

FILE: GUIDEA

Compliments of FRESS

A File Retrieval and Editing SyStem

Release 9.1 2 MAY 79

FRESS User's Guide

Andries van Dam
Carol L. Chomsky

Brown University
Providence, Rhode Island

February 2, 1975

1 FOREWORD

FRESS (File Retrieval and Editing SyStem) is a sophisticated and cost-effective text manipulation system. It is a vastly enhanced, multiconsole production version of the prototype HES (Hypertext Editing System). On IBM 2741 typewriter terminals, teletypes (TTY-33) and 2260 alphanumeric display consoles (and their equivalents), it allows authors and secretaries transcribing from marked-up manuscripts to create, review, edit, and print text files with unprecedented ease and power in an interactive, online environment.

The IBM 360/67 version of FRESS under the CP/CMS operating system has been in production use at Brown University since October, 1970, and users have dialed in from as far away as Toronto, Canada, and Washington, D.C. The VP/CSS version has been in commercial use on the NCSS timesharing service system (Stamford, Conn.) since December 1971. In addition, an OS/MVT version has been installed on a customer's 360/50 in Washington, D.C. containing a subset of the facilities described in this manual.

It should be noted that this manual is written primarily for users of the CP/CMS version of FRESS, with dial-up IBM 2741 typewriter terminals or teletypes. Minor differences in the command structure and default characteristics of the system are in effect for other terminals, while minor differences in logon procedure and hardcopy output are used in the OS/MVT version. These differences are noted in this manual where applicable.

It is assumed that the user has already read FRESS Concepts and Facilities Manual [2] for the basic methodology underlying FRESS and the format and specification of its commands. In this manual the user should read up to and including Section 3.6 for orientation and then read the sample editing session of Section 7. Reading the rest of the manual is then recommended. Because the system is so much richer and more powerful than other (commercially) available text editing systems, it takes a bit longer to learn to use it effectively. The "secretarial subset" of commands, however, may be assimilated within four to six hours of supervised practice, allowing useful work to be done after a single day of instruction. Any suggestions and comments both about the system and its documentation will be gratefully appreciated for future development.

2 INTRODUCTION

2.1 OUTLINE OF THE MANUAL

This manual describes the common base of FRESS, which includes all the information required to do ordinary (transcription-oriented) editing. Sections which contain information of secondary importance and which can be skipped on first reading are marked with a superscript plus sign ("^"). In Section 2 the general properties of FRESS are described as a prelude to the detailed description of the user command language in Section 3. This command language description is interspersed with simple examples of the most common editing functions. Section 4 describes various modes in which the user can request feedback from the system. In Section 5, formatting of the text for typewriter or line printer output is discussed. The user is introduced, in Section 6, to a few of the more advanced text structuring facilities; these features are designed to help in the more powerful editing operations that are unique to FRESS. Section 7 contains an annotated transcript of a typical FRESS editing session. Section 8 contains an alphabetical listing of the most frequently used FRESS commands.

Appendix 9 contains helpful hints about using the FRESS system. Appendix 10 contains a description of some of the operating system dependencies. Both these sections should be read before the user attempts his first FRESS session. Appendix 11 contains a list of the error messages which may be encountered. Appendix 12 classifies the FRESS commands explained in this manual into various categories: Section 12.1 lists the subset of commands which should be learned first, since they are necessary for editing; Section 12.2 places each command into one of the three main categories of editing, display, and file functions; and Section 12.3 lists all the commands which set the implied insert point (see Section 3.11).

The companion manual to this guide, FRESS Reference Manual: Structure and Commands [4] describes more advanced text structuring facilities such as jumps, splices, keywords, and automatic section numbering features. It also contains a complete listing of all FRESS commands, and is recommended for more advanced users of the system.

February 2, 1975

FRESS AS A STREAM EDITOR

2.2 FRESS AS A "STREAM-EDITOR"

2.2.1 PROGRAM EDITING VS. TEXT EDITING

Commercially available line oriented text editors are usually designed for program editing, and therefore allow the user to insert or delete whole lines, or to substitute one literal character string for another within a single line. Thus, a word in a line is deleted in such line editors by substituting a null string (one with no characters) for it; a phrase is changed by specifying it as it presently exists, followed by the desired new phrase.

While such a substitute command is optimal for the correction of typographical errors where the context specified need not be long, it has several disadvantages when real editing, as opposed to correction of spelling errors, is to be done. For example, when edits span more than half a line, an excessive amount of typing is required, so it is generally quicker to retype the entire line, even though half the line is not being changed.

As explained in detail below, FRESS has included several features designed to minimize such nonessential keystroking. One of these is the use of the "..." pattern when specifying context strings, with the same meaning as the ordinary ellipsis. FRESS also makes available separate, specialized commands which reflect the types of editing operations done by hand: insert, delete, substitute, and, especially, move (rearrange). Furthermore, these commands permit editing over more than one displayed line with automatic contraction and expansion of the surrounding text without regard to line boundaries. Thus one may, for example, delete from the middle of one line to the middle of a line three lines below in one operation, as explained below.

2.2.2 LINE EDITING VERSUS STREAM EDITING

In a typical commercially available "line" or line-based "context" editor like IBM's ATS or the CP-67/CMS editor, the user's file consists of a finite number of discrete lines, each from one character in width to some maximum width (e.g., 132). Typically, one may only examine and edit a single line at a time (Fig. 1).

INTRODUCTION

FRESS AS A STREAM EDITOR

February 2, 1975

While line orientation is natural for program text (which is primarily in card or line printer image format), it is most unnatural for free form manuscript text (reports, proposals, books, etc.). For example, a contiguous string of characters which happens to be split between lines cannot be located by the line oriented editing system since only one line is "current" at a time. What is worse is that text does not automatically expand or contract from one line to the next during editing. Loss of typed in characters may occur if the maximum line width is exceeded in the course of inserting new material. To avoid these problems, a FRESS file consists of a single indefinitely long "superline" or "stream" (Fig. 2). With a single operation one may edit any amount of this stream, e.g., deleting from "Boston" to "meadow" inclusive.

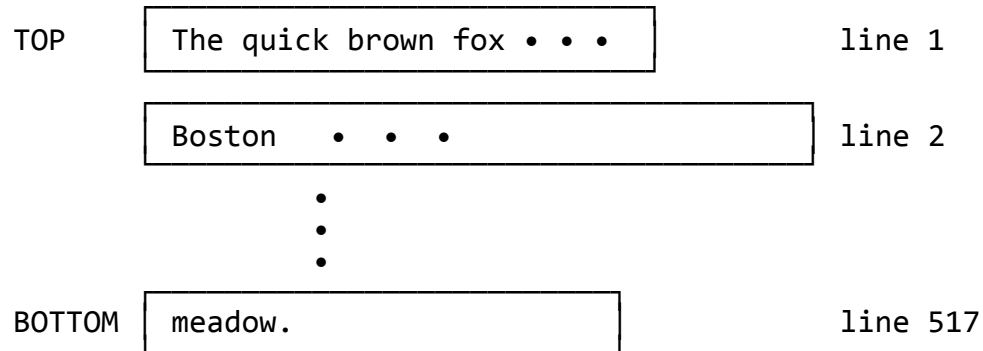


Figure 1: Model of a "Line or Line Oriented Context Editor" File

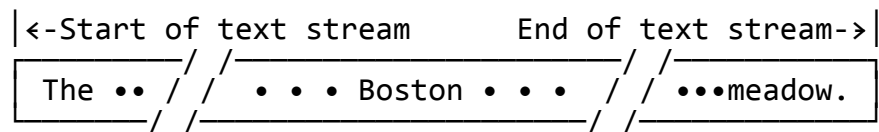


Figure 2: Model of the FRESS "Stream Editor" File

2.2.3 EDITING BUFFER, DISPLAY BUFFER AND DISPLAY WINDOW

On display consoles with special equipment, the user can supply the start and end points of edits by pointing at the relevant characters, called "Location Pointers" (LP's), with a Light Pen. In the case of typewriters and alphanumeric display terminals, one cannot point at LP's directly so one must

February 2, 1975

FRESS AS A STREAM EDITOR

indicate them by specifying "context patterns" (literal strings of characters occurring in the file) which inclusively surround the string to be edited. For example, the command

```
insert/fox/ jumped over the lazy dog
```

will insert the new string " jumped over the lazy dog" after the LP "x" specified by context pattern "fox".

The computer must then scan for such context strings in the text, which is expensive in terms of computer resources. Therefore, it is reasonable to restrict the amount of text to be scanned at one time. This prevents fruitless searches of a large file if the user specifies an incorrect context string. Thus one considers a "locality of reference" within the file called the "editing buffer" within which the context scanner is active (Fig. 3). This buffer is 2100 characters long.

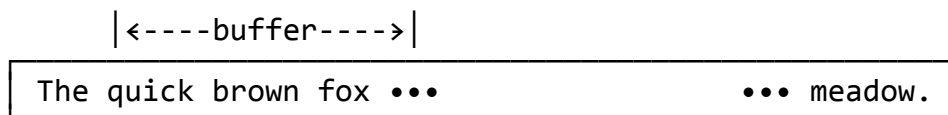


Figure 3: Editing Buffer Subset of File

Given a 2100 character display console, it is possible to see and manipulate the entire editing buffer at once. On a typewriter terminal, however, it would take a very long time to print an entire editing buffer. Since it is normal to work from "hard copy" (i.e., a paper manuscript) when using a typewriter console, one probably would wish to print only enough of the buffer to determine one's location in the text. Thus there exists a "display window" subsection of the editing buffer, which is the arrangement of text printed on the terminal, whether display console or typewriter, every time an editing or traveling function is used. The default size of the display window is determined by the version of FRESS being used, which is determined by the type of terminal (see Section 4.3). On a typewriter-like terminal, the default is 1 line of 50 characters; on a small display console it is 1 line of 70 characters. The user may change the size of the window by using the Set Display command to specify the number of lines and the line length.

The only restriction on the size of the display window is that the line length times the number of lines must, of course, be less than the editing buffer, and, in fact, less than the size of a third and intermediate unit, the "display buffer". The display buffer is the maximum number of characters which may be displayed on the terminal using the Print command. This is

500 characters in the version of FRESS used on typewriter-like terminals, 700 in the version for small display consoles such as Asciscopes or Hazeltines, and the full 2100 character editing buffer in the version for large displays (see Section 4.3). The size of the display buffer depends exclusively on the version of FRESS being used, and may not be changed by the user. Summarizing these three file subsections in Figure 4, the display window is a user-adjustable subset of the display buffer, which is fixed in size and a subset of the editing buffer of 2100 characters. Typically the user employs the default value of the display window and therefore need deal only with the editing buffer for all practical purposes.

The starting points of the editing and display buffers and the display window always coincide and may be positioned by the user anywhere in the text by "traveling" (e.g., scrolling, which means explicitly moving "n" lines forward or backward, or pattern scanning) while browsing and editing in the text. The editing buffer size limits the amount of text which may be affected in one editing operation. To affect an unlimited amount of text, "deferred LP's" and a combination of scrolling and locating can be used to re-position the editing buffer (See Section 3.8).

The end result of having an editing buffer is that, unlike line editors, one may edit "below" the line viewed, anywhere within the 2100 character editing buffer. Consider a two line display window of 40 characters per line on a 2741 terminal (Fig. 4).

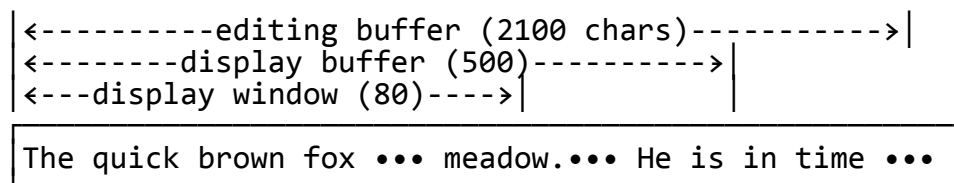


Figure 4: Symbolic Display Window

On the console this appears simply as in Figure 5.

```
L1: The quick brown fox jumped over the
L2: lazy dog asleep in the meadow.
```

Figure 5: Display Window on the Terminal

February 2, 1975

FRESS AS A STREAM EDITOR

The command "DELETE/He is in time" will delete that string even though the end points of that string are not in view. Furthermore, the display will move down in the file to one word in front of the edited portion of the text to provide the user with the context of the edit. This automatic movement in the file is called Transcription Mode, and is described in Section 4.1.

During the transcription process from hard copy, then, each edit moves the display window, display buffer and editing buffer forward. Successive edits are typically within an "editing buffer's worth" of 2100 characters of each other. Allowing editing operations in the entire editing buffer rather than just on the "current line" thus obviates most pattern scanning and scrolling so common to other text editors. Note that edits can be performed only below the start of the editing buffer and display window; to edit above, the user must explicitly pattern scan or scroll backwards (or use a negative line count; see Section 3.14).

2.2.4 LIMITS ON SCANNING

There are two types of character scanning in the text: context scanning and pattern scanning.

1) Context scanning is used by the system to find LP's specified by context in editing commands. In this case, the entire editing buffer of 2100 characters is searched for a specified LP pattern.

2) Pattern scanning is specified explicitly by the user with a Locate command to position the display at a particular location in the text. Up to 8000 characters are searched for the pattern unless the "long" qualifier is specified (see the Locate command), in which case the entire file will be searched.

3 COMMAND SPECIFICATION AND EXAMPLES

3.1 INPUT MODE VS. COMMAND MODE

When the user enters the FRESS environment, he is automatically placed in Command Mode. In this mode, the FRESS system expects him to type a command, e.g., "Insert", "Delete", "Substitute", "End". Most of the rest of this manual is devoted to a description of the use of Command Mode.

If the user wishes to insert a large amount of text in his file, he will not want to use the Insert command, since this allows him to add only one line of text at a time. Instead he should enter Input Mode. In this mode, all characters typed will be entered literally into the file. Then if the user types what he thinks are commands, they will be ignored and placed into the file as literal text. Thus the user should take care that he is in command mode when attempting to execute FRESS commands. Section 3.11 provides details on how to enter and leave Input Mode.

3.2 SUMMARY OF CAPABILITIES

FRESS provides a number of unique capabilities (discussed in detail in following sections) in addition to the normal text manipulation commands. Included among these are the following:

- 1) 'HOUSE FUNCTIONS' (Section 8.7) give the user some measure of control over how the commands are interpreted, causing the system to perform special functions or handle normal FRESS commands in special ways (e.g., repeating a previously specified command).
- 2) IMPLIED INSERTS (Section 3.11) allow the insertion of text without the explicit specification of an insert Location Pointer.
- 3) A LOCATION POINTER (abbreviated LP) in the text at which an edit is to take place may be specified in a variety of ways. On a specially equipped display console one may use a Light Pen. On a typewriter (or display console) one may use either a context pattern, the special LP separator character (|) accompanied by line and character

February 2, 1975

CLASSES

displacement specifications, or a combination of the two (Section 3.14).

- 4) DEFERRED Location Pointers (Section 3.8) can be specified, allowing the user to travel through his file before completing the specification of a command (e.g., Move) which he has initiated. This facility allows strings of text of arbitrary size to be edited in one command, irrespective of the size of the editing buffer.
- 5) Many commands may be QUALIFIED (Section 3.12) to indicate that the command applies specifically to a character, word or displayed line.
- 6) The default TRANSCRIPTION MODE (Section 4.1) causes the display to follow the user's editing operations through the text, obviating explicit pattern searches for the strings to be edited.
- 7) The Revert command allows the user to undo the effects of the most recent editing change (Section 8.6). This feature is especially valuable to a mistake-prone novice. Furthermore, the "saving" mechanism in FRESS automatically copies the next to the last editing change onto the archival copy on disk. This means that only the user's last change is lost if the computer system should "crash" (unexpectedly cease operation) or the user's terminal be dropped from the system (due to a break in the telephone connection), making it unnecessary to have the user do an explicit "save" every 10 minutes or so as in other systems.
- 9) Formatting ("typesetting") codes (Section 5) for eventual right justified and paginated hard-copy are inserted and edited as normal text. The common format codes are handled both online and offline on the line printer: paragraph (-P-), skip 1 line (-S1-), indent 5 (-I5-), etc. A macro capability (Section 5.6) is available for letting the user build more sophisticated codes.

3.3 COMMAND CLASSES

There are four main classes of commands in FRESS:

- 1) File manipulation: creating; opening; closing; copying;

- 2) Traveling: bi-directional scrolling and pattern scans (linear access); random accessing with labels; backtracking from such random accesses with Return;
- 3) Editing: Insert; Delete; Substitute; Move (rearrange destructively); Copy (non-destructively);
- 4) Structuring: Make Label, to aid in identifying and gathering related pieces of text.

3.4 COMMAND LANGUAGE INTERPRETER

The user communicates with FRESS via the Command Language INTERPreter (referred to in this manual as CLINTERP). CLINTERP decodes the commands specified by the user, calls the appropriate editing or traveling routines, and, in general, provides the user with feedback so he can observe the effects of his commands. It has been designed to assist the user in executing FRESS functions as easily and naturally as typewriter terminals allow.

Usually, the source of most command language errors is immediately obvious to the user and typing an error message at the terminal would waste time. Therefore, CLINTERP responds to an invalid command with a question mark, positioned under the start of the portion of the command line at which the error was detected. If an explanation of the error is desired, the user may type back a question mark himself and a message indicating the source of the error will be typed. The most frequent errors are mistyping context strings ("CONTEXT PATTERN NOT FOUND BY SCANNER"), misspelling command names, and specifying the wrong number of input values. A complete list of error messages will be found in Section 11.2.2.

3.5 DEFINITION OF COMMAND LANGUAGE TERMS

In general, a command consists of a mnemonic (abbreviated) command name followed by an optional blank, and a list of values (parameters) separated by user chosen delimiters ("/", ",", "\$", etc., or blank) and terminated by a carriage return (i.e., NOT the delimiter, as in some line editors - see Section 3.10).

In the examples and explanations below, capital letters (used to highlight command names) and special characters indicate characters the user actually types; words in lower case and contained in angle brackets (< and >) indicate generic names

February 2, 1975

DEFINITION OF COMMAND LANGUAGE TERMS

("types" of variables to be filled in by the user) such as <scope> and <text> which represent any text string the user can type in (see Example 1). All input lines in these examples end in carriage return characters, but these characters are not indicated visually in the examples, except for setting the input line off by itself in the text. To emphasize the presence of a single blank, the "Ø" symbol may be used.

Parameters are of several types. The following is a partial list; see Section 8.2 for a complete list.

1) a location in the file at which an editing operation is to take place (e.g., the point after which text is to be inserted). A pattern which specifies such a location is called a location pointer (<lp>). The location is the last character of the pattern unless the special LP character is used (see Section 3.15). Note the functional similarity of a Location Pointer specified by a pattern and a Light Pen detect: both are alternatives for picking a location character in the text. (The <lp> pattern must be either a literal character string or character string with an embedded ellipsis. Only one ellipsis may be embedded in a character string.)

2) an amount of a file (called a <scope>) which is to be operated on (e.g., deleted). The amount may be specified literally as a pattern or indirectly as a pair of <lp>'s identifying the begin and end points of the amount. One or both of these <lp>'s can be "deferred" (i.e., specified at a later time; see example 4). Note that the <lp>'s of a <scope> must be specified in the order in which they appear as one travels sequentially through the file from start (top) to end (bottom).

3) a literal input string (<text>) supplied, e.g., for an insert or substitute operation.

The position of a pattern in the parameter list for a given command determines whether the pattern is taken as an input string, or is used as an amount, or is interpreted as indicating a Location Pointer. Furthermore, if a string such as "fox" is used as an input string or an amount, it is taken literally, while as an <lp>, it specifies the position in the file coincident with the last character, here the "x". Similarly, "The...fox" indicates the string starting with "The" and ending with "fox" if it specifies an amount, or the location matching "x" if it specifies an <lp>. Examples below will make these distinctions clearer. For a complete description of the specification of commands, see Section 8.3.

3.6 UPPER AND LOWER CASE

Note: This section is applicable only when using FRESS on an upper and lower case terminal. For the differences when using other terminals, see Section 4.3.

In (<lp>) context specification, "character case" (upper or lower case) is taken into account. "Here" and "here" are thus distinct and it is possible to key on capital letters.

In the locate command, however, character case is disregarded when searching for context strings, so one cannot key on capital letters. This default was chosen based on user preference: users rarely are interested in character case if they are looking for a phrase in the text at which to start viewing or editing. Furthermore, the default can be overridden by qualifying the pattern with the "mixed" modifier "m". Thus,

lm pattern

would find the word "pattern" but not "Pattern".

3.7 SOME SIMPLE EXAMPLES

Example 1

DE/<scope>
e.g.: DE/The q...fox

deletes the first occurrence of a phrase which starts with "The q" and ends with "fox".

In this example, the user wishes to Delete a certain amount of text, the <scope>, from his file. The <scope> may contain any character, but some special "control" characters such as "=" and "?" need to be coded in special ways. (These codings are explained in Section 3.9.) The <scope> may even contain the "/" character: Delete takes only one "input value" (parameter) which, in this example, is delimited by the first "/" and by the carriage return. Thus any additional "/" characters in <scope> are taken literally since they cannot delimit additional parameters. For legibility, the user may put a single blank between the function name and the "/" delimiter. Thus he could have typed: "DE /amount".

February 2, 1975

LOCATION POINTERS

Example 2

S,<scope>,<text>
e.g.: S,quick,slow

This example differs from the previous one in one important aspect, namely that two input values are specified. The user requests the substitution of <text> for the amount of his file matched by <scope>. The two commas shown here have delimiting roles similar to the "/" character of the first example. <scope> may contain any character other than comma (and, as before, some special characters require special coding). In particular, <scope> cannot contain a comma because 1) the user chose to mark the beginning of the first input value (the <scope>) with a comma which means that 2) the first input value ends and the second input value begins at the second comma in the input line.

If the user, by mistake, does put a comma in, the amount would be shorter than the user intended and the remainder of the (intended) pattern would be part of the (actual) <text>. Likewise the "," which the user intended to be the second delimiter would be part of the (actual) <text>. For example, if the user specified

S,the quick, brown fox, the animal

"the quick" would be interpreted as the <scope>, and " brown fox, the animal" as the <text>.

3.8 LOCATION POINTERS AND THEIR DEFERRALExample 3

I/<lp>/<text>
e.g.: I/fox./ØFurthermore,

In this example the user requests that <text> be Inserted in his file after the location specified by <lp>. In other words, the first parameter required by Insert is an <lp> specifying where the input text is to be inserted. If the user is not sure yet where he wishes to insert <text>, he may defer specifying the location by typing

I/?/<text>

The system will respond with a "WAITING" reminder indicating that the location is still to be specified. When he has decided where to put the insertion (perhaps after some traveling in the file to find the location), the user resolves the deferred point by typing:

?/<lp>

and the completed Insert operation is then performed.

Example 4

In deferring the Insert operation (in the previous example), the user left unspecified (pending, deferred) the location in his file (i.e., the <lp>) at which the operation should take place. It is also possible for the user to defer specification of a <scope> when he specifies an operation which deals with such strings of text. The <scope> is specified either directly by a literal character string or indirectly by a pair of <lp>'s. Move, Delete, and Substitute are examples of operations which can take scopes as parameters.

If the user wishes to request a deletion but to specify nothing (as yet) about the amount to be deleted, he types

DE/?

The system will respond with a "WAITING" reminder to indicate a function with a pending <scope> remains to be resolved. The user would then resolve each of the two <lp>'s used to specify the scope, as in Example 3 above. The <lp>'s must be resolved in sequential order, as they appear in the file. Thus the user would travel to the start of the string he wished to delete, for example "The quick ... " and type

?/The

to indicate the first <lp> of the scope. The system responds with the "WAITING ISCOPE" reminder to indicate that an incomplete <scope> rather than an incomplete <lp> has been specified. The user would then travel to the end of the string to be deleted, for example "... in the meadow." and type

?/w.

Note that because the <lp>'s are actually specifying a <scope>, the character indicated by the first <lp> is actually the first character of the <lp> pattern, as in the specification of the first character of a <scope>; the character indicated by the second <lp> is the last character of the <lp> pattern, as in the specification of the last character of a <scope> and in standard <lp>'s.

A more common case is for the user to defer specifying one end of the <scope>; this is done using the equal sign control character as a separator:

February 2, 1975

LOCATION POINTERS

DE/<lp>=?

The user knows he wants the deletion to begin at the location in his file specified by the first character of the <lp> but has not decided where the deletion should end. For example, to delete a segment of initially unknown length beginning at the top of the current paragraph on this page, the user would type

DE/The user knows=?

The system responds with the "WAITING ISCOPE" reminder to indicate that an incomplete <scope> rather than an incomplete <lp> has been specified. To specify the opposite, namely giving the end of the <scope> (including the last character of <lp>) without specifying the beginning, the user types:

DE/?=<lp>

For example, to delete a segment of unknown length which ends at the bottom of the next paragraph on this page, the user would type

DE/?=resolved.

The user may do any travel function while both endpoints of a scope or an <lp> are deferred. All other commands are saved until all the deferred <lp>'s are resolved.

Example 5

Deferred <lp>'s are particularly useful with the Move command. This command is specified as

MOVE/<scope>/<lp>.

The following sequence will illustrate a typical rearrangement with multiple commands specified in between. It illustrates the method which might be used to move the text from the previous section, 3.7, after the current section, 3.8.

```

User:    M /Example 1=?/?
System:  WAITING ISCOPE
User:    20
System:  commas shown here have delimiting roles similar to
the
        WAITING ISCOPE
User:    L /animal"
System:  animal" as the <text>.
        WAITING ISCOPE

```

The move is specified with deferred scope (second endpoint) and deferred insert <lp>. Then two linear traveling functions are specified to locate the first deferral point to be completed - a scroll of 20 lines, which moves the buffer to an intermediate line with the insert point still "below", followed by a Locate. Three reminders are issued by FRESS, one after each of the three commands is interpreted (Move, Scroll, Locate). Next, another command sequence is specified to complete the <scope> (for the use of labels, see Section 6.1):

```
User:  ? /.
System: WAITING
User:  GL /example 5
System: %L(example 5)
        WAITING
User:  L /capability
System: capability is seldom required.
        WAITING
```

The first command completes the <scope> (the "." is at the end of the string 'the animal' as the <text>.); the entire string to be moved has now been specified. Next the user locates the general area to which the string is to be moved with a Get Label command. The Locate command pins down the exact place. Again, three reminders are returned by the system, one for each of these three functions. Because the effect of the first of the three commands was to complete the <scope>, only an ordinary <lp> is still pending and the reminder changes from waiting on an incomplete <scope> to waiting. Note that WAITING thus covers both an entire incomplete <scope> and an unspecified <lp>.

