth, and sixth Blocks are numbered "A," "B," "C" respectively.

- At<n>
- The first page printed is numbered <n> (overriding any format code specification in the file).
  - AT0 suppresses page numbering for the entire printout.
  - If this option is not specified, the page numbering will begin with "1" unless overridden by the FROM option.
- Declev<n>
- The lowest level Decimal Block to be printed during FULLPRINT.
  - Example: If "D3" is included as an option, Block 2.2.3.1 will not be printed; however, Blocks 2.2.3, 2.2, and 2 will be.
- From<n>
- The first page printed is physical page <n> (not numbered page <n>).
  - The portion of the file that would have appeared before physical page <n> is still processed, but printing is suppressed.
- Hyper
- Format codes are printed inline with the text and interpreted (although nested format macros are not interpreted clearly).
  - This option is useful for identifying format codes, although overlaying may occur with tab codes.
- Large
- The maximum physical page size is increased to a width of 132 characters and a depth of 136 lines.
  - The page alter code defaults to !+page132,128+, and the width alter code may be increased to !+width132+
  - This option is useful when producing originals for photo-reduction.
  - A user must have 576K of virtual storage to use this option.
- Macros
- The heading appearance and margins for each decimal Block encountered are controlled by a series of macro definition pairs, which must be inserted into the file by the user.
  - The headings of decimal level <n> are determined by the macros: "!.h<n>.". For example, a possible heading definition for level one is "!.h1=!-h1-.".
  - The margins for Blocks of decimal level <n> are defined by the macros: ".dm<m>.", where <m>=<n>-1. A common margin definition for level one is "!.dm0=!+margin0,0+."
  - Heading appearances (whether a particular heading level is capitalized, etc.) can be redefined using the !+HEAD+ alter code.
  - Format macros are fully described in Section 4.5.

- Offset<n>
- The output will be shifted <n> spaces to the right, on every page.
  - The sum of <n> and the value of the !+WIDTH+ alter code must not exceed 110 (unless Large is specified).
  - This option is in no way related to the !+OFFSET+ alter code for offsetting pages alternately on the left and right, it is just a way of increasing the left-margin at run-time.
- Pdisk
- Instead of writing the formatted file to the virtual printer, this option causes it to be written to the primary (A) disk, appended to the file "<file> LISTING A" (<file> is the same as the FRESS file).
  - If "<file> LISTING A" does not exist, it is created.
  - The user should have 576K of virtual storage if this option is to be used.
  - If a permanent formatted copy is not necessary, the command "CMS SPOOL PRT COPY <n>" can be used to obtain <n> copies instead of using this option.
- Stop
- Output is automatically routed to the terminal.
  - The system waits for a carriage return before each page.
  - This option is useful for printing on discontinuous forms (e.g., stationery on a typewriter-like terminal).
- To<n>
- The last page printed is physical page <n> (not numbered page <n>).
  - The total number of pages printed is TO<n> - FROM<n> + 1.
- Upper
- All output will be in upper-case.
- Video
- Special characters will translate to analogous characters displayable on a CRT-type (video) terminal (e.g., "[" prints as a "(" or "|" rather than a blank).
- 1403
- Output is printed on the offline printer.
  - The minimum abbreviation for this option is "1."
- 2741
- Output is routed to the terminal.
  - FULLPRINT waits for a carriage return before beginning output.
  - The difference between "2741" and "Stop" is that "Stop" temporarily stops before each page, and "2741" stops only at the beginning.
  - The minimum abbreviation for this option is "2."

<pass>.....The password on the file to be formatted [not necessary if <file> is the current file]. This parameter can be omitted if the password of <file> is "DEFAULT" (the default password).

<lp>.....The location at which processing is to begin. If this parameter is not specified, processing begins at the top of the file.

#### NOTES

- Macro definitions must be included in the portion of the file after <lp>, or the macros will be undefined and flagged as such.
- FULLPRINT runs independently of any online formatting in effect.
- If a formatting or Imbed error is detected during printing, the line containing the error will be marked on the far right by "\*\*\*\*," and an error message will be typed at the terminal.
- Undefined macros are ignored.
- FULLPRINT does not issue an implicit ACCEPT before execution.

#### RESTRICTIONS

- A comma (",") cannot be used as the Key delimiter for this command if <options> are to be specified.
- The Command Separator (">") cannot follow this command.

#### WARNINGS

- FULLPRINT may not work properly. If it fails in the vicinity of a footnote, try adjusting the file by inserting a few extra lines, rearranging some of the text, or skipping to a new page before the footnote.

#### RELATED COMMANDS

IMBED, OTYPE

EXAMPLES

fu/from1,to32000,at1,1403,offset0,declev6

This represents all the default settings.

fu/2,v///%L(ri)

This will print the current file online ("2"), starting at Label "ri" ("%L(ri)"), and will translate special characters ("v") so they will be represented at the terminal.

fu/a6,f10,t12/triple

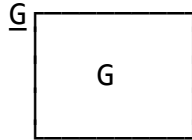
File "triple" (which does not have to be open) will be processed. Only the tenth, eleventh, and twelfth pages will be printed, and they will be numbered 6, 7, and 8

fu/o9

This is useful for centering output with a default width of 65 on NW forms.

MESSAGES

BOUNDARY ERROR  
DELIMITER ERROR  
FILE ERROR ON IMBED  
FILENAME?  
FINI  
FOOTNOTE(S) TOO LONG  
\*\*IMBED ERROR\*\*  
IMBED FILENAME TOO LONG  
IMBED NESTED TOO DEEPLY  
IMBEDDED FILE NOT FOUND  
INV OPTION  
INVALID CODE  
INVALID JUMP  
INVALID PASSWORD  
INVALID PARM, OFFSET=  
INVALID PASSWORD ON IMBED  
NO OR 3 OPEN FILES  
SYNTAX ERROR, OFFSET=  
UNDEFINED MACRO



- G      Get File

GFILE makes current the specified file.

Gfile <file> <wind>° <pass>°

#### NOTES

- GFILE must be used to initially access (open) a file. The display buffer is set to the "\*START OF TEXT AREA\*" line, but it is not displayed.
- <pass> need not be specified if the file has the system default password "DEFAULT."
- There is no limit, except the availability of computer memory, to the number of files that can simultaneously be open.
- Once a file is open, it can be SCROLLED OR JUMPed into from another file, if it has the password "DEFAULT."
- GFILE is useful when there are several open files, and the user wants to change the "current" file; the <pass> option need not be specified to make "current" an open file.
- Either GFILE or GDLABEL must be used to make "current" an open file when there is not already a "current" file.

WARNINGS

- If GFILE is issued with a <pass> parameter and the file is already open, <pass> is ignored. CPASSWORD should be used to change passwords.

EXAMPLES

g/file1

This command will retrieve the file "file1" if there is no password on it other than the default password "DEFAULT."

g/secret/viewer

Assuming the file "secret" has the password "viewer," this command will retrieve secret, but only allow the user to issue the commands specified by "viewer" (e.g., PRINT, TYPE). The password in effect could be changed later without issuing another GFILE if CPASSWORD is used.

MULTIWINDOW

- The current window is assumed if no other is specified.
- The buffer is displayed in the specified window when GFILE is issued.

MESSAGES

FILE NOT FOUND  
INVALID PASSWORD

RELATED COMMANDS

CPASSWORD, GDLABEL, GLABEL, MFILE



- GD    Get Decimal Label

GDLABEL locates the specified Decimal Block.

```
GDlabel <n> [<wind><file>°
                *
```

```
<n>.....the number of the Decimal Block to be retrieved
<file>...the open file in which the Block is searched for (if "*"
           is specified all open files will be searched)
```

## MULTIWINDOW

- The current window will default if no other is specified.
- The searching is done in the file displayed in the specified window.
- The block will be displayed in the specified window.

## MESSAGES

```

DECIMAL LABEL DOES NOT EXIST
FILE NOT FOUND OR NOT OPEN
INVALID NUMBER
NO CURRENT FILE

```

## RELATED COMMANDS

## GLABEL, MDBLOCK



GL

- GL    Get Label

GLABEL locates the point in the file where the specified Label is defined and moves display to that point.

```
GLabel <label> <wind>°    <file>°  
                          <current>  
                              *
```

#### NOTES

- If an asterisk ("\*") is specified for <file>, all open files will be searched for <label> instead of just the file indicated (which must be open).
- A substring of the Label may be specified, and the (alphabetically) first Label beginning with that substring will be retrieved.

#### RESTRICTIONS

- If there is no "current" file, this command may not be used.

#### EXAMPLES

- In a file with a Label at the top of each chapter ("chap1," "chap2," "chap3," ...) the command, "GL chap4" would move display to the top of chapter 4.

#### MULTIWINDOW

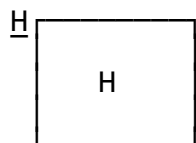
- The label will be displayed in the current window, unless another is specified.
- If <file> is not specified, the file displayed in the specified window will be searched for the label.

MESSAGES

NO CURRENT FILE: IGNORED  
LABEL SPECIFIED DOES NOT EXIST

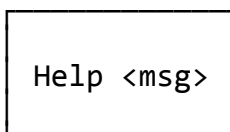
RELATED COMMANDS

MLABEL



- H Help

HELP allows the user to message the HELP virtual machine (at the Computer Center) for assistance.



<msg>...a message of up to 130 characters

#### NOTES

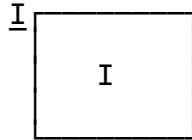
- The HELP consultant may not be able to help with specific FRESS questions. However, general assistance questions should be able to be answered.
- It may take the HELP consultant a few minutes to respond. The user should issue SLEEP so the terminal can receive messages.

#### MESSAGES

HELP NOT LOGGED ON

#### RELATED COMMANDS

SLEEP



- I     Insert Text

INSERT allows text to be added to the file after the specified location.

Insert <iip>° <text>

<iip>....the point after which the text is to go  
<text>...the character string to be inserted

#### NOTES

- If <text> is omitted, Input Mode will be entered after <iip>. Regular Input Mode validates input line by line searching for formatting errors. Swift Input should be used instead; it is faster and less expensive because it does not check for formatting errors.

#### RESTRICTIONS

- INSERT may be used in the Label or Structure Spaces, but only to modify existing Structure, not to insert new text.
- The total number of characters entered on a line when in input may not exceed 130 characters because of hardware limitations. All literal underscore and backspace characters are included in this count.

#### MESSAGES

EXPLAINERS ARE LIMITED TO 255 CHARACTERS  
INPUT  
LABELS LIMITED TO 16 CHARACTERS  
NO REGULAR TEXT ALLOWED IN STRUCTURE SPACE  
VIEWSPECS LIMITED TO 255 CHARACTERS

EXAMPLES

i/run/.

[before]=> See Spot run  
[after]=> See Spot run.

i /Spot/, Dick and Jane's pooch,

[before]=> See Spot run  
[after]=> See Spot, Dick and Jane's pooch, run.

[system]                   i/run/  
                  pooch, run.  
                  INPUT  
                  [Input Mode has been entered]

[user]        Øvery fast

[user]        [a null line is entered]

[before]=> See Spot, Dick and Jane's pooch, run.  
[after]=> See Spot, Dick and Jane's pooch, run very fast.

RELATED COMMANDS

BINPUT, CHANGE, IBEFORE, SINPUT, SITOP, SIBOTTOM, SUBSTITUTE,  
TINPUT

IA

- IA    Insert Annotation

IANNOTATION creates a Block in the Annotation Space which is referenced at the specified location.

IAnnotation <iip>° <keys>° <dkeys>° <text>

<iip>.....the place after which the tag goes  
<keys>....the Keywords to be placed on the Annotation Tag and Block  
<dkeys>...the Display Keywords to be placed on the tag (not the Block)  
<text>....the character string to be inserted in the Annotation Block

#### NOTES

- If <text> is left null, Input Mode will be entered in the Annotation Block.
- After this command is issued, display is in the Annotation Space. To get to the Tag (in the Text Space) referencing the new Block, use the RETURN command.
- Two Implied Insert Pointers are created by IANNOTATION: (see &POP for an example of how to utilize this feature).
- Display Keywords, if specified, allow the annotation to be potentially visible, though SKANNOTATION ultimately has control.

#### RELATED COMMANDS

MANNOTATION, &POP, RETURN, SKANNOTATION, SKDISPLAY

IB

- IB    Insert Before

IBEFOR inserts text before the specified location.

IBefore <lp> <text>

<lp>.....the place before which the text is to be inserted  
<text>...the string to be inserted

#### NOTES

- The <lp> is considered the first character specified in the context string.
- IBEFORE does not use or set the Implied Insert Point.

#### RESTRICTIONS

- IBEFORE can be used only in the main Spaces (Annotation, Text, and Work).
- Input Mode may not be entered using IBEFORE.

#### RELATED COMMANDS

BINPUT,   CHANGE,   INSERT,   SINPUT,   SITOP,   SIBOTTOM,   SUBSTITUTE,  
TINPUT

IBL

- IBL Insert Block

IBLOCK creates a Block containing <text> and inserts it into the file at the specified location.

`IBlock <iip>° <label>° <keys>° <dkeys>° <text>`

<iip>....the place after which the Block is inserted  
<text>...the string contained in the Block

#### NOTES

- If <text> is left null, Input Mode is entered, with the new text to be contained within a new Block.
- Two Implied Insert Pointers are created by IBLOCK: the first points to the character before the Block, and the second, to the last character of the new Block (see &POP for an example of how to utilize this feature).
- IBLOCK and MBLOCK are similar, except that MBLOCK acts on text already in the file.

#### RELATED COMMANDS

IDBLOCK, MBLOCK, MDBLOCK, &POP



EXAMPLES

## IBL/text string

"text string" is placed in a Block at the Implied Insert Point; there are no Labels, Keywords, or Display Keywords on the Block.

[before]=> here

[user] IBL/here/lab//

[system] here  
INPUT

[user] delayed input of text

[user] [null line]

[system] FRESS

[user] p 1

[system] %<(lab)delayed input of textØ%>

An unKeyworded but Labeled Block ("lab" is the Label) is created following "here." Input Mode is then entered and the string "delayed input of textØ" is inserted inside the newly created Block. The "Ø" follows "text" because the user did not append a logical hyphen (see Section 3) to the input line.

ID

- ID    Insert Decimal Block

IDBLOCK creates a Decimal Block containing the new text.

IDblock <iip>° <label>° <keys>° <dkeys>° <text>

<iip>...the point after which the Block is inserted  
<text>...the string to be contained in the Block

#### NOTES

- Two Implied Insert Pointers are created by IDBLOCK: the first points to the character before the Block, and the second, to the last character of the new Block (see &POP for an example of how to utilize this feature).
- IDBLOCK is similar to the IBLOCK command except that it creates a Decimal Block.
- IDBLOCK is the typical means used to input the sections of a file structured with Decimal Blocks.
- IDBLOCK always "pops" the Implied Insert Stack, even if <iip> is specified (see &POP for a further explanation).

#### EXAMPLES

id/AREA\*/Decimal Blocks are tops

```
[before]=> *START OF TEXT AREA*
[after]=>  *START OF TEXT AREA*
           %< '1' Decimal Blocks are tops%>
```

#### RELATED COMMANDS

IBLOCK, MBLOCK, MDBLOCK, &POP

IM

- IM Imbed

IMBED allows the FULLPRINT of one file to access and process another file.

IMbed <iip>° <dkeys>° <file>

<iip>...the place after which the file is to be inserted

#### NOTES

- IMBED creates an Imbed Tag at the specified location. The filename is an Explainer on the Tag. For example,

%Tmacfile%

indicates the file, "macfile," is to be included during FULLPRINT.

- After the expansion of all Imbed Tags, Decimal Blocks and format macro definitions and usages are treated as a "single stream," and are handled in the normal sequential manner.
- Reaching the end of the file signals the end of the Imbed operation.
- Interfile Jumps out of an Imbedded file will not affect the Imbed.
- If an error occurs when an IMBED is attempted, the IMBED is ignored and FULLPRINT continues. The offending Imbed Tag will be marked by "\*\*\*IMBED ERROR\*\*," in the margin of the printout. An error message corresponding to the last error detected (including footnote problems that cause early termination of the printout), will be printed at the terminal at the end of FULLPRINT.

- A separate file containing a standard set of format macros might be IMBEDded at the beginning of various files to avoid unnecessary duplication. However, the macros will be undefined for online display.
- The FMSG OFF command is useful for turning off "UNDEFINED MACRO" messages that occur when editing a file whose macro definitions are contained in a file imbedded in the current file.

#### WARNINGS

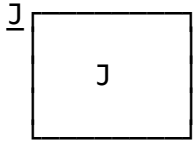
- Imbed Tags should be avoided if possible.
- FULLPRINT may not process an Imbed Tag properly.
- An Imbed Tag may be edited, but no error checking is done on the result.

#### RESTRICTIONS

- Decimal Reference Tags will be incorrect if a file is Imbedded. They are computed only for a single file, and thus the first level will not reflect the proper block number.
- Imbed Tags may not be nested more than ten levels deep.

#### RELATED COMMANDS

FMSG, FULLPRINT



- J      Jump

JUMP is a powerful command for travelling non-linearly within and/or between files.

Jump <lp> <wind>°

#### NOTES

- The old display point will be saved and can be returned to, by using RETURN.
- If the <lp> is in the Label Space, the effect is a GLABEL of the specified Label.
- If the <lp> is in the Structure Space, display will be at the corresponding order in a main Space (Annotation, Text, Work).
- If the <lp> is in a main Space, JUMP's function is as indicated in Figure 5.1.
- A useful application of JUMP is to scan through a file Block by Block, JUMPing from end to end until the desired Block is found.
- If there are several Labels in a file, it is often difficult to remember the name of each. Once a particular Label is located, it can be JUMPed on to get to the corresponding location in the Structure Space. Once there, the user has an online table of contents from which he can get an overview of the file, and then RETURN.

#### MESSAGES

INVALID VIEWSPEC STRING

#### MULTIWINDOW

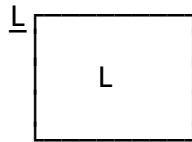
- The "jumped to" point is displayed in the specified window, or a window other than the current one, if none is specified.

RELATED COMMANDS

GDLABEL, GLABEL, RETURN, RING, SAVE

(JUMP from)	(JUMP "lands" at)
<u>MAIN SPACE LP</u>	<u>NEXT DISPLAY STARTS AT:</u>
Label [%L]	that Label in Structure Space
Area Order [***AREA***]	corresponding Structure Space location
Block-start(end) [%<, %>]	matched end (start) unless <lp> is of a Block-start in the Annotation Space, in which case the next display will be at the Tag which references the Block in the Text Space.
Jump(Pmuj) [%J, %P]	matched Pmuj(Jump)
Tag (Annotation, Decimal Reference Label, picture reference) [%T]	place referenced by the Tag
Keyword of order	same as <lp> of order
Explainer	same as Jump or Pmuj
other text	equivalent to a LOCATE, except the <lp> character is considered to be the <u>last</u> character rather than the first character if a literal string is specified.

Figure 5.1 -- Jumping in Main Spaces



- L      Locate (Pattern Scan)

LOCATE searches the file for the first occurrence of the pattern specified, without regard to upper and lower case, and moves the display to the start of the matched string.

Locate <lit>° <wind>°

<lit>...the literal character string to be located (must be less than 253 characters)

### MODIFIERS

LOCATE may be used with the modifiers "B", "L", or "M", in any combination or order:

B...Backwards: The search will be backwards towards the top of the file (up to 8000 characters).

L...Long: The entire file will be searched.

M...Mixed Mode: The case of each character must match exactly (rather than disregarding case).

### VARIANTS

LB            Locate Backwards

LL            Locate Long (to bottom of area)

LM            Locate in Mixed mode (take specified character case literally, without folding)

LBL,LLB      Locate Backwards Long (to top of area)

LBM,LMB      Locate Backwards in Mixed mode

LLM,LML      Locate Long to bottom of area in Mixed mode

LBLM,LBML,LLBM,LLMB,LMBL,LMLB  
Locate Backwards Long to top of area in Mixed mode



NOTES

- When the pattern scan is in the forward direction, it looks first at the second character in the editing buffer. This allows five consecutive LOCATES to find five different occurrences of <lit> (see WARNING below for an exception).
- If <lit> is not specified, it will default from the last LOCATE variant (of any form), and the resulting LOCATE is subject to the following:
  - The new variant of LOCATE is concatenated with the last to determine the type of pattern scan.
  - Once the search is going "backwards", it can't be switched to a forward direction, without respecifying <lit> (e.g., a "ll" that follows a "lb/dog", will have the effect of a "llb/dog").
- Ellipses may be used to specify <lit>, but will use considerably more execution time.

WARNINGS

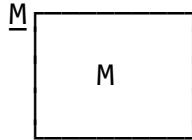
- If Hypertext is at the beginning of the editing buffer, the editor will probably start searching with the first character rather than the second character. Thus, five consecutive LOCATES will find the same occurrence five times.
- Line and Character Displacements (see Section 5.2) must not be used with this command.
- A LOCATE using the "L" modifier is liable to be expensive if the file is large.

MULTIWINDOW

- The pattern scan will be done in the current window unless another window is specified.

MESSAGES

INVALID PATTERN, TRY AGAIN  
NO SAVED PATTERN  
PATTERN NOT FOUND, EO AREA  
PATTERN NOT FOUND, EO COUNT  
TOO LONG CANNOT ADD HEADER  
UN LIGHTPENNABLE TEXT



- M     Move Text

MOVE deletes the text from its original location and inserts it at the specified <lp>.

Move <scope> <lp>

<scope>...the text to be moved  
<lp>.....the place the text is moving to

#### NOTES

- Deferred LP's are often used when moving large Blocks of text; either <scope> and/or <lp> may be deferred.
- Interfile moves (text only) are allowed.

#### EXAMPLES

m/ is easily/tence

[before]=> This is easily sentence edited.  
[after]=> This sentence is easily edited.

#### MESSAGES

HYPertext ENCOUNTERED: ACCEPT (A), REJECT(R), OR TEXT ONLY(T)  
MOVETO LP BETWEEN SCOPE LP'S  
SCOPE INCLUDES INCOMPLETE BLOCK

#### RELATED COMMANDS

COPY, CTLABEL, CTWORK, MFWORK, MTLABEL, MTWORK

MA

- MA    Make Annotation

MANNOTATION replaces <scope> with a Tag referencing a newly created Block in the Annotation Space that contains <scope>.

MAnnotation <scope> <keys>° <dkeys>°

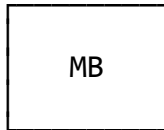
<scope>...the text to be made into an annotation

#### NOTES

- If regular Keywords are specified, they will be placed on the Block and the Tag, and must satisfy the Keyword string specified by SKANNOTATION for the annotation to be visible.
- Display Keywords, if specified, allow the annotation to be potentially visible, though SKANNOTATION ultimately has control over whether the annotation will appear online.
- Annotation Blocks will be printed as footnotes during a FULLPRINT, and typed in-line, if visible, during online display.
- After MANNOTATION is executed, display is where the Tag was established in the Text Space.

#### RELATED COMMANDS

IANNOTATION, MANNOTATION, SKANNOTATION, SKDISPLAY



- MB    Make Block

MBLOCK creates a Block containing the specified text.

```
MBlock <scope> <label>° <keys>° <dkeys>°
```

<scope>...the text to be included in the Block

#### NOTES

- Two Implied Insert Pointers are created by MBLOCK (see &POP).

#### RESTRICTIONS

- Empty Blocks cannot be made; <scope> cannot be null.

#### EXAMPLES

```
mb/This...ked./labx/graphics/visible
```

```
[before]=> This line should be Blocked.
```

```
[after]=> %<$visible$(labx)"graphics" This line should be  
Blocked%
```

"labx" is the Label, "visible" is a Display Keyword, and "graphics" is a regular Keyword. Note: This Block is only visible if the Display Keyword "visible" is enabled by SKDISPLAY.

#### MESSAGES

SCOPE INCLUDES INCOMPLETE BLOCK

#### RELATED COMMANDS

IBLOCK, IDBLOCK, MDBLOCK, &POP, SKDISPLAY

MC

- MC Macro Comment Control

MCOMMENT determines whether comments contained in Command Macros will be printed or not.

MComment <option>  
OFF | ON

### OPTIONS

OFF...messages will be suppressed

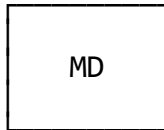
ON....messages will appear as normal

### NOTES

- When the "ON" option is specified, lines within Command Macros that contain a "&C'<text string>'" will print "text string" at the terminal during execution.
- See Section 7 for a full discussion of Command Macros.

### RELATED COMMANDS

FMSG



- MD    Make Decimal Block

MDBLOCK creates a Decimal Block around the specified text.

```
MDblock <scope> <label>° <keys>° <dkeys>°
```

<scope>...the text to be included in the Block

#### NOTES

- Aside from the added feature of its Decimal Labels, MDBLOCK is identical to MBLOCK.
- Two Implied Insert Pointers are created by MDBLOCK: the first points to the character before the Block, and the second, to the last character of the new Block (see &POP for an example of how to utilize this feature).
- If the relative location of a Decimally Labeled Block is changed, the Decimal Label of it and every other Decimal Block will be updated dynamically (as will any reference Tags).

EXAMPLES

md/N...y/num1

[before]=> Number this Block Decimally

[after]=> %< '1' (num1)Number this Block Decimally%>

"num1" is the Label and " '1' " is the Decimal Label

MESSAGES

SCOPE INCLUDES INCOMPLETE BLOCK

RELATED COMMANDS

IBLOCK, IDBLOCK, MBLOCK, &POP, SKDISPLAY

MDR

- MDR Make Decimal Reference

MDREF creates a Decimal Label Reference Tag in the text, referencing the Block whose Decimal Label is specified.

MDRef <iip>° <dkeys>° <n>

<iip>...the <lp> after which the Reference Tag is to be inserted  
<n>.....the current Decimal Label of the Block to be referenced

#### NOTES

- The Tags are dynamic; if the Decimal Block being referenced is renumbered, the Tag will be updated.
- If a Tag refers to a Block which currently is invisible because of splices or Display Keywords, the Tag will appear as "1.999". If neither of these conditions is true and the Tag appears as "1.999", there is a problem with the file, and the FRESS staff should be contacted.



EXAMPLES

mdr/see Section /1.2.3

[before]=> for more information, see Section .  
[after]=> for more information, see Section %T '1.2.3' .

This would appear during FULLPRINT as: "for more information,  
see Section 1.2.3."

MESSAGES

AREA ORDERS NOT ALLOWED IN BLOCKS  
DECIMAL LABEL SPECIFIED DOES NOT EXIST  
INVALID NUMBER

RELATED COMMANDS

MDBLOCK, MDRDEF, SKDISPLAY

MDRD

- MDRD Make Decimal Reference Deferred

MDRDEF differs from MDREF only in that its third parameter is a <lp> of the desired Block start, instead of its Decimal Label.

MDRDef <iip>° <dkeys>° <lp2>

<iip>...the <lp> after which the reference Tag is to be inserted  
<lp2>...an <lp> of the Block start, of the Block to be referenced

#### NOTES

- MDRDEF is typically used with <iip> deferred so the file can be scanned to find it.
- MDRDEF should be used when the current Decimal Label of the desired Block is not known, which is the usual case in dynamic, changing files.

#### MESSAGES

SECOND LP MUST BE A BLOCK START

#### RELATED COMMANDS

MDBLOCK, MDREF

MF

- MF    Make File

MFILE creates a FRESS file with the specified name and password.

MFile <file> <wind>° <pass>°

#### NOTES

- If <pass> is not specified, the password "DEFAULT" will automatically be used. The assignment of "DEFAULT" for <pass> will allow the file to be accessed in the future without password specification.
- After issuing a MFILE, the user is placed in Input Mode and can immediately start inserting text at the top of the file. Text input at this time is placed after the "\*START OF TEXT AREA\*" line.
- It is recommended that the user issue a null line to get back to Command Mode, then issue SINPOT to input text. This results in less expensive processing.

#### MESSAGES

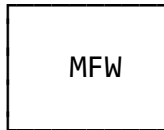
A-DISK IS READ/ONLY OR NOT LOGGED IN  
FILE ALREADY EXISTS  
NO SPACE ON A-DISK FOR NEW FILE

#### MULTIWINDOW

The current window will be used unless another is specified.

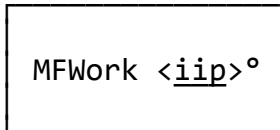
#### RELATED COMMANDS

APASSWORD, CFILE, CPASSWORD, DPASSWORD, FFILE, GFILE, SINPOT



- MFW Move from Work Space

MFWORK moves all the text from the Work Space to the specified <lp> in the Text Space.



<iip>...the location in the text Space after which the text will be moved

#### NOTES

- The text from all Areas in the Work Space is moved; Area lines themselves can not be moved. Thus, if the message "HYPERTEXT FOUND..." is typed, a text-only edit (T) should be specified.
- To move only portions of the text in the Work Space, the user should use the ordinary MOVE instruction with deferred <lp>'s.

#### MESSAGES

HYPertext ENCOUNTERED: ACCEPT(A), REJECT (R), OR TEXT ONLY (T)  
NOTHING IN WORK SPACE TO BE MOVED

#### RELATED COMMANDS

CFWORK, COPY, CTWORK, MOVE, MTWORK

MI

- MI    Make Identifier

MIDENTIFIER substitutes <string> for "&I," in all subsequent command macros that call the "&I" function (see Section 7.2.1.4).

MIdentifier <string>

<string>...a character string of four characters or less

#### NOTES

- If <string> is shorter than four characters, it will be padded on the right with blanks.
- The &I facility, through MIDENTIFIER provides a (limited) means of setting global variables.
- See Section 7 for a complete discussion of the Command Macro facility.

#### RELATED COMMANDS

MCOMMENT

MJ

- MJ    Make Jump

MJUMP creates a conditional Jump (Jump/Pmuj pair) in the file which may optionally be taken to skip a portion of text.

```
MJump <lp1> <lp2> <text1>° <text2>° <keys>°
      <dkeys1>° <dkeys2>° <vs1>° <vs2>°
```

```
<lp1>.....the place after which the Jump is to go
<lp2>.....the place after which the Pmuj is to go
<text1>....the Explainer on the Jump
<text2>....the Explainer on the Pmuj
<keys>.....the Keyword string for the Jump and Pmuj
<dkeys1>...the Display Keywords on the Jump
<dkeys2>...the Display Keywords on the Pmuj
<vs1>.....the Viewspec on the Jump
<vs2>.....the Viewspec on the Pmuj
```

#### NOTES

- If the Jump's (Pmuj's) Keyword string satisfies the Keyword Jump Request String set by SKJUMP, the Jump will be taken -- as though it is a Splice.
- Explainers are literal, editable text strings which appear within the file, as comments for the reader to explain where the Jump (Pmuj) goes to.
- The Jump's Explainer, Viewspec strings, and Display Keywords are independent of those on the Pmuj.
- The Keyword string is the same for both the Jump and the Pmuj; editing one will change the other.
- The Viewspec strings are the same as those used with SVIEW.
- The new Viewspecs take effect at the opposite end of the link. The old Viewspecs are saved by the system and can be invoked either by SCROLLing backwards over the link, or by RETURNing.

EXAMPLES

[given]=> If a Jump is made here\*all of this text will be hidden when the appropriate SKJUMP command is issued. In that case, the Jump would skip to the end marked by ##END##.

mj/here\*/marked by /to the boonies/from made here/dog/seen

The result of this command is the creation of (1) a Jump with the Keyword "dog", an Explainer "to the boonies" and the Display Keyword "seen"; and (2) a Pmuj with the Keyword "dog" and Explainer "from made here." If "SKJUMP/dog" and "SKDISPLAY/seen" are issued, the Jump/Pmuj pair will "splice" the text, so it appears online as:

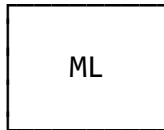
If a Jump is made here\*  
%J\$seen\$"dog"going off into the boonies%%&%P"dog"from made  
here"%%  
##END##.

RESTRICTIONS

- The Keyword string, Display Keyword string, and Viewspec string are each limited to 255 characters.

RELATED COMMANDS

JUMP, MSPLICE, SKDISPLAY, SKJUMP, SVIEW



- ML    Make Label

MLABEL Labels a point in the file, making it directly accessible using GLABEL.

MLabel <iip>° <dkeys>° <label>

<iip>...the place at which to make the Label

#### NOTES

- Labels may contain imbedded blanks.
- Labels can be retrieved even if <dkeys> are not in effect.
- A Label appears in-line as "%L(<label name>)"
- Each Label is entered alphabetically in the Label Space, accessible through the DSPACE command.
- Each Label is entered geographically in the Structure Space.
- Labels may be edited in the Label, Structure, or Text Spaces, but the result must conform to Label specifications.

#### RESTRICTIONS

- Labels must begin with an alphanumeric character.
- Labels must not contain an exclamation mark ("!").
- Labels may not be more than 16 characters.
- Labels are not permitted in the middle of format codes.



MESSAGES

LABEL ALREADY EXISTS  
LABELS LIMITED TO 16 CHARACTERS  
NULL LABEL IS ILLEGAL

EXAMPLES

ml/bozo/chapter2

[before]=> put a new Label after bozo  
[after]=> put a new Label after bozo  
%L(chapter2)

the Label "chapter2" is now directly after "bozo" in the Text  
Space

RELATED COMMANDS

DSPACE, GLABEL

MS

- MS    Make Splice

MSPLICE creates an unconditional jump in the file (Splice/Ecilps pair), taken automatically when reached.

```
MSsplice <lp1> <lp2> <text1>° <text2>° <keys>°
          <dkeys1>° <dkeys2>° <vs1>° <vs2>°
```

```
<lp1>.....the place after which the Splice is to go
<lp2>.....the place after which the Ecilps is to go
<text1>....the Explainer on the Splice
<text2>....the Explainer on the Ecilps
<keys>.....the Keyword string for the Splice and Ecilps
<dkeys1>...the Display Keywords on the Splice
<dkeys2>...the Display Keywords on the Ecilps
<vs1>.....the Viewspec on the Splice
<vs2>.....the Viewspec on the Ecilps
```

#### NOTES

- The text bypassed by a Splice is still in the file, but gone from view. It should be labeled so the user can access it later without deleting the Splice.
- The Splice will take effect as long as it can be "seen" (as governed by Display Keywords). If invisible, it will not be taken online, during OTYPE or FULLPRINT.
- A Splice-Ecilps pair appears online as:

```
%SP<text1>%&%EC<text2>%%
```

where the "&" represents all the text hidden by the Splice.

- If a Splice is specified within the scope of an editing operation (e.g., DELETE), all "Spliced-out" text will also be affected. Given the string, "The hidden material including the Splice will be gone", with "including the Splice" hidden behind a Splice:

s/ hidden...be/ lousy stuff is

[before]=> The hidden material %SP&EC will be gone.  
[after]=> The lousy stuff is gone.

- A Splice is actually the special case of a Jump. Editing the second letter of the Structure abbreviation (i.e., the "P" of "SP," "C" of "EC") is a special way of changing a Splice into a Jump.

d/P

[before]=> %SP"Keyword"explainer%%  
[after]=> %J"Keyword"explainer%%

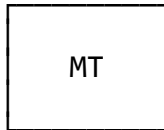
- Keywords on Splices are superfluous, unless the Splice is later to be made into a Jump.
- Splices can be made between files using deferred <lp>'s.

#### RESTRICTIONS

- The Keyword fields cannot be added once a Splice is made.
- Splices can be made into Jumps, but the process is irreversible, except via an immediate REVERT.

#### RELATED COMMANDS

MJUMP



- MT    Move to Label

MTLABEL moves a piece of text directly after the specified Label.

A rectangular box with a black border, containing the text "MTlabel <scope> <label>" in the center.

```
MTlabel <scope> <label>
```

<scope>...the amount of text to be moved

<label>...the Label after which the text will be placed

#### NOTES

- MTLABEL is similar to MOVE. It is useful for gathering diverse fragments at a common location (e.g., alphabetizing an unordered list by moving each entry in reverse order -- z, y, x, ..., a; which yields, the entries in the order a, ..., x, y, z when finished).

#### RESTRICTIONS

- <label> must be specified and cannot be deferred.
- MTLABEL does not work between files.

#### MESSAGES

LABEL SPECIFIED DOES NOT EXIST

#### RELATED COMMANDS

COPY, CTLABEL, MOVE, MTWORK

MTW

- MTW Move to Work Space

MTWORK moves the specified text to the bottom of the last Area in the Work Space.

MTWork <scope>

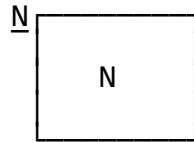
<scope>...the text to be moved

#### NOTES

- All normal FRESS functions may be performed upon the text in the Work Space.
- The Work Space is useful as a common collection point from which text can be copied to several points in the file (see CFWORK), or as a temporary gathering ground from which the text can be moved to the desired location in the file (see MFWORK).

#### RELATED COMMANDS

COPY, CTWORK, MFWORK, MOVE



- N    Next

NEXT moves the display <n> lines.

A diagram showing the command syntax for the NEXT command. It consists of a rectangular box containing the text "Next <n>° [<wind>]°" on the first line and the underlined number "1" on the second line.

<n>...the number of lines the buffer should move

#### NOTES

- The number of characters in each line is as set by the last SDISPLAY (or the default).
- See SCROLL for more information (an identical command).

#### RELATED COMMANDS

PRINT, SCROLL, SDISPLAY, TYPE

NA

- NA    New Area

NAREA creates a new Area at the bottom of the Space specified.

NArea <space>°  
M|T|W

M...main Text Space  
T...Text Space (same as "M")  
W...Work Space

#### NOTES

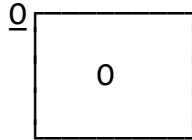
- If no <space> is specified, the Area will be made at the bottom of the Space currently being displayed.
- After NAREA is issued, display moves to the start Area line (i.e., "\*START OF TEXT AREA\*") of the new empty Area.
- It is recommended that immediately after its creation, Structure be imposed on the new Area or it will be accessible only by JUMPing on the Area line through the Structure Space, or by using BOTTOM (if the Area is the last one in the Space).
- Keeping material in separate Areas rather than separate files has the advantage of allowing easier Keyword and Label retrieval.

#### MESSAGES

WORK SPACE DOES NOT EXIST

#### RELATED COMMANDS

BAREA, SPLITAREA



- 0 Offline Read<sup>1</sup>

OREAD creates a FRESS file from one of two sources of input: cards in the virtual card reader, or a disk file.

```
Oread <options> <filename> <password>°
      B                $$$TEMP$$$    DEFAULT
      C<k|>
      D|R
      E<nn|80>
      H<k|>
      L|U
      S<nn|1>
```

### <options>

The <options> can be specified in any order, separated by commas. Conflicting options are resolved by using the one specified last.

B.....(bottom) The input is to go at the bottom of the FRESS file.

C<k>...(capitalize) The character following the capitalization character <k>, will be capitalized.

- A string of text enclosed by double capitalization characters will be all capitalized.
- Example: "C\*" indicates "\*" will be the capitalization character.

D[<.filename>[.filetype>[.filemode>]]]..(input expected from disk) Input will be read from the specified CMS file. The three optional parameters default to "FRESS," "SCRIPT," and "\*, " respectively.

E<nn>...(end in column <nn>) OREAD ignores input past column <nn>.

- Good only for card image files.
- Useful for ignoring sequence numbers.



H<k>...(logical hyphen) Specifies the logical hyphen character as <k>.

- Pertains only to line input when placed at the end of the line.
- Useful for breaking a word across line boundaries.

L.....(lower case) Input will be in both upper and lower case.

R.....(input from reader) Input is expected from the virtual reader.

- Text is considered as one long stream, except that trailing blanks on a card will be ignored.
- Blank cards will be ignored completely.
- Blanks are not inserted after each card.
- If the card reader is empty, "WAITING FOR CARD READER TO FILL" will be typed, and OREAD will wait before continuing.

S<nn>..(start in column <nn>) Input will start in column <nn>.

- Applies only to card image input.

U.....(upper case input) Input will be translated to all lower case, except as indicated with the capitalization character (see C<k>).

#### <filename>

The new FRESS file that is created as a result of this command.

#### <password>

The password on the new file (see APASSWORD for further information).

NOTES

- There are two types of input:
  - Card Image: from cards in the virtual reader, or from card images in a fixed length disk file.
  - Line: input from a variable length disk file.
- If no filename is given, but the D option is specified, <filename> will default from the CMS input file.
- Should neither the D option or <filename> be specified, <filename> will default to "\$\$TEMP\$\$".
- The file "\$\$TEMP\$\$ FRESS" is erased before a new one is created.
- The "\$\$TEMP\$\$" default filename is useful when adding a large Block of text to an existing file. It can be put into a temporary file, and then MOVED into the permanent file.

EXAMPLES

o/d.from.script.a/newfile

Lines are read from the SCRIPT file "FROM" on the A-disk, into a new FRESS file, "NEWFILE," which has the default password "DEFAULT."

MESSAGES

ENDING COLUMN > 80  
FILE ALREADY EXISTS  
FILE NOT FOUND  
STARTING COLUMN > ENDING COLUMN  
STARTING COLUMN < 1  
TOO MANY FILES OPEN  
WAITING FOR CARD READER TO FILL

OT

- OT Offline Type

OTYPE prints on the offline printer the entire current Space exactly as it appears online (under current Viewspecs and line width).

OType <lp>° <n>°  
          1|2|3

<lp>...the point after the current location at which OTYPE begins

<n>....the spacing desired:

1=single spacing

2=double spacing (1 blank line between lines of text)

3=triple spacing (2 blank lines between lines of text)

#### NOTES

- If <lp> is not specified, the entire Space will be printed.
- Backspaces in the file appear as spaces in the printout.
- The printed output will appear in upper and lower case even if an upper-case only terminal is being used.
- If the user wants an OTYPE of only a portion of the file, that section should be copied to the Work Space, from which the OTYPE can then be issued.
- OTYPE is useful for debugging format and Structure codes.
- It is recommended that the line width be maximized to save paper, by using the SDISPLAY command (e.g., SD 130).
- OTYPE should be used for minimally formatted hard copy which includes the format codes and Structure.

EXAMPLES

- To use single or triple spacing without specifying a <lp>, use a null parameter (two consecutive delimiters). For example, to specify a single-spaced OTYPE of the entire current Space, specify: "ot //1"

```
sd 130
ot
```

The current Space will be printed double-spaced maximizing the use of wide standard paper.

```
ds Label
ot //1
```

All the Labels in the file will be printed, single-spaced.

```
ds t
ot /%L(here)/2
```

The Text Space, starting at Label "here" will be printed double-spaced.

```
sv/print+m
ot //3
```

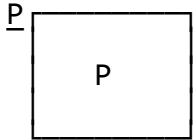
The entire Space will be printed triple-spaced, and minimally formatted with Labels printed ("m" Viewspec -- see SVIEW).

MESSAGES

```
FINIS
PLEASE FREE A FILE & TRY AGAIN
PRINTER TROUBLE
SPACING SPECIFIED INCORRECTLY
```

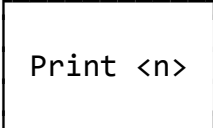
RELATED COMMANDS

SDISPLAY, SVIEW



- P     Print<sup>1</sup>

PRINT is used to type up to one "display buffer's worth" of text at the terminal.



Print <n>

### NOTES

- The maximum number of lines which may be printed is determined by dividing the buffer size by the current line length (see SDISPLAY).
- PRINT does not move the start of the display buffer.
- TYPE or FULLPRINT must be used for long online printouts.
- Issuing "PRINT 1" is the cheapest way to display the current location.
- "PØ" is equivalent to "P 1," but "P" is not!

### RESTRICTIONS

- PRINT should not be used immediately following QUERY.

### MESSAGES

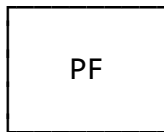
MAX EXCEEDED

### EXAMPLES

- With a line length of 50 (the default on a 2741 typewriter-like terminal), up to 10 lines may be printed.

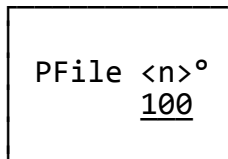
### RELATED COMMANDS

SCROLL, SDISPLAY, TYPE



- PF Pack File

PFILE recreates the current file so that each internal record is filled to the specified percentage.



<n>...the percentage each internal record is to be filled

#### NOTES

- PFILE is used primarily to make a file as small as possible when it is to be stored without editing for a long period of time.
- The new file created after this command is put on the A-disk; if the current file resides there, it is replaced.
- PFILE issues an implicit FFILE when it is finished.
- FRESS internal records are usually balanced at approximately 66% full.
- If the file to be packed is large, the user should consider creating a large temporary disk. One way to do this is to type "CMS TSEG E" The system responds with the message "E (005)...R/W." The user should type "CMS QCOPY <filename> FRESS A = E." The PFILE command can then be issued without fear of running out of disk storage.
- It is not advantageous to pack a file to 100% capacity if editing operations are still to be performed since any increase in amount of text at any point in the file will be likely to decrease editing efficiency.
- One might want to pack a file to perhaps 85% capacity if only minor typographical corrections were to be made subsequently.
- PFILE may be used to expand the file, by specifying a percentage smaller than the current packing level. This is useful for making major corrections more efficient.

WARNINGS

- If the file has Display Keywords, they should be displayed before PFILE is issued.

RESTRICTIONS

- No percentages below 50 are allowed.
- PFILE cannot be reversed using REVERT, but PFILE can be reissued using the original or different percentage.

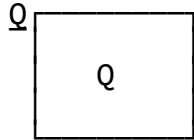
EXAMPLES

```
      [user]      PF 85  
      [system]    PACK SUCCESSFUL 0034/0025
```

This would restructure the current file so that each internal record is approximately 85% full. The message indicates the file previously occupied 34 CMS Blocks and now uses 25 Blocks.

MESSAGES

```
PACK SUCCESSFUL <bbbb>/<aaaa>  
PACKED PAGES MUST BE MORE THAN HALF FULL  
TOO MANY FILES OPEN
```



- Q Query System Status

QUERY describes the current location in terms of Space and Area, displays the three system keyword strings in effect, or lists all open files -- noting which file is current.

```
Query <option>° <wind>°
      Keywords
      Location
      Files
```

Files.....Display the names of all open files.

Keywords...Display the three system keyword strings; those set by SKANNOTATION, SKJUMP, and SKDISPLAY.

Location...Describe the current location in terms of Space name and Area number (which will always be 1 unless multiple Areas have been created).

#### NOTES

- QUERY is very useful when the user has forgotten the names of the files currently open, since the filename must be specified when using FFILE.

#### RESTRICTIONS

- "QUERY F" will list only up to ten files, although more than that may be open simultaneously.
- PRINT should not be used immediately following QUERY.

#### MULTIWINDOW

- QUERY is considered for the current window unless another is specified.
- QUERY F will list all open files, not just those open in that window.



MESSAGES

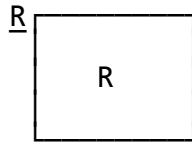
BAD DISP TO OPTION  
BAD OPTION PASSED TO QUERY

CURRENT: <file1>  
          <file>  
          <file>

FILE:     <file>  
SPACE:    <space>  
AREA:     <n>

ANNOTATION KEYWORDS: <bool>  
DISPLAY KEYWORDS:     <bool>  
JUMP KEYWORDS:        <bool>

NO FILE OPEN



- R      Return

RETURN moves the display buffer to the last display point saved on the Return Stack.

A rectangular box containing the text 'Return <wind>°'.

#### NOTES

- The Return Stack is a list of up to 64 display points constructed on a LIFO (last in - first out) basis. When RETURN is issued, the display moves to the display point on top of the Stack (the last one added).
- The current display point is added to the Return Stack whenever a FRESS command causes the display to move non-linearly (DSPACE, GLABEL, JUMP, BOTTOM, MOVE in Transcription Mode, GFILE, and MFILE commands).
- The current display point can be saved explicitly, using the SAVE command.