Now, the final command:

```
? /required.
```

serves to fill in the <lp>. The Move command, whose specification is completed by resolving the last <lp>, will now be executed.

Any editing operations specified while a previous edit is as yet incomplete will be similarly held until the incomplete one is finished; all commands are then executed in the specified order. While multiple commands can be deferred, this capability is seldom required.

February 2, 1975

3.9 REPRESENTING CONTROL CHARACTERSExample 6

It was mentioned above that, in order to avoid ambiguity, certain characters must be coded in special ways. Two of them are "=" and "?", since they each have a control function as illustrated above.

When the user wishes to represent a literal "=" in any parameter in Command Mode, he must use a pair of them. For example,

$$S/A==B/A+B$$

Here, the user wishes to substitute "A+B" for "A=B".

Example 7

When the user wishes to represent a literal question mark, nothing special need be done if the input value is neither an <lp> nor a <scope> specification. Even in an <lp> or a <scope> specification, nothing special need be done as long as there are characters other than question marks in the pattern specified (except if the question mark is preceded by an equal sign, see Example 6 above). If the specified input value contains only question marks, the user must specify the pattern with one extra question mark. Thus:

$$S/?\text{He};\text{h}$$

changes what is evidently two sentences into a single sentence.

Also

$$S/?./?$$

removes a period following a question mark. This could not be specified as

$$S/??./?$$

Since the <scope> pattern does not contain only question marks, the questions marks would be interpreted literally and the <scope> of this command would be "??" However:

$$S/???/?$$

replaces two question marks by a single one; the intended pattern consisted of only question marks, and the user added a third in order to encode the pattern properly. Next consider

S/?/?

This command defers specification of the <scope> to be replaced and specifies ? as the literal text replacing it. Note that literal input may not be deferred.

Other control character rules are discussed in Section 3.16.

3.10 THE KEY DELIMITER AND ITS USE

The user chosen character which delimits the beginning of parameters in a command is called the Key Delimiter. (In most of the preceding examples the Key Delimiter has been "/" but a blank or almost any other special character can be used. The exceptions are ">" and "-" which have special meanings as explained below. In some cases, an ordinary character can also be used (see below).) The Key Delimiter may thus be varied from command to command, and even within a single line of commands if more than one command is specified. (Particularly for beginners, it is recommended that the standard form in example 2 below be used in order to avoid confusion. The other forms exist to minimize keystrokes for the experienced user.)

CLINTERP examines the character immediately to the right of the function name (and to the right of the qualifier if one is present, e.g., "-w" in examples 6 and 7 below (see Section 3.12)). If this character is ">" (used for specifying multiple commands on one command line - see Section 3.13) or a carriage return character, then there are no input values for the function. Otherwise, if it is not a blank (examples 1, 6, 7), it is the Key Delimiter for this function (and therefore marks the beginning of the first input value). If it is a blank (examples 2, 3, 4, 5), the next character is examined. If this next character is also a blank (example 2), or a "special" character other than "%", ">" (example 2), or carriage return, then this character after the blank is the Key Delimiter. Otherwise, the blank (which was passed over a moment ago) is taken as the Key Delimiter (examples 4, 5).

February 2, 1975

KEY DELIMITER

Examples:

Input Line Reads:	Key Delimiter is:
1) functionname/Sam...	/
2) functionname /Sam...	/
3) functionname Sam	blank (since 2 blanks are present)
4) functionname Sam...	blank
5) functionname 35...	blank
6) functionname-W/Sam...	/
7) functionname-WORD...	The letter O (not standard delimiter)

Note 1: To specify an edit on a string which starts with a special character, for instance to delete a ".", one must use something other than a blank as the key delimiter: "de /." since "de ." would treat "." as a delimiter, not a literal character. Similarly to fill in a deferred <lp> after the end of a sentence, one must specify "?/.", not "? ."

Note 2: If more delimiters appear in a command line than are required to separate the parameters needed for that command, the extras will be taken literally as part of the last parameter:

de /text string/

will try to delete the pattern "text string/" which will probably result in "CONTEXT PATTERN NOT FOUND". This facility of treating "extra" delimiters literally is very handy for being able to specify them without special rules, as shown in example 8.

Note 3: Most commands can be typed using blank as the delimiter:

S Sam Charles

Example 8

As an additional aid to saving keystrokes, if blank is the key delimiter any blanks in the last parameter will be taken literally as long as the first parameter(s) contain(s) no blanks. Thus, for an insert:

I Here. The quick brown fox

CLINTERP will treat the first two blanks as delimiters but take the remaining blanks literally. Because of this facility the user should be careful not to put in any "extra" blanks at the end of his command line before he does a carriage return since they will be treated as literal blanks. Additionally, if the user prefers using blanks as his parameter delimiter, he may use the underscore

character ("_") to denote literal blanks and not key delimiter blanks:

`I_The_quick_brown_fox_jumped`

The " " will be converted to a literal blank, the first two blanks will be interpreted as delimiters, and "brown_fox_jumped" will be inserted after "The_quick".

3.11 INPUT MODE AND THE IMPLIED INSERT POINTER

To enter Input Mode, the user merely uses slightly different forms of Insert or Substitute or one of the special commands Top Input, Bottom Input, or Swift Input (see below).

While in Input Mode, the system automatically inserts a blank at the end of each line typed since it is assumed that the user breaks lines at word boundaries. Note that in Input Mode \$ and @ have their usual CP/CMS operating system dependent "line erase" and "character erase" functions, while >, ..., ?, _, & are taken literally. A literal % may be entered as "!/", and a literal exclamation point by doubling it, etc. (see Section 3.16).

Example 9

User: MF draft
System: INPUT

In this first method, the user creates a new file named "draft" and the system puts him in Input Mode, as its response indicates. The first line of text typed is placed after the start of the file (denoted by "*START OF TEXT AREA*"). Subsequent lines will be inserted in order after the first, until a line of zero length is typed.

Example 10

User: TI
System: *START OF TEXT AREA*
INPUT

Top Input causes the system to enter Input Mode at the top of the file, i.e., after the *START OF TEXT AREA* line.

February 2, 1975

INPUT MODE

Example 11

```
User:    BI
System:  word.
         INPUT
```

Bottom Input causes the system to enter Input Mode at the bottom of the file, i.e., after the last character before the *END OF TEXT AREA* line. The last word in the file is printed to give a small amount of context.

Example 12

The most common method for entering Input Mode is to specify an Insert or Substitute with the text to be inserted left null. The input is positioned not at the implied insert point but at the <lp> indicated. Again, subsequent lines of typed input will be inserted, until a null line is typed.

```
User:    I/location/
System:  the location
         INPUT
```

Since no text is specified between the second "/" and the carriage return, the system enters Input Mode. The first subsequent line of text typed in is placed immediately after "location", that is, with no intervening blanks.

Example 13

```
System:  the display buffer is fixed in size
User :    S/buffer/
System:  display is fixed in size
         INPUT
```

The amount of the file specified by "pattern" is deleted and Input Mode is entered. Subsequent typed input is inserted where the <scope> was deleted.

Example 14

Another method of entering Input Mode is to specify Insert with no parameters. Thereafter, all subsequent lines of typed input will be inserted into the file until a line of zero length (the "null" line, i.e., a carriage return only) is typed after which he is returned to Command Mode. The location of the insertion, called the "implied insert point", is the last position where text was added, or was intended to be added, to the file by one of the following commands: Bottom Input, Insert, Make File, Make Label, Substitute, Swift Input, Swift Input Bottom, Swift

Input Top, Top Input. Note that the user may not always remember where in the file his last edit function was executed; for safety's sake the form of input of example 12 may be used. If no implied insert point exists, Input Mode will be entered, but an error message (IMPLIED LP DOES NOT EXIST; LP IS DEFERRED: WAITING) will be printed when the first line of text is typed. At this point the user must either resolve the <lp> or use the &Clear command to clear the deferred pointer. In either case, the user will be back in Command Mode.

Example 15

There is a second form of Input Mode which is both faster and less expensive than that described above. This is called Swift Input Mode. Rather than inserting the text typed one line at a time, it gathers multiple lines before inserting them in the file. When this "insert" is done, FRESS types a blank line at the terminal.

The disadvantages of this form of Input Mode are that format codes (see Section 5) are not checked for validity, and system crashes could cause more text to be lost since saves are done less frequently.

Swift Input can be entered at the top or bottom of the file, or at any location in the file other than inside a piece of structure (e.g., a label). There are three forms of the command, depending upon which option is desired. To enter Swift Input Mode at the top of the file, the user should use the Swift Input Top command:

SIT

To enter at the bottom of the file, the command is Swift Input Bottom:

SIB

If the user desires to specify an <lp> as the insert point, he should use the command

SI <lp>

As in the case of normal Input Mode, the implied insert point may be used by specifying:

SI or SI/

The following table summarizes the relationships between Input Mode and Command Mode and the methods of moving from one to the other:

February 2, 1975

QUALIFIERS

<u>To Go From:</u>	<u>To:</u>	<u>Action:</u>
Input Mode	Command Mode	Type null line
Command Mode	Input Mode	Use:
		(1) Insert or Substitute and leave <text> null
		(2) Top Input or Bottom Input
		(3) Make File to start a new file
		(4) Swift Input, Swift Input Top, or Swift Input Bottom to enter Swift Input Mode

3.12 QUALIFIERS+

A "qualifier" is a keystroke saving way of specifying the amount of text to be affected by a command. A qualifier is specified by typing a hyphen (minus sign) and the qualifier code immediately after the command name, i.e., with no intervening blanks. Any command which has a <scope> parameter may be specified, instead, with a qualifier. When using the line qualifier the <scope> parameter is then omitted from the command line. When using any other qualifier, the <scope> is then specified by an <lp>.

The list of possible qualifiers is:

```
-c    character
-w    word
-l    line
-o    order
```

The character qualifier allows the user to specify an edit on a single character while still being able to give a longer text string to uniquely identify the character. For example, if the string

special blanks

appeared at the top of an editing buffer and the user wished to delete the 's' at the end of the word 'blanks', he could not specify

d/s

since this would result in

pecial blanks

Instead, he can type

d-c/ks

which will delete the correct s.

The word qualifier allows the specification of any part of a word to indicate an edit on the whole word. Thus, to change the word 'sufficient' to 'adequate', the user need only type

s-w/su/adequate

When used with the word qualifier, most of the commands containing <scope> parameters affect the blank before the word as well as the word itself. The exceptions to this are Capitalize, Substitute, and Uncapitalize.

The line qualifier specifies that the edit should apply to the first line of the online display and is especially useful when the text lines are cluttered with formatting and/or structure codes and it is difficult to indicate exactly what is wanted as the <lp>.

The order qualifier allows the user to specify an edit on an entire piece of structure (e.g., a label) by specifying any character in the structure. For example, to delete the label "%L(section)" the user need only specify:

d-o/s

3.13 COMBINING COMMANDS ON A SINGLE INPUT LINE+

Example 16

If the user wishes, he may place more than one command on an input line. This facility may also be used in conjunction with the "&&G" repeat facility (Section 8.7). To stack commands, he separates them by ">" characters:

SUB/amount1/text>DE,amount2>-5

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

The ">" terminates the command preceding it. Thus it acts like the carriage return character at the end of a line which contains only a single command.

February 2, 1975

LP DISPLACEMENTS

Note that all commands on a physical command line are executed on the same editing buffer even if transcription mode (see Section 4.1) is in effect. This means that the user does not have to worry about the display moving down to his edits and he therefore can specify them independently of position in the editing buffer. This means that

S /old pattern/new pattern>S amount text

will look for "amount" in the current editing buffer, not in the editing buffer starting at "new pattern".

Sometimes, however, it is desired to stack commands so that each command is fully executed, including moving the editing buffer, before the next command is handled. To do this, the CMS linend character (defaulted to "#") should be used instead of the FRESS command separator ">". As an example, if the user desired to specify three edits on one command line, but the context pattern in the third is more than 2100 characters from the start of the current editing buffer, he could use the CMS linend character to separate them. Then, the search for the context pattern in the third edit would be done in the 2100 character editing buffer following the context pattern from the second edit, where it might be found.

A pair of ">" characters is required to represent a single literal occurrence anywhere in Command Mode.

I/location/IF A>>B THEN

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

The user should take care to type all commands correctly when stacking them using the FRESS command separator. If any of the stacked commands cause errors in the command language interpreter, the whole command line will be ignored.

3.14 LINE AND CHARACTER DISPLACEMENT NUMBERS+

In addition to or in place of a context pattern, the user may specify line and/or character displacement numbers (optionally signed) when an <lp> or <scope> is required. This is particularly useful when the user is working on a display console which does not have a light pen facility (like the IBM 2260) or if there are two similar patterns in the display window, the second of which must be indicated. The numbers must be contiguous (i.e., intervening blanks are not allowed) and must be followed by the LP separator

(|) whether or not a context pattern follows. The LP separator notifies FRESS that line and character displacement numbers are being used. The first displayed line of the editing buffer is numbered zero. Thus, the line number specification is "equivalent" to a scroll of that amount before beginning context scanning. The first character of each line is numbered zero. Thus, a character displacement number means to move that number of characters into the current line or back into the previous line (after a scroll of the line number) before context scanning. If only one number is specified, it is assumed to be a line number. If a character number is present, it must be separated from the line number by either a plus or a minus sign.

Example 17

DEØ2|amount

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

Example 18

S/-1+10|amount/text

In this example, the user wishes to Substitute "text" for "amount" which begins somewhere after the tenth character in the line above the top of the display.

Example 19

DE/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

DE-w/2+2|

3.15 THE "&" LP CHARACTER+

The <lp> determined by a context pattern is the last character in the string unless the special LP character (&) is included in the pattern. If an ampersand is included, the character before the ampersand is used as the <lp>, no matter how long the context

February 2, 1975

CONTROL CHARACTERS

string extends beyond it. Thus "f&ather" LP's the f, not the r. To illustrate the use of this facility, suppose a character occurs twice in a line, with identical context preceding it in both cases, but different context after. It is possible to LP the proper one by giving merely the desired character, followed by an ampersand, followed by enough of the trailing context to uniquely identify the character.

A <scope> may also be specified within one context string by using two ampersands to identify the two points ("f&at&her" produces "fat" as the <scope>).

3.16 SUMMARY OF CONTROL CHARACTERS

Because of the use of standard keyboards on terminals, all editing systems are forced to use special characters as control characters to indicate special command language functions and format codes. The table below summarizes the definitions and conventions for representing literally such control characters in files, specifying them in command lines or in lines in Input Mode. Literal representations are given only for cases in which the character must be coded specially, typically only in Command Mode. Note that all special characters may be represented by format codes (see Section 5.8).

CHAR	NAME AND EXPLANATION
------	----------------------

any	key delimiter
-----	---------------

DEFINITION: see Section 3.10

LITERAL: in the parameters of a command, it can be used literally only in the last parameter; hence a different key delimiter should be used if this character is needed in previous parameters

>	command separator
---	-------------------

DEFINITION: see Section 3.13

LITERAL: in command lines only, 2n symbols represent n symbols

- qualifier separator (hyphen character)
 DEFINITION: see Section 3.12
 LITERAL: no restrictions

- _ blank fill (underscore character)
 DEFINITION: if the key delimiter in a FRESS command
 is a blank, all underscores are
 interpreted as literal blanks
 LITERAL: a non-blank key delimiter must be used

- = defer separator
 DEFINITION: see Section 3.8
 LITERAL: in command lines only, 2n symbols
 represent n symbols

- ? defer character
 DEFINITION: see Section 3.8
 LITERAL: in pattern parameters of command lines
 containing only ?'s, n+1 symbols
 represent n symbols

- ... ellipsis
 DEFINITION: see Section 2.2.3
 LITERAL: in pattern parameters of command lines
 only, n+1 periods represent n periods for
 n>2

- & LP character
 DEFINITION: see Section 3.15
 LITERAL: in pattern parameters of command lines
 only, 3n symbols represent n symbols

- | LP separator
 DEFINITION: see Section 3.14
 LITERAL: in pattern parameters of command lines
 only if it follows one or two signed
 numbers at the beginning of the

February 2, 1975

CONTROL CHARACTERS

parameter, n+1 symbols represent n symbols

! format code delimiter

DEFINITION: see Section 5.2

LITERAL: !! (see Section 5.8)

- special blank

DEFINITION: see Section 5.10

LITERAL: !- (see Section 5.8)

% structure code delimiter

DEFINITION: see Section 6

LITERAL: !/ (see Section 5.8)

% capitalize character

DEFINITION: in a <text> parameter (Substitute, Insert, Make Label 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

LITERAL: !/ (see Section 5.8)

% logical hyphen

DEFINITION: as the last character of a line in Input Mode, it prevents a word space from being inserted (and is not inserted itself)

LITERAL: !/ (see Section 5.8)

%% begin/end multicap

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 Offline Read or Swift Input Mode, then the input is capitalized until another %%

is encountered on another line or until
Input Mode is exited
LITERAL: !/!/ (see Section 5.8)

@ logical backspace, character delete

DEFINITION: in all user typed lines, n @ signs
"erase" the previous n non-@ sign
characters
LITERAL: !124 (see Section 5.8)

¢ line delete

DEFINITION: in all user typed lines, it "erases" all
characters preceding it on the current
line
LITERAL: !74 (see Section 5.8)

CMS linend

DEFINITION: equivalent to a carriage return for
stacking multiple commands on a user
typed line
LITERAL: !123 (see Section 5.8)

<backspace> character

DEFINITION: Entered via the backspace key on a Datel
terminal, or control-h on a TTY.
LITERAL: Always a literal character, but 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 a Fullprint (see
Section 8.6), and should normally be
entered in this context.

February 2, 1975

TEMPORARY DISPLAY MODES

4 DISPLAY MODES AND TERMINAL CHARACTERISTICS

Optional viewing modes have been incorporated into FRESS to provide concise, meaningful feedback and to allow suppression of redundant display.

4.1 DISPLAY MODES

There are several different displays which the user may elect for verification of an editing operation. The choices are display or brief mode and transcription or static mode.

DISPLAY MODE, the default setting, displays the contents of the display window (see Section 2.2) after every editing and traveling operation. If commands are stacked on a single line, multiple displays will be generated. BRIEF MODE suppresses all printing to the terminal.

TRANSCRIPTION MODE (default on typewriter terminals) 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 along as he edits. 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 (default on 2260's and the IMLAC) 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.

4.2 TEMPORARILY OVERRIDING DISPLAY MODES+

The Set Mode command establishes or changes the above viewing modes globally. To allow the user to control the modes for individual command lines, the following combinations may follow any command input line and affect the entire command line. (Note that multiple individual commands on a physical command line are separated by ">"). To distinguish them from a literal period as a

DISPLAY MODES AND TERMINAL CHARACTERISTICS

TERMINAL CHARACTERISTICS

February 2, 1975

parameter, these command modifiers must be preceded by exactly two blanks.

SYMBOL	MEANING
.	"brief mode": suppress print for all commands on the physical command line
.*	No print until last command on command line
.D	Use DISPLAY mode for all commands on this command line
.T	Use TRANSCRIPTION MODE for all commands on this command line
.S	Use STATIC mode for all commands on this command line

Individual commands in a multi-command line may be put in brief mode by preceding them with a "<" sign. The display can be suppressed completely by preceding a command with "(". The display will be suppressed until a ")" precedes a command.

4.3 TERMINAL CHARACTERISTICS

In order to allow FRESS to be used from different types of terminals, several versions of the system have been developed. When FRESS is entered from CMS the user types

FRESS <option>

where the specified <option> indicates which version of FRESS is desired. The following table lists the available versions and the types of terminals with which they are used. To invoke the "normal" version, the <option> is omitted.

<u><option></u>	<u>Terminals</u>
Normal	IBM 2741, Datel
T	Asciscop, Teleray 3311
A	IBM Communicating MCST, IBM 2741 with correspondence code
H	Hazeltine 1000
M	Brown University Graphics System - Vector General
I	Brown University's Imlac PDS-1D

The "normal" version is the one described throughout this manual. The special characteristics of the 'T' and 'H' versions are

February 2, 1975

described in the following sections. For information about the 'M', 'I' and 'A' versions, or additional information about any of the others, contact the FRESS staff, c/o Professor van Dam, Division of Applied Mathematics, Brown University. For information about the operating system dependent differences among terminals, see Section 10.1.

4.3.1 UPPER-CASE ONLY TERMINALS

4.3.1.1 Text Display

The 'T' version is for use on terminals which display only upper-case characters. The default display window is a single 70 character line and the display buffer size is 700 characters. When displaying a file or inputting text, those letters which are meant as capital letters should be preceded by a percent sign. Multiple upper case letters ("multicap strings") should be surrounded by double percent signs. All other letters are considered to be lower case. Thus, the sentence which would appear or be inserted on an upper/lower case terminal as:

The quick brown fox

would appear or be inserted on an upper case terminal as:

%THE QUICK BROWN FOX

Similarly

This text editor is FRESS.

would appear on an upper case terminal as:

%%THIS TEXT EDITOR IS %%FRESS.%%

Numbers and punctuation (except exclamation points) may be considered upper case characters when preceded by 2 or more upper case letters. Thus the period above is part of the "multicap string", but would not be considered an upper case character in the string

%P.%T. %BARNUM

This can sometimes affect pattern scanning (see below).

Note that the version of FRESS used affects only how the file appears on the terminal. The characters appear the same in the file independently of which type of terminal is used. The percent signs do not exist in the file but are merely used as a code in displaying upper case characters.

On upper/lower case terminals, labels and format codes (edit, alter, and macro codes - see Section 5) are displayed in the case in which they are entered. On upper case terminals, however, there is no distinction made when these three 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.

4.3.1.2 Pattern Matching

Care must be taken when pattern scanning on an upper case terminal using a 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 buffer (which is the text displayed on the terminal), not in the file itself. All characters in the display buffer appear in upper case, even though some of the characters are actually lower case in the file. For example, the string

%SENATOR %MC%GOVERN

would not be found if the user specified

I/MCGOVERN/text

or

L/MCGOVERN

because of the percent sign appearing in the display buffer. It would be found by

I/MC%GOVERN/text

OR

February 2, 1975

L/MC%G

As another example, consider the string

%THIS TEXT EDITOR IS %%FRESS.%%

Because the period at the end is considered part of the "multicap string", the last word could not be found by typing

I/%%FRESS%%/text

or

L/%%FRESS%%

since it does not appear that way in the display. It would be found by typing

I/%%FRESS.%%/text

or

L/%%FRESS.%%

or by

I/%%FRESS/text

or

L/%%FRESS

In fact, it would even be found by typing

I/FRESS/text

or

L/FRESS

The scan for pattern matching in Locate commands in the forward direction starts at the second character of the display buffer. 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 buffer. Therefore, leading percent or double percent signs should be included in the pattern to ensure the Locates will find separate instances.

4.3.1.3 Literal Text

When specifying <text> parameters in commands such as Insert, Substitute, Insert Before, and Uniform Substitute, 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 not necessary, since the multicap string is automatically ended by the carriage return.

4.3.1.4 Correcting Character Case Errors

It often happens that an error is made and 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, the Flip command may be used to correct the error. If, for example, an entire file is inserted in upper case accidentally, the Flip command, specifying the whole file as its <scope>, will flip all characters to lower case except those appearing after format codes or "sentence-ending" punctuation - periods, question marks and (literal) exclamation points. Flip will ignore labels, but will force format codes to lower case.

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 Capitalize and Uncapitalize commands must be used.

4.3.1.5 "Missing" Characters

Certain "special characters" do not appear on all types of terminals. Thus a not sign (~), which does not appear on certain terminals, can be typed in the 'T' version of FRESS as a backslash, which is a "capital L" on the keyboards of some teletype-equivalent terminals. Similarly an or bar (|)

February 2, 1975

can be typed as a circumflex, which is a "capital N" on some keyboards. A backspace may be inserted via "control H". Any character which does not appear on the keyboard may be inserted literally as an exclamation point followed by the decimal number representing that character code. See Section 5.8 for a list of some of these codes and a complete explanation of how to use them.

4.3.1.6 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 a Substitute command), 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, the carriage return at 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.

4.3.2 HAZELTINE VERSION

A third version of FRESS is available for use on Hazeltine 1000 terminals. Although these terminals are upper/lower case (with the TTY/TPWR switch set to TPWR), they are also missing the special characters mentioned above, i.e., not sign, or bar, and backspace.

Special characters may be typed in the 'H' version the same way they are typed in the 'T' version. Note, however, that when used in TPWR mode, shift-L is really a capital L and shift-N a

DISPLAY MODES AND TERMINAL CHARACTERISTICS

TERMINAL CHARACTERISTICS

February 2, 1975

capital N. When the user desires to insert a backslash or circumflex, he must flip the switch to TTY mode, type the character, and then flip back to TPWR mode. To alleviate some of this problem, the '#' character may also be used to insert a not sign. Control-H is the backspace and may be used in either mode. The 'H' version also makes the "line feed" (LF) and "rubout" keys the CMS line and character delete characters, respectively. The "esc" key becomes the CMS linend character, which allows "#" to be used as a not-sign. For a more complete discussion of these CMS dependent characters, see Section 10.1. These two characters may be typed in either TPWR or TTY mode.