#### MULTIWINDOW

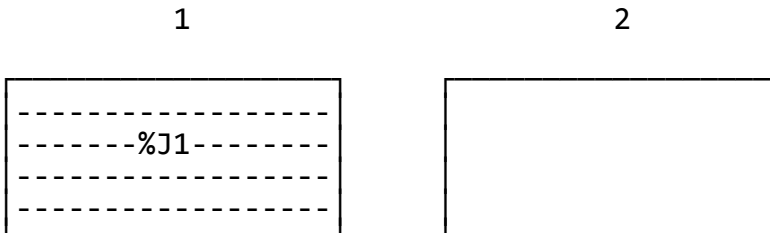
- A display point is saved only if it is replaced by the results of a non-linear travel other than a RETURN. That is, if a GLABEL from window 1 causes the text which was in window 2 to be replaced by the specified label and following text, the display point from window 2 would be saved in the LIFO stack.
- If a window number is specified, the return point will be displayed in that window.
- If no window number is specified, the system will try to place the return point in the window in which it originally appeared. If this is not possible (i.e., the window no longer appears on the screen), another window will be used.
- The window in which the return point is displayed becomes the current window.

EXAMPLES

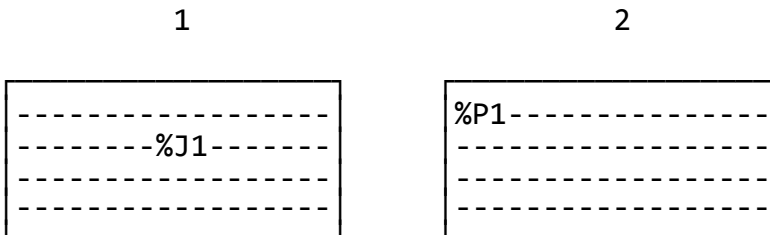
Two examples of the operation of the Return Stack in the multiple window version follow:

## Example 1:

The current window is #1. Window 2 is blank.



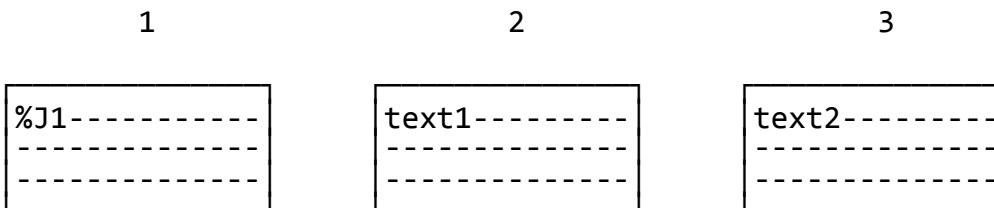
User types 'J/%'. Result is:



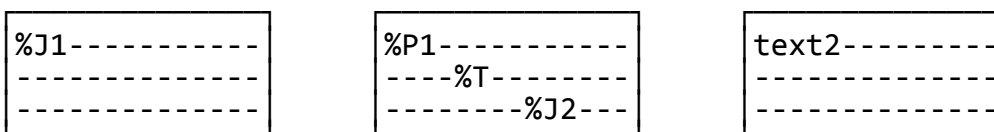
At this point, no locations have been saved in the return stack, as explained above. Issuing RETURN would produce the message 'NO MORE RETURN POINTS' and the display will remain unchanged.

## Example 2:

The current window is 1.



User types 'J/%'. Result is:



User JUMPs on the tag in window 2 using lightpen (without changing current window). Result is:

```
%J1-----
-----
-----
```

```
%P1-----
-----%T-----
-----%J2---
```

```
%<|annotation-
-----%>|-----
-----
```

User JUMPs on jump in window 2 using lightpen. Result is:

```
%J1-----
-----
-----
```

```
%P2-----
-----
-----
```

```
%<|annotation-
-----%>|-----
-----
```

Notice that the new display replaced window 2, not the current window which is still 1. User now types 'ret'. Result is:

```
%J1-----
-----
-----
```

```
%P1-----
-----%T-----
-----%J2---
```

```
%<|annotation-
-----%>|-----
-----
```

The current window is now window 2. User types 'ret/1'. Result is:

```
text2-----
-----
-----
```

```
%P1-----
----%T-----
-----%J2---
```

```
%<|annotation-
-----%>|-----
-----
```

The current window is now window 1. In this case, the display point replaced in window 1 is not saved, since it was replaced on a Return. User now types 'ret/3'. Result is:

```
text2-----
-----
-----
```

```
%P1-----
----%T-----
-----%J2---
```

```
text1-----
-----
-----
```

Window 3 is the current window. There are no more return points now.

### RESTRICTIONS

- RETURN will not work if there is no "current file."

### MESSAGES

```
NO CURRENT FILE: IGNORED
NO MORE RETURN POINTS
RETURN FILE NO LONGER OPEN
SPACE OF RETURN POINT HAS BEEN DELETED
```

### RELATED COMMANDS

RING, &POP, SAVE

REV

- REV Revert

REVERT undoes the last editing operation.

REVert

#### NOTES

- REVERT also resets the starting point of the display buffer to the point in the file at which the now REVERTed editing operation was specified (or completed in the case of a deferred specification).
- Only the last editing operation preceding REVERT may be undone.
- Traveling, both linear (SCROLLing, pattern scanning) and nonlinear (e.g., Jumps, GLABEL), may be done between an edit and a REVERT without negating the REVERT capability.
- ACCEPT makes the last edit permanent.

#### RELATED COMMANDS

ACCEPT

RI

- RI Ring

The Memory Return Ring allows the user to travel easily and repeatedly among specified locations, as opposed to the Return Stack which, through RETURN, enables the user to return to only the last stacked location, and only once.

```
RIng <option>° <wind>°  
A|B|C|D|E
```

#### OPTIONS

- A...(add) Adds current location to ring.
- B...(back) Starts display at previous ring entry.
- C...(clear) Erases all ring entries.
- D...(delete) Deletes the ring entry currently in use, or the one most recently used, and starts the display at the next entry in the ring.
- E...(forward) Starts display at next ring entry.

#### MESSAGES

The window number, if specified, indicates either which display point to add to the ring or where to display the next entry. It has no meaning when the Delete option is specified.

#### RELATED COMMANDS

&POP, RETURN, SAVE

RTA

- RTA Refer To Annotation

A Tag is made which references the indicated Block in the Annotation Space.

RTAnnotation <iip>° <keys>° <dkeys>° <lp2>

<iip>...the place after which the Tag is to go  
<lp2>...the Block to be referenced in the Annotation Space

#### NOTES

- A new Block-start/Block-end pair referenced by the new Tag is placed around the indicated Block.
- Any Keywords specified will be placed on the Tag and the Block.
- Display Keywords, if specified, govern whether the Tag can be viewed online. If the tag is not visible, the Annotation Block is not eligible to be viewed online or printed offline.

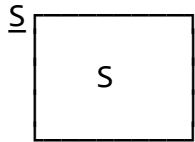
#### MESSAGES

ANNOTATION BLOCK TO REFERENCE NOT FOUND  
SECOND LP MUST BE A BLOCK START

#### RELATED COMMANDS

IANNOTATION, MANNOTATION, SKANNOTATION





- S     Substitute Text

SUBSTITUTE replaces a string of text in a file with a new string.

Substitute <scope> <text>

<scope>...The old text to be substituted  
 <text>....The literal string to be substituted

#### NOTES

- If <text> is omitted, Input Mode will be entered:

[before]=> sentences can be substituted for.

[user]     s/substituted for/  
 [system]   sentences can be .  
              INPUT

[user]     easily edited% [the "%" is to avoid having the  
                                  line padded with an  
 [user]     [null line]     extra blank (see Section 3)]  
 [system]   Sentences can be easily edited.

- Ellipses (...) can be conveniently used to designate long strings:

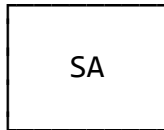
s/n amaz...ent/ sophisticated

[before]=> FRESS is an amazingly convenient text editor.  
 [after]=> FRESS is a sophisticated text editor.

- SUBSTITUTE is identical to CHANGE (see CHANGE for a list of system messages).

#### RELATED COMMANDS

CHANGE, INSERT, IBEFORE, USUBSTITUTE



- SA    Save Current Location

SAVE "pushes" the current display point onto the Return Stack (see RETURN).



SAve <wind>°

#### NOTES

- SAVE explicitly pushes the display point on the Return Stack, normally done only by non-linear traveling functions.
- SAVE is useful when examining some point in the file (by SCROLLing or pattern scanning), as it provides an easy return to the original point.

#### EXAMPLES

SA>T 20>R

will bring the user back to his original viewing point after TYPING 20 lines. Note the use of the FRESS Command Separator (">") (see Section 5.4).

#### MESSAGES

STACK CONTROL BLOCK ALL FILLED

#### RELATED COMMANDS

RETURN, RING, &POP

SC

- SC Scroll

SCROLL moves the display <n> lines of current line length (see SDISPLAY).

<n>        or        SCroll <n>° <wind>°  
                          1

#### NOTES

- SCROLLing may be done either forwards or backwards. To SCROLL forward, the user types just a number. SCROLLing backwards is specified by preceding the number with a minus sign.
- Only an integer (with the optional negative sign) need be specified to SCROLL:
  - 4 (travels forwards 4 lines)
  - 25 (travels backwards 25 lines)
- A SCROLL of zero lines will re-display the current line, however, it is more efficient to use the PRINT command ("P 1").

#### RESTRICTIONS

- No Spaces are allowed between the sign and digits.
- A maximum of 127 lines may be SCROLLed at a time.

MULTIWINDOW

The "SCROLL <n> <wind>" format must be used to SCROLL in other than the current window.

MESSAGES

MAX EXCEEDED

RELATED COMMANDS

NEXT, PRINT, SDISPLAY, TYPE

SD

- SD Set Display Window<sup>1</sup>

SDISPLAY specifies the number of lines of the Display Window that will be displayed after each command (under Display Mode), and the length of each line.

SDisplay <n><sup>o</sup> <len>  
1

<n>.....the number of lines in the Display Window  
<len>...the length of each line in the Display Window (preset to either 50 or 70, maximum is 130)

#### NOTES

- SDISPLAY determines the line length for OTYPE as well. To save paper, issue "SD 130" before OTYPE.

#### RESTRICTIONS

- The line length times the number of lines (<n> \* <len>), must be less than or equal to the Display Window size (see Section 2.2.2 for defaults and buffer sizes).
- It is not always possible to PRINT <n> lines of text for all <len>, even if <n> \* <len> is less than or equal to the size of the Display Window (e.g., the maximum values for <len> when <n> is 10 are 63 and 83 for the two different Editing Buffer sizes).

#### MESSAGES

BUFFER SIZE EXCEEDED  
LENGTH EXCEEDS DEVICE MAX

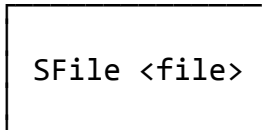
#### RELATED COMMANDS

PRINT, SMODE



- SF Scratch file

SFILE permanently erases the specified file from the user's disk storage.



#### NOTES

- If the file is not on the A-disk it will be FREEd but not scratched.
- The <file> parameter is required in order to avoid accidental erasure.
- GDLabel or GFile must be specified to access another file (even if there are other open ones) after this command is issued.

#### RESTRICTIONS

- SFILE can not be reversed using REVERT.
- The file must be the current file and be on a read/write A-disk in order to be scratched.

#### MESSAGES

FILE MUST BE CURRENT TO BE SCRATCHED  
SCRATCH FAILED  
SCRATCH SUCCESSFUL  
TOO FEW VALUES GIVEN

#### RELATED COMMANDS

FFILE, GDLabel, GFile, MFile

SI

- SI Swift Input

SINPUT puts the user into Swift Input Mode, which is faster and less expensive than Input Mode.

SInput <iip>° <options>°  
                  A|U

<iip>...the place after which the new text will be inserted.

<options>

Either <option> or both can be specified in any order. If neither <option> is specified, text is entered into the file as typed. Any invalid options will be ignored, but no error message will be printed. The options are:

A...(As-Is Mode) All blanks become special blanks and "!-S0-" is put before all input lines (only for this input session).

U...(upper case) All input is translated to upper-case.

NOTES

- FRESS performs no checking for format errors (e.g., on alter and edit codes) when SINPUT is used for inputting.
- While in Swift Input Mode, all text typed is interpreted as literal until a null line is typed (to return to Command Mode).
- A blank is automatically inserted into the file at the end of each inputted line. To avoid this, a logical hyphen ("%") should be typed as the last character on the line.
- Rather than inserting each new line of text into the file as it is typed, as INSERT does, several lines are grouped together before being saved. A blank line is typed at the terminal to indicate when they are safely in the file.

RESTRICTIONS

- The total number of characters entered on a line when in input may not exceed 130 characters because of hardware limitations. All literal underscore and backspace characters are included in this count.

EXAMPLES

si/here./au

[before]=> Let's swiftly input here.  
[system] INPUT

Swift Input Mode will be entered after "here." with both upper-case translation and As-is processing.

MESSAGES

INPUT

RELATED COMMANDS

&ASIS, INPUT, IBEFORE, SIBOTTOM, SITOP



SIB

- SIB Swift Input Bottom

SIBOTTOM puts the user into Swift Input Mode at the bottom of the text Space, just before the \*END OF TEXT AREA\* line.

SIBottom <options>°  
          A|U

<options>

Either <option> or both can be specified in any order. If neither <option> is specified, text is entered into the file as typed. Any invalid options will be ignored, but no error message will be printed. The options are:

A...(As-Is Mode) All blanks become special blanks and "!-S0-" is put before all input lines (only for this input session).

U...(upper case) All input is translated to upper-case.

RESTRICTIONS

- The total number of characters entered on a line when in input may not exceed 130 characters because of hardware limitations. All literal underscore and backspace characters are included in this count.

MESSAGES

\*END OF TEXT AREA\*  
INPUT

RELATED COMMANDS

&ASIS, INSERT, SINPUT, SITOP

SIT

- SIT Swift Input Top

SITOP puts the user into Swift Input Mode at the top of the Text Space, just after the \*START OF TEXT AREA\* line.

SITop <options>°  
A|U

<options>

Either <option> or both can be specified in any order. If neither <option> is specified, text is entered into the file as typed. Any invalid options will be ignored, but no error message will be printed. The options are:

A...(As-Is Mode) All blanks become special blanks and "!-S0-" is put before all input lines (only for this input session).

U...(upper case) All input is translated to upper-case.

RESTRICTIONS

- The total number of characters entered on a line when in input may not exceed 130 characters because of hardware limitations. All literal underscore and backspace characters are included in this count.

MESSAGES

\*START OF TEXT AREA\*  
INPUT

RELATED COMMANDS

&ASIS, INSERT, SINPUT, SIBOTTOM

SK

- SK Set Keyword Annotation Request String

SKANNOTATION specifies a Keyword request string against which Keywords on Annotation Tags are compared when the Tag is encountered for display or printout.

SKannotation <bool>° <wind>°

#### NOTES

- Specifying SKANNOTATION without a request string deletes the previous request so that no Tags will be "followed."
- See Appendix C.2 for Boolean Request Format.

#### DISPLAY EFFECTS

If the Keywords on the Tag satisfy the request string, the Annotation Block associated with the Tag is displayed inline after the Tag. When the end of the Block is reached, the display returns to the text after the Tag.

#### FULLPRINTING EFFECTS

The text in the Block will be printed as a footnote if the Keywords match.

#### MULTIWINDOW

- If a window number is specified, the request string will be applied only to that window.
- If no window number is specified, the Keyword Annotation Request String will be applied to all windows.

#### RELATED COMMANDS

IANNOTATION, MANNOTATION

SKD

- SKD Set Keyword Display Request String

SKDISPLAY sets the boolean Keyword Display String to which Display Keywords are compared. If they do not match, the entire Structure will not appear.

SKDisplay <bool>° <wind>°

#### NOTES

- If <bool> is not specified, or if SKDISPLAY is not issued, the Display Keyword Request String is set to the null string.
- Editing commands reference only what is displayed. Those commands that specify location pointers inside non-displayed Blocks, will not be executed. However, an editing command that includes invisible text in its scope will be executed (and the invisible text will also be affected).
- The Keyword Display String should always be set before creating Display Keyworded Structures. Structures with Display Keywords will be invisible if they don't match the Display Keyword Request String.
- SKDISPLAY should be issued before getting a file. FRESS will search the entire file for a displayable Block, after a GFILE. If no Keyword Display String is in effect, FRESS will create an empty display buffer. This search and subsequent failure consumes computer resources and should be avoided.
- Appendix C.2 describes the rules by which <bool> is specified.

#### MESSAGES

INVISIBLE TEXT

SKJ

- SKJ Set Keyword Jump Request String

SKJUMP sets the boolean Keyword Jump Request String which governs Keyworded Jumps.

SKJump <bool>° <wind>°

#### NOTES

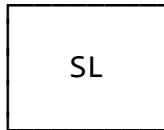
- A Jump encountered either during online display, or during offline printing will be taken as a Splice if its Keywords match the Keyword Jump Request String.
- The String may be replaced at any time with a new SKJUMP Request String and can be eliminated entirely by specifying a null request string.
- Appendix C.2 describes the rules by which <bool> is specified.

#### MULTIWINDOW

- The specified keyword string will be used in only the window indicated.
- If no window number is specified, the string is used in all windows.

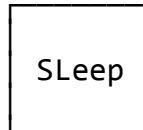
#### RELATED COMMANDS

MJUMP, MSPLICE



- SL Sleep

SLEEP puts the user into a wait state enabling messages to be received.



#### NOTES

- To return to the FRESS environment after issuing SLEEP, the user should press the interrupt, attn, or break key at the terminal.
- SLEEP is useful when the user wants to contemplate what to next, but still wants to receive messages as they are sent.
- It is a good idea to do an ACCEPT before issuing SLEEP, as a guard against later system troubles.

#### MESSAGES

(no response)

#### RELATED COMMANDS

HELP

SM

- SM Set Mode of Display

SMODE specifies the combinations of Display and Editing Modes desired.

SMode <option>°  
B|D  
S|I

### OPTIONS

<option> may be any string of the following symbols concatenated together (with no intervening blanks):

B...(Brief Mode) Display off, nothing printed.

D...(Display Mode) The amount of the Display Window specified by the last SDISPLAY is printed after each command.

S...(Static Edit Mode) Buffer does not move with editing operations (used primarily on display consoles).

I...(Transcription Edit Mode) Buffer moves to one word or code before the beginning of the editing operation performed (system default, used primarily on typewriter terminals)

### NOTES

- The <option> string is interpreted letter by letter from left to right.
- Any invalid letters are ignored, and conflicting Modes (ST or BD) are resolved by using the right-most Mode in the string.
- To temporarily override the current Display Mode, see Section 2.3.1.

### MESSAGES

INVALID MODE CHARACTERS WERE IGNORED

MULTIWINDOW

- The default for the multiple window version on the IMLAC is DS.



SPL

- SPL Split Area

SPLITAREA divides an existing Area within the main Text or Work Space of a file, into two Areas.

SPLitArea <iip>°

#### NOTES

- When SPLITAREA is issued, an \*END OF TEXT AREA\* and a \*START OF TEXT AREA\* line is inserted after the <iip>, and the user's display point is set to the \*START OF TEXT AREA\* line which is the top of the next Area (the one after the split point).
- The only way to access the new Area (once it is left) without a Label, Splice, or Jump into it, is through the Structure Space (unless the Area is the last one in which case BOTTOM could be used).
- SPLITAREA and NAREA differ only in that the former splits the current Area in two, while the latter creates a brand new, empty Area.
- SPLITAREA is useful when the user wishes to isolate text already in the file.

EXAMPLES

```
[before]=>  this is Area one this is Area two

[user]      spl/Area one
[system]    *START OF TEXT AREA*           [start of new
                                              Area]

[user]      print 3                        [display new
                                              Area]
[system]    *START OF TEXT AREA*
            this is Area two
            *END OF TEXT AREA*

[user]      ds                             [move to top of
                                              Space]
[system]    *START OF TEXT AREA*

[user]      print 6                        [display the top
                                              portion of the
                                              Text Space]
[system]    *START OF TEXT AREA*
            this is Area one
            *END OF TEXT AREA*
```

MESSAGES

AREA ORDERS NOT ALLOWED IN BLOCKS

RELATED COMMANDS

BAREA, NAREA

SUR

- SUR Surround Text

This command SURROUNDs the specified text, regular text or explainers, with the literal text.

SURround <scope> <lit1> <lit2>°

<scope>...the text to be surrounded  
<lit1>...literal text to go before <scope>  
<lit2>...literal text to go after <scope>

#### NOTES

- If <lit2> is omitted, <lit1> is used in both places.

#### EXAMPLES

sur/regular...ners/(/)

[before]=> text regular text or Explainers  
[after]=> text (regular text or Explainers)

sur this...marks "

[before]=> this sentence with quotation marks  
[after]=> "this sentence with quotation marks"

#### MESSAGES

FUNCTION NOT ALLOWED IN STRUCTURE  
TEXT TO INSERT MUST BE SPECIFIED  
TOO FEW VALUES GIVEN

#### RELATED COMMANDS

BARS, FOOTNOTE, UNDERSCORE

SV

- SV Set Viewspecs

SVIEW sets the Viewing specifications (Viewspecs) that determine the parts of a file displayed online, as well as offline, when an OTYPE is issued.

SView <vs> <wind>°

#### NOTES

- <vs> can either be a "system defined string", a "Viewspec", or combination thereof (see below).
- The original value of <vs> for the single-window version is EDIT + DK.
- Viewspecs control formatting, and determine which dynamic features (e.g., Keyworded Jumps, Viewspecs on Jumps) are utilized.
- To modify one of the system strings, the user should specify it first, followed by a sign ("+" or "-") which indicates whether the following Viewspec(s) should be added or deleted. For example,

sv /\*-ofp

issued just after logging-in to FRESS would display text formatted properly, but without any format delimiters and codes.

- Viewspecs stay in effect during the entire FRESS session (independent of the file in use), unless SVIEW is issued (or re-issued) or a Jump or Splice containing a Viewspec is SCROLLED over.
- The Viewspecs need not be specified as capital letters.

SYSTEM DEFINED STRINGS

NORMAL...(=, #, @, \$, AK, D, E, FP, FV, JK, M, O, P, SP, T):  
Online formatting, no justification, and format and  
Structure codes displayed, vertical formatting.

EDIT.....(NORMAL-FV): Minimal online formatting, no full  
justification, format codes displayed.

PRINT....(AK, FV, JK, JU, T): Online formatting, no format  
codes displayed, no Structure displayed, right  
justification. No editing is allowed.

\*.....Current Viewspecs.

VIEWSECS

<u>Viewspec</u>	<u>Explanation</u>
=	Block-starts ("%<") and Block-ends ("%>") are displayed
#	Tags ("%T") are displayed
@	Pmuj ("%P") and Pmuj Explainers ("%<text>%%") are displayed
\$	Jump ("%J") and Jump Explainers ("%<text>%%") are displayed
&	Jump connection symbols ("%&") are displayed
AK	Annotation Keywords will be compared to string specified by SKANNOTATION
B<k>	blanks will be displayed as the special character <k>
BL	all special character codes (of the form "!nnn") are displayed (this is actually a subset of the "O" Viewspec)
C	Keywords ("%<key>" and \$<key>\$) are displayed
CN	Splices ("%SP") and Ecilpses ("%EC") are <u>not</u> displayed
D	Decimal Block-start numbers (e.g., "%< '1' ") are displayed
DK	Keyword fields("..." and \$...\$) on Blocks are displayed

E	Explainers (" <code>&lt;text&gt;%%</code> ") are displayed
FL	Area lines (" <code>***AREA***</code> ") are <u>not</u> displayed
FN	Decimal Block-end numbers (e.g., <code>%&gt; '1' "</code> ), " <code>are displayed</code>
FP	format code delimiters (" <code>!</code> ") are displayed
FV	vertical formatting takes effect (e.g., <code>!-s3-</code> will skip 3 lines)
JK	Jump Keywords will be compared to the Keyword string defined by SKJUMP
JU	text is right justified
JV	Jump Viewsecs will be used
M	labels (" <code>%L(&lt;label&gt;</code> ") are displayed
O	format codes (e.g., <code>!-P-</code> , " <code>!+PARA5,1+</code> ", " <code>!.FMAC.</code> ", " <code>!175,</code> " " <code>!(0&lt;text&gt;!)</code> ") are displayed
P	Display Keywords are displayed
Q	Annotation symbols (" <code> </code> ") are displayed
SP	Structure code delimiters (" <code>%</code> ") displayed
S2	lines are typed double-spaced at the terminal
T	text is displayed

### MULTIWINDOW

- The default for the multiple window version is

NORMAL-@-&-O+BL+CN

This produces a display which is pleasant to read but also allows editing operations.