4.3.3 SUMMARY OF DEFAULT SETTINGS

The following table summarizes the default characteristics of the 3 most commonly used versions:

<u>Versions</u>	<u>Display window</u>	<u>Display</u> <u>bufferSpecial</u> <u>chars</u>
Normal	1 line, 50 chars	500 charsnone
'A'	1 line, 50 chars	500 chars =!
'T', 'H'	1 line, 70 chars	700 charsbackslash-
circumflex=		
ctrl-H=backspace		

February 2, 1975

FORMATTING CODE CONVENTIONS

5 FORMATTING CODES

5.1 OVERVIEW

Formatting codes allow the user to control the "layout" of his text in any desired way -- to create paragraphs, indentations, skipped lines, table of contents entries, footnotes, etc.

Format codes are entered into the user's file as part of the normal text stream and are edited as normal text. Like structure codes (labels -- Section 6.1), they are saved in-line in the data structure of the user's file. To format ordinary manuscripts for line printer or hard copy terminal output, only paragraph, skip, indent, special blank (blank fill), center, underscore and footnote need be specified since all other page layout decisions (column, width, depth, margin, head, etc.) have been defaulted by FRESS.

5.2 FORMATTING CODE CONVENTIONS

There are five general types of formatting codes within FRESS: EDIT codes, ALTER codes, MACRO codes, SPECIAL CHARACTER codes, and UNDERSCORE codes. EDIT codes have no range--they are in effect only at their unique place in the text. They are used to cause minor text formatting such as starting new paragraphs, line or pages, tabbing, indenting, and centering. ALTER codes do have scope since they remain in effect until respecified. They are used to cause more global formatting changes, such as setting tabs, changing margins, and changing the number of characters printed on each line (the width). MACRO codes let the user define his own codes using EDIT and ALTER codes. Special character codes allow the user to input system delimiters and characters not available on a standard keyboard.

The general format of a formatting code is:

!<DELIMITER><CODE><DATA><DELIMITER>

EDIT codes are delimited by dashes (-), ALTER codes by pluses (+) and MACRO codes by periods (.).

As an example, to skip two blank lines a user would specify:

!-s2-

where '-' is the delimiter, 's' is the code, and '2' is the data, while he would begin a new paragraph with

!-p-

which has no data field.

Several EDIT codes may be combined within one set of delimiters. For example, to skip four lines, indent 10 spaces and have a hanging indentation of 13 spaces, a user would specify

!-s4;i10;j13-

The ";" replaces the "-!-" string which would normally separate the edit codes. The user could further reduce the number of keystrokes required to issue this code and be able to redefine the effect of the code everywhere it occurs by entering it as a macro definition (see Section 5.6).

When a number of EDIT codes that perform the same function are concatenated (strung together), several rules are followed to determine the combined effect of the code:

- 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-).

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-).

ALTER codes, on the other hand, must appear by themselves - these codes may not be combined. To redefine the positions to which the user's tabs are set and to concurrently set the page width to 60 characters, he would have to specify two formatting codes:

!+settab1=28;4=30+!+width60+

These would set his first tab at 28 and his fourth at 30, leaving the others at their former values, and would set his page width to 60 characters.

February 2, 1975

EDIT CODES

5.3 NOTATION FOR FORMATTING CODE EXPLANATIONS

The codes in the table below are currently supported by the FRESS system. Their precise performance will become obvious after a few uses of each. The format code delimiter (!) is omitted in the descriptions.

The codes described are both those whose effect on text appears in the online display of text during editing operations and those whose effect appears in the output of the Fullprint formatting program which prints the entire file, typically produced offline on the 1403 line printer (see Fullprint command, Section 8.6). Only the simple codes are handled by online display but all are effective when a Fullprint to a 2741 terminal is done.

In the sections below the following conventions are followed for <DATA> fields:

n	numeric parameter (arbitrary number of digits)
m	additional numeric parameter where a distinction from n is necessary
x	indexed parameter - used as an index into a table
any upper case alphabetic	literal code or parameter (<u>may be specified by the user in lower case</u>)
a	a literal character
< >	optional parameters, choices are separated by blanks, and no more than one may be specified

All delimiters ("-", ";", ",", "*", etc.) are literal

5.4 EDIT CODES

The table below summarizes several properties and the default values of the codes explained in this section; it is for reference only and should be skipped on first reading. Codes handled by online display are marked with a (Y); currently, only line spacing, paragraphing, and indents of ten or less are handled.

FORMATTING CODES

EDIT CODES

February 2, 1975

Codes handled only by the Fullprint program are marked with an (N). Also indicated is whether or not the code begins a new line (X).

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

-B<n><,m>- draw box

Draws a box "n" characters wide and "m" characters deep whose upper left hand corner is positioned at the current column position on the current line. Both width and depth must be greater than one. If either parameter is unspecified its value will be the same as that of the previous box code. Note that a box may not be split over a page and hence the box code acts as a conditional page code (see !-Kn-). Consequently, if a box is preceded by a tab code and a new page is forced, the box will be left justified. The user should take care to allow for this, perhaps by adding his own conditional page format code before the box itself.

-C- center

Centers the following string of text up to the next code which begins a new line. If the original string occupies more than one line, each line, until the next code causing a new line, is centered.

February 2, 1975

EDIT CODES

-E- end heading

The -E- code ends a heading code without beginning a new line. The number of lines to be skipped after the heading which is implied by the heading code is ignored. This code is useful for attaching footnotes to heading text without having the footnote number appear in the table of contents.

-F<n>- footnote

Begins footnote text that will be set at the bottom of the current column (page) when a Fullprint is done. A second !-F- ends the footnote text. Specifying "n" resets the automatic footnote counter to "n" which is used to determine the value of the in-line footnote indicator. The value of the footnote counter appears as a superscript at the location of the starting !-F- code and at the start of the footnote. The system inserts a blank line and a horizontal line between the main text and footnote area. To insure proper formatting, footnotes should not be more than half a page long.

Note that if a careless user forgets the second !-f-, all of the text will become part of the footnote. This will probably cause a Fullprint to fail since it cannot handle footnotes of greater than a half page in length.

-Hn- heading

Causes the following string of text up to the next code which begins a new line to be formatted in accordance with the value of "n" (1 to 6) specified and the pre-defined heading format. See the !+HEAD...+ code for a description of heading formats. The text affected by the heading code appears in the table of contents in the same format as it is typed; see also !+TOFC+ code.

-In- indent

Begins a new line with an indentation of n spaces. To both skip a line and indent see the !-p- code.

-Jn- hanging indent

Specifies a hanging indentation of "n" spaces, i.e., the first line following this code is started at the current margin, while the succeeding lines are indented. No indentation occurs at the new line started by the -Jn- code unless it is concatenated to a -In- code. A hanging indentation is ended by the next format code which begins a new line.

-K<n>- conditional page

If less than "n" lines are left in the current column (page) then this code will begin a new column (page), otherwise it is ignored. This code should precede boxes, tables and grids to assure that they will not be split over a page boundary.

-N<n><C R>- new page

The !-N- code forces the following text to start a new page, while !-NC- forces a new column if multicolumn format has previously been specified. If "n" is specified the page number is reset to the value "n". Page numbering is suppressed if "n" is zero. The qualifier "R" denotes that page numbers are to be output as lower case roman numerals. (Note that multiple pages to be skipped must be specified by separate format codes with a special blank in between because of format code combination rules (see Section 5.2).)

-P- paragraph

Starts a paragraph by skipping one line and indenting five characters. These skip and indent values may be reset with the !+PARA...+ code.

-R- revision bar

This code will cause a revision bar (|) to be inserted in the margin until the next -R- code is encountered. In order to remove the revision bars the user would have to delete all !-R- codes. To avoid this, the revision bars can be entered instead as macro codes (see Section 5.6). The revision bars can then be removed merely by redefining the macro as null.

February 2, 1975

EDIT CODES

-S<n>- skip

Starts a "new line" and then skips "n" blank lines. The new line itself may cause some skipped lines if the file is double or triple spaced. For example, !-s2- will cause 3 lines to be skipped if the file is double spaced.

-T<*n x><L R C><,a ,n>- tab
-U<*n x>- justified tab

These lists indicate the positions of all the parameters available with the tab codes. All the parameters are optional, i.e., the simplest form of the tab code is !-T- which merely positions the following text at the next column position (see -T<x>-). For clarity each parameter is treated below as a separate code, although any combination of parameters is valid. Section 5.11 contains more detailed examples of the tab codes for building tables. Note that none of the tab codes start new lines.

-T<x>- indexed tab

Specifies a tab index into a one dimensional table of column positions. The text following the -T<x>- code is positioned at the column value of the xth table entry. "x" runs from one to ten. Its associated values are defaulted to every ten spaces (1=10,2=20,...,10=100), and may be re-specified using the !+SETTAB+ alter code. Thus T3 positions text to column 30.

-T<*n>- column tab

A column tab moves the following text to the absolute column position specified by "n" rather than indexing into the SETTAB table. For example, !-t*25- positions text to column 25.

-T<L R C>- modified tab

The tab indicated by this code specifies the left most (L), right most (R), or center (C) position of the text string following it. This command overrides any modifier specified with the SETTAB command.

-T<a ,n>- fill tab

This code will place the "fill character" specified by "a" or "n" into any spaces that were skipped over as a result of the tab part of the code. The fill character is either a literal character or, if "n" is specified, the decimal representation of the character (see special characters -- Section 5.8). For example, !-t1,96- indicates that "-" is to be used as the fill character. There must, however, be some kind of text (at least a special blank) following the tab.

As an example the next line was generated by a "!-T*60r,.-~" absolute column, right tab with "." as the fill character code.

.....

-U<x>- justified tab

This type of tab is used typically in tables where text must be left and right justified (see, for example, the table of example 3 in Section 5.11). The text following this code is right and left justified between the tab position specified by "x" and the right margin; or if Fullprint is in table mode (see +TABLE...+ code), the text is justified between left and right tabular column boundaries. This code can be viewed as establishing a temporary left margin at the tab position or if Fullprint is in table mode it can be viewed as establishing a temporary left margin at the tab position and a temporary right margin at the right table column boundary.

February 2, 1975

ALTER CODES

5.5 ALTER CODES

The table below summarizes several properties and the default values of the codes explained in this section; it is for reference only and should be skipped on first reading. Codes handled by online display are marked with a (Y); currently, only line spacing, paragraphing, and indents of ten or less are handled. Codes handled only by the Fullprint program are marked with an (N). Also indicated is whether or not the code begins a new line (X).

<u>LINE</u>	<u>ALTER CODE</u>	<u>MEANING</u>	<u>DEFAULT</u>	<u>NEW LINE</u>	<u>ON-</u>
	B	set mode for underscoring blanks	H		N
	COLUMN	set column mode	1	X	N
	DATE	print the date	-		N
	DEPTH	set page depth	66	X	N
	DI	set decimal label indent	3		N
	GRID	draw a grid	-		N
	GUTTER	set column separator	3	X	N
	HEAD	select heading table	B		Y
	JUST	set justification	C0;L0		N
	MARGIN	set margins	0,0,9,6	X	N
	OFFSET	set page offset	0	X	N
	PARA	set paragraph	5,1		Y
	SETTAB	set tabular columns	10*index		Y
	SPACE	set line spacing	1		N
	TABLE	set table mode	10*index by 10	X	N
	TITLE	set running titles	-		N
	TOFC	set table of contents	all headings		N
	WIDOW	set widow depth	2		N
	WIDTH	set page width	65	X	N

Figure 6 depicts the most common ALTER codes affecting page layout and their default settings (the codes are shown as they might be specified). In multicolumn form each column has identical margins (Fig. 6) within its allotted space (Fig. 7).

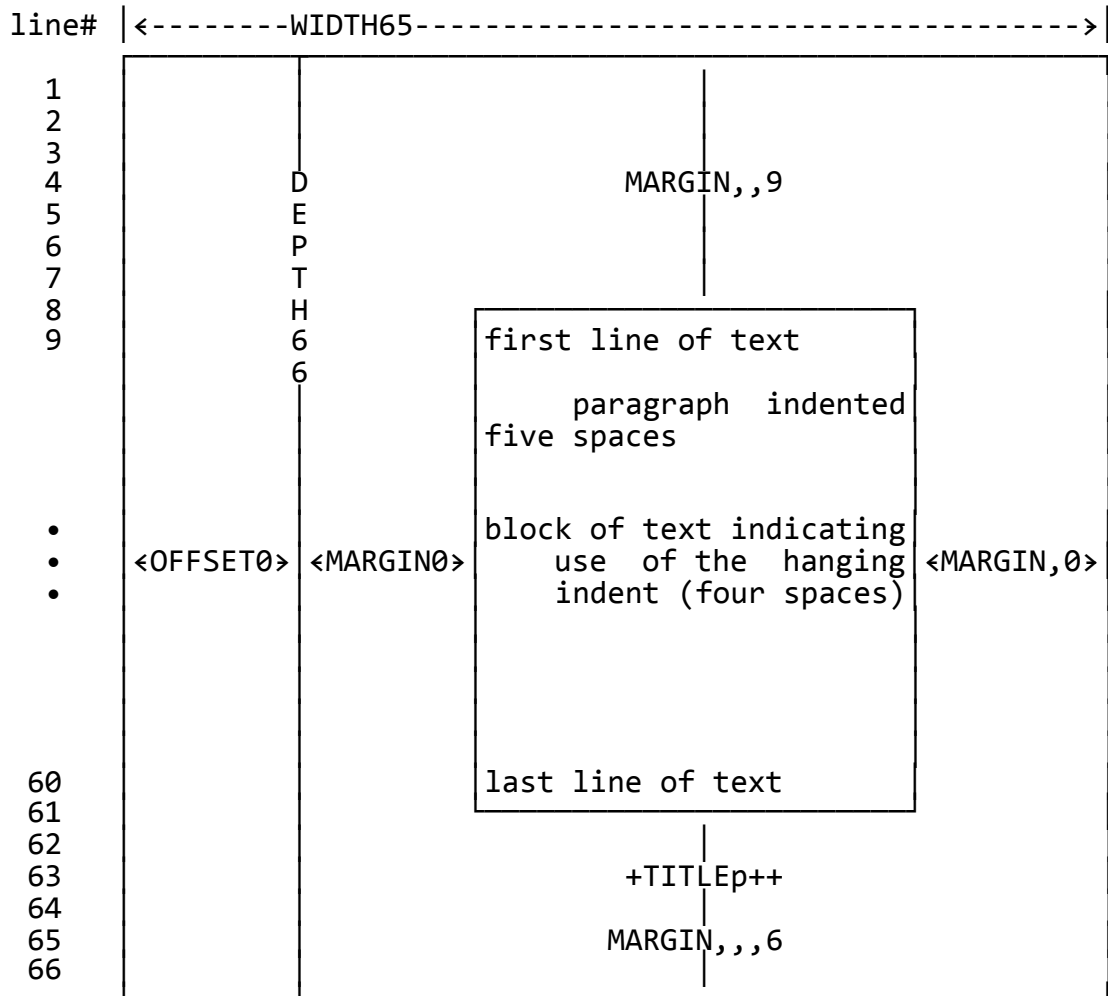


Figure 6 - Common Alter Codes effecting page layout

+B<H L U> set mode for underscoring blanks

This code determines which blanks will be underscored within an underscored text string. !+BH+ means only blanks in headings will be underscored. !+BL+ means no blanks will be underscored, e.g. an underscored text string. !+BU+ means all blanks will be

February 2, 1975

ALTER CODES

underscored, e.g. an underscored text string. The default mode is H.

+COLUMNn+

Begins an "n" column format (see Section 5.10 paragraph 2 for current restrictions on use of this code). The default width of the column text is determined by: $(\text{Page WIDTH} - \text{OFFSET} - \text{GUTTER} * (n-1)) / n$. The user is currently restricted to a maximum of five text columns on a page.

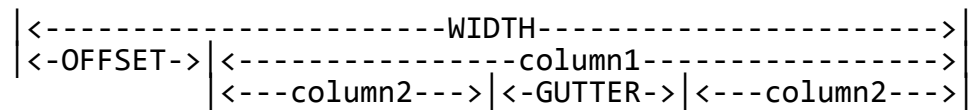


Figure 7 -- Effective column widths

+DATE<1 2>+ print the date

The date will be printed where this code appears in the text. The date can have two formats. For example,

!+DATE1+ will print 01/25/19.

!+DATE2+ will print January 25, 1919.

+DEPTHn+

This code determines the number of lines per page in a Fullprint. The actual number of lines available in the current column is the depth minus the top and bottom margins. The default page depth is 66 lines.

+DIn+

This code sets the decimal label level margin increment (see the FRESS Reference Manual [4] for a description of decimal labels). For each decimal label level, "n" spaces will be indented. The default is 3.

FORMATTING CODES

ALTER CODES

February 2, 1975

+GRID<grid>+

The GRID alter code is a facility which allows the FRESS user to represent graphs, tables, charts, etc. graphically. See Section 5.12 for a complete description.

+GUTTERn+

Sets the spacing between default column boundaries, i.e., by specifying !+GUTTER4+!+COLUMN2+ the user's column boundaries would be: first column position 0 to 30, second column position 34 to 64. The default gutter value is 3 characters wide. (See Figure 7.)

+HEAD<A B>+ select a heading table.

This code selects the table into which the !-Hn- codes are indexed, i.e., by specifying !+HEADB+ each following use of a !-H2- code would force the heading text to upper case, underscore it and skip three lines before and two lines after the heading text. The default for the heading table selection is table B.

The six heading types (!-Hn- codes) have the following characteristics:

	HEAD NO.	ALL CAPS	UNDER- SCORE	LINES SKIPPED BEFORE AFTER	
Table A:	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
Table B:	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

February 2, 1975

ALTER CODES

`+HEADn=a,b,c<;n=a,b,c;...;n=a,b,c>+`

This code will redefine heading table B. "n" is the heading level to be redefined. "a" is the sum of any combination of the following:

- 4 heading title will be capitalized
- 32 heading title will be underscored
- 1 skip to a new page before the heading
- 0 nothing is done to the heading.

"b" is the number of lines to skip before the heading, and "c" is the number of lines to skip after the heading.

For example, heading number 2 of table B was created with `!+HEAD2=36,3,2+` where $a=32+4=36$. Great care should be taken when using this code. Minimal error checking is done and invalid formats may cause the Fullprint to fail.

`+JUST<Cn><;Ln>+`

To understand the terms used to describe this code, it is advantageous to read Section 5.10 first.

This code sets the column and line justification modes according to:

- 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)
- 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

To specify no line justification, for example, a user would specify `!+JUSTL2+`. The default justification modes are C0 and L0.

`+MARGIN<lm><,<rm><,<tm><,<bm>>>>+`

"lm", "rm", "tm", "bm", are optional numeric parameters that indicate number of blank spaces left between the text and column boundaries (see Fig. 6). The parameters are positional and hence any unspecified parameter retains its original value. In multiple column mode the four margins set apply to each column.

Note that `!+MARGIN0,0,5,5` will reset the left and right margins to zero and the top and bottom margins to five; while `!+MARGIN,,5,5+` only resets the top and bottom margins. The default margin values are 0,0,9,6.

`+OFFSETn+`

This code adds an additional margin subtracted from the page width. It is generally used to leave room for binding the printed material. The offset is taken alternately from the left and right hand sides of the pages (see Fig. 8). To offset an entire document see the Fullprint command in Section 8.6.

`+PAGE<n><,m>+`

This code sets the output buffer page size. The maximum width of any page will be "n" and the maximum depth will be "m". The default is 110 by 66. This specification is installation dependent.

`+PARA<n><,m>+`

Resets the number of spaces indented by the `!-p-` code to "n" and the number of lines skipped to "m". The default is `!+PARA1,5+`

`+SETTABx=n<L R C><;x=n<L R C>>...<;x=n<L R C>>+`

When `!-Tx-` is specified, "x" is a number from one to ten used to index into a table of default values (every ten spaces). The SETTAB command resets any number of these default settings. For example,

`!+settab3=10;4=20+`

would reset the third tab stop (reached by a `!-T3-` or `!-U3-` code) to column 10 and the fourth tab stop to column 20.

The tab positions may be qualified by the modifiers "L", "R" or "C", indicating that the tab stop being defined is the leftmost (L), rightmost (R), or center position (C) of any tabular text. For example the format `!+settab5=30<mod>+` results in:

This when <mod> equals R,

February 2, 1975

ALTER CODES

This when <mod> equals C.
 This when <mod> equals L.

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.

+SPACEn+

Specifies that the printout is to be single (n equals 1), double (n equals 2) or triple (n equals 3) spaced. It also sets the number of allowed widow lines to twice "n". (See !+WIDOW+ code).

+TABLE<x=n,m<L R C>>...<;x=n,m<L R C>>+

The table command is similar to the !+SETTAB...+ command in that it is a table of column position pairs into which the !-Tx- codes are indexed (See Section 5.11.2 for an example). In other words, rather than specifying a single position, the indices "x" refer to two column positions, i.e., left and right boundaries of the table entry (slot), between which the text following the !-Tx- code is placed.

When in table mode (after a !+TABLE...+ code is encountered in a user's text), each new line code (!-Sn-) specifies the beginning of a new table row (as well as spacing "n" lines), whereas each tab code (!-Tx- or !-Ux-) specifies the beginning of a new table slot (but not a new line). Any text following a !-Tx- code which will not fit between the "n,m" columns is automatically moved to the next line (at the proper column position).

The user should leave table mode by issuing a null table command (i.e., !+TABLE+). If this is forgotten, subsequent tabs and tables may not work correctly.

+TITLEa<=m<L R C>>+<title string>+

Specifies a "title string" that is printed on every page starting with the current one (See Figure 8). The title, subtitle, and page number on this page were obtained by imbedding !+titleT+FORMATTING CODES+, !+titleS+ALTER CODES+ and !+titleF+:+ codes at the beginning of this section.

a - is an alphabetic character indicating one of the seven possible title strings (see the following table).

m - numeric parameter indicating the number of lines from the top of the page at which the title is to be positioned (see table below). The number may be modified by "L", "R" or "C" indicating that the title is to appear only on even (Left), odd (Right), or all (C) pages.

title string - this is the main body of the title text; it may not contain any format codes (this includes special blanks and underscores). A title may be cleared by indicating a null string (i.e. !+titleD++ will clear printing of the date field beginning on the page on which the code is encountered).

The quadding mode (margin alignment) for any title (see table below) may be overridden by specifying several title string segments. A segment is indicated by imbedding a "*" into the title string. A title code with a title string of "quad left*quad center*quad right" would position "quad left" against the lefthand side of the page, "quad center" in the middle of the page, and "quad right" against the right hand side of the page. Note that a text segment is quadded to the left, to the center, or to the right irrespective of the fact that the page may be even or odd format. Thus to cause a title string to always appear on the right side of the page, the code

!+TITLED+**title+

should be imbedded in the file.

The user may imbed a ":" into the title string to indicate substitution of the current page number in that string. He may also imbed "&1" or "&2" strings which will cause substitution of the current date in "MM/DD/YY" or "Month DD, 19YY" form.

The following table describes default title positions:

February 2, 1975

ALTER CODES

Keyword	Line Number	Quadding	
		Even page	Odd page
Date	7	right	left
Foot	63	left	right
Evenfoot ³⁶	63	left	-
Oddfoot ³⁶	63	-	right
Title	4	left	right
Subtitle	7	left	right
Pageno	63	center	center

<-----WIDTH----->		line#	<-----WIDTH----->	
TITLE	DATE	1	DATE	TITLE
		2		
		3		
		4		
		5		
		6		
		7		
		8		
		9		
SUBTITLE	<OFF>	D	<OFF>	SUBTITLE
		E		
		P		
		T		
		H		
FOOTING		62		FOOTING
		63		
		64		
		65		
		66		
(Even Page)			(Odd Page)	

Figure 8 -- Title Strings in relation to other ALTER Codes

+TOFC<n<=m>>...<;n<=m>>+

This code determines which heading levels "n" will appear in the table of contents. Also, the user may reset the indentation level in the table of contents by specifying "=m". For example,

!+TOFC1;2;3+

specifies the text following a !-H1-, !-H2- and !-H3- code is entered into the table of contents. Each entry is given the

default indentation of zero (for heading 1), one (for heading 2), etc.

Whereas, if the user specified

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

the respective heading entries are indented zero, three and five. Note if the +TOFC+ code is issued, only those headings specified will appear in the table of contents. If the code does not appear, all heading levels are listed in the table of contents.

+WIDOWn+

Sets the number of lines allowed to stand alone at the top and bottom of a column when column justification is in effect (see Section 5.10). The default widow value is twice the current spacing value as specified by the space code.

+WIDTHn+

Sets the user's page width to "n" spaces. When this code is issued, any default column boundaries are also reset. The default page width is 65 spaces and the maximum 110 spaces.

5.6 MACRO CODES

The user may define his own format codes by use of the macro code facility. A macro is defined by imbedding in the text the code:

!.macroname=any string.

thereafter any use of the code:

!.macroname.

causes the Fullprint program to substitute "any string". For instance if the user wishes to reduce the number of keystrokes required to specify:

!-s4;i10;j13-

he could define a macro !.SIJ. by specifying

February 2, 1975

MACRO CODES

!.SIJ=!-s4;i10;j13-.

Thereafter, every use of the code !.SIJ. would skip four lines, indent ten spaces, and start a hanging indent of thirteen spaces.

Macros are used not only to save keystrokes, but also to make global formatting changes easier. Suppose, for example, a file contains several hundred formulae, and each one is to appear indented 10 on a new line as follows:

!-i10-X = Y

Now suppose, after the entire file has been typed in, it was found that the formulae were required to be centered rather than just indented. This would require changing every one of the several hundred format codes. Had macros been used, however, only one change would be required. In this case, a macro named "form" would have first been defined as follows:

!.form=!-i10-.

Each formula would use this macro rather than the format code:

!.form.X = Y

To then center all the formulae, all that is required is to redefine "form":

!.form=!-c-.

Each macro definition must be less than 100 characters in length. In determining the length of a macro definition, any imbedded macros must be expanded, and their full length considered. The delimiters (!. and .) and macroname must also be counted. Therefore if a longer string is desired, the user must concatenate multiple macro usages.

Macros may also be used to generate frequently used strings of characters. For example, if a manuscript contains many bibliographic references to a particular magazine, the name of the journal could be entered as a macro.