- Special character codes will be interpreted for the IMLAC display while not showing splices, eclipses, or pmujs.
- The viewsec string will take effect in the specified window. If none is specified, it will take effect in all windows.

WARNINGS

- Do not use the FRESS Command Separator (">") with this command.

MESSAGES

ERROR IN STRING NEAR POSITION <nnn>  
TOO FEW VALUES GIVEN

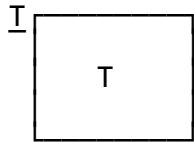
EXAMPLES

- The location of undesirable blanks in a file [e.g., extra underscores in a heading] can be easily discovered and removed by issuing "sv/\*+b\$" and deleting the offending "\$"s each representing a blank. The old Viewsecs can be restored by issuing "sv/\*-b\$".
- To obtain a short formatted printout of a section of a file, the user could locate the section and then do a "sv print" followed by "TYPE 100" (to see 100 lines).

RELATED COMMANDS

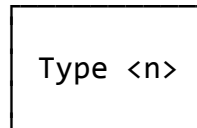
DVIEW, MJUMP, MSPLICE





- T      Type<sup>1</sup>

TYPE displays the specified number of lines at the terminal.



#### NOTES

- The display buffer is moved (unlike in PRINT), so that the last line typed is the start of the display after the completion of the command.

#### RESTRICTIONS

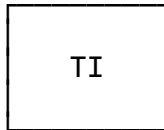
- <n> must be less than 1000.

#### MESSAGES

MAX EXCEEDED

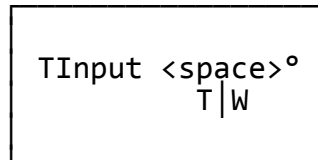
#### RELATED COMMANDS

NEXT, PRINT, SCROLL



- TI Top Input

TINPUT invokes Input Mode at the top of the specified Space.



SPACE

T...Text Space.

W...Work Space.

NOTES

- If <space> is not specified, Input Mode will be entered at the top of the current Space.
- SITOP should be used instead of TINPUT, because it is cheaper.

RESTRICTIONS

- The total number of characters entered on a line when in input may not exceed 130 characters because of hardware limitations. All literal underscore and backspace characters are included in this count.

MESSAGES

\*START OF TEXT AREA\*  
INPUT

RELATED COMMANDS

BINPUT, INSERT, SIBOTTOM, SINPUT, SITOP

TR

- TR Trail

TRAIL must be used to travel along the discrete Block Trail created by BTDISCRETE, and may also be used to travel along a Trail created by BTCONTINUOUS.

TRail <option>° <wind>°  
B|F

#### OPTIONS

B...Backwards along the Trail.

F...Forward along the Trail.

#### NOTES

- If a (continuous) Trail is left in the middle, the TRAIL command will rejoin the Trail where it was left.

#### MESSAGES

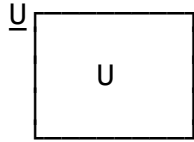
NO CURRENT BLOCK TRAIL  
NO MORE BLOCKS IN TRAIL

#### MULTIWINDOW

The next block will be displayed in the indicated window, or the current one if another is not specified.

#### RELATED COMMANDS

BTCONTINUOUS, BTDISCRETE



- U    Underscore

UNDERSCORE inserts underscore format codes ("!(0" and "!)") around the specified text.

Underscore <scope>

<scope>...the text to be underscored.

#### NOTES

- See Section 4.7 for a complete discussion of underscoring.

#### EXAMPLES

u/I...lined

[before]=> I want to be underlined.  
[after]=> !(0I want to be underlined!).

#### MESSAGES

FUNCTION NOT ALLOWED IN STRUCTURE

#### RELATED COMMANDS

SURROUND

UNC

- UNC Uncapitalize

All alphabetic characters in <scope> will become lower case.

UNCapitalize <scope>

#### NOTES

- If any literal "sentence-ending" punctuation (".", "?", or "!") is contained in <scope>, the next non-blank will be capitalized. This allows correct capitalization for multiple sentences.
- The next character after any alter code, edit code, or format macro, will also be capitalized if the code is included in <scope>.

#### EXAMPLES

unc/!-P-...FOX

[before]=> !-P-THE QUICK BROWN FOX  
[after]=> !-p-The quick brown fox

unc/THE...FOX

[before]=> !-p-THE QUICK BROWN FOX  
[after]=> !-p-the quick brown fox

#### MESSAGES

FUNCTION NOT ALLOWED IN STRUCTURE

#### RELATED COMMANDS

CAPITALIZE, FLIP

US

- US Uniform Substitute

USUBSTITUTE allows the user to repeat a normal text substitution an arbitrary number of times.

```

USubstitute <scope> <text> <options>°
                        A|I
                        B|D
                        L
                        <n>|4095
                        S|I

```

<scope>.....The character string to be changed.

<text>.....The new text.

<options>...The <options> determine how USUBSTITUTE is to be executed and displayed. The <options> string is scanned from left to right; <option>s should be specified in the order they are listed above. If conflicting options (e.g., AI) appear in the string, the rightmost option will be used. The allowable options are:

A...(automatic substitution) The substitution proceeds without user intervention and ends when no further occurrence of <scope> remains.

B...(brief display) Display of each substitution is suppressed.

D...(regular display) The portion of the display buffer as specified by the last SDISPLAY is printed after each command.

I...(inquire after substitution) The next occurrence of <scope> will be replaced by <text>, and the message

"OPTIONS="

will be typed. The user should respond with:

- A - accept the substitution, or
- R - reject this substitution, but continue execution
- X - reject this substitution and exit from USUBSTITUTE

If <scope> is not found, the message:

"PATTERN NOT FOUND, OPTIONS="

will be typed. To this the user should either respond:

- A - continue with a long scan to the end of the file, or
- R - exit from USUBSTITUTE and return to Command Mode
- X - exit from USUBSTITUTE and return to Command Mode

If "A" is selected, the user may specify new <options> characters immediately following the "A" on the same line.  
Note: If not specified, <n> will be reset to 4095.

L...(long scan) The entire file will be scanned for <scope>. If not specified, only 2100 characters will be searched.

<n>.(do <n> substitutions) This must be the last option specified in <options> and must be less than 4095.

S...(Static Mode) The Editing Buffer does not move.

I...(Transcription Mode) Buffer moves to one word before the editing operation.

## NOTES

- USUBSTITUTE is a complex command which should be fully understood before global changes are attempted.

## RESTRICTIONS

- REVERT will not undo any part of USUBSTITUTE, so it should be used with care.
- USUBSTITUTE cannot be repeated by using &GIN.

WARNINGS

- USUBSTITUTE will not function properly if the first substitution is rejected in Inquire Mode.

EXAMPLES

- If the user is in Display and Transcription Modes, specifying:

us/FRESS/the editor/a5

will automatically substitute "the editor" for the first five occurrences of "FRESS" starting at the current buffer location, and will display each line where a substitution occurs.

us/apples/oranges

Up to 4095 substitutions of "apples" for "oranges" will be attempted, allowing the user to accept ("A"), or reject ("R") the individual substitution; or reject it and end the execution of USUBSTITUTE ("X").

MESSAGES

ALTER OPTIONS=

END OF AREA

END OF COUNT

FRESS

HYPERTEXT ENCOUNTERED: ACCEPT (A), REJECT(R), OR TEXT ONLY(T)

INVALID PATTERN, TRY AGAIN

INVISIBLE TEXT ENCOUNTERED: REJECT(R) OR ACCEPT(A)

OPTIONS=

PATTERN NOT FOUND, OPTIONS=

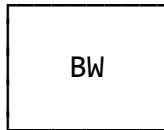
UNLIGHTPENNABLE TEXT

UNRECOGNIZED RESPONSE, PLEASE RESPECIFY



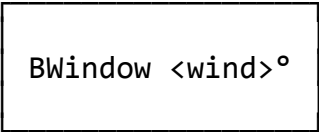
9.9 IMLAC VERSION COMMANDS

All commands in this section can only be used under "FRESS I"  
-- the special version of FRESS for the IMLAC PDS-1D  
mini-computer.



- BW    Blank Window

BWINDOW clears from the screen the contents of the specified window.

A rectangular box with a black border, containing the text "BWindow <wind>°" in a monospaced font, centered within the box.

BWindow <wind>°

#### NOTES

- If no window is specified, the current window is blanked.
- A non-linear travel function (e.g., DSPACE, GLABEL, etc.) must be used to return text to the window.
- BWINDOW is useful for blanking a window which would otherwise be regenerated (redisplayed) when an edit is done in another window.
- Usually only changed windows are regenerated, but those containing pictures and right-justified text are always regenerated.

#### RELATED COMMANDS

SWINDOW

COP

- COP Copy Picture

COPICT copies a picture from one file to another.

COPict < pict> <file1> <file2>

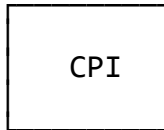
<pict>....the picture to be copied  
<file1>...the file the picture is to be copied from  
<file2>...the file the picture is to be copied to

#### NOTES

- <file1> remains unchanged.
- If <pict> already exists in <file2>, no copy is done.

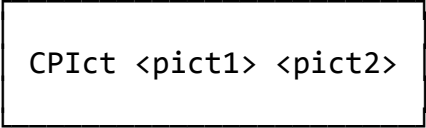
#### RELATED COMMANDS

CPICT, DPICT, LPICT



- CPI Change Picture Name

CPICT changes the name of the picture.

A rectangular box with a black border. Inside the box, the text 'CPIct <pict1> <pict2>' is centered in a monospaced font.

```
CPIct <pict1> <pict2>
```

<pict1>...the old name of the picture  
<pict2>...the new name of the picture

#### NOTES

- The name will also be changed in all references to the picture (tags made with MPREFERENCE)

#### RELATED COMMANDS

COPICT, DPICT, LPICT

CW

- CW    Change Current Window

CWINDOW makes the specified window the current window.

CWindow <wind>

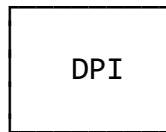
<wind>...the window which is to become current

#### NOTES

- All functions which default to the current window if no window is specified will subsequently use <wind>.

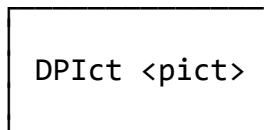
#### RELATED COMMANDS

BWINDOW, SWINDOW



- DPI Delete Picture

DPICT deletes the picture and all references (tags) to it.



<pict>...the picture to be deleted

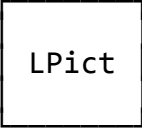
#### RELATED COMMANDS

COPICT, LPICT, PPICT

LP

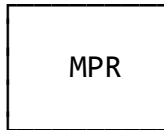
- LP List Pictures

LPICT lists the names of all the pictures in the current file.

LPict

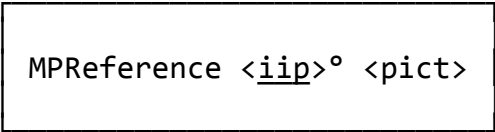
RELATED COMMANDS

CPICT, COPICT, DPICT



- MPR Make Picture Reference

MPREFERENCE creates a tag which refers to the picture specified.



MPReference <iip>° <pict>

<iip>...the place for the picture reference tag

#### NOTES

- The tag is displayed as:

%T'name'

- The picture itself can be displayed by JUMPing on the tag.

#### RELATED COMMANDS

SKETCH



PP

- PP    Print Picture

PPICT adds the picture specified to a file for Calcomp (offline plotter) output.

PPict < pict >

#### NOTES

- The filename is PICTOUT and the filetype FRESSP.
- To plot the picture(s), the following commands should be issued from CMS:

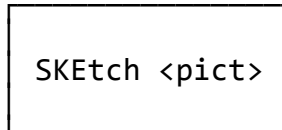
```
PLOTPICT <acct number>  
(wait for TAP1 ATTACHED msg)  
hit BREAK
```

After the plot is finished, the PICTOUT file is erased.



- SKE Sketch

SKETCH loads and starts the IMLAC drawing package; the screen will display the word 'LOAD' while this is done.



<pict>...the name of the picture.

#### NOTES

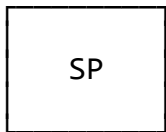
- If the picture is in the current FRESS file, it will be displayed and can be changed.
- If the picture does not exist, a new picture can be created and then added to the FRESS file.
- To save time when working with several pictures, window configuration 1B should be used (see SWINDOW). This allows the Sketch program to remain in the IMLAC, so it will not be reloaded with each picture.

#### RESTRICTIONS

- The picture name may not exceed eight characters.

#### RELATED COMMANDS

DPICT, LPICT, PPICT, SPICT, SWINDOW



- SP Scale Picture

SPICT sets the size at which a picture specified by < pict > will be displayed.

```
SPict <pict> <option>
              D|H|N|Q
```

#### OPTIONS

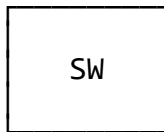
- D...double scale
- H...half scale
- N...normal scale
- Q...quarter scale

#### NOTES

- The size of the picture when it was originally drawn is considered to be the "normal" size.
- A picture changed using the SKETCH command will be reset to normal size.

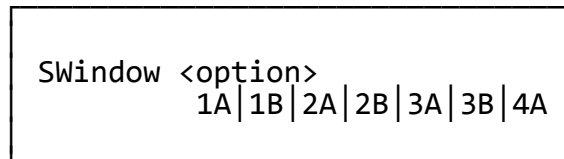
#### RELATED COMMANDS

CPICT, DPICT, LPICT, PPICT, SKETCH



- SW Set Window Configuration

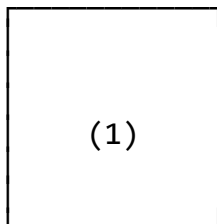
SWINDOW determines which of the seven default window configurations will be used for display.



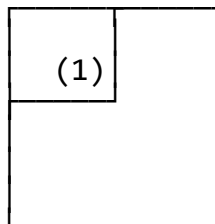
### OPTIONS

The following diagrams illustrate the window configuration described by each option:

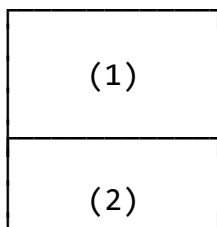
1A



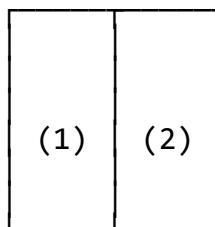
1B



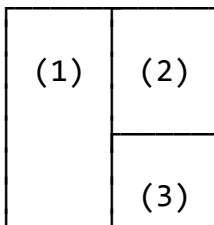
2A



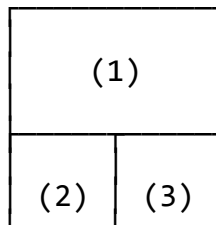
2B



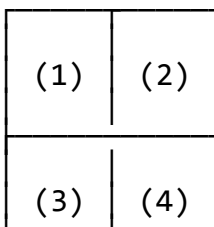
3A



3B



4A

RELATED COMMANDS

BWINDOW, CWINDOW, SWINDOW

## 10 FRESS HOUSE FUNCTIONS

### 10.1 OVERVIEW

"House" Functions are so named because they perform "housekeeping" Functions for the command language interpreter, the part of FRESS that parses lines from the terminal. These commands are executed by preceding the Function name with an ampersand("&").

### NOTES

- Minimum abbreviations for House Functions are indicated by capital letters.

### RESTRICTIONS

- House Function operands must follow the command name by one blank.
- House Functions may not be stacked on command lines using the FRESS Command Separator(">").

## 10.2 DESCRIPTIONS



&A

- &A    Enter As-is Mode

&ASIS causes all <text> parameters, and all lines in Input Mode, to be entered in As-is Mode.



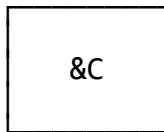
&A sis

## NOTES

- As-is Mode is useful for formatting tables (see the FRESS User's Guide for illustration).
- Each line is preceded by "!-S0-".
- Each occurrence of the physical tab key is replaced with "!-T-".
- As-is Mode is canceled when &NORMAL is issued.
- The "A" option of SINPUT, SIBOTTOM, and SITOP puts the user into As-is Mode only during that Swift Input session.

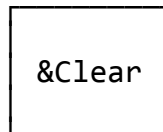
## RELATED COMMANDS

&NORMAL



- &C Clear Function Area

&CLEAR clears all pending functions, deferred <lp>s, and <scope>s.



#### NOTES

- &CLEAR should be used if the user has deferred a <lp> or <scope>, and decides not to complete the function.



&E

- &E    Execute

&EXEC executes a sequence of FRESS commands contained in the specified MEMO file.

&Exec <file>

#### NOTES

- The FRESS Command Macro facility (see Section 7) is a cleaner, more powerful extension of &EXEC and for most purposes, should be used instead.
- The sequence of commands is taken from the CMS file "<file> MEMO A".
- The format of the lines in the MEMO file is similar to the format of command lines typed individually into FRESS. Only one command or Command Macro should appear on each input line, unless the FRESS Command Separator (">") is placed between commands on the same line (see Section 5.4).
- A blank line is used to pass from Input Mode to Command Mode while using &EXEC.
- Any FRESS command, House Function (including &EXEC), or Command Macro may be included in a MEMO file processed by &EXEC.

#### WARNINGS

- Errors will be handled as usual, but no user intervention is allowed (to correct errors) until all commands in the file have been executed. Therefore, the user should exercise great caution when creating the MEMO file.

#### MESSAGES

ERROR READING (INPUT TERMINATED); FRESS READY

&G

- &G Repeat Last Command Line

&GIN repeats the entire last command line which was not "&GIN," the specified number of times.

&Gin <n>°  
1

<n>...the number of times to repeat

#### NOTES

- &GIN is useful for repeating an operation which could not have been carried out using USUBSTITUTE. It is especially handy for repeating multi-command lines.
- If an error is encountered during the iteration, execution of the command halts immediately.
- See Section 5.4 for a thorough discussion of multi-command lines and Section 2.3.1 on how to temporarily override the current Display or Editing Mode.

#### RESTRICTIONS

- Some commands cannot be repeated by &GIN (e.g., FULLPRINT, REVERT, USUBSTITUTE).

EXAMPLES

- While reading a small section of text, a user might want to PRINT 10 lines, read them, then SCROLL forward and PRINT the succeeding 10 lines. The sequence of commands would be:

```
P10
10>p10.*
&g
```

The first command line PRINTs 10 lines; the second command line SCROLLs 10 and then PRINTs 10 lines with only the last command on the line causing a display (see SMODE). After reading those ten lines, the user would then issue the third command line, which repeats the SCROLL and the PRINT.

MESSAGES

TRIED TO REPEAT A NON-REPEATABLE FUNCTION

&N

- &N    Leave As-is Mode

&NORMAL causes input values to be treated normally (as opposed to As-is Mode).

&Normal

#### RELATED COMMANDS

&ASIS

&P

- &P POP Implied Insert Pointer

&POP "pops" (removes) the specified number of Implied Insert Pointers from the top of the Stack.

&Pop <n>  
1

<n>...the number of pointers to pop from the Implied Insert Pointer Stack

#### NOTES

- The Implied Insert Stack is a stack of pointers used both in normal editing, and in the creation of Blocks. If the user does not specify an <lp> for a command which has an <iip> listed in its specification, the <iip> defaults to the pointer on top of the Implied Insert Stack (the Implied Insert Pointer).
  - Two entries are made in the Stack when IBLOCK, IDBLOCK, MBLOCK, or MDBLOCK is used: one points at the character before the Block-end, the other at the last character of the Block-end of the new Block. A new Block can thus easily be made nested in (inside of), or at the same level as (immediately after), the previous one.
- The Implied Insert Pointer (pointer on top of the Stack) points at the character before the Block-end of the Block just inserted or created.
- The pointer directly beneath those &POPped becomes the new Implied Insert Pointer.
- If there is only one pointer in the Stack, as with ordinary non-Blocked text editing, an error message is returned and the pointer stays in effect.

RESTRICTIONS

- The Implied Insert Stack currently holds only eight pointers (e.g., four successive MBLOCK commands with no intervening &POP's). When pointers are added to a full Stack, pointers from the oldest Block are deleted and the new pointers are added normally to the top.

EXAMPLESExample 1

This example uses non-Decimal Blocks to show the position of the Implied Insert Pointer(s) as pointers are created and popped. Each entry shows the position of the Implied Insert Pointer(s) (under the appropriate character(s) with numbers denoting their position in the Implied Insert Stack.

```
[given]=> "...text in old file..."

[user]      ibl/file/text inside the first Block
[system]    %<text inside the first Block%>
pointer(s)-->          1 2

[user]      mb/inside
[system]    %<inside%> the first Block%>
pointer(s)-->          1 2          3 4

[user]      i/ the second Block
[system]    %<inside the second Block%> the first Block%>
pointer(s)-->          1 2          3 4

[user]      &POP
[system]    %<inside the second Block%> the first Block%>
pointer(s)-->          1          2 3

[user]      i/popped text
[system]    %>popped text the first Block%>
pointer(s)-->          1          2 3

[user]      &POP
[system]    %>popped text the first Block%>
pointer(s)-->          1 2

[user]      &POP
[system]    %>popped text the first Block%>
pointer(s)-->          1
```

Example 2

This example explains the use of the Implied Insert Stack in manipulating Decimal Blocks. Consider the following:

ID/location

```
[before]=> location
[after]=> %< '1' Introduction%>
```

The Implied Insert Point is after the word "Introduction." An ordinary editing command making use of the Implied Insert Point, such as

I/word

would result in

```
%< '1' Introductionword%>
```

If, instead, the command

ID/word

were given, it would result in:

```
%< '1' Introduction
%< '1.1' word%>%>
```

A Decimal Block has been inserted at the Implied Insert Point. Note that the Block inserted is nested, that is, contained in, the previous Block (the "%>%>" indicates two consecutive Block-ends). If, instead, the command

ID/

were given, leaving both <iip> and <text> null, a Decimal Block would be created at the Implied Insert Point, as before, and the user would be placed into Input Mode inside that Block. In either case, the new Implied Insert Point is at the end of the text in Block 1.1. If a new Block is inserted using this Implied Insert Point it will be Block 1.1.1, nested inside the previous Block, 1.1.

The file would now look like:

```
%< '1' Introduction
%< '1.1' word
%< '1.1.1' text inside a Block%>%>%>
```

The Implied Insert Point is after the word "Block." Another Block inserted in the same manner as above would result in a Block numbered 1.1.1.1. If, however, the user does not wish this next Block to be nested inside the previous Block, but rather to be on the same level as the previous Block, &POP could be executed.

This "pops" the Implied Insert Stack to the next outermost level of decimal Blocks. Thus, in the example above, if the Implied Insert Point is after the word "Block" and &POP is given, the new Implied Insert Point is after the string:

Block%>

If, then, the command

ID/new one

were given, the result would be:

```
%< '1' Introduction
%< '1.1' word
%< '1.1.1' text inside a Block%>
%< '1.1.2' new one %>%>%>
```

The Implied Insert Point would then be after the word "one." If another Block on the same level as 1.1.1 and 1.1.2 were desired, the sequence of commands would be:

```
&POP
ID/this text will be on level 1.1.3.
```

Similarly, if the next Block to be inserted is to be on the same level as 1.1, two &POP commands must be given. The first will bring the Implied Insert Point to the same level as 1.1.3, the second to the level of 1.

## MESSAGES

TRIED TO POP TOO MANY TIMES

## RELATED COMMANDS

CFWORK, IANNOTATION, INSERT, IBLOCK, IDBLOCK, MDREF, MDRDEF, MFWORK, MLABEL, RTANNOTATION, SPLITAREA



&R

- &R Route

&ROUTE creates a CMS file which is a transcript of a portion of a FRESS session.

```
&Route <filename>° <filetype>° <option>°  
          FILE          FRESSOUT      C  
  
          or  
  
&Route OFF
```

<option>...(if "C") only commands typed by the user are put into the file.

(unspecified) user typed commands and FRESS responses are put into the file.

#### NOTES

- To specify any of the optional parameters, the user must also specify any preceding optional parameters.
- To <option> is specified as "C", the user must also specify a filename and filetype, as in: "&ROUTE FILE FRESSOUT C".
- To halt routing, the user should type "&ROUTE OFF".
- The file may be printed by issuing the CMS command "PRINT <filename> <filetype> (cc."