Macros may be redefined at any time. Each definition is in effect until another definition is encountered, either in online display or in a Fullprint. The user may sometimes desire to "undefine" or "null" a macro, that is, to cause nothing to happen when the macroname is encountered. To do this, he would imbed in the text the code

!.macroname=.

Note that unlike other format codes, character case is important in identifying macro names. Thus `!-p-` and `!-P-` would both produce paragraphs, but `!.sij.` would not be defined by the above macro definition. In addition, a terminal with only upper case characters will display upper and lower case macros the same way, i.e., there will be no capitalize character (%) indicating upper case. Therefore to avoid confusion it is wise to use only lower case characters when defining macro code names.

If during an editing session the message "UNDEFINED MACRO" arises, it means that either the user has not defined a macro code or he has not scrolled over the macro definition before encountering its use. The user may circumvent this problem by copying the macro definition to the head of each major section in his file because macros may be redefined at will. An alternative and recommended solution is to group all macro definitions together at the top of one's file, and scroll over these definitions at the start of an editing session.

5.7 FORMAT CODE ERROR MESSAGES

If an invalid format code appears in the current display buffer, one of the following error messages will be printed so that the user may substitute a correct code. Note that the editing operation that produced the error is completed even if one of these error messages is typed. These messages may also be typed at the terminal when the errors are encountered during a Fullprint.

- | | |
|----------------------|---|
| BOUNDARY ERROR | The dimensions of the current column or page as determined by the WIDTH, GUTTER, DEPTH, OFFSET, MARGIN, and COLUMN codes are in error. This message indicates that attempting a Fullprint hard copy printout may result in abnormal termination. |
| DELIMITER ERROR | Indicates a missing beginning or ending <code><DELIMITER></code> . Everything following the <code>!</code> is taken as literal text. |
| FOOTNOTE(S) TOO LONG | Fullprint is unable to properly format a footnote because of its excessive length (greater than half a page). The Fullprint is terminated at the point where the error occurred. The user must make the footnote smaller. This message will only appear during a Fullprint, not while the file is displayed online. |

February 2, 1975

SPECIAL CHARACTERS

- INVALID CODE The <CODE> part of the format specification is invalid.
- INVALID PARM, OFFSET= The <DATA> portion of the format code located at the specified character displacement from the start of the code is invalid. The offset number given equals the displacement from the ! (or last ";" -2) to the end of the invalid parameter.
- SYNTAX ERROR, OFFSET= This message indicates that an invalid delimiter was encountered in the <DATA> field. It applies to codes having multiple <DATA> fields (i.e. !+settab...+, !+table...+ etc. The offset value is defined as for INVALID PARM).
- UNDEFINED MACRO A macro code was encountered before its corresponding definition has been scrolled over; see Section 5.6.

5.8 SPECIAL CHARACTERS

Certain characters are used by FRESS to specify system parameters (see Section 3.16). The exclamation point (!), for example, is used as a format code delimiter, the percent sign (%) is the system structure code delimiter and the capitalization character, and the not sign (-) is the special blank character (see Section 5.10).

These three may be inputted and are represented in a user's file by the following format codes:

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

The following is a list of other FRESS control characters and their format code representations:

!74	¢	
!80	&	
!108	%	
!109	_	underscore
!111	?	
!123	#	
!124	@	
!126	=	

FORMATTING CODES

SPECIAL CHARACTERS

February 2, 1975

!127 "

The following is a list of special characters available on the 1403 line printer but not represented on a standard 2741 or Datel terminal:

!96	-	minus
!139	{	
!140	≤	
!141	(superscript (
!142	+	superscript +
!143	+	line intersection
!155	}	
!156	²³	
!157)	superscript)
!158	±	
!159	■	
!160	-	superscript -
!161	°	degree sign
!171	L	lower left corner
!172	[upper left corner
!173	[
!174	≥	
!175	•	
!176	0	superscripts
!177	1	
!178	1	
!179	3	
!180	4	
!181	5	
!182	6	
!183	7	
!184	8	
!185	9	
!187]	lower right corner
!188]	upper right corner
!189]	
!190	≠	
!191	-	horizontal line

The user may create his own special characters by making use of the backspace key on his terminal, i.e., the character "b" was "created" by typing "b" "backspace key" "/". Currently overprinting is restricted to a single 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.,!173*5... rather than ...!1735.... The asterisk will not appear in the text when Fullprinted. An additional restriction is that a character represented by a format code cannot be overprinted with another character represented by a format code. To overprint two

February 2, 1975

UNDERSCORES

characters one of which is in format code representation, the user must specify the format code first, followed by the backspace, followed by the ordinary character.

5.9 UNDERSCORES

Underscoring is specified by the codes

!(0underscored word!)

For example, to insert "An underscored word" the following could be used:

i /location/An !(0underscored!) word

Existing text can be underscored by inserting only the underscore codes before and after the text to be underscored, or by use of the Underscore command.

Underscores may also be input literally during an insert, using backspaces and underscores. The backspaces and underscores are removed and the text being underscored is surrounded with the proper format codes. Characters may be underscored one at a time, or all together. In the following example, "A" is a character and "b" is a backspace:

Ab_Ab_Ab_... or AAAbbb____... or Ab_AAAbb__

becomes:

!(0AAA!)

When literal underscoring is used, no formatting, structure, or other special codes may be included in the underscored text. If an ending underscore format code ("!") is accidentally omitted, all text to the next ending underscore code will be underlined except the first line of text after any new line format code.

NOTE: The total number of characters in a line entered at a 2741 typewriter terminal includes all underscores and backspaces. It should not exceed 130 characters because of hardware limitations.

5.10 JUSTIFICATION

1) Column/Page Justification

Whenever the end of a paragraph is reached, the Fullprint program decides whether or not the paragraph fits into the current column. If not, the following criteria are applied to determine where the paragraph lines will be positioned:

- 1) If there are less than n (n is set by the !+WIDOWn+ code) lines left in the column (page), 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 the Fullprint program 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 (see Section 5.4): heading two (first pass), heading three (second pass), etc.

2) Column Balancing (for multiple columns only)

If while in multi-column mode the user inserts a new page or single column code in a location that would cause the partially existing columns to be of unequal length, Fullprint will attempt to reformat the columns such that they are of equal length.

3) Line Justification

Currently there are four modes of line justification set by the !+JUST+ code:

- 0) 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.

February 2, 1975

- 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.

4) Special Blank

The "not" sign (~) is the SPECIAL BLANK or BLANK FILL (non-justifiable blank), i.e. during a Fullprint the ~ is translated to a blank but is regarded as a text character by the line justification algorithm.

As an example of its use 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.

The Fullprint procedure deletes extra blanks (more than one between words, more than two after punctuation) from the printout during all line justification.

5.11 TABLE FORMATTING EXAMPLES+

5.11.1 ENTERING TAB CODES

To create a simple table the user could enter &ASIS Mode (see Section 8.7). This automatically inserts a new line formatting code at the start of every input string or every line in Input Mode. For example, if the current display was set by

```
!+settab1=10;2=32+
```

then the user could create a table by issuing the following commands (CR corresponds to the physical carriage return and TB to the physical tab key):

command	explanation
&A	enter as is mode
i/32+/CR	system enters input mode
TBfirst entryTBsecond entryCR	first input line
TBthe second lineTBlast entryCR	second input line

The resulting input into the file would be translated to:

```
!-s0-!-t-first entry!-t-second entry
!-s0-!-t-the second line!-t-last entry
```

Hence, a Fullprint would give the following results:

first entry	second entry
the second line	last entry

Note that while it is easy to input a table using the physical tab key rather than by specifying tab format codes explicitly in the input string, our experience is that subsequent editing of complicated tables is much more difficult because of the need to count the number of !-t- codes preceeding the one to be edited.

5.11.2 TABLE MODE

For more complex tables (e.g., ones in which entries require multiple lines and/or various justification effects) as demonstrated below, the user would enter table mode by inserting, for example, the following format code into his file:

```
!+table1=1,15;2=17,20c;3=21,60+
```

This defines a table having three slots from column positions 1 to 15, from 17 to 20 and from 21 to 60. Note that all entries in the second slot are centered between the 17th and the 20th column. A typical row entry is then typed as follows:

```
!-s1;t1-structure code delimiter!-t2-!108!-u3-in a user's file, this character indicates the beginning of a structure code, such as a label
```

The resulting Fullprint is as follows:

structure code	%	in a user's file, this character
delimiter		indicates the beginning of a structure
		code, such as a label

February 2, 1975

Note that the first and third entries are wider than their slot widths and hence occupy more than one line. Also the third entry is justified both left and right as a result of the `!-u3-tab`.

Be sure to specify the skip (`!-Sn-`) codes when starting new lines. Using only tab codes to the first slot (`!-t1-`) when new lines are desired will cause strange results.

A null table mode code (`!+TABLE+`) not paired with another table code will result in ragged-right text justification during a Fullprint. Note that if a null table mode code is not placed after each completed table, strange things may happen to the print-out of subsequent tables.

5.11.3 OVERLAYED TEXT

A TAB code (`!-Tn-` or `!-Un-`) will position to the specified column, regardless of whether or not text is already there. This may result in tabular text concealing text which has already been positioned on the current line. For example, if the following line were part of the user's file:

```
!-T5-overlaid!-T7-HIDE IT
```

on output it would result in the following if tab stops five and seven were set only five characters apart:

```
overlHIDE IT
```

The situation can become even worse if the following is in the file:

```
!-t5-overlaid!-t7-HIDE IT!-t5-1234567890
```

This would result in:

```
1234567890IT
```

Overlay frequently occurs if the user sets his tabs improperly, forgets to start new lines at new tabular rows, or sets no tabs at all.

5.11.4 SOME MORE TAB CODE EXAMPLES

!-Tn- simple tab - n runs from one to ten and is defaulted to every ten spaces (1=10;2=20;...;10=100) - these values may be reset by using the !+settab...+ alter code.

Example:

0	10	20	30	40	50	60
	-C	"character"				
	-W	"word"				
	-L	"line"				

The above table was entered using the default setting 1=10;2=20;...;10=100. This table would be inserted as follows:

```
!-s1;t1--C!-t2-"character"!-s0;t1--W!-t2-"word"!-s0;t1--L!-t2-"line"
```

!-Tmod- modified tab - left = l, right = r, center = c.

This tab allows the user to change the appearance of his tabs. For example, the above table uses the default left setting. However, the user may want the words "character", "word", etc., to be centered. To do this he would use the center modification. The tabs would be inserted this way:

```
!-s1;t1--C!-t2c-"character"!-s0;t1--W!-t2c-"word"!-s0;t1--L!-t2c-"line"
```

and the table would look like this when printed:

0	10	20	30	40	50	60
	-C	"character"				
	-W	"word"				
	-L	"line"				

The right modified tab would normally be used with a list of figures in a budget. For example, the following table

0	10	20	30	40	50	60
	Office Supplies					Cost
	1 doz. erasers					1.98

February 2, 1975

2 doz. pencils	1.20
1 doz. pens	4.68
2 rulers	.59
3 reams bond paper	12.39

would be entered as follows:

```
...Cost!-s1-1 doz. erasers!-t6r-1.98!-s0-2 doz. pencils
!-t6r-1.20 !-s0-1 doz. pens!-t6r-4.68!-s0-2 rulers
!-t6r-.59!-s0-3 reams bond paper!-t6r-12.39...
```

!-T,fill- fill tab - will fill in blank space before the tab with character specified by "fill".

In the above example, if the user wanted to fill the space between "erasers" and "1.98" with periods, he would type:

```
...Cost!-s1-1 doz. erasers!-t6r,.-1.98!-s0-2 doz.
pencils!-t6r,.-1.20...
```

The table would then look like this:

	0	10	20	30	40	50	60
Office Supplies							Cost
1 doz. erasers.....							1.98
2 doz. pencils.....							1.20
1 doz. pens.....							4.68
2 rulers.....							.59
3 reams bond paper.....							12.39

5.12 THE GRID ALTER CODE⁺

5.12.1 DESCRIPTION

The GRID alter code is a facility which allows the FRESS user to represent graphs, tables, charts, etc. graphically. The GRID code can be seen as an extension of the b (BOX) format code; using the GRID code, the user can draw "boxes within boxes". Each of the tables below was drawn with one GRID code.

As a consequence of its powerful capabilities, the GRID code is rather complex and should not be attempted by beginners. It is also not, in its current state, "rock solid": incorrect specification may cause the Fullprint program to fail. Be that as it may, the paragraphs below will demonstrate the facilities of the code.

Formal definition

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 |

Grid character symbols

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. Table G1, below, lists the symbols used in the GRID

February 2, 1975

GRID CODES

code and the grid characters they represent. Note that those characters that can be typed in at the terminal are represented literally.

SYMBOL	REPRESENTS
A	┌
B	┐
C	└
D	┘
E	├
F	┤
G	┼
H	┴
I	┬
Ø	Ø
-	-

Table G1: Grid symbols

Expansion of symbols

An important concept employed by the GRID code is one of expansion. The grid character represented by the grid character symbol may be expanded in the horizontal direction. FRESS expands a grid character by placing one or more expansion characters after the grid character when the grid is printed. Table G2 lists the characters used to expand each grid character.

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. 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 "I2" results in "┬┐".

SYMBOL	REPRESENTS	HORIZONTAL EXPANSION
A	┌	—
B	┐	—
C	└	⌘
D	┘	—
E	+	—
F	┌	⌘
G	┐	—
H	└	—
I	┘	⌘
⌘	⌘	⌘
—	—	—
		⌘

Table G2: Horizontal expansion characters

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

February 2, 1975

GRID CODES

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 table G3.)

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.

SYMBOL	REPRESENTS	VERTICAL EXPANSION
A	┌	┌
B	└	└
C	├	├
D	┤	┤
E	┌	┌
F	└	└
G	├	├
H	┤	┤
I	┌	┌
Ø	└	└
-	├	├
	┤	┤

Table G3: Vertical expansion characters

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:

When making a grid, it is important to keep in mind that expansion takes place before iteration, so that it is the expanded grid character which is repeated.

The subgrid

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. Table G1 was produced with the following GRID code:

```
!+grid(a7,b11,c)2,(d7,e11,f)13,(g7,h11,i)+
```

Tables G2 and G3 were created with the following GRID code:

```
!+grid(a7,2b11,c)2,(d7,2e11,f)13,(g7,2h11,i)+
```

Placing text within the grid

As with the box format code, text following the GRID code begins in the upper left-hand corner of the grid. In most cases, then, a line is skipped by the user 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.

February 2, 1975

GRID CODES

5.12.2 REFERENCE TABLES

The tables which follow are designed to aid the user of the GRID code by summarizing much of the information given above. Table G4 lists the grid symbols and the characters they represent; table G5 lists all the expansion characters.

┌	┐	└
├	┤	┞
┬	┴	┴
⋈	-	

<=>

A	B	C
D	E	F
G	H	I
⋈	-	

Table G4

Summary of the grid characters
and character symbols.

Table G5

Summary of the vertical and
horizontal expansion characters.

Meaning of column headings:

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

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

5.12.3 GRID EXAMPLE

The following is a step-by-step example of how the square grid used in table G4 was produced. In each step, additions to the GRID code are underlined.

0. 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. (We are limiting our discussion to the grid on the right in table G4.) To position the grid correctly, the following edit and alter codes were used:

```
!+settab1=24;2=26;3=30;4=34+
!-s2;t1-!+grid...
```

1. 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)2+

Generates:

```
┌───┬───┬───┐
```

2. 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:

```
┌───┬───┬───┐
└───┴───┴───┘
```

3. 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)2+

Generates:

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

February 2, 1975

GRID CODES

4. 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:

5. 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:

6. 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:

7. 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. Additional tabs could, of course, have been used in place of the special blanks. Note that the Skip edit codes were used in such a way as to avoid overlaying the

lines of the grid. (See the section above on placing text within the grid.)

Miscellaneous tips

A GRID format code acts as a conditional page. Consequently, if a GRID is preceded by a tab code and a page eject is forced, the grid will be left justified. The user should take care to allow for this, perhaps by adding his own conditional page format code before the GRID itself.

A horizontal line, 32 characters long, could be created with the GRID code

!+grid(32-)+


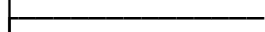

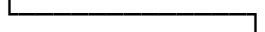
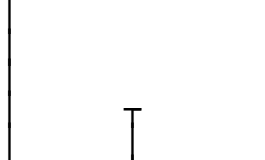
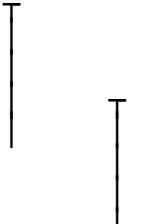





and a vertical line of the same length could be created with the code

!+grid(|)32+

Besides generating the various tables shown above, the grid code is also useful for constructing diagrams such as flowcharts and decision trees. The following examples show various "pieces" of grids which may be used in this way along with the grid codes used to generate them.

February 2, 1975

GRID CODES

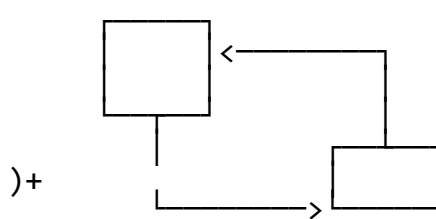
	<code>!+grid(15-)+</code>
	<code>!+grid(d15)+</code>
	<code>!+grid(d14,f)+</code>
	<code>!+grid(g14,c)+</code>
	<code>!+grid()5+</code>
	<code>!+grid(b)5+</code>
	<code>!+grid(b)4,(g2)+</code>
	<code>!+grid(a15)4+</code>
	<code>!+grid(14,)3,(14-,i)+</code>
	<code>!+grid(7-,b8-)4+</code>
	<code>!+grid(14,)3,(g14,i)+</code>

The following example shows how boxes and grid codes can be intermixed. Assume in this example that the leftmost line is in column 0.

FORMATTING CODES

GRID CODES

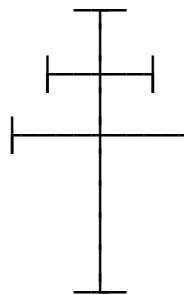
February 2, 1975



!-s0;b7,4-
!-s0;t*7-<+grid(8-,c)3+

!-s1;t*3-!+grid(b)2+
!-s0;t*13;b7,3-!-t*16-!+grid(h-

!-s0;t*3-!+grid(g9)+!-t*12->



!+grid(4,-,b2)2,(2,d3,e3,f),(5,|)+
!-s3-!+grid(d5,e5,f),(5,|)4,(4,-,h2)+

February 2, 1975

6 ADDITIONAL FEATURES6.1 LABELS

A facility for remembering "important points" in a file is very convenient in order to avoid lengthy (and costly) pattern scans or scrolls. "Labels" may be placed (with a Make Label (MLABEL) command) anywhere in a file by the user and later retrieved via a Get Label command (GLABEL) to provide instant relocation of the display window to the specified label. To provide this "random access" facility, FRESS stores each label (and its internal location in the file) in an alphabetized "label space", which may also be viewed by the user (see the DSPACE command). The label appears in the text encoded as "%L(label)", where "%" is the FRESS system delimiter specifying that structure (i.e., nonliteral text) follows, and the "L" specifies that this particular type of structure is a label. It is recommended for saving user and machine time that the user place labels in his file at major section headings or places where repeated editing takes place. One label per four or five hardcopy output pages provides a convenient "online table of contents".

The system will remember the current location in the file with a pointer if the user moves from there with a Get Label. He may stack these pointers, and use them to reverse his traveling, returning to previous locations in inverse order (see the Return command).

6.2 HYPERTEXT EDITING

If a label is part of a <scope> to be deleted, substituted, moved or copied, FRESS will give a feedback message:

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

The first (A) allows an edit of both text and structure. The second alternative (R) rejects the command, the third (T) will leave all the structure orders and edit just the text around them. The Copy command cannot be used to copy a label since each label must remain unique. Also, interfile moves of labels are not allowed.

If the user wishes to edit hypertext he may specify this by putting a "%" in front of the command name. For instance:

%DE <scope>

will allow hypertext deletions without giving the user a feedback message. There is no way to specify text-only editing in the original command.

6.3 FILES AND SPACES+

Any number of files may be in use simultaneously. The user "opens" a file by doing a Get File command, specifying a password if the file was created originally with a password. Using deferred Location Pointers (Section 3.8) he may edit freely between files (typically moving or copying text fragments).

Each file in FRESS is divided into several arbitrarily long, disjoint subportions, called spaces. The text space is the segment into which the literal text of the file (and any embedded structure) is stored, while the label space contains the alphabetized list of user defined labels. The structure space contains all of the structure in a file.

The work space is used as a storage and collection area. The functions Move To Work (MTWORK), Copy To Work (CTWORK), Move From Work (MFWORK), and Copy From Work (CFWORK) allow pieces of text to be collectively added to the work area, where they may be viewed or edited as if they were in the text space. The entire contents of the work area may also be moved or copied to some place in the user's file. The value of the work area is to allow "boilerplating", that is, constructing new documents from segments picked out of existing documents. It is also a convenient "attic" in which to store pieces of text not currently needed in the manuscript but still potentially useful. The text in the work space is ignored when a Fullprint is done but can be printed out by using the Offline Type command.

The label space is a system area in which the user's labels are stored alphabetically. It may be viewed by using the Display Space command (DSPACE L) and may be scrolled through to review the labels that have been made previously, or even to edit them. The user may use the Get Label command to travel within or between open files, using the Return function to retrace his steps.

The structure space of a file contains, in geographical order, all of the hypertext in a file. This includes labels and area lines, as well as other types of structure discussed in FRESS

February 2, 1975

VIEWSPECS

Reference Manual [4]. It reflects, in order, structure of the main text and work spaces. Displaying this space can be a useful way to get an overall picture of the structure of a file. For instance, if labels are associated with the format code headings of a file, the structure space becomes a readable outline of the file.

A complete description of the spaces in a FRESS file may be found in the FRESS Reference Manual [4].

6.4 VIEWSPECS+

Viewspecs (shorthand for "viewing specifications") are used to specify how the file is to be displayed and formatted online. The Set Viewspec (SV) command is used to specify which of the various viewspecs are to be in effect, and the Display Viewspec (DV) command to find out which viewspecs are currently in effect.

There are four "system-defined" viewspec strings. These are:

edit	minimal online formatting, no right justification, format codes displayed (the default)
print	online formatting, no formatting codes displayed, no structure displayed, right justification
normal	online formatting, no justification, and format and structure codes displayed
*	current viewspecs.

Following one of these standard strings, the user may optionally include a list of specific viewspecs to be added or deleted from the standard string. Any number may be specified by giving the mnemonic codes preceded by a sign (+ or -) which indicates whether the viewspec is to be added or deleted. A given sign has scope over all viewspecs following it, until a new sign is encountered. Normally, the user specifies "*" plus or minus selected viewspec codes, giving him his current viewspecs with only a few changes. For example, had he just logged in to FRESS, issuing

```
sv /*-ofp
```

would display text formatted properly but without any format delimiters and codes. (Note that a non-blank delimiter should be used in this case. If the command were issued as "sv *-ofp" the "*" would be interpreted as the delimiter and the viewspec string would be "-ofp".)

ADDITIONAL FEATURES

VIEWSPECS

February 2, 1975

For a complete listing of viewspecs, refer to the FRESS Reference Manual [4]. A list of some of the more useful viewspecs follows. The viewspecs need not be specified as capital letters.

- B use special character for blanks
- FP format code delimiters displayed
- FV formatting off
- JU text is justified
- M labels displayed
- O format codes displayed
- SP structure code delimiters (%) displayed
- S2 double space

February 2, 1975

7 TRANSCRIPT OF AN EDITING SESSION

7.1 ANNOTATION FOR SAMPLE SESSION

This section is for demonstration purposes and describes, by examples, how to use FRESS. It is not a manual and omits many important features (and restrictions). In this explanation of the basic features, generic names to be filled in by the user will be enclosed within angle brackets (<>) to distinguish them from literal text, which is typed as shown; explanatory notes are in parentheses; and entire commands and responses are in quotes. The carriage return (CR) terminates all commands on a single line. Typically, blanks are used to separate parameters, but if literal blanks are present in any of the parameters, a delimiter such as "/" is used to separate them. For clarity, "Ø" sometimes is used to indicate a blank.

Each paragraph number corresponds to an example in Section 7.5, entitled "Sample Session". Typically, user and machine responses are on alternate lines. The sample text file used is basically this section itself.

7.2 INITIALIZATION EXAMPLES

For a user to begin a session on FRESS, it is first necessary to LOGON to the system.

1) The user types in "l(ogon) <user-identification>". The system will respond "ENTER PASSWORD:" at which point he types "<password>". The system will come back with general logging on information.

2) Then to call up FRESS, he types: "fress". When FRESS responds with "FRESS READY", the system is ready to accept commands.

3) To create a new file, the following command is used: "mf (Make File) <filename> <password>". The system responds "INPUT" and a "proceed" is given (see Section 10.1). The user is now in Input Mode: he may type as many lines of literal text into his file as desired. When finished, he types a null line (i.e., a line containing only a carriage return) and the system responds "FRESS" to indicate the user has now returned from Input Mode to Command Mode. Typing "Ø" (scroll zero lines) determines the current

February 2, 1975

position in the file: in Transcription Mode, the default on a 2741, it is one word before the last line typed in. Determining one's position in the file can also be done (at less expense) by typing "p 1" to print one line. "ff <filename>" frees (closes) a file and makes the last change permanent. The system will respond "PROCEED".

4) To access an existing file the following command is used: "gf (Get File) <filename> <password>". A "proceed" is given and the user's window is positioned at the system provided "start line" of the file, "*START OF TEXT AREA*", although this line is not typed out. Text is put into a file at the top by typing: "ti(nput)".

5) The body of the inserted text may be typed continuously (type a line, CR; type a line, CR; etc.) until all the necessary information has been entered in the file. A null line with only a carriage return indicates the end of the inputting phase to FRESS. Input Mode could also have been entered by typing "si(nput)/t" or "i/<insert point>/", in this case "i/*/" . The "null" second parameter indicates Input Mode is to be entered after the point specified; note that the "file." located would be the first occurrence of that pattern in the text.