#### MESSAGES

ERROR WRITING TO FILE - ROUTING TERMINATED

&T

- &T    Type on, off

&TYPE controls all typing at the terminal during normal processing.

&Type <OFF|ON>

#### NOTES

- &TYPE is particularly useful with &EXEC. &TYPE OFF entered before &EXEC <filename> will process the commands in the MEMO file but will not type the effects of the commands (including errors) at the terminal. At the termination of the &EXEC function, the type is automatically turned back on (if it was off) in order to let the user reassume control.
- &TYPE may be turned on manually by entering &TYPE ON.

## 11 COMMAND MACROS

### 11.1 OVERVIEW AND USE

A Command Macro is a series of FRESS commands grouped together by the user to be invoked with a single command name. Command macros help to reduce the number of keystrokes needed to perform standard operations or to make a complicated operation easier.

Command Macros are created (see Section 7.2.1) and then grouped together in macro libraries (<maclib>s) (see Section 7.2.2). FRESS should be invoked as follows to make command macro libraries available for a given session (see Section 2.1):

```
FRESS [NOS] [<version>] [<maclib>...]
```

<maclib>...Up to nine <maclib>s may be specified -- duplicate macro definitions will be resolved in favor of that residing in the library specified earliest. The system macro library will be searched last.

#### Specification

Command Macros are invoked or "called" much as ordinary commands are. The differences between the two methods are:

- A Command Macro name must be preceded by a period (".") or a comma (",") to indicate it is a macro. If a comma is used, each expanded command line is printed at the terminal, preceded by a colon, before execution. This is useful when "debugging" a new macro (figuring out why it does not work). If a period is used the expanded lines are not typed.
- A Command Macro must be the first command on a line and must not be preceded by any blanks.
- If a Command Macro is followed by a command separator and an ordinary FRESS command, the entire expanded macro will be executed before the ordinary FRESS command is parsed and executed.

## Parameters

- Parameters may be passed to the macro in the same way parameters are specified in an ordinary FRESS command.
- Key delimiters are used to separate the parameters for the macro. These delimiters bear no direct relation to the delimiters used for each line within the macro definition and need not use the same character.
- A parameter is left null by specifying a key delimiter, but no value for it. For example, to omit the parameter in DELETE, type "d"; to leave it null, type "d/".
- Parameters in excess of those required by the macro are considered to be part of the last parameter.
- All parameters are required and may only be omitted if an explicit default is given for each occurrence of it within the macro definition.

## Errors

- The following messages indicate macro expansion errors (see Appendix E.3 for explanations):

```
INVALID DEFAULT LITERAL
MACRO EXPANSION TOO LONG
MACRO NAME TOO LONG
MACRO NOT FOUND
MISSING SYMBOLIC PARAMETER
&R<n> HAS NOT BEEN DEFINED
```

- If one of the errors above occurs during expansion or execution of a Command Macro, the relevant error message is typed, the expansion of the line causing the error is typed, and so is the message:

COMMAND MACRO: ACCEPT (A) OR REJECT (R) REMAINING FUNCTIONS

A...the execution of the macro will proceed, but the line containing the error will be skipped

R...(or anything else) expansion and execution of the macro immediately halts. Remaining functions include the rest of the Command Macro and any commands that were on the same line as the macro invocation (after a command separator).

## 11.2 CREATION

### 11.2.1 INDIVIDUAL MACROS

Note: The "\$"s in the titles for this section represent "&"s.

#### Format

A Command Macro is created as a CMS MEMO file with the same name as that desired for the macro. These files are created and changed using the CMS editor (see TM17 "CMS Editor Primer"). When complete, the MEMO file form is merged into a <maclib> by using either FMACADD or FMACREP (see Section 7.2.2).

- Each line of the MEMO file is considered to be a FRESS command line and may contain multiple commands separated by the FRESS logical command separator ">" (see Section 5.4), but may not contain nested macro calls.
- Trailing blanks are removed from lines during processing unless followed by the FRESS Command Separator.
- Each command in the macro consists of literal character strings, special functions (described below) and symbolic parameter indicators which represent values to be filled in when the user invokes the macro.
- Literal ampersands ("&" -- those not used as special function indicators, &C or &I, and those not used as symbolic parameters &<n>, &R<n>) must be doubled.

#### Continuation Column

It is possible for a logical command line of the MEMO file to occupy up to 3 physical lines by placing a "\*" in column 80.

- The next line will then be concatenated to the 79th column of the first line.
- Although the macro definition may be 3 lines long (including macro comments, prompts, etc.) the resulting expanded FRESS command line must be less than 130 characters in length.
- Symbolic parameters (&<n>, &R<n>), special function indicators (&C, &I) and literal ampersands (&&) all consist of two or more characters which must not be split over card boundaries.

### 11.2.1.1 Symbolic Parameters (Index Variables)

The symbolic parameters specified on a command line indicate that a read will be done from the terminal and the line read in can be used throughout the macro. One form is:

`&<n>[|<default>|]`

<n>.....a number from 1 to 9

<default>...a value for the parameter that is used if the parameter is omitted or left null. Defaults may contain any of the symbolic parameters, any of the special functions, or regular text as well as any combination of these. The only thing a default cannot contain is another default.

#### Notes

- If a parameter is omitted when no default is present an error message will be typed and the Command Macro will not be executed.
- Parameters may be left null even when no default is present. The null string will be substituted in its place.
- Each separate occurrence of a symbolic parameter indicator may have its own particular associated default string.

#### Example

```
i/&1/&3|&2 and &R'second half of inserted text:'|
```

In this macro, the default will be used if &3 is either omitted or left null. If the default is used, &2 must previously have been specified, since it can have no default.

### 11.2.1.2 &READ

This function can be used to prompt the user for required parameters which are not specified initially and for which no standard default can be used. An &R function inside a macro will cause the character string enclosed in apostrophes to be printed and a read to be done at the user's terminal (i.e., the terminal will wait for a response). The string typed by the user will replace the &R function in the macro expansion.

&R[<n>]['<prompt>']

or

&R<n>

<n>.....A number from 1 to 9. If <prompt> is specified, the line read in will substitute the &R function in the command line.

- Any &R<n> may be redefined at any time by another &R<n>'<prompt>'.
- If <prompt> is not specified, the last line read by &R<n>'<prompt>' will substitute &R<n> in the command line.

<prompt>...A line typed at the terminal before the read is done.

- If null, or if both <n> and <prompt> are not specified, no line will be typed, but a read will still be done.

### Notes

- If the &R<n>'<prompt>' occurs as part of a default the &R<n> is not defined unless the default is used. A null prompt may be specified, for example, by &R1''.
- The definitions do not have to proceed in numerical order, that is, &R2 may be defined and used even if &R1 is undefined.
- Both the <text string> and the input from the user are considered literal text; no parameter substitution is done.
- An &R<n> must be defined before it is used or an expansion error will occur and macro execution will terminate.
- &R<n> cannot itself take a default, but it can be included in a default for a symbolic parameter (&<n>).

Example

- ```
(1) l/&R1'What to change:'
(2) s/&R1/&R2'Change to:'
(3) l/&R2
```

This macro locates a pattern, changes it to a new pattern, and then makes the pattern the first text in the buffer.

- (1) The macro first prompts for the LOCATE pattern.
- (2) In SUBSTITUTE the same pattern is then used as the <scope> parameter and the macro prompts for the text string.
- (3) The text string is used again as a LOCATE pattern.

Example

- ```
(1) mc off>          &C'turn macro comments off'
(2) g/&1|&R1'filename?'|
(3) l/&1|&R1|>      &C'to find end of macros'
(4) ib/&1|&R1|/&R2'new macro?'
(5) s/&1|&R1|/&R1'new end of macro delimiter?'
(6) g/macfile>      &C'GET universal macro file'
(7) l/&R1>          &C'Find end of macro delimiter'
(8) ib/&R1/&R2>      &C'put new macro in universal'
(9) mc on>          &C'turn macro comments back on'
```

This macro enables the user to insert a new macro in both the file specified and in a special file that holds all his macros. The comments (&C) are just internal documentation so they are turned off at the top (1) and back on at the bottom (9) [mc off, mc on]. Note the use of the Command Separator (">") so comments can be put on the same line (see Section 7.2.1.3).

- (2) &R1 is defined to be the filename if the first symbolic parameter is omitted or left null. This same line will then get the file.
- (3) The filename is LOCATED. It was used as a delimiter to denote the end of the format code macro definitions in the file.
- (4) &R2 is defined to be a new format code macro and it is inserted before the filename. Note that if the user typed a null line in response to "new macro?" this would define &R2 as null and the command would put the user into Input Mode. The subsequent input would not be saved as &R2.
- (5) The saved filename is used in place of &R1 in its first instance and then &R1 is redefined to be the new end of macros delimiter. The old delimiter (the filename) is then replaced with the new delimiter.



- (6) A file (macfile) set up to store all format code macro definitions is retrieved. The next two lines locate the new end of the format macros delimiter, and the new macro defined in line four is inserted.

#### 11.2.1.3 &COMMENT

An easy way to optionally monitor progress through a macro or to document what a macro does, is by including this function on a command line:

&C'<comments>'

<comments>...the literal text to be typed at the terminal (i.e., no parameter substitution will occur). No apostrophes ( "' ") are allowed in <comments>.

#### Notes

- The MCOMMENT command controls when these "macro comments" will be typed.
- For clear internal documentation of a Command Macro it is often desirable to include a comment on the end of a command line. To make sure that intervening blanks between the end of the command and the &C'<comments>' are not included in the command line the FRESS command separator (">") should be used (see example in the section above).
- When the Display Mode is to be temporarily overridden (see Section 2.3) on lines in which there are <comments>, the "&C'<comments>'" must immediately follow the period or modifier (if present) with no intervening blanks.

Examples

i/&1/NOT      &C'negate it'  
the text string "NOT"      " is inserted after &1

i/&1/NOT>      &C'negate it'  
the text string "NOT" is inserted after &1

10>p 10    .\*&C'print next 10 lines'  
this is valid

10>p 10    .\*    &C'print next 10 lines'  
10>p 10    .\*> &C'print next 10 lines'  
these are invalid because the last characters on the command  
line are not " .\*"

11.2.1.4 &IDENTIFIER

A way of setting a global variable is to include the "&I" function in a command line:

&I

Notes

- The value of &I is set by the MIDENTIFIER command.
- If the string set by MIDENTIFIER is shorter than 4 characters it will be padded on the right with blanks.

Example:

Consider the macro:

&I/&1/&2///&3

This macro can be used to insert either a Decimal Block, or a regular Block, if &I is set to either ID, or IBL respectively. (The three parameters are for the <lp>, <label>, and <text>.

### 11.2.1.5 Additional Macro Examples

#### Example:

Consider the macro:

```
SA
T &1|15|
RET
```

This macro will TYPE a specified number of lines and then RETURN the display to the position before the TYPE was performed. If no parameter is specified in invoking this macro or if the parameter is left null the number of lines typed defaults to 15.

#### Example:

If, instead, the macro had been specified as:

```
SA
T &1
RET
```

then leaving out the parameter would result in an error message and none of the three commands in the macro would be executed. If the parameter were left null the commands would be executed, but an error would occur because TYPE has a required parameter. For example, consider a macro which retrieves a file, locates a pattern at the end of the format code macro definitions (to make sure they are all defined) and then travels to a particular Decimal Block. This might be written as:

```
g/&1|&r'Filename?'|
l/!.end of macros
gdl/&2|&r'Decimal Block#?'|
```

If &1 or &2 are not specified when the macro is invoked, the message "Filename?" or "Decimal Block#?" will be printed when that line in the macro is reached.

### 11.2.2 MACRO LIBRARIES

Command macros must be contained in a macro library (<maclib>).

#### Notes

- <maclib>s may contain any number of Command Macro definitions.
- The name of the system Command Macro library is "FMAC". If the user names a <maclib> "FMAC", the system <maclib> will not be invoked.
- All <maclib> functions must be executed in CMS and not the CMS Subset or FRESS and the <maclib> must reside on the A-disk.

#### 11.2.2.1 Editing Functions for <maclib> Manipulation

The following four <maclib> editing functions must be used to create and modify <maclib>s. Each can manipulate up to 10 macros (<cmac>) in or out of one <maclib>.

FMACADD <maclib> <cmac1> [<cmac2>...]

The specified MEMO files are added to <maclib>. If <maclib> did not previously exist it will be created. If <maclib> already contained a copy of any of the specified macros those particular "adds" will be ignored. All MEMO files which are successfully added to <maclib> will be erased.

FMACDEL <maclib> <cmac1> [<cmac2>...]

The specified macros are deleted from <maclib>. Unlike FMACSPLT, no MEMO file is created.

```
FMACREP <maclib> <cmac1> [<cmac2>...]
```

The specified macros in <maclib> will be replaced by their current MEMO file representation. The MEMO files will then be erased.

```
FMACSPLT <maclib> <cmac1> [<cmac2>...]
```

The specified macros are recreated as MEMO files but are not deleted from the macro library. The MEMO files then be edited.

#### 11.2.2.2 Printing Functions for <maclib>

```
FMACLIST <maclib>
```

The names of the macros contained in <maclib> will be printed at the user's terminal.

```
FMACPRNT <maclib>
```

The contents of the specified macro library is formatted into a file called "<maclib> MEMO A". Individual macros are separated by a line of asterisks and each macro is preceded by its name. This function is expensive so it should be used judiciously. Because of this, the user is given the option to retain the disk file. Note: MEMO files on the A-disk that have the same names as members of the macro library will be erased. The function requires no more disk space than twice that occupied by the <maclib>.

### 11.3 SYSTEM COMMAND MACROS

All users have access to the FRESS System Command Macros which are described in Section 7.3.1 and listed in Section 7.3.2.

#### 11.3.1 INDIVIDUAL DESCRIPTIONS

##### .DS <space> [<n>]

Description: goes to the top of the specified <space> and PRINTs <n> lines (<n> defaults to 10).

##### .END

Description: ends the FRESS session, queries the virtual printer, and issues the CMS command "log hold."

##### .FRSCOMM

Description: Will send a message and/or a file to the FRESS staff (see Section 1).

Prompts:

"FILENAME ( ENTER NULL LINE FOR NO FILE ) ?"  
FRESS file to be sent (a null line will send no file). If the specified file is not found, the user is given another opportunity to specify it correctly.

"ENTER NAME, PHONE NUMBER, AND PROBLEM OR COMMENT, UP TO 80 CHARACTERS PER LINE; TYPE A NULL LINE WHEN DONE."

Type the message, and when through type a null line.

##### .GDL <dec#>

Description: Goes to one line before the decimal block <dec#> and PRINTs 3 lines.

.GET

Description: INSERTs a FRESS file into the current FRESS file.

Prompts:

Where should the file be inserted?

Specify a <lp>.

"Enter name of file to be inserted?"

Enter the desired filename.

.GF <file> [<n>]

Description: <file> becomes the current file, the "FN" Viewspec is turned on (Decimal Block numbers will also be typed at the end of each Decimal Block), the string "!.eom." is searched for, and <n> lines (defaults to 10) are PRINTed.

.HELP

Description: see FRSCOMM.

.IDB <dec#> <label> <text>

Description: A new Decimal Block is INSERTed after Decimal Block numbered <dec#>.

.IDBT <lit> <label> <text>

Description: <lit> is LOCATED, and then a new Decimal Block is INSERTed after it.

.L <lit>

Description: <lit> is LOCATED, a SCROLL -1 is done, and then PRINT 3.

.MOB <moved-block> <move-to-block>

Description: The Decimal Block numbered <move-block#> is moved so it directly follows <move-to-block#>.

.MOBT <block#> <lp>

Description: The Decimal Block <block#> is moved so that it follows <lp>.

.P [<scroll#>] [print#>]

Description: <scroll#> line(s) are SCROLLed (defaults to 10) and then <print#> lines are PRINTed (defaults to 10). If only one parameter is specified, it is taken as <scroll#>. If <scroll#> is negative, it must be preceded by a non-blank (otherwise the "-" is taken as the delimiter).

.PRACTICE

Description: A copy of the system DEMOFILE is placed on the user's A-disk (the computer center consultant has the FRESS LECTURE handouts in which an editing session using this file is transcribed).

.PUT <file> <lp> <command> <lp>

Description: Writes a part of the current FRESS file into a new file

Prompts:

"Enter name of file to be created:"

Type the name of the file that is to contain part of the current file.

"Enter starting point:"

Type a unique character string that identifies the beginning of the section of text to be written to the new file.

"Enter travel command or null line:"

Type a travel command if one is necessary to find the endpoint for the copy. If traveling is not necessary, type a null line.

"Enter ending point:"

Type in the unique character string that identifies the end of the section of text that is to be written to the new file.

.REMOTEFU <file> [<options>]

Description: simplifies FULLPRINTing to the terminal.

Prompts:

"Filename?"

"Options (separated by commas)"

See the FULLPRINT description for a discussion of these options.



11.3.2 LISTING

The following listing is the result of using the FMACPRNT command on the system Command Macro library "FMAC".

\*\*\*\*\*

FMAC : FRESS COMMAND MACROS

\*\*\*\*\*

MACRO: DS

DS &1 .

P /&2|10|

\*\*\*\*\*

MACRO: END

cm/\$end 1

\*\*\*\*\*

MACRO: FRSCOMM

CM EXEC FRSCOMM

\*\*\*\*\*

MACRO: GDL

gdl/&1 .

-1 .

p 3

\*\*\*\*\*

MACRO: GET

&C'when asked for text or not type t'

co/?=?/&R'Where should the file be inserted?'

gf/&R'Enter name of file to be inserted:'

?/\* .

bo .

1 .

?/\*

\*\*\*\*\*

\*\*\*\*\*

MACRO: GF  
SV/\*+FN  
GF &1 .  
l /!eom. .  
P &2|10|

\*\*\*\*\*

MACRO: HELP

CM EXEC FRSCOMM

\*\*\*\*\*

MACRO: IDB

GDL &1 .  
J/% .  
IDB/%>>/&2| |/&3

\*\*\*\*\*

MACRO: IDBT

l /&1 .  
idb \$&1\$&2| |\$&3| |

\*\*\*\*\*

MACRO: L

l/&1 .  
-1 .  
P 3

\*\*\*\*\*

MACRO: MOB

GDL &1  
M-B/&1/? .  
GDL &2  
J/% .  
?/%>> .  
P 3

\*\*\*\*\*

\*\*\*\*\*

MACRO: MOBT

```
ri/c .
ri/a .
GDL &1
M-B/&1/? .
ri .
?/&2 .
P 3
```

\*\*\*\*\*

MACRO: P

```
&1|10|
P /&2|10|
```

\*\*\*\*\*

MACRO: PRACTICE

```
cf frsteach demofile .
```

\*\*\*\*\*

MACRO: PUT

```
o d.bl.memo &R'Enter name of file to be created:' .
-1 .
co/?=?/* .
ret .
?/&R'Enter starting point:' .
&R'ENTER TRAVEL COMMAND OR NULL LINE:'
?/&R'Enter ending point:' .
ret
```

\*\*\*\*\*

MACRO: REMOTEFU

```
gf$&R'ENTER FILENAME:' .
fu &R'ENTER OPTIONS (IF ANY):',2&C'POSITION PAPER; HIT CR' .
```

\*\*\*\*\*

## 12 JUSTIFICATION

### 12.1 Column/Page Justification

Whenever the end of a paragraph is reached, the FULLPRINT program determines whether or not the paragraph fits into the current column (page). If it does not fit, the following criteria are applied to determine where the paragraph lines will be positioned (in the order shown):

- 1) If there are less than <n> lines left in the column (page) (set by the !+WIDOW+ alter code), the entire paragraph is moved to the next column.
- 2) If the paragraph cannot be split so that <n> lines appear in the current and next column, the entire paragraph is moved to the next column.
- 3) If less than <n> lines will appear in the next column, the current column is padded with blank lines until <n> lines appear in the next column.

If the justification algorithm described above results in several blank lines being left at the bottom of a column, then FULLPRINT tries to distribute these lines within the column in order to preserve the depth of the column. This is done by making five passes through the column in order to insert a blank line in front of any heading code (!-H-): heading two (first pass), heading three (second pass), etc.

### 12.2 Column Balancing (for multiple columns only)

If while in Multi-Column Mode the user inserts either a new page code (!-N-) or single-column code (!+COLUMN1+) in a location that would cause the partially existing columns to be of unequal length, the FULLPRINT program will attempt to reformat the columns so they are of equal length.

### 12.3 Line Justification

There are four modes of line justification set by the !+JUST+ code (see Section 4.3):

- 0) fully justified (the default)--blanks are inserted into the line until the text is flush with both the left and right margin. A blank is inserted only where another blank already exists in the line. If the existing blank is underscored, the inserted blank will also be underscored.
- 1) bell--the text on each line is centered between the margins; no blanks are inserted.
- 2) flush left--the text appears flush left, i.e., with a ragged right margin; no blanks are inserted.
- 3) flush right--the text appears flush right, i.e., with a ragged left margin; no blanks are inserted.

### 12.4 Formatting Blanks

It is grammatically proper to leave two blanks after each end of sentence period. The FULLPRINT program deletes extra blanks (more than one between words, more than two after punctuation) from the printout during all line justification.

The "not" sign ("-") is the Special Blank or Blank Fill (non-justifiable blank) character. During FULLPRINT the "-" is translated to a blank but is regarded as a text character by the line justification algorithm.

Consider the following indentation/ hanging indentation situation:

- 1) This is text where the user wants the following lines to begin flush with the first character in the first line.

If the justification were L0, the chances are that an additional blank would be inserted between the ")" and the "T" in the first line, thus throwing the alignment off. By changing the intervening blank into a special blank, "1)-This" is treated as a single word (with the "-" translated to a single blank on output only), and hence no additional space will be inserted.

13 HYPERTEXT CHARACTERISTICS

FRESS uses the percent sign ("%") as a delimiter, to differentiate specially created Structure (generically referred to as Hypertext) from ordinary literal text online. The first character(s) following a "%" defining a piece of Structure indicates which type it is. The generic format of a complete Structure code is:

```
%<Structure-ID symbol>[<data>]
```

NOTES

- Figure B.1 summarizes all possible Structure as each appears online. The meaning and function of these Structure types are explained in the "Structure Data Fields" table. That table lists the various types of data fields, which pieces of Structure they may be associated with, and how they are displayed online. The explanations of individual commands should be referenced for information on how to create and edit each data field.

Label	%L
Block-start	%<
Block-end	%>
Jump	%J
Pmuj	%P
Splice	%SP
Ecilps	%EC
Tag	%T

Figure B.1 -- Structure-ID Symbol Representations

- If more than one data field appears on a particular piece of Structure, they appear in the order indicated in the "Structure Data Fields" table.
- The appearance of Structure is important so that the reader may recognize it in the text and because of the specialized rules which apply to editing Structure.

- Unlike format codes, none of this Structure can be entered as part of literal text input. Because of the fact that FRESS maintains complex control information, specialized commands (e.g., MJUMP, MSPLICE) must be used.

<u>Data Field Type</u>	<u>How Displayed</u>	<u>May be attached to:</u>
Display Keyword	within dollar signs	Annotation Tags, Blocks, Jumps/Pmujs, Imbed Tags, Splices/Ecिल्pses
Decimal Label	within single quotes	Decimal Label Tags, Decimal Blocks
Label	within parentheses	locations, Block starts
Viewspeccs	within parentheses	Jumps/Pmujs, Splices/Ecिल्pses
Keyword	within double quotes	Annotation Tags, Blocks, Jumps/Pmujs, Splices/Ecिल्pses
Explainer	followed by %%	Jumps/Pmujs, Splices/Ecिल्pses
Picture name	within single quotes	picture reference tag

Figure B.2 -- Structure Data Fields

### EXAMPLES

%< '1.3' \$all;dkey\$(label)"regkey1;regkey2"text in label%>  
 Decimal Block 1.3, with label "label", Keywords "regkey1" and "regkey2," and Display Keywords "all" and "dkey"

%J(print)"keyed"this is the explainer%%  
 A Jump with Viewspec "print", Keyword "keyed," and Explainer "this is the explainer"

14 KEYWORD STRINGS

## W A R N I N G

The features described in this section do not always work properly, particularly Weights and Values. They have a higher incidence of bugs than other FRESS facilities.