7.3 TRAVELING EXAMPLES

6) "ds (Display Space) t (text)" (re)positions the display at the top of the text space.

There are three principal methods of moving (traveling) through text in a file. The first is by scrolling. This is done by indicating a number of display lines to be moved, either forward or backward.

7) For example, 3 would move forward three lines in the text, while -3 would move three lines backward.

The second method is a locate. A locate searches the text for a specified string of characters.

8) For example, "l(ocate) text", would search from the current point in the file for the character string "text". When the pattern is located (the first time it occurs in the file, no more than 8000 characters from the current location) the display pointer will be positioned at this string. Scanning backwards is done by typing "lb (locate backwards) <pattern>".

These first two methods are sequential movement, i.e., the user moves through the material in the file in the order in which

February 2, 1975

it was typed. The third method of traveling through a file is non-linear travel, such as retrieving labels (which the user has previously made at appropriate locations in the text). This facility parallels transfers to labels in a computer program.

9) For example, "gl (Get Label) fress", would locate the label "fress" (if it exists), and the display would start at that point in the user's file. If the label does not exist, a message is typed.

10) A new label is made by typing: "ml (Make Label) <lp> (point at which the label is to be inserted) <labelname> (identifier chosen by user)". The system puts in a label at that point in the text and automatically updates a list of all labels made for future reference by the user. (Note: one needs to specify only a unique beginning substring for a Get Label; in this sample session, "ed" suffices.)

11) To see the list of labels, the user may type "ds (Display Space) l (label)" which will display the file's "label space", an alphabetized index of all labels.

12) "r(eturn)" will cause the display to go back to the text of the file. This command will retrace "Get Label" and "Display Space" traveling but not scrolling or pattern scanning.

Labels and the label space are used primarily as an online table of contents to travel to a desired part of the file quickly and economically. Using labels (and returns) provides random access, allowing the user to jump anywhere in the file, not necessarily in a sequential manner.

7.4 EDITING AND FORMATTING EXAMPLES

FRESS allows the user to both edit and format his text. Editing changes are changes to the actual text in the file, whereas formatting changes only affect the layout of the text. The five principal editing commands are shown below. Note that in the sample session, we have marked the text with daggers (†) to delimit the area of change.

13) INSERT: inserts the text typed in by the user at the point indicated: i <insert point lp> <text to be inserted>. Typing "i at one" would insert the character string "one" immediately after the character string "at" (leaving no blank).

February 2, 1975

14) In FRESS, blanks can separate the parameters of commands. In order to include blanks in a literal string, the user must use a different delimiter (any special character except % or >):

i /at this /point I insert.

15) If an insert is more than one line (about 130 characters), it is much easier to enter Input Mode to do the insert. An insert point is established with a normal insert (in this case, "i/./ Note") and Input Mode is entered with the command "i", or, alternatively, "i/<insert point>/" with null text may be used. After leaving Input Mode, a scroll zero determines the current position in the file: the word before the last line inputted.

There are two methods of using each of the following commands (Delete, Substitute, Move, and Copy). The first method, used when there is a small amount of text to be maneuvered, is to specify a short literal string as the <scope> of the operation.

16) SUBSTITUTE: s(ubstitute) <amount to be substituted for> <new string>. For example, "s(ubstitute) dog beagle" substitutes (i.e., changes) the characters "beagle" for "dog", as in all other editing systems. To delete, substitute, move or copy large quantities of text (over 15 characters) it is convenient to use the second method, omitting the middle of the string by typing an ellipsis instead, with no extra spaces.

Note that in FRESS, unlike in other editors, one may edit more than one displayed line in one operation. Thus, the lefthand side of a "...", for example, may be on the displayed line, and the right on the third line below it on the printout of the file. (With "deferred" <lp>'s one may edit even more; see Section 3.8.)

17) DELETE: d(elete) <amount to be deleted>; removes the text specified by the user. Typing "de text" would remove the first occurrence of the character string "text" from the current line (or below). Deleting a larger string, such as the previous sentence, could be done by specifying "de /Typ...low). ". (Note in the sample session: "lm" (locate mixed) means to scan for the pattern exactly as typed since normally the pattern is scanned for without regard to upper and lower case.)

18) MOVE: m(ove) <amount to be moved> <location to be moved to>; rearranges the text from the location indicated by the user to another point in the file. For example, typing "m point indicated" would place the character string "point" immediately after the character string "indicated", with no blank between the two strings. Move will delete the string from its original location in the text. Moving the string "I can't think of another thing" to follow a string on the line below, e.g., "however," would look like: "m /I...hing/ver, ".

February 2, 1975

19) COPY: co(py) <amount to be copied> <location to be copied to>; copies the text from the location indicated by the user to another point in the file. For example, typing "co user point" would place the character string "user" after the character string "point". Unlike move, copy does not delete the string from its original location in the text.

20) REVERT is another command which is very useful in editing since it allows the user to undo any mistakes he may have made in editing. For example, if he had typed in "i edit text" and the string should have been inserted at "format", then by simply typing "rev" the text will change back to what it originally was and the user may try again.

Formatting allows the user to shape his document using headings, indentations, paragraphs, skipped lines, centered material, tables, etc. Embedded format codes are used to do this.

21) For example, inserting "!-p-" will indicate to the system that the text following the ! sign is a format code (in this case the code for a paragraph). Format codes may be edited just like normal text. The effect of format codes is typically indicated online in a minimal fashion, e.g. a paragraph code will cause a new line to start and an indent of a few spaces. If the user wants to see the full effect online, he can ask the system to do so with the "S(et)V(iewspecs) normal" command; the "P(rint) 3" command shows the old line, the skipped line and the indented line. Also, codes themselves are not printed on final printouts such as this manual (unless the "hypertext" option is specified). After finishing his editing, he might then typically ask for his file to be printed with full formatting on the upper case printer by typing "fu upper".

22) Once the user has completed all his work, he should save his last change and leave FRESS by typing "end". The system will return with "R;" after which the user types: "cp logout". The session is ended and all work done has been saved on disk.

February 2, 1975

7.5 SAMPLE SESSION

User input is preceded by "U:" and system responses by "S:".

1)

U: 1 edit
S: ENTER PASSWORD:
S: ~~XXXXXXXX~~
S: USED=\$10.00, AVAIL=\$1,990.00
S: READY AT 15.33.13 ON 7/1/74
S: CMS VERSION 3.28 - 5/16/74

2)

U: fress
S: FRESS VERSION 7.00 - 5/6/74
S: EXECUTION BEGINS...
S: FRESS READY

3)

U: mf fresdemo sample
S: INPUT
U: A short phrase.
U: This line will later be the last line in the file.
U:
S: FRESS
U: 0
S: phrase. This line will later be the last line in the
U: ff fresdemo
S: PROCEED

4)

U: gf fresdemo sample
U: ti
S: *START OF TEXT AREA*
S: INPUT

February 2, 1975

5)
 U: This section is for demonstration purposes and describes,
 U: by examples, how to use FRESS. It is not a manual
 .
 .
 U: is ended and all work done has been saved on disk.
 U:
 S: FRESS

6)
 U: ds t
 S: *START OF TEXT AREA*

7)
 U: 2
 S: This section is for demonstration purposes and describes, by
 U: 3
 S: will be enclosed within angle brackets (<>) to distinguish
 U: -3
 S: This section is for demonstration purposes and describes, by

8)
 U: 1 text
 S: text, which is typed as shown; explanatory notes are in
 U: 1 text
 S: text file used is basically this writeup itself. For clarity,

9)
 U: gl fress
 S: %L(fress)FRESS allows the user to both edit and format his
 U: gl aardvark
 S: LABEL SPECIFIED DOES NOT EXIST.

10)
 U: m1 edit editlabel
 S: both edit%L(editlabel) and format his text. Editing changes
 U: gl ed
 S: %L(editlabel) and format his text. Editing changes are

EDITING SESSION

February 2, 1975

11)

$$U: \quad ds \rightarrow 1$$

S: editlabel fress logon sample

12)

```
U:  ret
```

S: %L(editlabel) and format his text. Editing changes are

$$U: \mathbb{R}^n \rightarrow \mathbb{R}^n$$

S: %L(fress)FRESS allows the user to both edit and format his

13)

U: 1 INSERT

S: INSERT: inserts the text typed in by the user at the point

U: i at one

S: user at one the point indicated: <insert point lp> <text

+

14)

U: i /at/Ø

S: user at one the point indicated: <insert point lp> <text

+

U: i /at/ ~~bone~~ ~~more~~

S: user at one more one the point indicated: <insert point lp>

+

+

15)

U: 1 /null text may

S: null text may be used. After leaving Input Mode, a scroll

U: i /./ Note

S: null text may be used. Note: After leaving Input Mode, a scroll

 \vdash \vdash

U: i

S: INPUT

```
U: that spaces are taken literally in this mode, without
```

U: the use of other delimiters . . .

U:

S: FRESS

U: -2

S: null text may be used. Note that spaces are taken literally

+

February 2, 1975

16)

U: l dog

S: dog beagle" substitutes (i.e., changes) the characters

U: s dog beagle

S: "s(substitute) beagle beagle" substitutes (i.e., changes) the

U: s /sub...cters/changes the letters

S: beagle" changes the letters "beagle" for "dog", as in the

17)

U: lm DE

S: DELETE: d(etele) <amount to be deleted>; removes the text

U: de text

S: removes the specified by the user. Typing "de text" would

U: de /Typ...low).~~ll~~

S: user. Deleting a larger string, such as the previous

18)

U: l move

S: MOVE: m(ove) <amount to be moved> <location to be moved>

U: m point indicated

S: location indicatedpoint by the user to another in the file.

U: p 7

S: location indicatedpoint by the user to another in the file.

For example, typing "m point indicated" would place the character string "point" immediately after the character string "indicated", with no blank between the two strings. Move will delete the string from its original location in the text.

Moving the string "I can't think of another thing" to follow a string on the line below, e.g., "however," would look

U: m /I...hing~~l~~/however,"~~l~~

S: e.g., "however," I can't think of another thing would

19)

U: l copy

S: COPY: co(py) <amount to be copied> <location to be copied>

U: co /copies...text~~l~~/text~~l~~

S: the text copies the text from the location indicated by the

EDITING SESSION

February 2, 1975

```

20)
U: gl fress
S: %L(fress)FRESS allows the user to both edit %L(editlabel)
U: i /edit/ text
S: both edit text%L(editlabel) and format his text.
      +      +
U: rev
S: %L(fress)FRESS allows the user to both edit %L(editlabel)
      +

```

```
21)
U:  lm Formatting
S:  Formatting allows the user to shape his document
U:  i /user /!-p-
S:  the user
U:  sv normal
U:  p 3
S:  the user

      !-p-to shape his document using headings,
U:  fu upper
S:  FINI
```

```
22)
U:  end
S:  R;

U:  cp log
S:  CONNECT= 00.13.42 VIRTCPU= 000.01.30 TOTCPU= 000.03.38
S:  PRT=130, PUN=0, XFR=0, READ=0, DED SIO=70, PAGEIO=164
S:  LOGOUT AT 15.46.55 on 7/1/74
```

February 2, 1975

TYPES OF PARAMETERS

8 FRESS COMMANDS8.1 FORM OF COMMANDS IN MANUAL

The section for each command begins with a designation of the format of the command. Parameters to be specified by the user take certain standard forms, explained in Section 8.2; they are enclosed in syntactic brackets (" \langle " and " \rangle ") in the command descriptions. The user must specify parameters in the order indicated, using the key delimiter of his choice (blank, /, etc.). For rules on specifying parameters in commands, see Section 8.3. Also, command mnemonics are shown below in upper case for ease of reading; they may be typed in upper or lower case, using any substring of the command name beginning with at least the underscored portion. Parameters followed by a raised o (o) are optional (see below). The symbol \emptyset indicates a blank.

8.2 TYPES OF PARAMETERS

There are certain standard types of parameters in FRESS commands. These are listed below with an explanation of the format and meaning of each one.

 $\langle lp \rangle$

This parameter is 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 1 or more characters which may contain an embedded ellipsis; the last character matched by the string is considered the $\langle lp \rangle$ except in the case of the Insert Before (IB) command or when & is used (see Section 3.15).

 $\langle scope \rangle$

This parameter designates an amount of text to be affected by the command. It is specified either as a context string or as a pair of $\langle lp \rangle$ s identifying the beginning and ending points of the amount, one or both of which may be deferred (see Section 3.8).

<text>

This parameter is a literal character string. It is a piece of text meant to be inserted in the file as specified in the particular command.

<lit>

This parameter is a literal character string not meant to be inserted in the file. Its use and format is determined by the particular command.

<n>

This parameter is a number. Unless otherwise indicated, it is an unsigned number to be considered positive. Its use is determined by the particular command.

<pass>

This is a literal character string of no more than 7 characters which acts as a password to a file. Passwords are used to protect a file from particular commands, for instance to allow a user to scroll through but not edit a file. The password in effect is the one specified in the Get File (GF) command when the file was opened. Any number of passwords may be associated with a file. If no password is specified when the file is created, the password DEFAULT is used and all commands are allowed.

<label>

This is a literal character string of no more than 16 characters which acts as a label in a file. See Section 6.1 for further explanation of labels.

<space>

This parameter is specified as a single character, upper or lower case, representing one of four spaces (see Section 6.3). These are:

T	main text space
W	work space
L	label display space
S	structure space

February 2, 1975

SPECIFYING PARAMETERS

<file>

This is a filename, which is a literal character string of no more than 8 characters. If there is no <file> parameter in a command, the command is executed on the current file.

<vs>

This parameter is a viewspec string. Viewspecs are used to specify how the file is to be displayed and formatted online. For further information on the specification of viewspecs, see Section 6.4.

<options>

This is a character string which may contain certain characters defined as indicating certain options available. The <options> list is unique to and defined in any command which uses it.

8.3 SPECIFYING PARAMETERS

Some functions allow the user to omit values for some parameters. These parameters are called optional and are indicated in the command descriptions by a degree sign (°). There are a few forms which commands may take, and these are described below with examples of the assignment of parameters.

No matter what form a command takes, however, it is logically impossible for the user to specify too many input values (except for functions which take no 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. For example, the Insert (I) command takes only 2 parameters. If the user typed

I/LP/text/more text

then the <text> string to be inserted would be

text/more text

It is possible to specify too few input values. If the number of values specified is less than the number of required parameters, an error message is printed.

All parameter assignment is done in the order in which the input values are specified on the command line. Also, all required parameters are assigned values before any optional parameters are assigned values. Thus when using the Get File command, which is of the form

GF <file> <pass>⁰

if only one parameter is specified, e.g.,

GF/example

the input value is assumed to be the required parameter, namely the <file>. Similarly consider the Set Display command which is of the form:

SD <n1>⁰ <n2>

If the user typed

SD/60

the input value would be assigned to the required parameter, in this case, <n2>.

A few other simple rules may be applied when specifying parameters. As mentioned above, all parameter assignment is done left to right, in the order in which the input values appear on the command line. Thus when a command has a string of optional parameters:

FU <options>⁰ <file>⁰ <pass>⁰ <lp>⁰

the user must specify with his key delimiters which of these optional parameters are to be filled. For example, if the user wished to specify only a filename on a Fullprint, he would type

FU//filename

The extra key delimiter is necessary to indicate the <options> parameter is being omitted. This rule applies only to the omitted optional parameters to the left of the one(s) specified. Optional parameters to the right may be totally ignored, as in the above example.

February 2, 1975

INPUT MODE

8.4 IMPLIED INSERT POINTS

A few commands have an optional <lp> parameter. Except for the Fullprint command, if no value is specifically given to this parameter by the user, the implied insert point is used. For example, the Make Label command is of the form

```
ML <lp>0 <label>
```

If the user specified

```
ML/part2
```

the label 'part2' would be created at the implied insert point.

The user may not always remember where the implied insert point is. To prevent errors, it is wise to use explicit insert points in most cases.

8.5 ENTERING INPUT MODE

Besides the special commands for entering Input Mode (Make File (MF), Bottom Input (BI), Top Input (TI), Swift Input (SI)), certain other commands may be used to enter Input Mode. If the <text> parameter is left null in the Substitute (S) or Insert (I) commands, the user will be placed in Input Mode at the point at which the <text> would be inserted if it had been included explicitly in the command. For example, the Substitute command is of the form

```
S <scope> <text>
```

If the file contained the string 'The word was' and the user specified

```
S/word/
```

he would be placed in Input Mode after 'The '.

When the user leaves Input Mode, the display point is one word before the last line inputted.

February 2, 1975

8.6 ALPHABETICAL LISTING OF FRESS COMMANDS

This section contains descriptions of the most commonly used FRESS commands, arranged alphabetically. Section 12.1 contains a list of the subset of commands which are necessary for editing, and which therefore should be learned first. A full listing of all FRESS commands may be found in the FRESS Reference Manual [4].

Following some of the command descriptions is a list of messages which FRESS may type at the terminal. Some of these messages merely indicate the successful completion of the command, while others are error messages. Many commands do not have a specific message indicating successful completion. In these cases either the text comprising the display window is typed, or a "proceed" is given. Error messages from editing commands and those which may be responses to more than one command appear in Appendix 11 along with those listed in this section.

A

- Accept

ACCEPT

Normally, FRESS makes permanent the result of the previous editing operation each time the current edit is executed. Accept, however, explicitly makes permanent the current editing operation, writing it in the user's file on disk, and thereby disabling the Revert function. This command should be used if a user has to leave his terminal for an extended period of time so that the last edit will not be lost if the computer should drop him.

Messages:

PROCEED

The command has been executed.

February 2, 1975

- Add Password

APASSWORD <pass> <lit>

<pass> is the password to be added
 <lit> is the 'allowable functions list' (see below)

Each password assigned to a file has associated with it an 'allowable functions list', that is, the list of FRESS functions (commands) which the user is permitted to execute when viewing the file using that password. When a file is created with the Make File (MF) command, the password assigned to the file (DEFAULT if none is specified) has an 'allowable functions list' which includes all FRESS commands. Later on, the owner of the file may wish to allow others to use the file (for editing, viewing, etc.) but under protected conditions. For example, he may wish a certain user to be able to display and scroll through all portions of the text but not be able to make any changes in the text.

To accomplish this, the user adds new passwords to his file (a password may be up to seven characters long and contain any characters), designating for each password the functions he desires the particular user who accesses the file with that password to be able to perform.

The following format for the <allowable functions list> is used:

```
<standard string>    <sign>    <function> ... <function>    <sign>
<function> ... function>
```

Blanks must be used to separate the <functions>'s not separated by <sign>'s. Each <function> is indicated by any substring which would be considered legal when actually specifying the command.

System defined <standard strings> are specified by:

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

The <sign> mentioned above specifies whether the function is to be permitted (+) or prohibited (-). A given sign has scope over all function mnemonics encountered before the next sign. Thus, if the user opened a file with a password allowing all functions, he could create a password (in the example below, "pass") which does not allow insert and delete as follows:

```
ap /pass/*-i d
```

February 2, 1975

It is possible to add the system password "DEFAULT" to one's file, allowing it to be accessed in the future without password specification. Specify:

AP default <allowable functions>

Messages:

PASSWORD ALREADY EXISTS

The specified password is already defined in the current file.

INVALID FUNCTION

The <options> list contains an invalid function.

February 2, 1975

B

- Bottom of Space

BOTTOM <space>⁰

This command moves the display pointer to the last display line of the space indicated by <space>, and displays that line. If <space> is not specified, the display will move to the bottom of the space currently being viewed.

- Bottom Input

BINPUT <space>⁰

This command causes input mode to be entered at the bottom of the specified space. Valid spaces are text and work. If no space is specified, the current space is assumed.

Messages:

END OF TEXT AREA

INPUT

The user is now in Input Mode. All text typed at the terminal will be interpreted as literal input until a null line is typed.

February 2, 1975

C• Capitalize Text

CAPITALIZE <scope>

<scope> is the string to be capitalized

This command capitalizes the indicated text. All regular text can be capitalized. As an example, the following display line:

All regular text or jump explainers

and the command line:

CAP reg...ners

would result in:

ALL REGULAR TEXT OR JUMP EXPLAINERS

If a format code is encountered in <scope>, it will be forced to lower case. This is to avoid confusion on upper-case only terminals (see Section 4.3.1). However, the beginning delimiter of the format code (!) must be included in <scope>. Thus given the string

!-P-!.M.the dog

the command

cap/!...g

would result in

!-p-!.m.THE DOG

but the command

cap/P...g

would result in

!-P-!.m.THE DOG

February 2, 1975

- Copy File

```
CFILE <file1> <file2>
```

<file1> is the file to be copied
<file2> is the file to be copied into

CFILE allows the copying of one file into another. Under CMS this command is used strictly for making backup copies, saving intermediate drafts or generating multiple versions of a document.

Messages:

COPY SUCCESSFUL

The indicated copy was completed. The user now has two identical files, <file1> and <file2>.

FILE ALREADY EXISTS

A file by the name of <file2> already exists. The copy was not done.

FILE TO COPY NOT FOUND

<file1> does not exist. The copy was not done.

INVALID FILENAME

The name of either <file1> or <file2> contains an invalid character, generally caused by a telephone line error. The copy was not done, so try again.

- Copy From Work Space

```
CWORK <lp>°
```

<lp> is the point to copy after; if omitted, the implied insert point is used

This command copies all the text currently in the work space (see Copy To Work and Move To Work) into the main text after the point specified. The text also remains unaltered in the work space. If the user does not wish the whole space to be copied, an ordinary Copy should be used, with the <lp> deferred.

February 2, 1975

Messages:

NOTHING IN WORK SPACE TO BE MOVED
The work space was empty.

- Execute CMS Command

CMS <lit>

<lit> is the CMS command to be executed

This command will cause the specified CMS command to be executed. The user must take care not to specify any command, such as LOAD, COMBINE, MOVE, COPY, or COST, which will overlay the FRESS program. For examples of some commands which can be executed using this command see Section 10.2.

Messages:

CMS: NO COMMAND SPECIFIED
The <lit> parameter was omitted.

CMS: <CMS error code>
An error was returned from the specified CMS command.
The error code is printed in decimal. Negative error codes indicate the command was not recognized.

- Copy Text

COPY <scope> <lp>

<scope> is the amount of text to be copied
<lp> is the point to copy after

The string of text specified by the first parameter is copied after the point indicated by the last parameter. The text remains unchanged in its original location.

Applied to the previous sentence, for example, the command

co /unchangedØ/ginalØ

would result in

"The text remains unchanged in its original unchanged location"

February 2, 1975

Format codes may be included in the text being copied; however, to preserve uniqueness, structure codes (i.e. labels) in the original string may not be copied.

To copy text from one file or space to another, the user may defer the "to" point, travel to another file or space using either Get Label, or Get File, and then resolve the last point. For example, to copy the previous paragraph to the start of an existing file (just after the start line, "*START OF TEXT AREA*"), the following sequence could be used:

```
co /Format...copied./?
gf existing
? /*
```

Messages:

HYPERTEXT INTERFILE MOVE NOT SUPPORTED

Labels and other structure may not be moved or copied interfile.

MOVETO LP BETWEEN SCOPE LP'S

that is, the <lp> specified is contained in the <scope> specified.

• Change Password

CPASSWORD <pass>

Once the user has accessed a file, he may switch to a different password under which he wishes to operate on the file, that is, he may change his 'present' password. For example, if he has accessed the file with a password which permits all functions and wishes to let someone view his file without being able to perform any alterations, he may switch to a restricted-function password without the bother of freeing the file and getting it again with the different password. (This lesser password would hopefully not allow a Change Password function!) Note that this function merely switches to a password previously created (using Add Password); it does not create a new password.

Messages:

PROCEED

The command has been executed as specified.

February 2, 1975

INVALID PASSWORD

The <pass> field specified a password not associated with the file.

- Copy to Label

CTLABEL <scope> <label>

<scope> is the amount to be copied
<label> is the label to copy the text after

Copy To Label copies the specified text to the point immediately after the specified label (see Make Label). As with all copy operations, the text also remains in its original location. To gather miscellaneous notes on a single subject, for example, the user might create an appropriate label, then scan through his text copying relevant sections to this label. This command cannot be used to do interfile copies.

Messages:

LABEL SPECIFIED DOES NOT EXIST

The <label> specified does not exist in the current file.

- Copy To Work Space

CTWORK <scope>

<scope> is the amount to be copied

When the user specifies a Copy To Work command, the desired text will be copied to the bottom of the work space, and will also remain in the main text body.

For the difference between Copy and Move, consult those functions themselves.

February 2, 1975

D

• Delete Text

```
DELETE <scope>
```

<scope> is the amount to be deleted

The FRESS Delete command allows the user to delete an arbitrary portion of his FRESS file. Given the string

```
SYSTEM SHUTDOWN AT 2200 FOR TWO HOURS..
```

the command line

```
de TWOØ
```

would result in the string

```
SYSTEM SHUTDOWN AT 2200 FOR HOURS..
```

The use of the ellipses ("Dot Dot Dot") specification simplifies the deletion of longer strings of text. For example,

```
de / SH...WO
```

would result in

```
SYSTEM HOURS..
```

Delete is an example of a function for which qualifiers are useful; for example, if the top of the display buffer were positioned at the start of this paragraph, "de-w us" would delete "useful"; "de-c unct" would delete the "t" in "function". See Section 3.12 for further information on qualifiers.

For a discussion of Hypertext deletion see Section 6.2, Hypertext Editing.

Messages:

BOTH LP'S MUST BE IN SAME DATA FIELD

One cannot delete a string starting in a data field and ending outside of that data field. For the purposes of this manual, data field means label field.

February 2, 1975

CANNOT DELETE START OR LAST END LINE OF SPACE

The <scope> of the specified Delete included the *START OF TEXT AREA* line or the last *END OF TEXT AREA* line in the file.

- Delete Password

DPASSWORD <Pass>

The Delete password command deletes the specified password from the current file. The default password, "default" can be deleted using this command. Care must be taken not to delete all passwords in one's file, after which the file cannot be opened.

Messages:

PASSWORD DOES NOT EXIST

The specified password has not been defined in the current file.