14.1 Keyword Specification

Keyword strings are made up of separate Keywords or Attribute-Value pairs, separated by semicolons.

- Values are like Keywords, but are used in conjunction with modifying Attributes.
- Attributes are separated from their Values by a colon. Only one Value is allowed with an Attribute and vice versa.
- Keywords and Values can also be weighted according to their importance to the section in which they are used. Weights range from 1-15 with 15 the most important. They must follow the Keyword or Value, separated by a comma (e.g., "composer:Berlin,3" or "Berlin,4"). Only one Weight is allowed per Keyword or Attribute-Value pair. A Weight of 0 or no Weight at all indicates that the Keyword is universal and will cause a match regardless of weight.



RESTRICTIONS

- Keyword strings can be no longer than 255 characters in total.
- Each Keyword, Attribute, or Value is limited to 16 characters.
- The following characters are not allowed in a Keyword:

<u>SYMBOL</u>	<u>NAME</u>	<u>REASON</u>
;	semi-colon	Keyword separator
:	colon	Attribute-Value delimiter
,	comma	Weight delimiter
&, ,-,(,)	boolean operators	must be distinguishable for a retrieval request
␣	blank	internal restriction

Figure C.1 -- Illegal Keyword CharactersEXAMPLES

- In a file consisting of popular song titles a user might wish to distinguish between those for which an author wrote the music and those for which he wrote the lyrics. The user could assign the Attribute-Value pairs:

composer:Irving\_Berlin  
lyricist:Irving\_Berlin

"design;graphics:interactive;text-processing,6"

this is a Keyword string containing the Keywords "design" and "text-processing" (the latter with a Weight of 6); it also contains the Attribute-Value pair "graphics:interactive."

## 14.2 Boolean Request Format

The rules for retrieving Blocks (via BTCONTINUOUS, BTDISCRETE) and for matching Keywords (SKANNOTATION, SKDISPLAY, SKJUMP) are the same. They are as follows: With either type of request, a Keyword in the request will match either a Keyword or the value part of an Attribute-value pair in the file; e.g., a request for the Keyword "Irving\_Berlin" would be satisfied by either Attribute-value pair shown in Appendix C.1. An attribute-value pair will match only another attribute-value pair. In this way the user can select only that music for which Berlin wrote the lyrics with no regard to the composer. If the "keys" (Keywords or attribute-value pairs) match, then Weights are compared (see Appendix C.1 for a description of Weights). If the request or the Keyword in the file is unweighted the match is successful. If both are weighted then the Keyword in the file must have a Weight equal to or higher than the Keyword in the request in order to satisfy the request. Boolean combinations of requested Keywords are satisfied using standard interpretations of the logical operators "-" (not), "&" (and), and "|" (or), in order of descending precedence. The precedence may be changed by the use of parentheses, and is recommended for clarity (see the examples below).

### EXAMPLES

bt/Keyword1&key2|key3

retrieves all Blocks with both Keyword1 and key2, as well as all Blocks with key3

bt/Keyword1&(key2|key3)

retrieves all Blocks with both Keyword1 and key2, and all Blocks with both Keyword1 and key3

bt/-key1

is invalid since it requests all Blocks which do not have key1 and it is not permitted to enumerate all Keyworded Blocks (possibly hundreds).

skd/all& imlac& double

"turns off" all Blocks which do not have the display keywords "all." Furthermore, any Block which has either of the display Keywords "imlac" or "double" is also "turned off".

### 14.3 Keyword Space

The Keyword Space is a system-defined Space in which the user's Keywords are entered alphabetically by Keyword and Attribute, and alphabetically by Value for each Attribute. Values are listed both with the Attribute, and separately. The number of pieces of each type of Structure on which each Keyword or value is entered on the same line.

- The Keyword Space is useful for making global changes on a particular Keyword. If a Keyword is edited in this Space, the change will be reflected in all places in the file where that Keyword is used. However, the Keyword Space can not be used as a point from which JUMP would allow non-linear travel to the Structure which contains each Keyword.

### EXAMPLES

A sample Keyword Space might appear like this:

```

design          3 %<
graphics
  design       3 %<
  hardware     1 %<
  interactive  2 %<
  passive      1 %<
  software     1 %<
graphics       1 %<
hardware       1 %<
interactive     2 %<
passive        1 %<
software       3 %<
sysproglang    1 %<  2 %J  2 %P
text-processing 2 %<                2 %T

```

- The Attribute "graphics," has 5 values in this file; the Keyword "sysproglang" appears on 2 Jump/Pmuj pairs as well as on a Block.

## 15 COMMAND LISTS

Commands are listed alphabetically by abbreviation in this Section.

### 15.1 Ordered by Type

#### 15.1.1 Display Commands

Block Trail Continuous	BT
Block Trail Discrete	BTD
Display Space	DS
Change Window	CW
Display Viewspecs	DV
List Pictures	LP
Macro Comment Control	MC
Next	N
Print	P
Scroll	SC
Set Display	SD
Set Keyword Annotation String	SK
Set Keyword Display	SKD
Set Keyword Jump	SKJ
Set Mode	SM
Set Viewspecs	SV
Set Window Configuration	SW
Type	T
Scroll	<integer>

#### 15.1.2 Editing Commands

Accept	A
Bars	BA
Bottom Input	BI
Change	C
Capitalize	CA
Copy From Work	CFW
Copy	CO
Copy Picture	COP
Change Picture Name	CPI
Copy To Label	CT
Copy To Work	CTW
Delete	D
Delete Picture Name	DPI
Flip	FL

Footnote	FO
Insert	I
Insert Annotation	IA
Insert Before	IB
Insert Block	IBL
Insert Decimal Block	ID
Imbed	IM
Move	M
Make Annotation	MA
Make Block	MB
Make Decimal Block	MD
Make Decimal Reference	MDR
Make Decimal Reference Deferred	MDRD
Move From Work	MFW
Make Jump	MJ
Make Label	ML
Make Picture Reference	MPR
Make Splice	MS
Move To Label	MT
Move To Work	MTW
New Area	NA
Revert	REV
Refer To Annotation	RTA
Substitute	S
Swift Input	SI
Swift Input Bottom	SIB
Swift Input Top	SIT
Sketch	SKE
Scale Picture	SPI
Splitarea	SPL
Surround	SUR
Top Input	TI
Underscore	U
Uncapitalize	UNC
Uniform Substitute	US

### 15.1.3 Travel Commands

Bottom	B
Bottom Area	BA
Display Space	DS
Get Decimal Label	GD
Get File	GF
Get Label	GL
Jump	J
Locate	L
Next	N
Return	R
Ring	RI
Save	SA
Scroll	SC
Type	T

Trail  
Scroll

TR  
<integer>

15.1.4 System Commands

Add Password	AP
Copy File	CF
Change Password	CP
Delete Password	DP
Execute CMS Command	CM
End FRESS Session	E
Free File	F
Find Error	FE
FRESS Message Control	FMSG
Fullprint	FU
Get File	G
Make File	MF
Set Macro Identifier	MI
Offline Read	O
Offline Type	OT
Pack File	PF
Query	Q
Scratch File	SF

15.2 Implied Insert Commands

Each one of the following commands sets the Implied Insert Pointer when issued. Implied Insert Pointers are explained in Section 1.3 under Implied Insert Point.

Copy From Work	CFW
Insert	I
Insert Annotation	IA
Insert Block	IBL
Insert Decimal Block	ID
Make Decimal Reference	MDR
Make Decimal Reference Deferred	MDRD
Move From Work	MFW
Make Label	ML
Make Picture Reference	MPR
Refer To Annotation	RTA
Splitarea	SPL

16 PROBLEMS AND MESSAGES16.1 Types of Problems

As with all large-scale software systems, the FRESS user may at some point encounter a "bug" in the FRESS program. This problem may manifest itself in several ways:

- A command may not work as expected. Before seeking aid, check the appropriate FRESS manual and be sure the nature of the command used is fully understood. If there are still questions, contact a member of the FRESS staff (see Section 1.2).
- If any messages are received and the user does not feel responsible for the indicated error, it may also indicate a problem in the file. The command should first be repeated to make certain it was not mistyped or that a transmission error between the terminal and the computer was not the cause. The user should then make a note of the last few commands executed (saving the listing from the terminal session if possible) and contact a FRESS staff member.
- If the explanation of a message below says "contact FRESS staff" there is a possibility that one or more active files have something wrong with them. The user should always do a REVERT in case the previous edit was the cause of the problem and make a note of the last commands executed. A member of the FRESS staff should then be contacted to determine the nature of the problem.



## 16.2 System Errors

If the message "SYSTEM ERROR ..." is typed, FRESS has detected an internal error. A REVERT is usually performed automatically in this case, but the user should do a REVERT as a matter of course in the event of a system error. If a message of one of the following forms is encountered, the user should note all <number>s or <name>s, and then send a ".frscomm":

DMS<number> <message>

DMS<number> <message> AT <number> IN ROUTINE FRESS

SYSTEM ERROR <number> - PLEASE CONTACT FRESS STAFF

SYSTEM ERROR <number> - SAVE TERMINAL SESSION

Three types of problems that occur more frequently than others, cause the following messages:

DMSBWR109S VIRTUAL STORAGE EXCEEDED

DMSITP141T OPERATION EXCEPTION AT <n> IN ROUTINE FRESS

These are actually CMS messages. The user must take care to maintain enough free space on disk for operations which will extend the size of a file or create a new file. The message is followed by a return to CMS, but all work before the next-to-last input or edit has been saved. If the operation which caused this message was creating a new file (e.g., CFILE, MFILE, PFILE), a partial work file may be left on the user's segment. This file may either have the name specified in the command or have the name "SYSUT6" or "SYSUT7." These files should be erased using the "CMS ERASE <file> FRESS" command.

Note: These two messages will sometimes occur if several read/write disks are logged in. Try accessing only the disk that the file resides on.

DMSABN148T SYSTEM ABEND 80A CALLED FROM 101F0C

The user's virtual machine must be at least 512k to invoke the FRESS system. Issue the CMS "DEFINE STORAGE 512k" command, and then after being put in CP, type "IPL CMS" (see Section 2).

### 16.3 Messages

A list of command abbreviations precedes those messages associated with just a few commands. Messages resulting from command macro functions either begin "FMAC:...", or are noted with the "(cmac)" abbreviation. Unless otherwise stated, when one of the following messages is received the last command typed has been ignored.

(no response)

(SL) The user is now in a wait state able to receive messages. Hit the break key to return to FRESS. If this does nothing, the system has crashed.

A-DISK IS READ/ONLY OR NOT LOGGED IN

(MF) New files are always created on the A-disk, which must be read/write.

ACCEPT (A) OR REJECT (R) REMAINING FUNS.

One section of the command line separated with the FRESS Command Separator (">") could not have its <scope> or <lp> resolved within the current Editing Buffer (see Section 5.4). No editing commands specified were executed.

ALTER OPTIONS=

(US) A USUBSTITUTE in Inquire Mode has been performed. The user must now specify one of the options: ("A") accept the substitution, ("R") reject it, but continue with the command, or ("X") reject it, and stop the command.

&, |, AND ~ NOT ALLOWED IN KEYWORDS

The <key> or <dkey> parameter of an editing command either contained an invalid Keyword or Keyword String, or the edit would have changed a Keyword or Keyword String making it invalid.

ANNOTATION BLOCK TO REFERENCE NOT FOUND

(RTA) A <key> but no <lp> was specified in the command. No Annotation Block with the specified <key> was found.

AREA ORDERS NOT ALLOWED IN BLOCKS

(SPL, MB, MD) A SPLITAREA was attempted inside a Block, or a MBLOCK or MDBLOCK was attempted with an Area line included in its <scope>.

BAD DISP TO OPTION

(Q) Contact the FRESS staff.

## BAD OPTION PASSED TO QUERY

(Q) QUERY was specified with an <option> other than "L" or "F".

## BOTH LP'S MUST BE IN SAME DATA FIELD

(C, D, S) A string that starts in one data field and ends outside of that field cannot be deleted (e.g., a string that is part of a label, but goes beyond the label structural boundaries).

## BOTH LP'S MUST BE IN SAME FILE

The pair of <lp>'s comprising a <scope> must be in the same file.

## BOTH LP'S MUST BE IN THE SAME WINDOW

The pair of <lp>'s comprising a <scope> must be in the same window.

## BOUNDARY ERROR

FRESS has detected conflicting demands on the dimensions of the current column or page. The codes that may be related to this problem are !+COLUMN+, !+DEPTH+, !+GRID+, !+GUTTER+, !+MARGIN+, !+OFFSET+, and !+WIDTH+ alter codes. For example, a !+GRID+ code may specify a drawing wider than the current !+WIDTH+. Attempting FULLPRINT may result in abnormal termination (or at least some error). If this message occurs after issuing FULLPRINT, the command was executed.

## BTL BUTCHERY

Contact the FRESS staff.

## BUFFER BUTCHERY

Some problem is preventing the file from being displayed at the current display point. Usually the file is all right, but contact the FRESS staff anyway.

## BUFFER SIZE EXCEEDED

(SD) The line length <len2> times the number of lines <n> is greater than the display buffer size. Use smaller limits.

## CANNOT DELETE START OR LAST END LINE OF SPACE

(C, D, S) The <scope> specified, included the \*START OF TEXT AREA\* line or the \*END OF TEXT AREA\* line in the file.

## CANNOT HIT IN A DATA FIELD

Contact the FRESS staff.

## CLOSET BUTCHERY

Contact the FRESS staff.

## CMS ERROR CODE &lt;nnnnn&gt;

(CM) An error was returned from the specified CMS command. The error code is printed in decimal.

## COMMAND MACRO: ACCEPT(A) OR REJECT(R) REMAINING FUNCTIONS

(cmac) An error was made either in the expansion or operation of the command macro being executed. If "A" is typed, the execution will proceed, skipping the line containing the error. If anything else is typed, expansion and execution of the command macro will halt immediately.

## COMMAND NOT SUPPORTED

The specified command has been designed but is not ready for use as yet.

## CONTEXT PATTERN NOT FOUND BY SCANNER

After a request for a context scan as part of an editing command, the <scope> or <lp> parameter of the editing command specified a pattern which could not be found in the 2100 characters following the start of the display. Frequent causes of this error are mistyping, difficulty in indicating capitalized strings as <lp>'s or <scope>'s on upper-case terminals (see Section 2.2.1), or having already SCROLLED or pattern scanned past the pattern being searched for now.

## COPY SUCCESSFUL

(CF) The command was executed successfully; there are now two identical files, <file1> and <file2>.

CURRENT: <file1>  
<file>  
<file>

(Q) This is the format of the reply to a "QUERY F"; <file1> is the current file; other <file>s may also be open.

## CURRENTLY UNSUPPORTED

The specified command has been designed but is not ready for use as yet.

## DECIMAL LABEL DOES NOT EXIST

(GD) There is no Decimal Block numbered <n> in the current file.

## DECIMAL LABEL SPECIFIED DOES NOT EXIST

(MDR) There is no Decimal Block numbered <n> in the current file.

## DELETE INCLUDES TOO MANY BLOCKS

(C, D, S) An internal restriction; try deleting fewer Blocks.

## DELETE SCOPE MUST BE WITHIN KEYWORD

(D) Only a single Keyword can be included in <scope> while deleting text in the Keyword Space.

## DELIMITER ERROR

A <delimiter> is missing at the beginning or end of a format code (see Section 4.1). The text that formed a partial format code following the exclamation point ("!") in question, is taken as literal text.

## DISPLAY KEYWORDED BLOCKS NESTED TOO DEEPLY

Remove the Display Keywords from the innermost level Blocks.

## EDIT COMMAND IGNORED; FILE IS READ/ONLY

Either the file is on a read/only disk, or it is passworded against editing. Either change the password (CPASSWORD) or access the disk as read/write.

## EDIT LIMITED TO ONE LABEL OR KEYWORD

When editing in the Label or Keyword Spaces, only one Label or Keyword may be included in the <scope> of a single edit.

## END OF AREA

(US) The pattern scan has encountered an "\*\*\*\*AREA\*\*\*\*" line; no more substitutions can be done.

## END OF COUNT

(US) The <n> option has been specified, and all <n> substitutions have been completed successfully.

## \*END OF TEXT AREA\*

## INPUT

(BI) The user is now in Input Mode. (SIB) The user is now in Swift Input Mode.

## ENDING COLUMN &gt; 80

(O) The E<nn> parameter must be less than 81. OREAD was not executed.

## ERROR IN COMPOUND SCOPE

An error has been made in the format for a deferred <scope>. For example, this message would be received if two question marks were used instead of one (e.g., "m/string=??/x") or if neither of the two <lp>'s were actually deferred (e.g., "m/start=end/new loc").

## ERROR IN STRING NEAR POSITION &lt;nnn&gt;

(SV) An invalid Viewspec has been included in <vs>. The error occurs near the character with displacement <nnn> from the beginning of the string.

## ERROR READING (INPUT TERMINATED); FRESS READY

(&E) There was an error in the line of the MEMO file being read with &EXEC. The user is now in Command Mode.

## ERROR WRITING TO FILE - ROUTING TERMINATED

(&R) FRESS has encountered some problem in writing the transcript file. The user is now in Command Mode with &ROUTE OFF, and &TYPE ON in effect.

## EXPLAINERS ARE LIMITED TO 255 CHARACTERS

(C, I, S) The intermediate or end result of the command would have been longer than the maximum length of Explainers.

## FERROR: ERROR NOT FOUND, END OF AREA

(FEL) No formatting error was found between the current position in the file, and the bottom of the Area.

## FERROR: ERROR NOT FOUND, END OF COUNT

(FE) No formatting error was found within 8000 characters of the current position of the file.

## FILE ERROR ON IMBED

(FU) FULLPRINT was unable to retrieve the IMBEDded file. Check whether the file specified by the Imbed Tag exists, then contact the FRESS staff.

FILE: <file>

SPACE: <space>

AREA: <n>

(Q) This provides the requested information for "QUERY L." <file> is the current file; <space> is an abbreviation for the current system Space; <n> is the Area number (counting from the top). The abbreviations for <space> are: "ANNOTATI" (Annotation), "KEYWORD" (Keyword), "LABEL" (Label), "MAIN" (Text), "STRUCTUR" (Structure), and "WORK" (Work).

## FILE ALREADY EXISTS

(CF, MF, O) A FRESS file with the given filename already exists on a read/write disk. In the case of CFILE, it is a file by the name of <file2> which already exists.

## FILE MUST BE CURRENT TO BE SCRATCHED

(SF) The user should retrieve the file with the GFILE command and then attempt the SFILE again.

## FILE NOT FOUND

(G, O) The specified file does not exist. In the case of OREAD, it is the CMS file specified in the Disk <option> which does not exist.

## FILE NOT FOUND OR NOT OPEN

(GD) The <file> specified must be open.

## FILE NOT IN USE

(F) The file specified was not an open file.

## FILE TO COPY NOT FOUND

(CF) <file1> does not exist.

## FILE WITH PMUJ NOT FOUND -- EDIT CANNOT BE DONE

Keyword(s) on an interfile Jump may not be edited if the other end of the Jump cannot be found (see SKDISPLAY).

## FILENAME?

(FU) <file> was not specified, or specified incorrectly -- try again (do not use a comma as the Key delimiter).

FINI

(FU) FULLPRINT was completed successfully.

FINIS

(OT) OTYPE was completed and has been sent to the virtual printer.

FMAC: <file> COPY NOT FOUND  
Respecify.

FMAC: ERROR IN LINE &lt;n&gt;

Only the characters "&amp;", "C", "I", "R" or a digit from "1" to "9" are valid when preceded by an ampersand. This message is also caused if there is a bad character in the MEMO file or if an error occurs while reading the MEMO file.

FMAC: ERROR WHILE READING <file> MEMO  
Contact the FRESS staff.FMAC: ERROR WHILE WRITING TO <file> COPY  
Try again.FMAC: FATAL MACLIB ADD ERROR <n>  
If respecifying does not work, contact the FRESS staff.FMAC: FATAL MACLIB COMP ERROR <n>  
If respecifying does not work, contact the FRESS staff.FMAC: FATAL MACLIB DEL ERROR <n>  
If respecifying does not work, contact the FRESS staff.FMAC: FATAL MACLIB ERROR <n>  
If respecifying does not work, contact the FRESS staff.FMAC: FATAL MACLIB GEN ERROR  
<maclib> did not exist and is being created.FMAC: FATAL MACLIB LIST ERROR <n>  
If respecifying does not work, contact the FRESS staff.FMAC: FMACCVTC ERROR <n> ON <macname> MEMO  
If respecifying does not work, contact the FRESS staff.FMAC: FMACCVTM ERROR <n> ON <macname> COPY  
If respecifying does not work, contact the FRESS staff.FMAC: INVALID CHAR AFTER & IN <file> MEMO  
Only the characters "&", "C", "I", "R" or a digit from "1" to "9" are valid if preceded by an ampersand. This message is also caused if there is a bad character in the MEMO file or if an error occurs while reading the MEMO file.

FMAC: LMEMBERS ERROR <n>

If respecifying does not work, contact the FRESS staff.

FMAC: MACLIB NOT FOUND ON A-DISK

The specified macro library could not be updated because it is not on a read/write A-disk.

FMAC: <file> MEMO CONTAINS CHARS WHICH ARE NOT ALLOWED

Only the characters "&", "C", "I", "R" or a digit from "1" to "9" are valid if preceded by an ampersand. This message is also caused if there is a bad character in the MEMO file or if an error occurs while reading the MEMO file.

FMAC: <file> MEMO NOT FIXED LENGTH 80

The logical record length of the MEMO file must be 80. Specify "LRECL 80" while in the Editor.

FMAC: <macname> MEMO NOT FOUND

The specified macro does not exist as a MEMO file. Execution of the FMACADD continues for the other macros.

FMAC: MORE THAN 10 MACROS TYPED; FIRST 10 USED

The limit of 10 <macname>s has been exceeded. All but the first 10 have been ignored.

FMAC: NO MACLIB TYPED

No <libname> was specified.

FMAC: NO MACROS TYPED

No <macname>s were specified.

FMAC: <macname> NOT FOUND

The specified macro was not found in the specified macro library.

FMAC: ONE OR MORE MACROS NOT IN MACLIB; NOTHING DONE

One of the specified <macname>s is not in the specified library. The command has been ignored.

FMAC: ONE OR MORE MEMO FILES NOT FOUND

Nothing was done.

FOOTNOTE(S) TOO LONG

(FU) The file specified a footnote longer than a full page, which cannot be properly formatted. FULLPRINT terminated at the point where the error occurred. Note: this message only appears during FULLPRINT, not while the file is being edited.

FRESS

The user is [back] in Command Mode.



## FRESS READY

The user has successfully invoked FRESS [current state is Command Mode].

## FUN. ALREADY REPEATED, OR NO FUN TO BE REPEATED

Respecify the command in full.

## FUNCTION NOT ALLOWED

The command is password-protected.

## FUNCTION NOT ALLOWED AT END LINE

An edit has been attempted on or after the \*END OF TEXT AREA\* line.

## FUNCTION NOT ALLOWED IN SPECIFIED SPACE

There are many commands which may be executed only in a particular Space or Spaces. For example, FLIP and SURROUND may not be used in the Structure Space.

## FUNCTION NOT ALLOWED IN STRUCTURE

(CAP, FL, FO, SUR, U, UNC) These commands may not be used in a piece of Structure (e.g., a Label).

## HELP NOT LOGGED ON

(H) The HELP consultant is not currently logged-on. Try this command again later.

## FUNCTION TAKES NO PARAMETERS

A parameter was specified in a command which takes no parameters (e.g., DVIEW).

## HYPERTEXT ENCOUNTERED: ACCEPT (A), REJECT (R), OR TEXT ONLY (T)

("A") allows the edit of both text and Structure; ("R") rejects the command; ("T") leaves all the Structure orders and edits just the text around them. This message will be suppressed and an implicit accept ("A") will be done if the command is preceded by a percent sign ("%").

## HYPERTEXT INTERFILE MOVE NOT SUPPORTED

Labels and other Structure may not be moved or copied interfile. Text may be copied between files, however, if no command modifier is used.

## \*\*IMBED ERROR\*\*

(FU) When FRESS can not imbed the file specified by the Imbed Tag, this comment is printed inline where the Imbed would have taken place. This line is marked by four asterisks ("\*\*\*\*") to the right of the formatted text.

## IMBED FILENAME TOO LONG

(FU) A filename is limited to eight characters.

## IMBED NESTED TOO DEEPLY

(FU) The nesting limit is ten levels; try combining files.

## IMBEDDED FILE NOT FOUND

(FU) The file specified in an Imbed Tag was not found.

## IMPLIED LP DOES NOT EXIST; LP IS DEFERRED: WAITING

The <lp> parameter of the command has been left null and an Implied Insert Point does not exist (either no edit has been done or all the <iip>s have been "popped" using &POP). If the user does not wish the command to be executed, he must type "&C" to clear the waiting function. If Input Mode has been entered at the same time, the user must return to Command Mode before typing this command.

## IMPROPER LP

This message occurs if the user attempts to indicate as an <lp> the special characters which surround data fields (e.g., "(" and ")" for Labels) or other fields which should not be edited.

## INCOMPLETE RETRIEVAL, NOT ENOUGH ROOM

Contact FRESS staff.

## INCORRECT VALUE IN PARM POSITION

A <scope> has been specified with one or both endpoints deferred when an <lp> parameter was expected.

## INPUT

The user is now in Input Mode or Swift Input Mode. All text typed at the terminal will be interpreted as literal input until a null line is typed.

## INTERNAL FORM OF REQUEST TOO LONG

Contact FRESS staff.

## INTERNAL LIMIT REACHED -- EDIT CANNOT BE DONE

(C, D, S) Specify again with a smaller <scope>.

## INTERNAL LIMIT REACHED - TRY SMALLER DELETE

Specify again with a smaller <scope>.

## INV OPTION

(FU) One of the <options> was invalid. FULLPRINT was not executed.

## INVALID ATTRIBUTE SPECIFIED (MISPLACED :)

(BT) A Block Trail or Keyword function was specified incorrectly.

## INVALID CHARACTER IN COMMAND LINE - CONVERTED TO BLANK

## INVALID CHARACTER IN COMMAND LINE - TRANSLATED TO BLANK

Possibly because of a transmission error, the terminal has transmitted an invalid character somewhere in the command line (a control key might have inadvertently been hit, these

include the characters: "{", "}", "[", "]"). Following the command a line will be typed containing a single or-bar ("|") under the invalid character. Unless this is accompanied by another error message, the command has been executed after having changed the invalid character to a blank.

**INVALID SUBSET COMMAND**

(CM) The CMS command may not be issued from the CMS subset environment. The user should RETURN to FRESS, then END the FRESS session, thus returning to the complete CMS environment where the command may be issued successfully.

**INVALID CODE**

The <code> part of a format code specification is incorrect (see Section 4.1).

**INVALID DEFAULT LITERAL**

(cmac) The ending or-bar ("|") was not found for a default literal within a given line of the macro.

**INVALID FILENAME**

(CF) The name of either <file1> or <file2> contains an invalid character (possibly caused by a transmission error).

**INVALID FUNCTION**

(AP) The <options> list contains an invalid command.

**INVALID FUNCTION SEPARATOR**

Only ">" may be used as a function separator - see Section 5.4. This message may occur if a house function is followed by another command with no function separator between them.

**INVALID JUMP**

(FU) FULLPRINT stopped at the bad Jump. Check to see whether it was specified correctly, then contact FRESS staff.

**INVALID MODE CHARACTERS WERE IGNORED**

(SM) Conflicting options, or characters other than B, D, S, or T were specified in the <option> string. The invalid Modes were ignored.

**INVALID LIBRARY TYPE SPECIFIED**

(routine lmembers) Contact the FRESS staff.

**INVALID NUMBER**

(GD, MDR) The value given for the Decimal Block number (<n>) is not a valid number.

**INVALID OR ILLEGAL QUALIFIER**

An undefined qualifier was used (see Section 5.5).

## INVALID PARAMETER

A Block Trail or Keyword function was specified incorrectly.

## INVALID PARM, OFFSET=

A format code contains an invalid <data> portion (see Section 4.1). The offset value is equal to the difference from the last "!" or ";", to the beginning of the invalid parameter of the line returned when FELONG is issued.

## INVALID PASSWORD

(CP, FU, G) For CPASSWORD, this message indicates that the <pass> field specified a password not associated with the file. In the case of GFILE, it means the <pass> specified is not one of the valid passwords for the given <file>. If no <pass> was specified, the specified <file> is password-protected and the system-default password is considered invalid. If the user does not think he placed a password on the file, it may indicate that something is wrong with the file itself. In this case, contact a FRESS staff member.

## INVALID PASSWORD ON IMBED

(FU) The password specified does not allow FULLPRINTing of the indicated file.

## INVALID PATTERN, TRY AGAIN

(L, US) This message occurs if the <scope> pattern started with an ampersand ("&"), which is not allowed.

## INVALID PLACEMENT OF WEIGHT (MISPLACED ,)

(BT) A Block Trail or Keyword function was specified incorrectly.

## INVALID SUBSET COMMAND

(CM) Only some CMS commands may be executed from the CMS SUBSET environment. See the IBM Virtual Machine Facility/370: CMS Command and Macro Reference for a complete list of allowable subset commands.

## INVALID VIEWSPEC STRING

(J) The <lp> was in a Jump or Pmuj which contained an invalid Viewspect string.

## INVALID WEIGHT SPECIFIED

A Block Trail or Keyword function was specified incorrectly.

## INVALID WINDOW CONFIGURATION

(SW) A valid option must be specified (1A, 1B, 2A, 2B, 3A, 3B, 4A). This pertains only to the multiple-window version.

## INVALID WINDOW NUMBER SPECIFIED

A current window number must be specified. This pertains only to the multiple-window version.

## INVISIBLE TEXT

Non-linear travel (e.g., GLABEL or JUMP) into a Block which is protected from display by display Keywords will result in this message. A non-linear travel commands (e.g. RETURN) must then be used to get back to displayable text. This is true even if the appropriate SKDISPLAY is issued to "turn on" the previously "invisible" Block.

## INVISIBLE TEXT ENCOUNTERED: REJECT(R) OR ACCEPT (A)

(US) If the user accepts this substitution, a context string which is currently hidden in a Block governed by Display Keywords not in effect, will be changed.

## I/O ERROR WHILE READING DICTIONARY FROM &lt;fn&gt; &lt;ft&gt;

(cmac) Something is wrong with the specified file. Recreate it, or contact the FRESS staff. (routine lmsetd)

## I/O ERROR WHILE READING HEADER RECORD FROM &lt;fn&gt; &lt;ft&gt;

(cmac) Something is wrong with the specified file. Recreate it, or contact the FRESS staff. (routine lmsetd)

## JUSTIFICATION: EDIT IGNORED

Editing may not be done while on-line justification is in effect. Reset the viewspecs (see SVIEW)

## KEYWORD STRING INCLUDES TWO DELIMITERS IN A ROW

The <key> or <dkey> parameter of an editing command either contained an invalid Keyword or Keyword String, or the edit would have changed a Keyword or Keyword String making it invalid.

## KEYWORD STRING LIMITED TO 255 CHARACTERS

The specified edit would result in a Keyword string greater than maximum length.

## KEYWORDS LIMITED TO 16 CHARACTERS

A Block Trail or Keyword function was specified incorrectly.

## LABEL ALREADY EXISTS

(ML) The Label specified in MLABEL already exists.

## LABEL MUST BE WITHIN A BLOCK

The Block qualifier (-b) was specified, but the specified <lp> was not in a Block.

## LABEL NOT FOUND IS SPECIFIED FILE(S)

There is no Block labelled <n>.

## LABEL SPECIFIED DOES NOT EXIST

(CT, GL, MT) <label> is not defined in <file>, or in the current file if <file> is specified (GLABEL).

## LABELS LIMITED TO 16 CHARACTERS

(C, I, ML, S) Either the <label> specified in MLABEL was longer than 16 characters or the end or intermediate result of the edit (INSERT or SUBSTITUTE) would produce a Label longer than 16 characters.

## LAST COMMAND WILL CAUSE &lt;n&gt;-DISK OVERFLOW; SYSTEM WILL IPL

FRESS has recognized a possible disk overflow on the <n> disk. The user should clear more room on the appropriate disk before attempting the command again.

## LENGTH EXCEEDS DEVICE MAX

(SD) The line length is greater than the maximum line length for the terminal.

## LITERAL CONTAINS INVALID TEXT

This message appears when the second parameter of FOOTNOTE is anything other than a number. It can also be caused if any <text> parameter contains more backspaces than ordinary characters.

## LP FOR WORD EDIT MAY NOT BE A BLANK OR PUNCTUATION

The user specified an edit using a command qualifier with an <lp> of a blank or some kind of punctuation.

## LP MAY NOT BE WITHIN A STRUCTURE CODE OR DATA FIELD

If the word qualifier (-w) is specified, the text to be edited must be literal.

## LP MUST BE IN AN ORDER

The order qualifier (-o) was specified but the specified <lp> was not in an order (i.e., a piece of Hypertext).

## LP MUST BE WITHIN A BLOCK

The Block qualifier ("B") was specified but the specified <lp> was not in a Block.

## LP'S ARE IN THE WRONG ORDER

The two <lp>'s of a <scope> must be specified in the order in which they appear when traveling sequentially through the file from start to end. This problem occurs only when at least one end of the <scope> has been deferred and later resolved in an incorrect position with respect to the other <lp>.

## LP'S IN SPACE STILL EXIST - SPACE NOT DELETED

The Space could not be deleted because the Implied Insert Point is pointing inside it. However, that the command has been executed.

## MACLIB ERROR

One of the libraries specified in the FRESS invocation does not exist.

## MACRO EXPANSION TOO LONG

(cmac) Each command line in a command macro, when expanded, must not exceed 130 characters. Either use abbreviations or break the line into more than one logical line.

## MACRO NAME TOO LONG

(cmac) Macro names may not exceed eight characters.

## MACRO NOT FOUND

The macro specified is not a member of any of the macro libraries included when FRESS was invoked.

## MAX EXCEEDED

(P, SC, T) For PRINT, this message means the number of lines in the Display Window is less than <n>. Specify a smaller number. For SCROLL, it means a SCROLL of more than 127 lines forward or backward was specified. For TYPE, more than 100 lines was specified.

## MESSAGE AND FILE SENT

The ".frscomm" along with a copy of the specified file has been sent to the FRESS staff.

## MESSAGE SENT

The ".frscomm" has been sent to the FRESS staff.

## MI COMMAND WAS NOT USED BEFORE &amp;I

(cmac) The MIDENTIFIER command must be issued prior to a Command Macro that uses the &I global variable.

## MISMATCHED PARENTHESIS

(BT, BTd) The boolean request string is syntactically incorrect.

## MISSING SYMBOLIC PARAMETER

(cmac) There were no default values for one or more unspecified symbolic parameters. It is possible to specify default values for any parameter in a Command Macro. This value may be the null string, which is specified by including a Key delimiter but no explicit value for the parameter.

## MOVETO LP BETWEEN SCOPE LP'S

(CO, M) The <lp> is (illegally) contained in <scope> (i.e., one cannot copy or move a string within itself).

## NO BLOCKS SATISFY REQUEST

(BT, BTd) No Blocks have the given characteristics.

## NO CURRENT BLOCK TRAIL

(TR) A request was made with no existing Block Trail.

## NO CURRENT FILE

A display command (SCROLL, BOTTOM, etc.) has been attempted without a current file.

## NO CURRENT FILE: IGNORED

An editing or traveling command has been attempted without a current file.

## NO DECIMAL LABEL SPECIFIED

(GD) The <n> was left null.

## NO FILE OPEN

(Q) Either no GFILE has been issued, or all open files have been freed using FFILE, or all open files have been freed using FFILE.

## NO LIBRARY NAME SPECIFIED

Contact the FRESS staff. (routine lmembers)

## NO LIBRARY TYPE SPECIFIED

Contact the FRESS staff. (routine lmembers)

## NO MESSAGE SENT

The first line typed of the ".frscomm" was null.  
Re-".frscomm".

## NO MORE BLOCKS IN TRAIL

(TR) A request was made at the last Block of the Trail.

## NO MORE RETURN POINTS

(R) There are no more entries in the Return Stack, and therefore no more RETURNS can be done, until a non-linear travel command is issued.

## NO OR 3 OPEN FILES

(FU) FULLPRINT was not executed; specify <file> explicitly.

## NO OUTSTANDING INCOMPLETE FUNCTIONS

(&C) There are no deferred <lp>'s or <scope>'s or pending commands to be cleared.

## NO REGULAR TEXT ALLOWED IN STRUCTURE SPACE

(C, I, S) The user attempted to insert text directly into the Structure Space and was not merely editing what was already there (e.g., fixing the spelling of a Label).

## NO REQUEST SPECIFIED OR LEFT FROM BEFORE

(BT, BTD) The command was issued without specifying a parameter, and none was previously specified.

## NO SAVED PATTERN

(L) No previous LOCATE had been issued from which the <lit> parameter could default, and it wasn't specified.



## NO SPACE ON A-DISK FOR NEW FILE

(MF) Use SFILE or the "CMS ERASE <file> FRESS" command to clear some room for new files.

## NO TEXT IN SCOPE OF TEXT-ONLY EDIT

If "T" (text-only) was specified in response to the "HYPERTEXT ENCOUNTERED..." message, <scope> must contain some regular text to edit.

## - AND ( MUST FOLLOW &amp; OR |

A Block Trail or Keyword function was specified incorrectly.

## NOTHING IN WORK SPACE TO BE MOVED

(CFW, MFW) The Work Space was empty when a MFWORK or CFWORK was specified.

## NOTTED KEYS MUST BE ANDED WITH UNNOTTED ONES

The user can request "key1&-key2", but not simply "-key2".

## NULL LABEL IS ILLEGAL

(ML) The user attempted a MLABEL with a null <Label> parameter.

## ONE OF REQUESTED KEYS NOT REFERENCED IN FILE

A Keyword specified in a boolean request does not exist.

## OPTIONS=

(US) A substitute in Inquire Mode has been performed. The user must now specify one of the options: ("A") accept the substitution, ("R") reject it, but continue with the command, or ("X") reject it, and stop the command.

## OUT OF ROOM IN PUSHDOWN STACK

Contact FRESS staff.

## OUT OF WORKING STORAGE

Contact FRESS staff.

## PACK SUCCESSFUL &lt;bbbb&gt;/&lt;aaaa&gt;

(PF) The specified file has been successfully packed; it formerly occupied <bbbb> CMS Blocks and now occupies <aaaa> CMS Blocks.

## PACKED PAGES MUST BE MORE THAN HALF FULL

(PF) A minimum percentage of 50 must be specified.

## PAGING ERROR

The file was accessed incorrectly by FRESS, contact the FRESS staff.

## PARSE TREE TOO LONG

Contact the FRESS staff.

## PASSWORD ALREADY EXISTS

(AP) The password specified is already defined in the current file.

## PASSWORD DOES NOT EXIST

(DP) The password specified has not been defined for the current file.

## PASSWORDED FILE NOT OPEN

(C, D, S) One end of an interfile Jump has been deleted (the command has been executed). The other end wasn't deleted because it was in a file which did not have the "DEFAULT" password.

## PATTERN NOT FOUND, EO AREA

(L) The specified pattern was not found within the limit of the Area in the direction specified.

## PATTERN NOT FOUND, EO COUNT

(L) The specified pattern was not found within 8000 characters of the current display point. This message will not arise when the "L" modifier is used.

## PATTERN NOT FOUND, OPTIONS=

(US) A short scan failed to match the <scope> pattern with any text in the editing buffer. At this point the user may reject the command ("R" or "X") or continue with a long scan ("A").

## PICTURE DOES NOT EXIST. DO AN LPICT TO LIST NAMES

This message only has meaning for the Imlac version of FRESS and should be ignored by other users.

## PICTURE DELETED

This message only has meaning for the Imlac version of FRESS and should be ignored by other users.

## PICTURE HAS BEEN DELETED

This message only has meaning for the Imlac version of FRESS and should be ignored by other users.

## PICTURE NAMES LIMITED TO EIGHT CHARACTERS

This message only has meaning for the Imlac version of FRESS and should be ignored by other users.

## PICTURE SPECIFIED DOES NOT EXIST

This message only has meaning for the Imlac version of FRESS and should be ignored by other users.

## PLEASE FREE A FILE &amp; TRY AGAIN

(OT) OTYPE had insufficient work space to complete. As indicated, the user should free one or more files (using FFILE) and try the OTYPE again.

## PRINTER TROUBLE

(OT) There is some difficulty in communication with the Offline Printer. Contact a FRESS staff member.

## PROCEED

This is the response to a number of FRESS commands and indicates that the command was completed successfully.

## PROCESSING NEEDED TOO MUCH SPACE

The user stacked too many commands on a single line or had too many pending (deferred) functions. If the former case is true, stack fewer commands. In the latter case, complete the deferred <lp> or <scope> to allow the pending functions to complete.

## PUSHDOWN STACK FULL, NO KEYWORD CHECK POSSIBLE

Contact the FRESS staff.

?

This message is often typed when it is impossible for FRESS to decipher a particular command line. The question mark will appear under the section of the line causing the problem. The user may type "?", in response, for a more complete explanation of the problem. Sometimes the explanation will be typed after the next command is issued, even if the new command produces no errors. No harm is done to the file or to the execution of the new command (this is only an occasional malfunction in the procedure).

## &amp;R&lt;n&gt; HAS NOT BEEN DEFINED.

(cmac) &R<n> cannot be referenced before its initial definition is encountered within the macro. Macro expansion and execution are terminated.

## REQUEST MUST END WITH ) OR ;

(BT) A Block Trail or Keyword function was specified incorrectly.

## RESULTING LABEL ALREADY EXISTS

The specified edit will produce a Label that already exists.

## RETURN FILE NO LONGER OPEN

(R) The file of the next display point in the stack has been FREEd. The user may continue popping the stack with additional RETURNS.

## SCOPE AND LP MUST BE IN SAME SPACE

When deferring <scope> and <lp>, be sure to stay in the same Space (e.g., Text, Structure, etc.).

## SCOPE INCLUDES INCOMPLETE BLOCK

(M, MB, MD) A new Block cannot be made which includes only one end of a Block, nor is a Hypertext MOVE which includes only one end of a Block permissible. A Block is a unit and must be treated as such.

## SCOPE QUALIFIER USED NOT SUPPORTED

A qualifier of -d, -p, or -s (designed but not yet implemented) was specified. (These qualifiers are meant to represent data field, paragraph, and sentence.)

## SCRATCH FAILED

(SF) Contact a member of the FRESS staff.

## SCRATCH SUCCESSFUL

(SF) The file has been erased.

## SCROLL OR CHAR LENGTH &gt; 127 IN POSITION STR

The line or character displacement number is too large (see Section 5.2).

## SECOND INCOMPLETE FUNCTION ENCOUNTERED

A command must be completely resolved before another command can be issued in which <scope> or <lp> is deferred.

## SECOND LP MUST BE A BLOCK START

(MDRD, RTA) The second <lp> of the command must be a Block start.

## SEMANTICS CHECK

A restriction on a parameter has been ignored, e.g., a password longer than 7 characters has been given.

## SOMETHING WENT WRONG

Contact FRESS staff.

## SPACE OF RETURN POINT HAS BEEN DELETED

(R) The Space (Work or Label) which contained the next display point in the stack has been deleted. The user may continue popping the stack with additional RETURNS.

## SPACING SPECIFIED INCORRECTLY

(OT) The spacing parameter for OTYPE must be 1, 2, or 3.

## SPECIFIED SPACE DOES NOT EXIST

(DS) The specified Space (Work or Label) does not exist. The Structure Space and Text Space always exist.

## \*SPLICE CANNOT BE COMPLETED\*

If its corresponding Ecilpse cannot be located, this message will appear in the display buffer immediately after a Splice. This condition arises when one end of an interfile Jump or Splice is not available (e.g., file has been erased or deleted). If this is not the case, the user should contact the FRESS staff.

## STACK CONTROL BLOCK ALL FILLED

(SA) Before each non-linear travel command is executed, the current location is saved in the Stack Control Block. This Block is now full, and RETURNS should be issued to clear the

stack. See the description of RETURN or SAVE for more information.

\*START OF TEXT AREA\*

INPUT

(SIT) The user is now at the top of the file, in Swift Input Mode. (TI) The user is now at the top of the file, in Input Mode.

STARTING COLUMN < 1

(0) The S<nn> parameter must be greater than 0; the command was not executed.

STARTING COLUMN > ENDING COLUMN

(0) S<nn> must be greater than E<nn>.

SYNTAX ERROR, OFFSET=

An invalid delimiter was encountered in the <data> field of a format code (see Section 4.1). This message applies to codes having multiple <data> fields (e.g., !+SETTAB+, !+TABLE+). The offset value is equal to the difference from the "!" (or last ";"), to the beginning of the invalid parameter of the line returned when FELONG is issued. If this message appears after FULLPRINT, the command was executed, but it contained an invalid format code (marked by asterisks to the right of the formatted text).

TEXT TO INSERT MUST BE SPECIFIED

(SUR) Either <lit1>, or <lit1> and <lit2> must be specified

TOO FEW VALUES GIVEN

The command takes one or more required parameters that were not specified.

TOO LONG CANNOT ADD HEADER

(LL) The pattern that is to be scanned for cannot exceed 253 characters.

TOO MANY CHARACTERS IN KEYWORDS (IMPL. RESTRICT.)

Implementation restriction, contact the FRESS staff.

TOO MANY FILES OPEN

(0, PF) Because of space limitations, it is necessary to close some files (using FFILE). PFILE should be repeated once this is done.

TOO MANY OPEN FILES

The specified edit caused a new file to be opened and there is insufficient Space for FRESS to complete the operation. Free one or more files and try again.

## TOO MANY OPERATORS (IMPL. RSTRCT.)

Contact FRESS staff.

## TOO MANY PARAMETERS SPECIFIED

Contact the FRESS staff. (routine lmembers)

## TRIED FUNCTION WITH IMPLIED POINT BUT NONE SET

A command which sets the Implied Insert Point must be specified before it can be used (see Appendix D.2).

## TRIED TO POP TOO MANY TIMES

(&P) There were not <n> pointers, or there was only one pointer in the Implied Insert Stack. The stack remains unchanged.

## TRIED TO REPEAT A NON-REPEATABLE FUNCTION

(&G) Some commands such as FULLPRINT and REVERT cannot be repeated using the &GIN function. Specify the command again.

## TWO DELIMITERS IN A ROW

(BT, BTD) A Block Trail or Keyword function was specified incorrectly.

## TWO KEYS SPECIFIED WITH NO OPERATOR

(BT) A Block Trail or Keyword function was specified incorrectly.

## TWO VALUES SPECIFIED WITHIN ONE KEYWORD

The <key> or <dkey> parameter of an editing command either contained an invalid Keyword or Keyword String, or the edit would have changed a Keyword or Keyword String making it invalid.

## TWO WEIGHTS GIVEN ON ONE KEYWORD OR VALUE

The <key> or <dkey> parameter of an editing command either contained an invalid Keyword or Keyword String, or the edit would have changed a Keyword or Keyword String making it invalid.

## UNDEFINED MACRO

A format code was encountered and its corresponding definition has not been SCROLLED over. It is also possible that the macro in question is imbedded with another macro which is not defined.

## UNDEFINED SPACE

(BI, DS) For BI, Input is not allowed in the specified Space. For DS, the <space> indicated has not been created (used) yet.

## UNKNOWN COMMAND NAME

FRESS doesn't know what you are trying to tell it.

## UNKNOWN CP/CMS COMMAND

(CM) Non-existent CP or CMS subset command.

## UNKNOWN SPACE

(DS) An invalid symbol was specified as <space>.

## UNLIGHTPENNABLE TEXT

(L, US) Either the "JU" viewspec is in effect, or the <lit> parameter of LOCATE specified a piece of text in the display buffer which does not correspond to any real text in the file and which therefore cannot be located or edited. An example is the "\*\*\*\*END OF LABELS\*\*\*\*" line in the Label Space.

## UNRECOGNIZED RESPONSE, PLEASE RESPECIFY

(US) The <options> string contains invalid option characters.  
Warning: USUBSTITUTE may infinite loop under this condition. It is suggested that the command be exited by specifying "X." USUBSTITUTE can then be safely re-issued with the proper <options>.