- Display Space

DSPACE <space>°

Display Space is a quick way of traveling to the start of certain system-defined spaces in a FRESS file. The spaces which are currently available (see Section 6.3 for a description of each):

<space>	(description)
T	main text space
W	work space
L	label display space
S	structure space

If no <space> is specified, the current space is assumed.

Messages:

UNKNOWN SPACE

A letter other than T, W, L, or S was specified as <space>.

February 2, 1975

SPECIFIED SPACE DOES NOT EXIST

The specified space (W or L) does not exist. The structure space (S) and text space (T) always exist.

- Display Viewspecs

DVIEW

This command allows the user to see what the current "viewspecs" are (see <vs> parameter definition). The viewspecs determine what the user will and will not see when displaying his file. For example, he can turn off the Online Formatting capability of the system, or prevent the display of formatting or structure codes.

The viewspecs are expressed in terms of one of 4 standard viewspec strings and additions or deletions to these strings. These four standard strings correspond to the standard strings given in the Set Viewspecs command:

VS1	normal
VS2	print
VS3	edit
VS4	null string

February 2, 1975

E

- End FRESS Session

END

The End command closes all open FRESS files (equivalent to free-filing (see FFILE below) all files being used) and ends the FRESS editing session. The user is returned to the CMS command environment.

February 2, 1975

F

• Free File

```
FFILE <file>
```

This command makes permanent the last editing operation specified and "frees" the specified file. Note that Accept makes the change permanent also but does not close the file.

Messages:

PROCEED

The file has been freed as specified.

FILE NOT IN USE

The file specified was not open, that is, had not been retrieved.

• Flip Case

```
FLIP <scope>
```

<scope> is the text to be 'flipped'

The character case of each character of the string specified by <scope> will be flipped; that is, upper case characters will become lower case and vice versa. There are two exceptions to this rule:

- 1) Any format codes and macro names encountered will be uncapitalized to avoid confusion on upper-case only terminals (see Section 4.3.1). However, the beginning delimiter of the format code (!) must be included for this to be true. See the Capitalize command for an example.
- 2) 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 will be in upper case. However, the "sentence-ending" punctuation or the format code must be included in the <scope> for this to be true. See the Uncapitalize command for an example.

February 2, 1975

- Make Footnote

FOOTNOTE <scope> <n>^o

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

This command causes the specified text to become a footnote by inserting footnote format codes (!-f-) around the text. If a footnote number (n) is specified, !-fn- will be inserted in front of the text. For example, if the display were located at the start of this paragraph:

fo/command...become/3

would result in:

This !-f3-command causes the specified text to become!-f- a footnote.

Messages:

LITERAL CONTAINS INVALID TEXT

The second parameter of the command is not a number.

February 2, 1975

- Full Printout

FULLPRINT <options>⁰ <file>⁰ <pass>⁰ <lp>⁰

This command is used for paginated and fully formatted hard copy output. If no parameters are specified the current open file is printed to the offline printer.

<options> = <option,option,...,option>

<u>A</u> T<n>	the first page printed is numbered <n>. If AT<n> is not specified the default value is FROM<n>. AT0 suppresses all page numbering for that printout.
<u>F</u> ROM<n>	the first page printed on the output device is page <n> i.e., output from previously formatted pages is suppressed. <n> is the nth <u>physical</u> page, not the nth <u>numbered</u> page.
<u>H</u> YPER	format codes are printed along with the text and are also interpreted. This command is useful when one wants to debug one's format codes, but overlaying may occur with tabs (see section 5.11).
<u>M</u> ACROS	macros (inserted by user in the file) are used to determine the format of each decimal level (number of spaces indented, heading format.) See section 2.6 in the <u>FRESS Reference Manual</u> [4] for further explanation.
<u>O</u> FFSET<n>	shifts the output <n> spaces to the right on the output device. Note that this offset is performed on all pages and is not related to the OFFSET ALTER code. <n> plus the value of the WIDTH ALTER code should not exceed 110.
<u>P</u> DISK	creates a file of filetype "print" on the user's CP/CMS P-DISK. The filename is the same as that of the FRESS file printed. The output file must be printed on the offline printer using the CMS "offline printcc" command.
<u>S</u> TOP	waits for a carriage return before printing the next page. If STOP is specified, the user's output is automatically routed to his terminal. This option is useful for printing on non-continuous forms, e.g., letterhead on a typewriter-like terminal.
<u>T</u> O<n>	the last page printed is <n>, i.e., the total number of pages printed is TO<n>-FROM<n>+1
<u>U</u> PPER	translates to upper case
<u>V</u>	translates special characters so that they print on a terminal as an analogous character (i.e., "[" prints as "(") rather than as a blank when using the 2741 option (below).

February 2, 1975

2741 print to user's terminal. When this option is specified the user must wait for a 'proceed' signal, meaning the Fullprint program is ready to start printing. The terminal paper should be adjusted if necessary. The user must hit carriage return before the Fullprint will start.

1403 print to 1403 (offline) printer

Defaults:

fu /FROM1,T032000,AT1,1403,OFFSET0

For minimally formatted hard copy output which includes both format codes and structure, use Offline Type. For further information, see the explanation of that command.

<file> <pass>

The user may override printing of the current open file by specifying a filename and, if the file is password protected, a password.

<lp>

The user may begin printing at any location in his file by specifying a lightpen hit. Page numbering will begin at 1. Because the Fullprint program runs independently of any online formatting, the user must be sure that Fullprint knows about any macro definitions he has embedded in the file (see Section 5.6). The optional <lp> parameter is not defaulted to the implied insert point.

Messages:

FINI

The Fullprint completed and has been sent to the printer.

INV OPTION

One of the <options> was invalid. The command was ignored.

Any format code error message (see Section 5.7).

February 2, 1975

G

• Get File

```
GFILE <file> <pass>°
```

This command has two purposes. First, it is used to gain initial access to an existing file after entering the FRESS environment. (If creating a new file, a user will issue the Make File command.) FRESS will open the specified file, and the display buffer is set at the "*START OF TEXT AREA*" line, but no text is printed. The user may then utilize any FRESS function within the capabilities of his password. No <pass> need be specified if the file has the system default password ("DEFAULT").

The second purpose is to access a file which has previously been opened with a Get File. This is done when switching among multiple open files.

There is currently no limit on the number of open files other than the availability of machine 'core'. This may be altered in other versions.

Messages:

INVALID PASSWORD

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.

FILE NOT FOUND

The specified file does not exist.

• Get Label

```
GLABEL <label> <file>°
```

To travel quickly and conveniently from place to place in a FRESS file, one can label various points (using Make Label; see below). Get Label causes FRESS to locate the point in the file at which the label is defined and to start the display at that point.

February 2, 1975

A substring of the label may be specified, and the (alphabetically) first label beginning with that substring will be retrieved.

The second parameter (<file>) is normally omitted, which means that only the FRESS file currently being displayed is searched for the label. However, if a filename is specified, that file (which must be open at the time) is searched for the label. If an asterisk (*) is given for the filename, then all open files are searched for the label. Thus Get Label provides a convenient means of interfile traveling.

Messages:

LABEL SPECIFIED DOES NOT EXIST
 <label> is not defined in <file> or the current file if
 none is specified.

February 2, 1975

H

- Help

HELP <option>°

The Help command provides online information on new or recently changed facilities in FRESS. If "LIST" is specified as the <option>, FRESS will print at the user's terminal a list of the items available. If one of these items is specified as the <option>, the available information concerning that item will be printed at the user's terminal. Specifying "PRINTALL" as the <option> will cause all available information to be printed on the offline printer. If no <option> is given, FRESS will print at the user's terminal an explanation of how to use the Help command.

After the specified information has been typed, the display point remains where it was before the Help command was used.

Messages:

NO HELP AVAILABLE FOR THAT

An <option> was specified which does not correspond to any available information.

February 2, 1975

I• Insert Text

INSERT <lp>⁰ <text>

<lp> is the point to insert after; if omitted, the implied insert point is used
<text> is the character string to be inserted; if left null, Input Mode is entered

The Insert command allows the insertion of an arbitrary length string of text after a specified location in the user's file. Given the fragment

See Spot run

the command

i/run/.

will create the sentence

See Spot run.

Of course, longer inserts are possible. One might specify:

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

yielding

See Spot, Dick and Jane's pooch, run.

For multiple line inserts, the user may omit the <text> parameter, which will place him in Input Mode after the indicated <lp>. If no <lp> is specified, the implied insert point is used.

Messages:

INPUT

The user is now in Input Mode. All text typed at the terminal will be interpreted as literal input until a null line is typed.

LABELS LIMITED TO 16 CHARACTERS

The result of the Insert would produce a label longer than 16 characters.

NO REGULAR TEXT ALLOWED IN STRUCTURE SPACE

The user attempted to insert text directly into the structure space and was not merely editing what was

February 2, 1975

already there, for example, fixing the spelling of a label.

- Insert Before

<u>I</u> BEFORE <lp> <text>

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

The Insert Before command is similar to the Insert command, except that it inserts the text before the specified <lp> instead of after it. In this case, the <lp> is considered the first character specified in the context string. Insert Before does not use the implied insert point as does Insert, nor does it set the implied insert point. The user may not enter Input Mode using Insert Before. Insert Before may not be used in the label space.

February 2, 1975

L

- Locate (Pattern Scan)

<u>LOCATE</u> <lit> ^o

<lit> is the literal character string to be located

Other Locate commands may be created using the modifiers Long, Backwards, and Mixed, in any order:

<u>LB</u>	<u>Locate Backwards</u>
<u>LL</u>	<u>Locate Long</u> (to bottom of area)
<u>LM</u>	<u>Locate in Mixed mode</u> (take specified character case literally, without folding)
<u>LBL,LLB</u>	<u>Locate Backwards Long</u> (to top of area)
<u>LBM,LMB</u>	<u>Locate Backwards in Mixed mode</u>
<u>LLM,LML</u>	<u>Locate Long to bottom of area in Mixed mode</u>
<u>LBLM,LBML,LLBM,LLMB,LMBL,LMLB</u>	<u>Locate Backwards to top of area in Mixed mode</u>

The pattern scanning facilities of FRESS allow the user to search a FRESS file for the first occurrence of the pattern specified and to begin the display at the start of the matched string. If the simple form of locate is specified without a text parameter, i.e., just typing 'l', the previous locate command, regardless of form, is repeated. If any other form of locate is specified without a <lit> parameter, e.g. just typing 'lb' or 'lm', the pattern specified in the previous locate command (of any form) will be used. If no previous locate was done, an error message is printed. As with the specification of Location Points and amounts of text (Scopes), the pattern may contain the ellipses (...) construct, but this will take considerably more execution time!

By use of the various forms of the locate command, the user is able to control how the pattern is searched for. For Locate, the pattern is matched without regard to upper and lower case ("folded"), and the scan is in the forward direction for up to 8000 characters. If a "B" is used in the command name, the scan is done in a backwards direction for 8000 characters. If a second "L" is used in the command name, the scan will continue until the end of the area (top or bottom) regardless of length. If an "M" is used in the command name, the pattern is scanned for exactly as typed (in "mixed mode"), i.e. without folding.

February 2, 1975

Messages:

PATTERN NOT FOUND, EO COUNT

The specified pattern was not found in 8000 characters from the current display point. This condition will not occur when the "L" modifier is used.

PATTERN NOT FOUND, EO AREA

The specified pattern was not found within the limit of the area in the direction specified.

INVALID PATTERN, TRY AGAIN

Certain configurations of ampersands with other characters are considered invalid patterns. The ampersand is used to indicate a modifier to the pattern, a facility which is not yet fully implemented and will be explained in a future update of this manual.

UN LIGHTPENNABLE TEXT

The <lit> parameter of a Locate command 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.

February 2, 1975

M• Move Text

MOVE <scope> <lp>

<scope> is the amount of text to be moved
<lp> is the place to move the text to

This command behaves much like Copy, except that the specified text is deleted from its original location. A Move of a large block of text is often done using DEFERRED LP's, discussed in Section 3.8. Considering a small example, modifying:

This is a DATEL terminal.

with the command

mo/ DATEL/al

will result in

This is a terminal DATEL.

Labels can not be moved between files.

Messages:

HYPERTEXT INTERFILE MOVE NOT SUPPORTED

Labels and other structure may not be moved or copied
interfile.

MOVETO LP BETWEEN SCOPE LP'S

that is, the <lp> specified is contained in the <scope>
specified.

• Make File

MFILE <file> <pass>⁰

This command creates a FRESS file with the specified name and password. 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.

February 2, 1975

After issuing a Make File, the user is placed in Input Mode and can immediately start inserting text into the start of the file, which will be placed after the "*START OF TEXT AREA*" line.

Messages:

INPUT

The user is now in Input Mode. All text typed at the terminal will be interpreted as literal input until a null line is typed.

FILE ALREADY EXISTS

A file by the specified name already exists.

• Move From Work Space

MFWORK <lp> ⁰

<lp> is the point to move the work space text to; if omitted, the implied insert point is used

Move From Work allows the user to move ALL the text in his work space to a specified place in his file. If the user desires to move only sections of the text in the work space, he should use the ordinary Move instruction with deferred <lp>'s.

Messages:

NOTHING IN WORK SPACE TO BE MOVED

The work space was empty.

• Make Label

MLABEL <lp> ⁰ <label>

<lp> is the place to make the label; if omitted, the implied insert point is used

Labels, as mentioned under Get Label, provide a convenient means of identifying important points in a FRESS file which can later be accessed directly without having to pattern-scan for them. The label appears in-line in the text as "%L(LABEL)", and is entered into the Label Display Space, which is accessible through the Display Space command. A label may be from 1 to 16 characters, may contain embedded blanks, and may be edited.

February 2, 1975

Messages:

LABEL ALREADY EXISTS

(ML) The label specified in a Make Label command already exists.

LABELS LIMITED TO 16 CHARACTERS

The <label> specified was longer than 16 characters.

NULL LABEL IS ILLEGAL

The user attempted a Make Label with a null <label> parameter.

- Move to Label

MTLABEL <scope> <label>

<scope> is the amount of text to be moved

<label> is the label after which to move the text

This command allows the user to move a piece of text to a position beginning immediately after the specified label. The command behaves precisely like Move in all respects and is useful for gathering diverse fragments at a common origin, similar to work space functions in the work space. Move To Label does not work interfile.

Messages:

LABEL SPECIFIED DOES NOT EXIST

The <label> specified does not exist in the current file.

- Move To Work Space

MTWORK <scope>

<scope> is the amount of text to be moved

A work space is available to the FRESS user (see Section 6.3). The Move To Work command allows the user to move a specified piece of text to the bottom of in the work space. Within the work space, the user may perform all normal FRESS functions upon the text he has there.

February 2, 1975

When finished manipulating text in the work space, the user may return the entire collection to the main space of his file using the Move From Work or Copy From Work command.

February 2, 1975

0• Offline Read

OREAD <options> <filename> <password>⁰

The OFFLINE READ command creates a FRESS file from one of two sources of input as a means of cheap input mode: 1) from cards in the virtual card reader, or 2) from some specified disk file. Input to the command is of two main types, card image input and line input. Card image input is input coming from cards in the virtual card reader or from card images in a fixed length disk file. Line input is input coming from a variable length disk file. Note that some of the options listed below apply only to input of one of the above types. The options may be specified in any order, separated by commas. Each option specified cancels any previously specified option with which it conflicts. Only the first character of each option is looked at. The following is a list of options:

Reader

specifies that the input to the file is to come from the virtual card reader. No blank inserting is done after each card. That is, the text is considered to be one continuous stream, except that all trailing blanks except one on a card will be ignored. Blank cards will be ignored completely.

Disk[.<filename>[.<filetype>[.<filemode>]]]

specifies that the input to the FRESS file is to come from the specified CMS file. All three parameters are optional and default to the FRESS filename, SCRIPT, and * respectively.

Lower

specifies that the input to the file will be in both lower and upper case. This is the default input mode.

Upper

specifies that the input is in upper case only, which means that the input will be translated to lower case and capitalized as indicated.

CapitalizeK

specifies that the capitalize character will be K, which can be any character at all. Example: C* specifies that the capitalize character is to be a *. In the input stream, a single capitalize character specifies that the next character is to be capitalized, which is useful when U is specified as an option, since

February 2, 1975

everything is translated to lower case. A string of text enclosed by double capitalize characters is all capitalized. The default capitalize character is the %.

HyphenK

specifies that the logical hyphen character is to be the character K, which may be any character. The logical hyphen character pertains only to line input, and when placed at the end of an input line, causes no blank to be inserted after the line. This is useful when breaking a word across a line boundary. The default hyphen character is the %.

Startnn

specifies that input starts in column nn for a card image input item. The default starting column is 1. This option applies only to card image input.

Endnn

specifies that input ends in column nn for a card image input item. This option is useful for ignoring sequencing numbers at the end of a card. The default ending column is 80. This option applies only to card image input.

Bottom

specifies that the input is to go at the bottom of the current FRESS file.

The complete list of default options is: d,l,c%,h%,s1,e80 on an upper/lower case terminal, and d,u,c%,h%,s1,e80 on an upper case only terminal.

If the FRESS filename is not specified, it will be defaulted from the CMS filename if d is specified. If no filename at all is specified it will be defaulted to ((TEMP)). The file ((TEMP)) FRESS is erased before the new one is created.

If r is specified and the card reader is empty, the Offline Read command will wait for the card reader to fill before continuing. When this occurs, the message WAITING FOR CARD READER TO FILL will be typed. It should be noted that a multicap field will be treated as a stream by Offline Read, as opposed to the line-oriented treatment of multicaps by normal Input Mode. In other words, a multicap field is implicitly ended by Input Mode, even if the closing delimiters are not included, whereas Offline Read will not end the field until a new capitalize delimiter is encountered.

February 2, 1975

Messages:

STARTING COLUMN > ENDING COLUMN

An error has been made in the specification of column positions from card image input.

STARTING COLUMN < 1

An error has been made in the specification of column positions from card image input.

ENDING COLUMN > 80

An error has been made in the specification of column positions from card image input.

FILE NOT FOUND

The CMS file specified in the Disk <option> does not exist.

FILE ALREADY EXISTS

A FRESS file with the given filename already exists.

- Offline Type

O^UTYPE <space>°

This command will print on the offline printer the entire specified space exactly as it is displayed online (under the current viewspecs and line width). If no <space> is specified, the space currently being displayed is used. This command is useful for debugging one's format and structure codes. Note that the printed output will appear in upper and lower case to reflect the actual contents of the file even if an upper case terminal is being used.

Messages:

FINIS

The Offline Type was completed and has been sent to the printer.

PLEASE FREE A FILE & TRY AGAIN

Offline Type had insufficient space to complete. As indicated, the user should free one or more files (using Free File) and try the Offline Type again.

February 2, 1975

PRINTER TROUBLES

There is some difficulty in communication with the
Offline Printer. Contact a FRESS staff member.

February 2, 1975

P• Print

PPRINT <n>

This command is used to print online at most "one display buffer's worth" of text. Type or Fullprint must be used for longer online printouts. Print does not move the start of the display buffer. The maximum number of lines which may be printed is determined by dividing the buffer size (500 for a typewriter-like terminal, 700 for a display console) by the current line length (set by a Set Display command). Thus with the default line length of 50 on a 2741, up to 10 lines may be printed.

Messages:

MAX EXCEEDED

The number of lines in the display buffer is less than <n>. Specify a smaller number.

• Pack File

PFILE <n>°

<n> is the percentage each page is to be filled

This function causes the internal structure of the current file to be altered such that each internal page or record is filled according to the percentage specified. The file must be open and is freed after packing. If no percentage is specified, 100% is assumed. No percentages below 50% are allowed. Internal FRESS pages are usually balanced at approximately 66% full.

A typical command would be:

PF 85

which would re-structure the current file so that each internal page would be as close as possible to 85% full.

Pack File 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. 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 require

February 2, 1975

the formation of new internal pages. However, one might want to pack a file to perhaps 85% capacity if only very minor typographical corrections were to be made subsequently.

When the packing has been completed, the message 'PACK SUCCESSFUL' will be typed, followed by two numbers indicating the size of the file in pages before and after packing. Thus the response might look like:

PACK SUCCESSFUL 0034/0025

which would indicate the file had 34 pages before packing, 25 pages after packing.

Note that this command can not be reversed using the Revert command.

Other Messages:

PACKED PAGES MUST BE MORE THAN HALF FULL
A percentage (<n>) of less than 50 was specified.

TOO MANY FILES OPEN
Because of space limitations, it is necessary to close some files, using the Free File command. The Pack File command may then be repeated.

February 2, 1975

Q• Query Current Files

QUERY <option>°

The Query command is used to display the file name of and location in the user's current FRESS file (<option> = L, the default). The location is specified in terms of space name and area number. It can also be used to display the file names of all currently open files (<option> = F). This command is very useful when the user has forgotten the names of the files he currently has open, since the file name must be specified in order to free the file (see Free File command).

Messages:

FILE: <filename>
SPACE: <space name>
AREA: <n>

This provides the requested information from a "Q L".

CURRENT: <filename>
<filename>
<filename>

This is the format of the reply to a "Q F".

BAD OPTION PASSED TO QUERY

Query was specified with an <option> other than "L" or "F".

NO FILE OPEN

Either the user has not executed a Get File, or all open files have been freed using Free File.

February 2, 1975

R

- Return

RETURN

When a FRESS command causes the user's display to move nonlinearly (Display Space, Get Label, Bottom, Move in Transcription Mode, Get File, and Make File commands) or if the user explicitly saves a display point (see Save), the system automatically "pushes" the current display point into a LIFO (last in - first out) stack called the Return Stack. Specifying Return causes the system to "pop" the stack and to relocate the display buffer at the popped point. Using the stack to mark his path through a text, the user can retrace this path.

Messages:

RETURN FILE NO LONGER OPEN

The file of the next display point in the stack has been freed. The user may continue popping the stack with Return.

SPACE OF RETURN POINT HAS BEEN DELETED

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 Return.

NO MORE RETURN POINTS

There are no more entries in the Return stack, and therefore no more Returns can be done.

- Revert

REVERT

After performing a FRESS editing operation, the user is able to undo the edit by issuing Revert. Only the last editing operation preceding the Revert command may be undone, and Accept neutralizes the ability to revert.

Traveling, both linear (scrolling, pattern scanning) and nonlinear (getlabels), may be done between an edit and a revert without negating the revert capabilities. Revert also reverts the starting point of the display buffer to the point in the file at

February 2, 1975

which the now reverted editing operation was specified (or completed in the case of a deferred specification).

February 2, 1975

S

• Substitute Text

```
SUBSTITUTE <scope> <text>
```

<scope> is the text to be substituted for
 <text> is the literal string to be substituted; if omitted, Input Mode is entered

Substitute allows the user to replace a string of text in a file by a new string. This command is useful for typographical corrections. For example, to change "useful" to "useless" in the previous sentence, the user could type

```
s useful useless
```

Substitute is also useful for editing longer strings of text with an ellipsis. Consider the following:

```
FRESS is an amazingly convenient text editor.
```

To substitute "a reliable" for "an amazingly convenient", the user need only specify:

```
s /a...nt/a reliable
```

See also Uniform Substitute.

Messages:

INPUT

The user is now in Input Mode. All text typed at the terminal will be interpreted as literal input until a null line is typed.

BOTH LP'S MUST BE IN SAME DATA FIELD

One cannot substitute for a string starting in a data field and ending outside of that data field. For the purposes of this manual, data field means label field.

CANNOT DELETE START OR LAST END LINE OF SPACE

The <scope> of the specified Substitute included the *START OF TEXT AREA* line or the last *END OF TEXT AREA* line in the file.

February 2, 1975

LABELS LIMITED TO 16 CHARACTERS

the end and/or intermediate result of the Substitute would produce a label longer than 16 characters.

NO REGULAR TEXT ALLOWED IN STRUCTURE SPACE

The user attempted to insert text directly into the structure space and was not merely editing what was already there, for example, fixing the spelling of a label.

- Save Current Location

<u>SAVE</u>

Save pushes the current display point into the Return Stack (see Return). It is useful if the user wants to look for some other point in his file (by scrolling or pattern scanning) and still be able to return easily to the original point. (Normally, only nonlinear traveling functions causes his original location to be remembered.) For example, specifying "SA>20>R" will bring him back to his original viewing point after scrolling forward 20 lines.

- Scroll

<n>	or	<u>SCROLL</u> <n>
-----	----	-------------------

<n> is the signed or unsigned number of lines to scroll

Scroll moves the display the specified number of lines of the current line length (see Set Display). 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.

No command name need be specified to scroll. Examples:

4 (travels forwards 4 lines)
-25 (travels backwards 25 lines)

NOTES: A scroll of zero lines is a way to re-display the current line. No spaces are allowed between sign and digits. A maximum of 127 lines may be scrolled at a time. Forward and backward scrolls are not counted in exactly the same manner. Thus, a forward scroll of n lines followed by a backward scroll of n lines may not return the display point to the same place.

February 2, 1975

Messages:

MAX EXCEEDED

A scroll of more than 127 lines forward or backward was specified.

• Set Display Window

SDISPLAY <n1> ⁰ <n2>

<n1> is the number of lines in the display window

<n2> is the length of each line in the display window

Using this command the user may specify the number of lines and the line length of the text which is displayed after each function is executed (assuming the DISPLAY mode is in effect; see Section 4 and Set Mode below). The default of the system for typewriter-like terminals is one line of 50 characters. The command

sd 2 120

would result in two lines of 120 characters being displayed every time a display buffer was generated. The only restriction is that the line length times the number of lines must be less than the display buffer size (500 characters for typewriter terminals). See Section 4.3.3 on terminal characteristics for defaults and buffer sizes for other terminals.

Messages:

BUFFER SIZE EXCEEDED

The line length (<n2>) times the number of lines (<n1>) is greater than the display buffer size (see Section 4.3.3). Use smaller limits.

LENGTH EXCEEDS DEVICE MAX

The line length is greater than the maximum line length for the terminal.

February 2, 1975

- Scratch file

SFILE <file>

Scratch File erases the specified file. The file must be the current file in order to be scratched. A Get File must be specified to access another of the open files. Note that this command can not be reversed using the Revert command.

Messages:

SCRATCH SUCCESSFUL

The command was executed as specified.

FILE MUST BE CURRENT TO BE SCRATCHED

The user should retrieve the file with a Get File command and then attempt the Scratch again.

SCRATCH FAILED

A serious error has occurred; contact a member of the FRESS staff.

- Swift Input

SINPUT <lp>

<lp> is the place to insert text; if omitted, the implied insert point is used

This command places the user in Swift Input Mode, which is a faster and less expensive form of Input Mode. See Section 3.11 for a complete discussion of the differences. See also Swift Input Bottom (SIB) and Swift Input Top (SIT) commands.

Messages:

INPUT

The user is now in Input Mode. All text typed at the terminal will be interpreted as literal input until a null line is typed.

February 2, 1975

- Swift Input Bottom

SIBOTTOM

This command will put the user into Swift Input Mode at the bottom of the current file, that is, just before the ****END OF TEXT AREA**** line.

Messages:

END OF TEXT AREA

INPUT

The user is now in Input Mode. All text typed at the terminal will be interpreted as literal input until a null line is typed.

- Swift Input Top

SITOP

This command will put the user into Swift Input Mode at the top of the current file, that is, just after the ****START OF TEXT AREA**** line.

Messages:

START OF TEXT AREA

INPUT

The user is now in Input Mode. All text typed at the terminal will be interpreted as literal input until a null line is typed.

- Set Mode of Display

SMODE <option>°

Set Mode allows the user to select which combinations of DISPLAY or BRIEF, and STATIC or TRANSCRIPTION modes he desires (see Section 4).

<option> may be any string of the following symbols concatenated together (with no intervening blanks):

February 2, 1975

SYMBOL	MODE	MEANING
B	Brief	display off, nothing printed
D	Display	the amount of the display buffer specified by the last Set Display command (or system default) is printed after each command
S	Static	buffer does not move with editing operations (used primarily on display consoles)
T	Transcription	buffer moves to one word or code before the beginning of the editing operation performed (system default, used primarily on typewriter terminals)

"DT" is the default option. The <option> string is interpreted letter by letter from left to right. Any invalid letters are ignored. In addition, conflicting modes (ST or BD) are resolved by using the right-most mode in the string. For example, if the user meant to set his viewspecs (SV) but typed "SM print" by mistake, the string "print" would set transcription mode because of the ending "t", while all other letters would be ignored, since they are invalid mode characters.

The mode set may be temporarily overridden by using the mode indicators at the end of a command line, after two blanks and a "." (see Section 4.2). To suppress display for one command, for example, just use this "blank blank period" convention.

Messages:

INVALID MODE CHARACTERS WERE IGNORED

Conflicting options, or characters other than B, D, S, or T were specified in the <option> string. The invalid modes were ignored.

• Surround Text

```
SURROUND <scope> <text1> <text2>^
```

<scope> is the text to be surrounded

This command surrounds the specified text with the specified literal text as follows. For the display line:

text regular text or explainers

the command line:

February 2, 1975

sur regular...ners ()

would result in:

text (regular text or explainers)

If <text2> is omitted, <text1> is used in both places.

Messages:

TEXT TO INSERT MUST BE SPECIFIED

Either <text1> or <text1> and <text2> must be specified.

• Set Viewspecs

SVIEW <vs>

The Viewspecs determine how the text displayed online is presented and formatted. The system default (in the single window version) is for no online formatting, with all formatting codes beginning a new line and all structure codes (i.e., labels) shown in-line.

The viewspec string is in the format explained above in Section 6.4. The default for the single window (non-IMLAC) version is EDIT viewspecs.

To obtain a short formatted printout of a section of a file, the user should locate the section and then do a "sv print" followed by a "type 100" to see 100 lines.

Messages:

ERROR IN STRING NEAR POSITION <number>

This means an invalid viewspec has been included in <vs>. The error occurs near the character with displacement <number> from the beginning of the string.

February 2, 1975

I• Type

TYPE <n>

<n> is the number of lines to be typed

The TYPE command allows the user to have any number (less than 1000) of lines typed at his console. Unlike Print, the last printed line is the start of the display after the completion of the command.

• Top Input

TINPUT <space>⁰

This command causes input mode to be entered at the top of the specified space. Valid spaces are text and work. If no space is specified, the current space is assumed.

Messages:

START OF TEXT AREA

INPUT

The user is now in Input Mode. All text typed at the terminal will be interpreted as literal input until a null line is typed.

February 2, 1975

U

- Underscore

UNDERSCORE <scope>

<scope> is the amount of text to be underscored

The Underscore function inserts underscore format codes (i.e., `!(0...!)`) around the specified text.

- Uncapitalize

UNCAPITALIZE <scope>

<scope> is the character string to be put in lower case

The specified string will be put in lower case. However, the first non-blank character after any "sentence-ending" punctuation (period, question mark, or literal exclamation point) or after any format (macro, alter, or edit) code will be capitalized so the <scope> may include multiple sentences where the first character of the sentence is to be capitalized. However, if the <scope> does not include the format code or sentence-ending punctuation, the following letter will not be capitalized. Thus given the string

!-p-THE QUICK BROWN FOX

the command

unc/!...X

would result in

!-p-The quick brown fox

but the command

unc/TH...X

would result in

!-p-the quick brown fox

Note that this command ignores the original case of the string, so it may find a string which is already in lower case.

February 2, 1975

- Uniform Substitute

USSUBSTITUTE <lit> <text> <options>°

<lit> is the amount of text to be substituted for
 <text> is the character string to replace it

Uniform Substitute allows the user to repeat a normal text substitute an arbitrary number of times. The user can specify that the substitute should be made a specified number of times or uniformly throughout the current file, he can optionally accept or reject each substitute and escape from the command at any time.

The first two parameters are the same as for Substitute. The first parameter is the pattern to be substituted for. Uniform Substitute will do a Locate for the pattern when the next substitute is to be done. The <options> parameter then determines how the Uniform Substitute is to be executed and displayed. The allowable options are:

- I Inquire after each substitution
- A perform each substitution Automatically
- D use Display mode to show the effect of each substitution
- B use Brief mode to suppress display of each substitution
- L do a Long scan for the <lit> pattern
- n perform the substitution "n" times only

The Inquire option means that the user must explicitly accept or reject each substitution in order of occurrence, and may exit from the command at any time. The Automatic option means that the substitution proceeds without user intervention, and ends only when no further occurrence of <lit> remains. Option "n", if used, must be the last option in <options> and should be less than 4095. An important note, regardless of which options are in effect, is that the Revert command can not be used to undo a Uniform Substitute.

As an example, 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 editing buffer location, and displaying each line where the substitution occurs.

If no <options> are specified in the command, Uniform Substitute will use defaults of Inquire, Display, and a count of 4095. If <options> are specified, these options will be used in addition to the static/transcription mode currently being used (see

February 2, 1975

Set Mode command). The <options> string is scanned from left to right. If conflicting options (e.g., AI) appear in the string, the rightmost option is used.

In Inquire mode, after each replacement, "OPTIONS=" will be displayed at the terminal. The user then responds with either A (accepting the substitute) or R (rejecting it). Immediately following A or R, the user may type X to exit from the Uniform Substitute command and return to normal Command Mode. If X is not specified, the user can optionally follow the A or R with new <options> characters.

If the 'l' option is not specified, only 2100 characters will be scanned for <lit>. If the pattern is not found, the message

PATTERN NOT FOUND, OPTIONS=

will be typed. The user may reject the command ("X" or "R") or continue with a long scan to the end of the file ("A").

Messages:

INVALID PATTERN, TRY AGAIN

This message occurs if the <scope> pattern started with an ampersand (&), which is not allowed.

UNLIGHTPENNABLE TEXT

The <scope> pattern has matched some text in the display buffer which does not reflect any real text in the file. An example is the "***END OF LABELS***" line at the bottom of the label space.

PATTERN NOT FOUND, OPTIONS=

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).

END OF AREA

The pattern scan has encountered the "*END OF TEXT AREA*" line; no more substitutes can be done.

HYPERTEXT ENCOUNTERED: ACCEPT(A), REJECT(R), OR TEXT ONLY(T)
See Section 6.2.

UNRECOGNIZED RESPONSE, PLEASE RESPECIFY

The options string typed by the user contains invalid option characters.

February 2, 1975

END OF COUNT

The user specified the <option> giving the number of times to perform the substitution. The specified number have now been done.

OPTIONS=

A substitute in Inquire mode has been performed. The user must now specify one of the options described above.

FRESS

The Uniform Substitute has completed and control has returned to the user in Command Mode of FRESS.

8.7 FRESS HOUSE FUNCTIONS

A certain group of commands are called "house" functions (because they perform "housekeeping" functions for the command language interpreter). These commands are specified by using an ampersand (&) followed by the name of the command. As with other FRESS commands, only the first few letters of the command name (as shown by underscores below) are needed. If an operand is necessary, the command name is followed by one blank and then the operand. House functions may not be stacked on command lines using the FRESS command separator (>).

- Enter Asis Mode

&ASIS

Issuing this command causes the following to happen to input values (lines in Input Mode, and all <text> parameters): each is preceded by !-s0- and each occurrence of the tab key character (from the physical tab key) is replaced by !-T-. See Section 5.11 for the use of &A with table formatting.

- Clear Function Area

&CLEAR

Issuing this command clears all pending functions and deferred <lp>s and <scope>s. It should be used if the user has a function with a deferred <lp> or <scope> and decides that he does not want to complete the function.

February 2, 1975

Messages:

NO OUTSTANDING IMCOMPLETE FUNCTIONS

There are no deferred <lp>'s or <scope>'s or pending functions to be cleared.

• Repeat Last Command Line

&GIN <n>°

<n> is the number of times to repeat

This command repeats the entire last command line (which was not an '&g') for the specified number of times. If no operand is specified, '1' is assumed. This function is useful for iterating a pattern scan, a substitute, or a delete. If an error is encountered during the iteration, execution of the command halts immediately.

An example of this command follows. If the user is reading a small section of text, he might print 10 lines, read them, then desire to scroll forward and print the succeeding 10 lines. The sequence of commands would be

```
p10
10>p10p.*
&g
```

The first command 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 Section 4.2). After reading those ten lines, the user would then type the third command line above, which would repeat the scroll and the print.

Note that the CMS linend character (defaulted to #) marks the end of a logical command line although multiple commands may physically be on the same line. Thus if the user typed the sequence

```
10#p 10
&g
```

only the Print command would be repeated by the &G.

February 2, 1975

- Leave Asis Mode

&NORMAL

This command causes input values to be treated normally. See Enter Asis Mode above.

- Route Terminal Session

&ROUTE <filename>° <filetype>° <"c">°
or
&ROUTE OFF

&Route allows the user to save his terminal session, or any part of his session, in a CMS file. An inexperienced FRESS user might wish to do this if he is using a display console rather than a hard copy terminal. If he has questions about the session, he then would have a record of what he had done. The filename and filetype are optional and default to FILE FRESSOUT if not specified. If "c" is specified, only commands typed by the user are routed to the file. If "c" is not specified, FRESS responses will be routed as well. If the user wishes to specify any of the optional parameters, he must also specify any preceding optional parameters. For example, if he wishes to specify the "c" option, he must also specify a filename and filetype. The file may be printed using the CMS command Offline Printcc (o printcc <filename> <filetype>).

To halt routing, the user should type "&R OFF".

February 2, 1975

9 APPENDIX A: HELPFUL HINTS

This section provides a number of suggestions about the "best" (least expensive or fastest) way to perform certain operations. It does not claim that certain ways are right or wrong, but the incorporation of these suggestions will make FRESS both easier and less expensive to use. Some of the points made here have been covered elsewhere in this manual, but they are repeated for emphasis and easy referral.

1) Labels

In a file of any length it is advisable to use labels (see Section 6.1) frequently. The user can then employ the Get Label command to position the display instead of always using the more expensive Scroll and Locate commands. All labels should be in lower case to avoid confusion on upper-case only terminals (see Section 4.3.1).

2) Viewing the text

To view a section of text, the Print or Type commands should be used instead of scrolling line by line. The Print command should be used when less than a "display buffer's worth" of text (see Section 2) is desired, or when editing is to be done on the text displayed.

3) Pattern and Context Scanning

It is usually not necessary to do explicit Locates (called pattern scanning) or scrolling to find a string to be edited. FRESS will search 2100 characters (about 1-1/2 manuscript pages) from the current display point to find the string specified as a <scope> parameter (context scanning). Thus the user can often complete an editing operation with a single command rather than with two -- a Locate or Scroll followed by the edit.

4) Input Mode

Whenever the user intends to input more than 5 to 10 lines of text, he should use Swift Input Mode (see Section 3.11). It is both less expensive and faster than ordinary Input Mode.

February 2, 1975

5) Macros

Format macro codes (see Section 5.6) can save a lot of work and typing. Their names should all be typed in lower case to avoid confusion on upper-case only terminals (see Section 4.3.1). It is generally wise to put all macro definitions at the top of the file, possibly with some unique string following them. When the file is retrieved (using Get File) the user should scroll or locate (using the unique character string) over these definitions to avoid "UNDEFINED MACRO" messages later.

6) Pack File

If a file is to be saved with little or no editing to be done on it, the user should pack it, using the Pack File command.

7) &CLEAR

Often a new user of FRESS will mistype a few commands and find himself in a very confusing situation. Messages which he does not understand will be thrown back at him by FRESS and he will not know how to continue. One of the most common mistakes is to accidentally omit the <lp> parameter from a function in such a way that FRESS believes the user has deferred the <lp>. WAITING and WAITING ISCOPE messages will be printed and all commands other than those to resolve the deferred pointers will be temporarily ignored and saved to be done later. Therefore, if the user notices WAITING or WAITING ISCOPE messages which he does not understand, he should use the &C command to clear all these pending functions and deferred <lp>'s. (Note that &C can not be used to solve the problem which results in the STACK CONTROL BLOCK ALL FILLED message. Multiple Returns should be done in this case or use the End command and then re-enter FRESS.

8) Printing online and offline

To print all or part of a file on the offline printer, Fullprint or Offline Type can be used. Fullprint provides complex formatting features, and thus should always be used to print final copies. Offline Type can be used to print a copy of the file with any desired format which can be set up by the Set Viewspecs command. Although Fullprint can be used to print the file with all format codes displayed, it is less expensive to use Offline Type to do the same thing. In addition, Fullprint can not be used to print anything but the main text space. To obtain hard-copy of the label, structure, or work spaces, Offline Type must be used.

To print some or all of a file online, Fullprint, Type, or Print can be used. Fullprint provides fully formatted copy, and is

February 2, 1975

generally used online only if the user desires to have the file printed on special paper, for instance, letterhead. Type may be used to type up to 999 lines of the file, formatted in any way as specified by the Set Viewspecs command. After a Type is done, the start of the display buffer has moved to the start of the last "display window's worth" typed. Print can be used to display up to a "display buffer's worth" of text, again as formatted by the Set Viewspecs command. When the Print has completed, the start of the display buffer has not moved. The Print command is generally used to display small portions of the file while editing is being done.

9) Command line lengths

On any kind of terminal, the maximum line length is 130 characters. Certain terminals, e.g., Datels, can actually print 130 characters across. Other terminals, e.g., Hazeltines and Asciscopes, can display only 80 characters across on their screens. However, even on these terminals, the user can still type up to 130 characters. When the right margin is reached at 80, the characters will begin to overlay each other on the screen, but they will transmit normally.

February 2, 1975

10 APPENDIX B: 360/67 OPERATING SYSTEM ENVIRONMENT

The CP-67 operating system time-shares a number of so-called "virtual machines". These are simulated IBM System/360's, and CP-67 insulates each virtual machine from other virtual machines in the system. Each typewriter console is the operator's console of a virtual machine, on which the user may IPL (load) his own operating system (e.g., CMS) and problem program (e.g., FRESS). It is necessary therefore for the user to interact with both CP-67 and CMS before he can reach the FRESS environment. For more detailed information than is provided here, see the CP/CMS User's Guide [5] and the Brown University Interactive User's Guide [6].

10.1 LOG-ON PROCEDURE

The procedure for logging onto (invoking) FRESS is as follows:

1. Dialing the number for the computer causes it to answer with a high-pitched whistle. The phone's receiver should be quickly and firmly placed in the acoustic coupler attached to the terminal, which must have already been turned on. If you are using a hard-wired terminal (one without an acoustic coupler), simply turn the terminal on. On a Datel, you must wait for the green proceed light to come on.

2. On a Datel, depress the carriage return key. On a teletype or teletype-equivalent terminal (Asciscope, Hazeltine 1000, Teleray), you must type "S" for 110 baud and "O" for 300 baud, both followed by a carriage return. The computer will type out the message:

cp-67 online xd.65 qsyosu

for a Datel and

CP-67 ONLINE

for a teletype-equivalent.

3. Pressing the "attn" (attention) or "int" (interrupt) key at the upper right of the keyboard (for a Datel), or "break" (for a teletype) unlocks the keyboard and alerts the computer that someone is about to log in.

4. Typing "l(ogin) <userid>" or "l(ogin) <account number>" tells the computer the name of the virtual machine on which FRESS is to be run.

February 2, 1975

5. When the computer asks for a password, the password of the virtual machine in which the user wants to run FRESS must be typed in. If either the machine name or password is misspelled, the computer will print a message followed by "restart nur@pn@" on a Datel, or simply RESTART on a TTY, at which point "l <userid> must be typed again.

6. After the day's log messages (notes from the computer's operations staff to all the users) have been printed, the computer will print "CMS VERSION 3.29 - 7/18/74" or a similar message. From this point on, every time the computer is ready to accept user input from the terminal it will signal this with a "proceed". On Datels this means the green proceed light goes on. On teletype-equivalents a "carat" (">") is typed at the terminal. On a Datel the keyboard is locked until this "proceed" is given, so no characters may be typed. On the other terminals, the keyboard is not locked, but hitting any character before the "proceed" is given is equivalent to hitting the "break" or "attn" key: CP will be entered (see below).

7. Once CMS has been entered, certain characters are automatically defined as logical character and line "delete" or "erase" characters both for CMS and for FRESS. This means that when these characters are typed the single previous character, in the case of the character delete, or all the previous characters, in the case of the line delete, typed on the current line will be ignored. On a 2741 or equivalent (e.g. Datel), the line delete is $\$$ and the character delete is @. On a teletype-equivalent the line delete is [and the character delete is @.

In addition, a specific character will be considered the CMS "linend" character. Typing one of these will cause the two halves of the line on either side to be considered as two separate lines by CMS, and thus by FRESS. The default linend character is "#". Any of these CMS dependent characters may be changed using the CMS VSET command. See the CP/CMS User's Guide [5] for details.

8. FRESS is invoked by typing "fress <option>". The option may be any of the following:

- "T" for the teletype version,
- "I" for the multiple-window IMLAC version,
- "M" for the Brown University Graphics System (BUGS) version,
- "H" for the Hazeltine 1000 version,
- "A" for a magnetic card device.

February 2, 1975

If no option is given, the normal (Datel) version is assumed. If an invalid option is given, FRESS will type "INVALID DEVICE" and return to CMS.

9. The computer will print:

```
FRESS VERSION 7.0 - 5/16/74
FRESS READY
```

Before these messages, others left by the FRESS maintenance programmers or systems manager may be printed.

10. The computer is now in command mode, waiting for any FRESS command, customarily a Get File.

10.2 CP COMMANDS

Very often it is useful to leave FRESS temporarily to use CP's facilities (listed below). This can be done by depressing "attn" (or "break" or "int", depending on the type of terminal). The same functions can be executed from inside FRESS by using the FRESS "CMS" command and preceding the function names listed below with "CP" when they are typed. This action interrupts FRESS in whatever it was doing, passing control to CP. There are four CP functions which are helpful to a FRESS user:

1. "q n" (query names) prints out the name of each virtual machine currently running under CP (not necessarily using FRESS), as well as a three digit "address" which the computer uses to identify each terminal. This command is useful for finding out whether the machines to be sent messages (see below) are logged in.

2. "msg <userid> <message>" causes <message> (up to one line) to be printed on the terminal being used by <userid> (a virtual machine name). If that virtual machine is not being used, CP will return an appropriate error message.

3. "q u" (query users) prints the number of users currently running under CP. It is a count of the names that a "q n" would print.

4. The "spool" command can be used to select a special print form. The format of this command is

```
SPOOL 00E AS <form-type> <train-type>
```

February 2, 1975

The <form-type> is the kind of paper used. The system-defined options are STD (standard wide computer print-out paper), NW (narrow 8-1/2 x 11 paper), NC (narrow white centered--see below), NC3 (narrow white centered with three holes punched), and LBLS (mailing labels). The <train-type> specifies the character set to be used in printing. The available options are QN (upper-case only), TN (upper and lower case), QT (QN or TN, whichever is available first), and ALX (upper and lower case plus some special linguistic characters). FRESS output is normally spooled as STD TN.

Because of certain characteristics of printed output from a Fullprint, form types NC and NC3 center FRESS output whose width is 63; that is, the width alter code must be set to 63. To center output of other widths or to specify other non-system defined forms, the user must make arrangements with the operator at the computing lab for a special spooling class number, which is the <form-type>. To do this, the user may call the operator or use the "msg" command described above with a <userid> of CP.

When output of width 63 is centered, it means the perforations on the right side of the paper have been positioned at the righthand side of column 75 on the printer. The formula for determining the alignment of the perforation in other cases is

$$P = 86 - ((85 - W) / 2)$$

where P is the printer position on whose right side the perforation is to be aligned, and W is the FRESS line width. If P is a fraction, it means the perforation should be aligned in the middle of the next printer position.

Use of the Offset <option> on the Fullprint command will affect the centering of the output since it shifts the entire printout to the right. This is most often used to provide extra white space on the left when proofreading hard-copy and not when printing final copies on narrow white paper. If this option is used when printing on narrow white paper, the number of positions in the Offset should be added to the formula above to calculate the printer position for centering.

The !+OFFSET+ alter code has no effect on the printer position for centering since the extra space is taken alternately from left and right.

5. The "q p" (query print) command can be used to list the number and types of files waiting to be printed. If there is no system response to this command it means all files previously Fullprinted have been printed (or are currently

February 2, 1975

printing) on the offline printer. Otherwise the response is a list of the form

<form-type> <train-type> <n>

where <form-type> and <train-type> have the same meaning as in the "spool" command described above and <n> is the number of files of that type still waiting to be printed. Printed FRESS files are usually of the type "STD TN".

CP will return control to FRESS exactly where it was interrupted when the command "b" (begin) is typed in.

Since the terminal is connected to the computer through regular telephone lines, there is occasionally some electrical noise on the line that causes a jump back from FRESS into CP (a "line error"), as if "attn" were hit. A "b" causes an immediate return to FRESS.

10.3 CMS COMMANDS

It should be noted that, unlike FRESS commands, parameters for CMS commands are always separated by blanks.

Another useful function is the CMS "cost" command. This will print at the terminal the total cost of the current terminal session up to the present time. This command should not be executed using the FRESS "CMS" command since it causes a "load" of a program. One typically uses the "cost" function after leaving FRESS but before logging off to determine how much the current session cost.

To list the names of all or some of this files, the user may use the CMS List command. The format of this command is

L <filename> <filetype> <filemode>

FRESS files are of type FRESS and mode P5. Replacing any of the three parameters with an asterisk will list all files which meet the other requirements specified. Leaving out all parameters will list all the files.

If the user wishes to keep track of how much execution time is being spent on his commands he may employ the CMS "Blip" command. This is particularly useful in controlling "infinite loops" (see Section 11). This command will cause a user-chosen string of characters to be printed on the terminal whenever the computer has spent two CPU (vs. "wall clock") seconds in execution of a program.

February 2, 1975

If the "Blip" command is not used, the character typed after 2 seconds of CPU time is an unprintable character. On a Datel or other typewriter-like terminal this manifests itself as a "jump" of the type ball. On display console terminals, there is no manifestation, and thus no way to tell when two CPU seconds have been expended. If using one of these terminals, the user should always use the "Blip" command. The format of the Blip command is

BLIP <chars> <count>

where <chars> is the string of characters to be typed and <count> is the number of characters, up to 8, in that string. <count> defaults to the length of <chars>. If <count> is greater than the length of <chars>, the string will be padded with blanks to the right. Thus to cause a '\$' to be printed when two seconds have been expended, the user should type, in CMS:

blip \$

The user may then continue normal use of the FRESS system.

10.4 SUMMARY OF ENVIRONMENT RELATIONSHIPS

Should it become unclear which level (CP, CMS, or FRESS) is in control, a carriage return will cause the name of the control environment to be typed.

The following table summarizes the relationships among the various environments and how to move from one to the other:

<u>To get from:</u>	<u>To:</u>	<u>Action:</u>
CP	previous environment	type "b"
CMS	CP	hit "attn", "break", or "int" key
CMS	FRESS	type "fress <option>"
FRESS	CP	hit "attn", "break", or "int" key
FRESS	CMS	type "end"
CP	CMS	type "CMS"

It is possible to stop FRESS from completing the printing of one or more lines. Hitting "attn" once gives control to CP. If a line is being printed, it will be stopped. By hitting "attn" again control is returned to FRESS. At this point typing "kt" tells FRESS to ignore any other lines left to be printed for the current command.

February 2, 1975

10.5 ARCHIVAL STORAGE

At Brown, each user has a "segment" or a "minidisk" on which his FRESS files are stored. Because of the limited size of this storage and because it is not desirable to leave currently unused files there, any user who intends to save FRESS files which he does not currently need, or who intends to create many files which will involve a great deal of storage, should maintain some form of archival storage for his files. This is usually in the form of a tape. Any user who desires more information about the use of archival storage should contact a member of the FRESS staff.

February 2, 1975

11 SYSTEM ERRORS AND OTHER PROBLEMS

As with all large-scale software, the FRESS user may at some point encounter a "bug" in the FRESS program. This problem may manifest itself in several ways.