## UNSUPPORTED COMMAND

Certain house functions have been designed but not yet implemented. If the command name for one of these is typed, this message will be returned.

## UNSUPPORTED STRUCTURE FOUND

The file may be bad; contact FRESS staff.

## VIEWSPECS LIMITED TO 255 CHARACTERS

(C, I, S) The command resulting viewspec string on a Jump or Splice, would have exceeded the maximum length.

## WAITING

There is an outstanding <lp> waiting to be resolved. To do this, type "?/<lp>." To clear all outstanding functions, type "&C."

## WAITING FOR CARD READER TO FILL

(O) The "R" option was specified, but the reader is empty. OREAD will wait for it to fill before continuing.

## WAITING ISCOPE

There is an outstanding <scope> waiting to be resolved. To do this, type "?/<scope>". To clear all outstanding functions, type "&C".

## WARNING: INTERFILE JUMP FOUND WITHOUT PMUJ

(C, D, S) One end of an interfile Jump has been deleted (the command has been executed); the other end could not be found.

## WARNING: TEXT IN ANNOTATION BLOCK NOT DELETED

(C, D, S) The only Tag referencing a particular Annotation has been deleted; the Block around the Annotation has been deleted, but the text remains.

October 1, 1979

PROBLEMS AND MESSAGES  
MESSAGES

WEIGHT GIVEN IN WRONG PLACE

A Block Trail or Keyword function was specified incorrectly.

WORK SPACE DOES NOT EXIST

(NA) The user attempted to create a new Area in the Work Space, but the Space did not exist.

WRONG SYNTAX IN PATTERN

Respecify.





TABLE OF CONTENTS

## AVAILABLE FRESS DOCUMENTATION

PREFACE.....	i
HOW TO READ THIS MANUAL.....	ii
Organization.....	ii
Notation.....	iii
TABLE OF CONTENTS.....	iv
TABLE OF FIGURES.....	x
1 INTRODUCTION.....	1
1.1 History.....	1
1.2 Support.....	3
1.3 Overview.....	4
• The FRESS Environment.....	4
• Stream Editor.....	5
• Editing Buffer/Display Window.....	5
• Getting Output.....	5
• Hypertext.....	6
• Online Display.....	7
• Key Delimiter.....	7
• Deferrals.....	7
• File Structure.....	8
• Implied Insert Point.....	10
• As-Is Mode.....	10
2 USING FRESS.....	11
2.1 Invoking FRESS.....	11
2.2 FRESS Characteristics.....	12
2.2.1 Explanation of the "T"-Version.....	13
2.3 Display and Edit Modes.....	16
2.3.1 Overriding Display and Edit Modes.....	17
3 CONTROL CHARACTERS.....	18
3.1 Summary Table.....	18
3.2 Descriptions.....	20

4	FORMATTING.....	26
4.1	Overview.....	26
4.2	Notation for Formatting Code Specifications.....	28
4.3	Alter Codes.....	29
	!+B+ set the <u>blank</u> underscoring mode.....	31
	!+COLUMN+ set the number of text <u>columns</u> .....	32
	!+DATE+ print the current <u>date</u> .....	34
	!+DEPTH+ set the number of text <u>lines</u> .....	35
	!+DI+ set the <u>decimal</u> block <u>indentation</u> level.....	36
	!+GRID+ draw a <u>grid</u> .....	37
	Reference Tables.....	38
	Grid Construction.....	39
	Horizontal Expansion.....	40
	Vertical Expansion.....	41
	!+GUTTER+ set <u>gutter</u> size between columns.....	47
	!+HEAD+ define <u>heading</u> .....	48
	!+HEAD+ select <u>heading</u> table.....	49
	!+JUST+ set <u>justification</u> mode.....	51
	!+MARGIN+ set text <u>margins</u> .....	53
	!+OFFSET+ set binding <u>offset</u> .....	54
	!+PAGE+ define output buffer page size.....	55
	!+PARA+ define <u>paragraph</u> .....	56
	!+SETTAB+ set <u>tab</u> stops.....	57
	!+SPACE+ set <u>line spacing</u> .....	59
	!+TABLE+ define <u>table</u> of tab column boundaries.....	60
	!+TITLE+ define running <u>title</u> strings.....	62
	!+TOFC+ define <u>Table of Contents</u> .....	66
	!+WIDOW+ set <u>widow</u> level (stand-alone lines).....	68
	!+WIDTH+ set <u>width</u> of text line.....	69
4.4	Edit Codes.....	70
	!-B- draw <u>box</u> .....	72
	!-C- <u>center</u> line.....	74
	!-E- end <u>heading</u> .....	75
	!-F- start/end <u>footnote</u> .....	76
	!-H- start <u>heading</u> .....	78
	!-I- <u>indent</u> line.....	79
	!-J- hanging indent.....	80
	!-K- <u>conditional</u> page (column).....	81
	!-N- <u>new</u> page (column).....	82
	!-P- start <u>paragraph</u> .....	84
	!-R- start/end <u>revision</u> bars.....	85
	!-S- <u>skip</u> line(s).....	86
	!-T- <u>tab</u> .....	87
	!-U- <u>justified</u> tab.....	90
	!-X- <u>expand</u> line.....	91
4.5	Format Macro Codes.....	92
4.6	Special Character Codes.....	95
4.6.1	ALX Print Train.....	97
4.6.2	TN Print Chain.....	98
4.7	Underscore Codes.....	99

5 FRESS COMMANDS.....	101
5.1 Specification.....	101
5.2 Line and Character Displacement.....	102
5.3 The Special "&" LP character.....	104
5.4 Specifying Multiple Commands On One Line.....	105
5.4.1 The FRESS Command Separator.....	105
5.4.2 The CMS Linend Character.....	107
5.5 Command Qualifiers.....	108
5.6 Parameter Specification.....	111
5.7 Parameter Listing.....	112
5.8 Command Descriptions.....	114
A.....	114
• A    Accept.....	114
• AP   Add Password.....	115
B.....	117
• B    Bottom of Space.....	117
• BA   Bottom Area.....	118
• BARS Bars.....	119
• BI   Bottom Input.....	120
• BT   Block Trail Continuous.....	121
• BTD  Block Trail Discrete.....	123
C.....	124
• C    Change.....	124
• CA   Capitalize Text.....	125
• CF   Copy File.....	126
• CFW  Copy From Work Space.....	127
• CM   Execute CMS Subset Environment Command.....	128
• CO   Copy Text.....	129
• CP   Change Password.....	131
• CT   Copy to Label.....	132
• CTW  Copy to Work Space.....	133
D.....	134
• D    Delete Text.....	134
• DP   Delete Password.....	135
• DS   Display Space.....	136
• DV   Display Viewspecs.....	137
E.....	139
• E    End FRESS Session.....	139
F.....	140
• F    Free File.....	140
• FE   Find Error.....	141
• FL   Flip.....	142
• FMSG FRESS Message Control.....	144
• FO   Make Footnote.....	145
• FU   Full Printout.....	146
G.....	151
• G    Get File.....	151
• GD   Get Decimal Label.....	153
• GL   Get Label.....	154
H.....	155
• H    Help.....	155
I.....	156
• I    Insert Text.....	156
• IA   Insert Annotation.....	158

• IB	Insert Before.....	159
• IBL	Insert Block.....	160
• ID	Insert Decimal Block.....	162
• IM	Imbed.....	163
J.....		165
• J	Jump.....	165
L.....		167
• L	Locate (Pattern Scan).....	167
M.....		169
• M	Move Text.....	169
• MA	Make Annotation.....	170
• MB	Make Block.....	171
• MC	Macro Comment Control.....	172
• MD	Make Decimal Block.....	173
• MDR	Make Decimal Reference.....	175
• MDRD	Make Decimal Reference Deferred.....	177
• MF	Make File.....	178
• MFW	Move from Work Space.....	179
• MI	Make Identifier.....	180
• MJ	Make Jump.....	181
• ML	Make Label.....	183
• MS	Make Splice.....	185
• MT	Move to Label.....	187
• MTW	Move to Work Space.....	188
N.....		189
• N	Next.....	189
• NA	New Area.....	190
O.....		191
• O	Offline Read.....	191
• OT	Offline Type.....	194
P.....		196
• P	Print.....	196
• PF	Pack File.....	197
Q.....		199
• Q	Query System Status.....	199
R.....		200
• R	Return.....	200
• REV	Revert.....	201
• RI	Ring.....	202
• RTA	Refer To Annotation.....	203
S.....		204
• S	Substitute Text.....	204
• SA	Save Current Location.....	205
• SC	Scroll.....	206
• SD	Set Display Window.....	207
• SF	Scratch file.....	209
• SI	Swift Input.....	210
• SIB	Swift Input Bottom.....	212
• SIT	Swift Input Top.....	213
• SK	Set Keyword Annotation Request String.....	214
• SKD	Set Keyword Display Request String.....	215
• SKJ	Set Keyword Jump Request String.....	216
• SL	Sleep.....	217
• SM	Set Mode of Display.....	218
• SPL	Split Area.....	219

• SUR	Surround Text.....	221
• SV	Set Viewspecs.....	222
T.....		226
• T	Type.....	226
• TI	Top Input.....	227
• TR	Trail.....	228
U.....		229
• U	Underscore.....	229
• UNC	Uncapitalize.....	230
• US	Uniform Substitute.....	231
6	FRESS HOUSE FUNCTIONS.....	234
6.1	Overview.....	234
6.2	Descriptions.....	235
• &A	Enter As-is Mode.....	235
• &C	Clear Function Area.....	236
• &E	Execute.....	237
• &G	Repeat Last Command Line.....	238
• &N	Leave As-is Mode.....	240
• &P	POP Implied Insert Pointer.....	241
• &R	Route.....	245
• &T	Type on, off.....	246
7	COMMAND MACROS.....	247
7.1	Overview and Use.....	247
7.2	Creation.....	249
7.2.1	Individual Macros.....	249
7.2.1.1	Symbolic Parameters (Index Variables).....	251
7.2.1.2	&READ.....	251
7.2.1.3	&COMMENT.....	253
7.2.1.4	&IDENTIFIER.....	254
7.2.1.5	Additional Macro Examples.....	255
7.2.2	Macro Libraries.....	256
7.2.2.1	Editing Functions for <maclib> Manipulation.....	256
7.2.2.2	Printing Functions for <maclib>.....	257
7.3	SYSTEM COMMAND MACROS.....	258
7.3.1	Individual Descriptions.....	258
	.DS <space> [<n>].....	258
	.END.....	258
	.FRSCOMM.....	258
	.GDL <dec#>.....	258
	.GET.....	259
	.GF <file> [<n>].....	259
	.HELP.....	259
	.IDB <dec#> <label> <text>.....	259
	.IDBT <lit> <label> <text>.....	259
	.L <lit>.....	259
	.MOB <moved-block> <move-to-block>.....	259
	.MOBT <block#> <lp>.....	259
	.P [<scroll#>] [print#].....	259
	.PRACTICE.....	260
	.PUT <file> <lp> <command> <lp>.....	260
	.REMOTEFU <file> [<options>].....	260
7.3.2	Listing.....	261

A JUSTIFICATION.....	264
A.1 Column/Page Justification.....	264
A.2 Column Balancing (for multiple columns only).....	264
A.3 Line Justification.....	265
A.4 Formatting Blanks.....	265
B HYPERTEXT CHARACTERISTICS.....	266
C KEYWORD STRINGS.....	268
C.1 Keyword Specification.....	268
C.2 Boolean Request Format.....	270
C.3 Keyword Space.....	271
D COMMAND LISTS.....	272
D.1 Ordered by Type.....	272
D.1.1 Display Commands.....	272
D.1.2 Editing Commands.....	272
D.1.3 Travel Commands.....	273
D.1.4 System Commands.....	274
D.2 Implied Insert Commands.....	274
E PROBLEMS AND MESSAGES.....	275
E.1 Types of Problems.....	275
E.2 System Errors.....	276
E.3 Messages.....	277

TABLE OF FIGURES

Figure 1.1 -- System-Defined Spaces.....	9
Figure 2.1 -- Different Version Characteristics.....	12
Figure 3.1 -- Control Character Summary Table.....	19
Figure 4.1 -- Alter Code Properties.....	29
Figure 4.2 -- Page Layout Showing Alter Codes.....	30
Figure 4.3 -- Grid Characters and Character Symbols.....	38
Figure 4.4 -- Vertical and Horizontal Expansions of Grid Characters.....	38
Figure 4.5 -- Grid Code Formal Definition.....	39
Figure 4.6 -- Heading Table Characteristics.....	49
Figure 4.7 -- Title String Default Settings.....	63
Figure 4.8 -- Default Title String Locations.....	64
Figure 4.9 -- Edit Code Properties.....	72
Figure 5.1 -- Jumping in Main Spaces.....	166
Figure B.1 -- Structure-ID Symbol Representations.....	265
Figure B.2 -- Structure Data Fields.....	266
Figure C.1 -- Illegal Keyword Characters.....	268



TABLE OF CONTENTS

## AVAILABLE FRESS DOCUMENTATION

PREFACE.....	i
HOW TO READ THIS MANUALL+: -- How to Read**+.....	ii
Organization.....	ii
Notation.....	iii
1 INTRODUCTION.....	1
1.1 History.....	1
1.2 Support.....	3
1.3 Overview.....	4
• The FRESS Environment.....	4
• Stream Editor.....	5
• Editing Buffer/Display Window.....	5
• Getting Output.....	5
• Hypertext.....	6
• Online Display.....	7
• Key Delimiter.....	7
• Deferrals.....	7
• File Structure.....	8
• Implied Insert Point.....	10
• As-Is Mode.....	10
2 USING FRESS.....	11
2.1 INVOKING FRESS.....	11
2.2 FRESS CHARACTERISTICS.....	12
Figure 2.1 -- Different Version Characteristics.....	12
2.2.1 Explanation of the "T"-Version.....	13
2.3 DISPLAY AND EDIT MODES.....	16
2.3.1 Overriding Display and Edit Modes.....	17
3 CONTROL CHARACTERS.....	18
3.1 Summary Table.....	18
Figure 3.1 -- Control Character Summary Table.....	19
3.2 Descriptions.....	20
4 FORMATTING.....	26
4.1 Overview.....	26
4.2 Notation for Formatting Code Specifications.....	28
4.3 Alter Codes.....	29
Figure 4.1 -- Alter Code Properties.....	29
!+B+ set the <u>blank</u> underscoring mode.....	31
!+COLUMN+ set the <u>number</u> of text <u>columns</u> .....	32
!+DATE+ print the current <u>date</u> .....	34
!+DEPTH+ set the <u>number</u> of text <u>lines</u> .....	35
!+DI+ set the <u>decimal</u> block <u>indentation</u> level.....	36
!+GRID+ draw a <u>grid</u> .....	37
Reference Tables.....	38
Figure 4.3 -- Grid Characters and Character Symbols.....	38

Figure 4.4 -- Vertical and Horizontal Expansions of Grid Characters.....	38
Grid Construction.....	39
Figure 4.5 -- Grid Code Formal Definition.....	39
Horizontal Expansion.....	40
Vertical Expansion.....	41
!+GUTTER+ set <u>gutter</u> size between columns.....	47
!+HEAD+ define <u>heading</u> .....	48
!+HEAD+ select <u>heading</u> table.....	49
Figure 4.6 -- Heading Table Characteristics.....	49
!+JUST+ set <u>justification</u> mode.....	51
!+MARGIN+ set text <u>margins</u> .....	53
!+OFFSET+ set binding <u>offset</u> .....	54
!+PAGE+ define output buffer <u>page</u> size.....	55
!+PARA+ define <u>paragraph</u> .....	56
!+SETTAB+ set <u>tab</u> stops.....	57
!+SPACE+ set <u>line spacing</u> .....	59
!+TABLE+ define <u>table</u> of tab column boundaries.....	60
!+TITLE+ define running <u>title</u> strings.....	62
Figure 4.7 -- Title String Default Settings.....	63
Figure 4.8 -- Default Title String Locations.....	64
!+TOFC+ define Table of Contents.....	66
!+WIDOW+ set <u>widow</u> level (stand-alone lines).....	68
!+WIDTH+ set <u>width</u> of text line.....	69
4.4 Edit Codes.....	70
Figure 4.9 -- Edit Code Properties.....	71
!-B- draw <u>box</u> .....	72
!-C- <u>center</u> line.....	74
!-E- end <u>heading</u> .....	75
!-F- start/end <u>footnote</u> .....	76
!-H- start <u>heading</u> .....	78
!-I- <u>indent</u> line.....	79
!-J- hanging indent.....	80
!-K- <u>conditional</u> page (column).....	81
!-N- <u>new</u> page (column).....	82
!-P- start <u>paragraph</u> .....	84
!-R- start/end <u>revision</u> bars.....	85
!-S- <u>skip</u> line(s).....	86
!-T- <u>tab</u> .....	87
!-U- justified tab.....	90
!-X- <u>expand</u> line.....	91
4.5 Format Macro Codes.....	92
4.6 Special Character Codes.....	95
4.6.1 ALX Print Train.....	97
4.6.2 TN Print Chain.....	98
4.7 Underscore Codes.....	99
5 .....	101
6 .....	102
7 .....	103
8 .....	104

9 FRESS COMMANDS.....	101
9.1 Description.....	101
9.2 Line and Character Displacement.....	102
9.3 The Special "&" LP character.....	104
9.4 Specifying Multiple Commands On One Line.....	105
9.5 Command Qualifiers.....	108
9.6 Parameter Specification.....	111
9.7 Parameter Listing.....	112
9.8 Command Descriptions.....	115
A.....	115
• A    Accept.....	115
• AP   Add Password.....	116
B.....	118
• B    Bottom of Space.....	118
• BA   Bottom Area.....	119
• BARS Bars.....	120
• BI   Bottom Input.....	121
• BT   Block Trail Continuous.....	122
• BTB  Block Trail Discrete.....	124
C.....	125
• C    Change.....	125
• CA   Capitalize Text.....	126
• CF   Copy File.....	127
• CFW  Copy From Work Space.....	128
• CM   Execute CMS Subset Environment Command.....	129
• CO   Copy Text.....	130
• CP   Change Password.....	132
• CT   Copy to Label.....	133
• CTW  Copy to Work Space.....	134
D.....	135
• D    Delete Text.....	135
• DP   Delete Password.....	136
• DS   Display Space.....	137
• DV   Display Viewspecs.....	139
E.....	141
• E    End FRESS Session.....	141
F.....	142
• F    Free File.....	142
• FE   Find Error.....	143
• FL   Flip.....	144
• FMSG FRESS Message Control.....	146
• FO   Make Footnote.....	147
• FU   Full Printout.....	148
G.....	153
• G    Get File.....	153
• GD   Get Decimal Label.....	155
• GL   Get Label.....	156
H.....	158
• H    Help.....	158
I.....	159
• I    Insert Text.....	159
• IA   Insert Annotation.....	161
• IB   Insert Before.....	162
• IBL  Insert Block.....	163
• ID   Insert Decimal Block.....	165

• IM	Imbed.....	166
J.....		168
• J	Jump.....	168
Figure 5.1 --	Jumping in Main Spaces.....	170
L.....		171
• L	Locate (Pattern Scan).....	171
M.....		173
• M	Move Text.....	173
• MA	Make Annotation.....	174
• MB	Make Block.....	175
• MC	Macro Comment Control.....	176
• MD	Make Decimal Block.....	177
• MDR	Make Decimal Reference.....	179
• MDRD	Make Decimal Reference Deferred.....	181
• MF	Make File.....	182
• MFW	Move from Work Space.....	183
• MI	Make Identifier.....	184
• MJ	Make Jump.....	185
• ML	Make Label.....	187
• MS	Make Splice.....	189
• MT	Move to Label.....	191
• MTW	Move to Work Space.....	192
N.....		193
• N	Next.....	193
• NA	New Area.....	194
O.....		195
• O	Offline Read <sup>1</sup> .....	195
• OT	Offline Type.....	198
P.....		200
• P	Print <sup>1</sup> .....	200
• PF	Pack File.....	201
Q.....		203
• Q	Query System Status.....	203
R.....		205
• R	Return.....	205
• REV	Revert.....	209
• RI	Ring.....	210
• RTA	Refer To Annotation.....	211
S.....		212
• S	Substitute Text.....	212
• SA	Save Current Location.....	213
• SC	Scroll.....	214
• SD	Set Display Window <sup>1</sup> .....	216
• SF	Scratch file.....	217
• SI	Swift Input.....	218
• SIB	Swift Input Bottom.....	220
• SIT	Swift Input Top.....	221
• SK	Set Keyword Annotation Request String.....	222
• SKD	Set Keyword Display Request String.....	223
• SKJ	Set Keyword Jump Request String.....	224
• SL	Sleep.....	225
• SM	Set Mode of Display.....	226
• SPL	Split Area.....	228
• SUR	Surround Text.....	230
• SV	Set Viewspecs.....	231

T.....	236
• T Type <sup>1</sup> .....	236
• TI Top Input.....	237
• TR Trail.....	238
U.....	239
• U Underscore.....	239
• UNC Uncapitalize.....	240
• US Uniform Substitute.....	241
9.9 Imlac Version Commands.....	244
• BW Blank Window.....	245
• COP Copy Picture†.....	246
• CPI Change Picture Name†.....	247
• CW Change Current Window.....	248
• DPI Delete Picture†.....	249
• LP List Pictures†.....	250
• MPR Make Picture Reference†.....	251
• PP Print Picture†.....	252
• SKE Sketch.....	253
• SP Scale Picture†.....	254
• SW Set Window Configuration.....	255
10 FRESS House Functions.....	257
10.1 Overview.....	257
10.2 Descriptions.....	258
• &A Enter As-is Mode.....	258
• &C Clear Function Area.....	259
• &E Execute.....	260
• &G Repeat Last Command Line.....	261
• &N Leave As-is Mode.....	263
• &P POP Implied Insert Pointer.....	264
• &R Route.....	268
• &T Type on, off.....	269
11 COMMAND MACROS.....	247
11.1 OVERVIEW AND USE.....	247
11.2 CREATION.....	249
11.2.1 Individual Macros.....	249
11.2.1.1 Symbolic Parameters (Index Variables).....	250
11.2.1.2 &READ.....	251
11.2.1.3 &COMMENT.....	253
11.2.1.4 &IDENTIFIER.....	254
11.2.1.5 Additional Macro Examples.....	255
11.2.2 Macro Libraries.....	256
11.2.2.1 Editing Functions for <maclib> Manipulation.....	256
11.2.2.2 Printing Functions for <maclib>.....	257
11.3 SYSTEM COMMAND MACROS.....	258
11.3.1 Individual Descriptions.....	258
.DS <space> [<n>].....	258
.END.....	258
.FRSCOMM.....	258
.GDL <dec#>.....	258
.GET.....	259
.GF <file> [<n>].....	259
.HELP.....	259

.IDB <dec#> <label> <text>.....	259
.IDBT <lit> <label> <text>.....	259
.L <lit>.....	259
.MOB <moved-block> <move-to-block>.....	259
.MOBT <block#> <lp>.....	259
.P [<scroll#>] [<print#>].....	259
.PRACTICE.....	260
.PUT <file> <lp> <command> <lp>.....	260
.REMOTEFU <file> [<options>].....	260
11.3.2 Listing.....	261
12 JUSTIFICATION.....	264
12.1 Column/Page Justification.....	264
12.2 Column Balancing (for multiple columns only).....	264
12.3 Line Justification.....	265
12.4 Formatting Blanks.....	265
13 HYPERTEXT CHARACTERISTICS.....	266
Figure B.1 -- Structure-ID Symbol Representations.....	266
Figure B.2 -- Structure Data Fields.....	267
14 KEYWORD STRINGS.....	268
14.1 Keyword Specification.....	268
Figure C.1 -- Illegal Keyword Characters.....	269
14.2 Boolean Request Format.....	270
14.3 Keyword Space.....	271
15 COMMAND LISTS.....	272
15.1 Ordered by Type.....	272
15.1.1 Display Commands.....	272
15.1.2 Editing Commands.....	272
15.1.3 Travel Commands.....	273
15.1.4 System Commands.....	275
15.2 Implied Insert Commands.....	275
16 PROBLEMS AND MESSAGES.....	276
16.1 Types of Problems.....	276
16.2 System Errors.....	277
16.3 Messages.....	278