First, a command may not work as expected. Before seeking aid, check the appropriate FRESS manual and be sure you fully understand the nature of the command used. If, at that point, you still have questions, contact a member of the FRESS staff. To do this, call Professor van Dam's office at Brown University (401 863-3088, 305 Applied Mathematics Building).

Secondly, the message "SYSTEM ERROR <number> - PLEASE CONTACT FRESS STAFF" may be typed. In this case, 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. The user should then end his session and do a FRESS BUG (see below).

Thirdly, the message "ABEND" or "BOMB" may be typed. In this case a severe error has occurred and the user is returned to CMS. Unless the last processed command was Revert or Accept, the last editing change made will be lost. The user should always do a "FRESS BUG" in this case.

Lastly, the user may encounter an "infinite loop". This means that FRESS will indefinitely continue attempting to execute a command but will never succeed. In order to recognize this, the user should employ the CMS "Blip" command (see Section 10.3). This command will cause a user-chosen string of characters to be printed on the terminal whenever the computer has spent two CPU (vs. "wall clock") seconds in execution of a program. Since computers work at tremendous speeds, two seconds should be enough to execute many FRESS commands (with certain exceptions noted below). If the "Blip" command is not used, the character typed after 2 seconds of CPU time is an unprintable character. On a Datel or other typewriter-like terminal this manifests itself as a "jump" of the type ball. On display console terminals, there is no manifestation and thus no way to tell when two CPU second have been expended. If using one of these terminals, the user should always use the "Blip" command. Certain FRESS commands, in particular Fullprint, Offline Type, Copy File, Pack File, and any of the Locate Long commands in a large file, will take a great deal of time, perhaps several multiples of two seconds, to complete. If, however, several blip characters should appear consecutively after the user types any other command, he should immediately hit the "attn" key to stop FRESS. He should then go back to CMS by typing "CMS" and then

February 2, 1975

do a FRESS BUG (see below). He may attempt to repeat his last command, but if the same problem occurs he should consult a member of the FRESS staff before continuing.

11.1 FRESS BUGS AND SESSION RECORDING

All FRESS sessions are automatically monitored for the use of the FRESS staff. The statistics gathered are designed to help improve FRESS itself. They include only such general information as the number of FRESS users, the number of times each command was used in each session, the total number of times each command was used by all FRESS users, the length of FRESS sessions, and the errors encountered by FRESS users. They do not include the names or contents of files. These statistics are available for public inspection, and under no circumstances will a FRESS project member ever examine a FRESS file without the knowledge and consent of the file's owner.

The standard mode of communication between the FRESS staff and the user community is the CMS command FRESS BUG. This command should be used if one has problems or questions regarding FRESS in general or a particular file. This procedure will ask for the name of a file (to be entered if a problem concerns a particular file), and then for a description of the problem or a message. This information is automatically sent to the FRESS staff along with a copy of the named file to aid the staff in determining and fixing the problem. Information concerning the problem which would be helpful to the staff includes the position of the display of the file when the error occurred and the sequence of the last few commands executed by the user to the best of his recollection. It is requested that a name and phone number or address of whom to contact regarding the problem or question be included with this information.

11.2 FRESS ERROR MESSAGES

FRESS will attempt to inform the user of both his own (user) errors and FRESS (system) errors in the most coherent fashion possible. The rest of this section is devoted to a list of the error messages in FRESS and accompanying explanations where necessary. The messages are divided into three categories: internal and system errors; "question mark" errors; and others. Those messages which relate to a particular command appear also in the alphabetical command listings of Section 8.6. Some of the messages involve situations which may arise using either the

February 2, 1975

advanced facilities listed in the FRESS Reference Manual [4] or the subset of the facilities described in this manual. References to unknown facilities should be ignored. Should a user discover a message which is not listed, or should he desire a more complete description of a particular error condition, he is urged to contact a member of the FRESS staff.

Unless stated otherwise, when one of the following messages is received the last command typed has been ignored. If the explanation of a message below says "contact FRESS staff" this means 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. He should then do a FRESS BUG and contact a FRESS staff member to determine the nature of the problem.

In addition, if any of these messages are received and the user does not think he made the indicated error, it may also indicate a problem in the file. He should first repeat the command to make certain it was not mistyped or that a line error, i.e. a bad character transmitted by the terminal, was not the cause. Then the user should make a note of the last commands executed and contact a FRESS staff member.

11.2.1 FRESS INTERNAL AND SYSTEM ERRORS

The user should contact the FRESS staff immediately if he should encounter one of these messages, noting the particular message and <number> or <name>.

ABEND AT <name>

BOMB AT <name>; PSW= <digits>

SYSTEM ERROR <number> - PLEASE CONTACT FRESS STAFF

Any messages of the form:

TSI <three numbers>

February 2, 1975

11.2.2 "QUESTION MARK" ERRORS

?

This message is often typed when it is impossible to decipher a particular FRESS command. The question mark will appear under the section of the command causing the problem. The user may type "?" in response, in which case a more complete explanation of the problem will be typed. This may be any of the following:

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 4.3.1); or having already scrolled or pattern scanned past the pattern being searched for now.

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. "mo/b=??/x") or if neither of the two <lp>'s were actually deferred (e.g. "mo/b=r/x").

FUNCTION TAKES NO PARAMETERS

A parameter was specified in a command which takes no parameters, e.g., Display Viewspeccs or Accept.

INCORRECT VALUE IN PARM POSITION

A <scope> has been specified with one or both endpoints deferred when an <lp> parameter was expected.

INVALID FUNCTION SEPARATOR

Only ">" may be used as a function separator - see Section 3.13. This message may occur if a house function is followed by another command with no function separator between them.

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

February 2, 1975

latter case, complete the deferred <lp> or <scope> to allow the pending functions to complete.

SEMANTICS CHECK

A restriction on a parameter has been ignored, e.g., a password longer than 7 characters has been given.

11.2.3 OTHER MESSAGES AND ERRORS

If one of the following messages is associated with a small number of commands, the list of the abbreviations for these commands follows the error message.

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

(D, S) One cannot substitute for or delete a string starting in a data field and ending outside of that data field. For the purposes of this manual, data field means label field.

BOTH LP'S MUST BE IN SAME FILE

The pair of <lp>'s comprising a <scope> must be in the same file.

BUFFER BUTCHERY

Some problem is preventing the file from being displayed at the current display point. Usually the file is all right, but contact FRESS staff anyway.

BUFFER SIZE EXCEEDED

(SD) The line length (<n2>) times the number of lines (<n1>) is greater than the display buffer size (see Section 4.3). Use smaller limits.

CANNOT DELETE START OR LAST END LINE OF SPACE

(D, S) The <scope> of the specified Delete or Substitute included the *START OF TEXT AREA* line or the last *END OF TEXT AREA* line in the file.

CMS: <CMS error code>

(CM) An error was returned from the specified CMS command. The error code is printed in decimal.

February 2, 1975

Negative error codes indicate the command was not recognized.

CMS: NO COMMAND SPECIFIED

(CM) The <lit> parameter was omitted.

COMMAND NOT SUPPORTED or
CURRENTLY UNSUPPORTED

The specified command has been designed but is not ready for use as yet.

COPY SUCCESSFUL

(CF) The indicated copy was completed. The user now has two identical files, <file1> and <file2>.

CURRENT: <filename>
<filename>
<filename>

(Q) This is the format of the reply to a "Q F".

EDIT LIMITED TO ONE LABEL OR KEYWORD

When editing in the label space, only one label may be included in the <scope> of a single edit. (Keywords are explained in the FRESS Reference Manual [4].)

END OF AREA

(US) The pattern scan has encountered the "*END OF TEXT AREA*" line; no more substitutes can be done.

END OF COUNT

(US) The user specified the <option> giving the number of times to perform the substitution. The specified number have now been done.

ENDING COLUMN > 80

(O) An error has been made in the specification of column positions from card image input.

ERROR IN STRING NEAR POSITION <number>

(SV) This means an invalid viewspec has been included in <vs>. The error occurs near the character with displacement <number> from the beginning of the string.

FILE: <filename>

SPACE: <space name>

AREA: <n>

(Q) This provides the requested information from a "Q L".

February 2, 1975

FILE ALREADY EXISTS
(MF, O, CF) A FRESS file with the given filename already exists. In the case of Copy File, 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 a Get File command and then attempt the Scratch again.

FILE NOT FOUND
(G, O) The specified file does not exist. In the case of Offline Read, it is the CMS file specified in the Disk <option> which does not exist.

FILE NOT IN USE
(F) The file specified was not open, that is, had not been retrieved.

FILE TO COPY NOT FOUND
(CF) <file1> does not exist. The copy was not done.

FINI
(FU) The Fullprint completed and has been sent to the printer.

FINIS
(OT) The Offline Type was completed and has been sent to the printer.

FUNCTION NOT ALLOWED
The function 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 functions 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
(FL, SUR, UND, FO, CAP, UNC) These commands may not be used in a piece of structure, e.g., a label.

HYPERTEXT: ACCEPT (A), REJECT (R), OR TEXT ONLY (T)
See Section 6.2.

February 2, 1975

HYPERTEXT INTERFILE MOVE NOT SUPPORTED

Labels and other structure may not be moved or copied interfile.

IMPLIED LP DOES NOT EXIST; LP IS DEFERRED: WAITING

The <lp> parameter of the command has been left null before an implied insert point has been established, i.e., before any edit has been done. 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 '(' and ')' for labels).

INPUT

The user is now in Input Mode. All text typed at the terminal will be interpreted as literal input until a null line is typed.

INV OPTION

(FU) One of the <options> was invalid. The command was ignored.

INVALID CHARACTER IN COMMAND LINE - TRANSLATED TO BLANK

Because of a "line error" (a malfunction of the telephone connection to the computer) the terminal has transmitted an invalid character somewhere in the command line. 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 changing the invalid character to a blank.

INVALID FILENAME

(CF) The name of either <file1> or <file2> contains an invalid character, generally caused by a telephone line error. The copy was not done, so try again.

INVALID FUNCTION

(AP) The <options> list contains an invalid function.

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.

February 2, 1975

INVALID OR ILLEGAL QUALIFIER

An undefined qualifier (see Section 3.12) was used.

INVALID PASSWORD

(G, CP) For Change Password, this message indicates that the <pass> field specified a password not associated with the file. In the case of Get File, 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 PATTERN, TRY AGAIN

(L, US) Certain configurations of ampersands with other characters are considered invalid patterns. The ampersand is used to indicate a modifier to the pattern, a facility which is not yet fully implemented and will be explained in a future update of this manual.

JUSTIFICATION: EDIT IGNORED

Editing may not be done while on-line justification is in effect.

LABEL ALREADY EXISTS

(ML) The label specified in a Make Label command already exists.

LABEL SPECIFIED DOES NOT EXIST

(CTL, MTL, GL) <label> is not defined in the current file or, in Get Label, in the specified <file>.

LABELS LIMITED TO 16 CHARACTERS

(ML,I,S) Either the <label> specified in the Make Label command was longer than 16 characters or the end or intermediate result of the edit (Insert or Substitute) would produce a label longer than 16.

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 the Footnote command is anything other than a number. It can also be caused if any <text> parameter contains more backspaces than ordinary characters.

February 2, 1975

LP FOR WORD EDIT MAY NOT BE A BLANK OR PUNCTUATION

The user specified an edit using the word qualifier with an <lp> of a blank or some kind of punctuation.

LP MUST BE IN AN ORDER

The order qualifier (-o) was specified but the specified <lp> was not in an order (e.g., a label).

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

e.g., all text was removed from the work space but the implied insert point was still pointing inside the space so it could not be deleted. Note that the command has been executed.

MAX EXCEEDED

(P, SC) For the Print command, this message means the number of lines in the display buffer is less than <n>. Specify a smaller number. For the Scroll command, it means a scroll of more than 127 lines forward or backward was specified.

MOVETO LP BETWEEN SCOPE LP'S

(M, CO) i.e., one cannot move a string to within itself.

NO CURRENT FILE

A display function (a scroll, Bottom, etc.) has been attempted without a current file.

NO CURRENT FILE: IGNORED

An editing function has been attempted without a current file.

NO FILE OPEN

(Q) Either the user has not executed a Get File, or all open files have been freed using Free File.

NO HELP AVAILABLE FOR THAT

(H) An <option> was specified which does not correspond to any available information.

February 2, 1975

NO MORE RETURN POINTS

(R) There are no more entries in the Return stack, and therefore no more Returns can be done.

NO OUTSTANDING IMCOMPLETE FUNCTIONS

(&C) There are no deferred <lp>'s or <scope>'s or pending functions to be cleared.

NO REGULAR TEXT ALLOWED IN STRUCTURE SPACE

(I,S) The user attempted to insert text directly into the structure space and was not merely editing what was already there, for example, fixing the spelling of a label.

NO TEXT IN SCOPE OF TEXT-ONLY EDIT

An edit specified as Text-only (see Section 6.2) must have some regular text in the <scope>.

NOTHING IN WORK SPACE TO BE MOVED

(MFW, CFW) The work space was empty when a Move From Work or Copy From Work was specified.

NULL LABEL IS ILLEGAL

(ML) The user attempted a Make Label with a null <label> parameter.

OPTIONS=

(US) A substitute in Inquire mode has been performed. The user must now specify one of the options described above.

PACK SUCCESSFUL <n>/<m>

(PF) This message indicates that the specified file has been successfully packed. The numbers <n> and <m> are the size of the file in internal pages before and after packing, respectively.

PACKED PAGES MUST BE MORE THAN HALF FULL

(PF) A percentage (<n>) of less than 50 was specified.

PASSWORD ALREADY EXISTS

(AP) The specified password is already defined in the current file.

PASSWORD DOES NOT EXIST

(DP) The specified password has not been defined in the current file.

February 2, 1975

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 in 8000 characters from the current display point. This condition will not occur 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 E) or continue with a long scan (A).

PDISK IS FULL

Actually a message from CMS. 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 re-ipl of 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., Pack File, Copy File, Make File), a partial file may be left on the user's storage. This file may either have the name specified in the command or have the name 'SYSUT6' or 'SYSUT7'. These files should be erased.

PLEASE FREE A FILE & TRY AGAIN

(OT) Offline Type had insufficient space to complete. As indicated, the user should free one or more files (using Free File) and try the Offline Type again.

PRINTER TROUBLES

(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.

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 Return.

February 2, 1975

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) A serious error has occurred; contact a member of the FRESS staff.

SCRATCH SUCCESSFUL

(SF) The command was executed as specified.

SCROLL OR CHAR LENGTH >127 IN POSITION STR

The line or character displacement number (see Section 3.14) is too large.

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 Return.

SPECIFIED SPACE DOES NOT EXIST

(DS) The specified space (W or L) does not exist. The structure space (S) and text space (T) always exist.

STACK CONTROL BLOCK ALL FILLED

Occurs during the saving of a location in a file; RETURNS should be done to clear the stack. See the description of the Return command in Section 8.6 for more information about this stack.

STARTING COLUMN < 1

(O) An error has been made in the specification of column positions from card image input.

STARTING COLUMN > ENDING COLUMN

(O) An error has been made in the specification of column positions from card image input.

TEXT TO INSERT MUST BE SPECIFIED

(SUR) Either <text1> or <text1> and <text2> must be specified.

TOO MANY FILES OPEN

(PF) Because of space limitations, it is necessary to close some files, using the Free File command. The Pack File command may then be repeated.

February 2, 1975

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.

UNKNOWN SPACE

(DS) A letter other than T, W, L, or S was specified as <space>.

UN LIGHTPENNABLE TEXT

(L) The <lit> parameter of a Locate command 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 typed by the user contains invalid option characters.

UNSUPPORTED COMMAND

Certain house functions (see Section 8.7) 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

File may be bad; contact FRESS staff.

February 2, 1975

12 LISTS OF COMMANDS ACCORDING TO TYPE

12.1 NECESSARY SUBSET OF COMMANDS FOR EDITING

Bottom	B
Copy	CO
Delete	D
Display Space	DS
End	E
Fullprint	FU
Get File	G
Insert	I
Locate	L
Move	M
Make File	MF
Print	P
Revert	REV
Substitute	S
Scroll	SC or < \pm n>
Swift Input	SI
Type	T

February 2, 1975

12.2 EDITING, DISPLAY, AND FILE COMMANDS

The following FRESS commands are Editing commands:

Accept	A
Bars	BA
Bottom Input	BI
Capitalize	CA
Copy From Work	CFW
Copy	CO
Copy To Label	CT
Copy To Work	CTW
Delete	D
End	E
Footnote	FO
Insert	I
Insert Before	IB
Move	M
Move From Work	MFW
Make Label	ML
Move To Label	MT
Move To Work	MTW
Offline Read	O
Revert	REV
Substitute	S
Swift Input	SI
Swift Input Bottom	SIB
Swift Input Top	SIT
Surround	SUR
Top Input	TI
Underscore	U
Uncapitalize	UNC
Uniform Substitute	US

The following FRESS commands are Travel or Display commands:

Bottom	B
Display Space	DS
Display Viewspecs	DV
Fullprint	FU
Get Label	GL
Locate	L
Offline Type	OT
Print	P
Query	Q
Return	R
Save	SA
Scroll	SC
Set Display	SD

February 2, 1975

Set Mode	SM
Set Viewspecs	SV
Type	T

The following FRESS commands are File commands:

Add Password	AP
Copy File	CF
Change Password	CP
Delete Password	DP
Free File	F
Get File	G
Make File	MF
Pack File	PF
Scratch File	SF

February 2, 1975

12.3 COMMAND NAME ABBREVIATIONS

The following is a list of FRESS command alphabetized according to their abbreviations.

A	Accept
AP	Add Password
B	Bottom
BI	Bottom Input
CA	Capitalize
CF	Copy File
CFW	Copy From Work
CM	Execute CMS Command
CO	Copy
CP	Change Password
CT	Copy To Label
CTW	Copy To Work
D	Delete
DP	Delete Password
DS	Display Space
DV	Display Viewspects
E	End
F	Free File
FL	Flip
FO	Make Footnote
FU	Fullprint
G	Get File
GL	Get Label
H	Help
I	Insert
IB	Insert Before
L	Locate
M	Move
MF	Make File
MFW	Move From Work
ML	Make Label
MT	Move To Label
MTW	Move To Work
O	Offline Read
OT	Offline Type
P	Print
PF	Pack File
Q	Query
R	Return
REV	Revert
S	Substitute
SA	Save
SC	Scroll
SD	Set Display
SF	Scratch File

February 2, 1975

SI	Swift Input
SIB	Swift Input Bottom
SIT	Swift Input Top
SM	Set Mode
SUR	Surround
SV	Set Viewspecs
T	Type
TI	Top Input
U	Underscore
UNC	Uncapitalize
US	Uniform Substitute
&A	Enter Asis Mode
&C	Clear Function Area
&R	Repeat Last Command Line
&N	Leave Asis Mode
&R	Route Terminal Session

February 2, 1975

BIBLIOGRAPHY

- [1] Carmody, et. al.: "A hypertext editing system for the 360," Proc. Conference in Computer Graphics, 1969, Univ. of Illinois, 291-330.
- [2] FRESS Concepts and Facilities Manual, Text Systems, Inc., Barrington, R.I., 1972.
- [3] van Dam, Andries and David Rice: "Online Text Editing: A Survey", ACM Computing Surveys, September, 1971.
- [4] van Dam, Andries and Carol L. Chomsky, FRESS Reference Manual: Structure and Commands, Providence, R.I., July, 1974.
- [5] IBM Corp., Control Program-67/Cambridge Monitor System (CP-67/CMS) -- User's Guide, Form GH20-0859, WHITE PLAINS, N.Y., 1972.
- [6] Brown University Computing Laboratory's Interactive User's Guide, Manual Number BCL09-001-00, Providence, R.I., September, 1972.

February 2, 1975

TABLE OF CONTENTS

1 FOREWORD.....	1
2 INTRODUCTION.....	2
2.1 Outline of the Manual.....	2
2.2 FRESS as a "Stream-editor".....	3
2.2.1 Program Editing vs. Text Editing.....	3
2.2.2 Line Editing Versus Stream Editing.....	3
2.2.3 Editing Buffer, Display Buffer and Display Window.....	4
2.2.4 Limits on Scanning.....	7
3 COMMAND SPECIFICATION AND EXAMPLES.....	8
3.1 Input Mode vs. Command Mode.....	8
3.2 Summary of Capabilities.....	8
3.3 Command Classes.....	9
3.4 Command Language Interpreter.....	10
3.5 Definition of Command Language Terms.....	10
3.6 Upper and Lower Case.....	12
3.7 Some Simple Examples.....	12
3.8 Location Pointers and Their Deferral.....	13
3.9 Representing Control Characters.....	17
3.10 The Key Delimiter and Its Use.....	18
3.11 Input Mode and the Implied Insert Pointer.....	20
3.12 Qualifiers+.....	23
3.13 Combining Commands on a Single Input Line+.....	24
3.14 Line and Character Displacement Numbers+.....	25
3.15 The "&" LP Character+.....	26
3.16 Summary of Control Characters.....	27
4 DISPLAY MODES AND TERMINAL CHARACTERISTICS.....	31
4.1 Display Modes.....	31
4.2 Temporarily Overriding Display Modes+.....	31
4.3 Terminal Characteristics.....	32
4.3.1 Upper-case Only Terminals.....	33
4.3.1.1 Text Display.....	33
4.3.1.2 Pattern Matching.....	34
4.3.1.3 Literal Text.....	36
4.3.1.4 Correcting Character Case Errors.....	36
4.3.1.5 "Missing" Characters.....	36
4.3.1.6 Summary of Rules.....	37
4.3.2 Hazeltine Version.....	37
4.3.3 Summary of Default Settings.....	38
5 FORMATTING CODES.....	39
5.1 Overview.....	39
5.2 Formatting Code Conventions.....	39
5.3 Notation for Formatting Code Explanations.....	41
5.4 Edit Codes.....	41

February 2, 1975

5.5	Alter Codes.....	47
5.6	Macro Codes.....	56
5.7	Format Code Error Messages.....	58
5.8	Special Characters.....	59
5.9	Underscores.....	61
5.10	Justification.....	62
5.11	Table Formatting Examples ⁺	63
5.11.1	Entering Tab Codes.....	63
5.11.2	Table Mode.....	64
5.11.3	Overlaid Text.....	65
5.11.4	Some More Tab Code Examples.....	66
5.12	The GRID alter code ⁺	68
5.12.1	Description.....	68
5.12.2	Reference tables.....	73
5.12.3	GRID example.....	74
6	ADDITIONAL FEATURES.....	79
6.1	Labels.....	79
6.2	Hypertext Editing.....	79
6.3	Files and Spaces ⁺	80
6.4	Viewspecs ⁺	81
7	TRANSCRIPT OF AN EDITING SESSION.....	83
7.1	Annotation for Sample Session.....	83
7.2	Initialization Examples.....	83
7.3	Traveling Examples.....	84
7.4	Editing and Formatting Examples.....	85
7.5	Sample Session.....	88
8	FRESS COMMANDS.....	93
8.1	Form of Commands in Manual.....	93
8.2	Types of parameters.....	93
8.3	Specifying Parameters.....	95
8.4	Implied Insert Points.....	97
8.5	Entering Input Mode.....	97
8.6	Alphabetical Listing of FRESS Commands.....	98
A.....		98
• Accept.....		98
• Add Password.....		99
B.....		101
• Bottom of Space.....		101
• Bottom Input.....		101
C.....		102
• Capitalize Text.....		102
• Copy File.....		103
• Copy From Work Space.....		103
• Execute CMS Command.....		104
• Copy Text.....		104
• Change Password.....		105
• Copy to Label.....		106
• Copy To Work Space.....		106

February 2, 1975

D.....	107
• Delete Text.....	107
• Delete Password.....	108
• Display Space.....	108
• Display Viewspects.....	109
E.....	110
• End FRESS Session.....	110
F.....	111
• Free File.....	111
• Flip Case.....	111
• Make Footnote.....	112
• Full Printout.....	113
G.....	115
• Get File.....	115
• Get Label.....	115
H.....	117
• Help.....	117
I.....	118
• Insert Text.....	118
• Insert Before.....	119
L.....	120
• Locate (Pattern Scan).....	120
M.....	122
• Move Text.....	122
• Make File.....	122
• Move From Work Space.....	123
• Make Label.....	123
• Move to Label.....	124
• Move To Work Space.....	124
O.....	126
• Offline Read.....	126
• Offline Type.....	128
P.....	130
• Print.....	130
• Pack File.....	130
Q.....	132
• Query Current Files.....	132
R.....	133
• Return.....	133
• Revert.....	133
S.....	135
• Substitute Text.....	135
• Save Current Location.....	136
• Scroll.....	136
• Set Display Window.....	137
• Scratch file.....	138
• Swift Input.....	138
• Swift Input Bottom.....	139
• Swift Input Top.....	139
• Set Mode of Display.....	139
• Surround Text.....	140

February 2, 1975

• Set Viewspecs.....	141
T.....	142
• Type.....	142
• Top Input.....	142
U.....	143
• Underscore.....	143
• Uncapitalize.....	143
• Uniform Substitute.....	144
8.7 FRESS House Functions.....	146
• Enter Asis Mode.....	146
• Clear Function Area.....	146
• Repeat Last Command Line.....	147
• Leave Asis Mode.....	148
• Route Terminal Session.....	148
9 Appendix A: Helpful Hints.....	149
10 Appendix B: 360/67 Operating System Environment.....	152
10.1 Log-on Procedure.....	152
10.2 CP Commands.....	154
10.3 CMS Commands.....	156
10.4 Summary of Environment Relationships.....	157
10.5 Archival Storage.....	158
11 System Errors and Other Problems.....	159
11.1 FRESS BUGS and Session Recording.....	160
11.2 FRESS error messages.....	160
11.2.1 FRESS Internal and System Errors.....	161
11.2.2 "Question Mark" Errors.....	162
11.2.3 Other Messages and Errors.....	163
12 Lists of Commands According to Type.....	173
12.1 Necessary Subset Of Commands For Editing.....	173
12.2 Editing, Display, and File Commands.....	174
12.3 Command Name Abbreviations.....	176
BIBLIOGRAPHY.....	178