CONTROL DATA
CORPORATION

CONTROL DATA®
CYBER™ 70 SERIES MODELS 72/73/74
6000 SERIES
COMPUTER SYSTEMS

KRONOS® 2.1
TEXT EDITOR
REFERENCE MANUAL

SUMMARY OF AVAILABLE COMMANDS AND FORMATS

Command	Page
ADD(S)	4-9
ADD(S);n	4-9
ADD(S):/string/	4-9
ADD(S):/string/;n	4-9
ADD(S):/string1/, /string2/	4-9
ADD(S):/string1/, /string2/;n	4-9
ALIGN	4-26
ALIGN;n	4-26
ALIGN:/string/	4-26
ALIGN:/string/;n	4-26
ALIGN:/string1/, /string2/	4-26
ALIGN:/string1/, /string2/;n	4-26
BLANK(S)	4-14
BLANK(S);n	4-14
BLANK(S):/string/	4-14
BLANK(S):/string/;n	4-14
BLANK(S):/string1/, /string2/	4-14
BLANK(S):/string1/, /string2/;n	4-14
CHANGE(S)	4-17, 18
CHANGE(S);n	4-17, 18
CHANGE(S):/string/	4-17, 18
CHANGE(S):/string/;n	4-17, 18
CHANGE(S):/string1/, /string2/	4-17, 18
CHANGE(S):/string1/, /string2/;n	4-17, 18
CLEAR	4-22
DEFTAB	4-31
DEFTAB:/tabchar/	4-31
DELETE(S)	4-13
DELETE(S);n	4-13
DELETE(S):/string/	4-13
DELETE(S):/string/;n	4-13
DELETE(S):/string1/, /string2/	4-13
DELETE(S):/string1/, /string2/;n	4-13
END	4-38
EXTRACT(S)	4-22
EXTRACT(S);n	4-22
EXTRACT(S):/string/	4-22
EXTRACT(S):/string/;n	4-22
EXTRACT(S):/string1/, /string2/	4-22
EXTRACT(S):/string1/, /string2/;n	4-22
FIND(S);n	4-3
FIND(S):/string/;n	4-3
FIND(S):/string1/, /string2/;n	4-3
INSERTS:/string1/, /string2/;n	4-9
LENGTH;n	4-25
LINE	4-5
LIST(S)	4-2
LIST(S);n	4-2
LIST(S):/string/	4-2
LIST(S):/string/;n	4-2
LIST(S):/string1/, /string2/	4-2
LIST(S):/string1/, /string2/;n	4-2
LISTAB	4-31
MERGE:/lfn/;n	4-33
MERGE:/lfn/, /string/;n	4-33
NUMBER(S)	4-36
NUMBER(S):/string/	4-36
NUMBER(S):/string1/, /string2/	4-36
REPLACES:/string/;n	4-18
REPLACES:/string1/, /string2/;n	4-18
RESET	4-4
SET;n	4-4
SET;-n	4-4
SET:/string/	4-4
SET:/string/;n	4-4
TAB	4-31
TAB:/$t_1, \ldots, t_n$/	4-31
WIDTH;n	4-25

CONTROL DATA
CORPORATION

CONTROL DATA®
CYBER™ 70 SERIES MODELS 72/73/74
6000 SERIES
COMPUTER SYSTEMS

KRONOS® 2.1
TEXT EDITOR
REFERENCE MANUAL

REVISION RECORD

REVISION	DESCRIPTION
A	Manual released. This manual obsoletes the KRONOS 2.0 Text Editor Reference Manual,
(3-15-73)	Pub. No. 59150700.
Publication No. 60408200	

Address comments concerning this manual to:

Control Data Corporation
Technical Publications Department
4201 North Lexington Avenue
St. Paul, Minnesota 55112

or use Comment Sheet in the back of this manual.

© 1973
by Control Data Corporation
Printed in the United States of America

PREFACE

The Text Editor (also known as the EDIT program) is a part of the KRONOS VI, Version 2.1, Time-Sharing System (hereinafter called KRONOS) for CONTROL DATA® CYBER 70, Model 72, 73, and 74 computer systems. Its purpose is to effect character-oriented data manipulations from a remote terminal.

This manual contains the information a user must have to use the Text Editor.

Section 1 introduces the Text Editor and summarizes its general capabilities.

Section 2 describes certain features of the KRONOS system that are necessary to the Text Editor user. It is not a detailed description of system software or hardware, and is included as a general guide for the inexperienced user.

Section 3 defines or describes the basic concepts and terminology inherent in Text Editor usage.

Section 4 contains descriptions of each of the EDIT commands, including contextual illustrations for each general type of command. Certain commands are mutually related and an effort is made to depict these relationships clearly in the examples.

Cross-references to other Control Data manuals are usually made by citing the entire manual; a section name or index heading is also given if not obvious. This method is necessary because some manuals are updated frequently.

For additional information about KRONOS VI software, refer to the current versions of the following manuals.

Control Data Publications	Publication Number
6400/6500/6600 Computer Systems Reference Manual	60100000
KRONOS 2.1 Reference Manual	60407000
KRONOS 2.1 Time-Sharing User's Reference Manual	60407600
KRONOS 2.1 BASIC Reference Manual	19980300
KRONOS 2.1 FORTRAN (Time-Sharing) Reference Manual	60408600
KRONOS 2.1 Terminal User's Instant Manual	60407800

This product is intended for use only as described in this document. Control Data Corporation cannot be responsible for the proper functioning of undescribed features or undefined parameters.

CONTENTS

FIGURES

INTRODUCTION 1

GENERAL

The KRONOS 2.1 Text Editor (EDIT) performs data manipulations on a file as specified by the time-sharing terminal user. It is an interactive package; that is, the user enters a command, EDIT interprets the command and executes it, after which the user can enter another command to be executed, and so forth.

Manipulations are performed on a mass storage file allocated to the user; this is called the edit file. The edit file can contain various data forms, such as a body of prose text, a table of numeric data, or a program written in source code (such as BASIC, FORTRAN, or COMPASS, etc.).

TEXT EDITOR CAPABILITY

Using Text Editor commands, the user can manipulate edit file data in the following ways (appropriate command words are shown in parentheses).

- Print a file, either in part or in entirety (LIST, FIND)
- Erase information from a file (DELETE, BLANK)
- Add information to a file (ADD, INSERTS)
- Replace information in a file with other information (CHANGE, REPLACES)
- Move information to a temporary holding area for subsequent insertion (EXTRACT, CLEAR)
- Combine the contents of two files (MERGE)
- Obtain a count that reflects the number of times a specified combination of characters occurs in the edit file (NUMBER)
- Direct Text Editor activity to a specific area of the edit file (SET, RESET, FIND, LINE)
- Control edit file and page format (LENGTH, WIDTH, ALIGN, DEFTAB, LISTAB, TAB)
- Terminate the editing session (END)

EDIT OPERATIONS

An edit command consists of a command word, followed optionally by a string specification and an n parameter.

The Text Editor command repertoire allows three basic types of operations.

1. Line mode operations are addressed to one or several entire lines of text in the edit file.

2. String mode operations are addressed to a sequence of characters, as indicated by a string specification that follows the command word. A string mode command word always ends with an S. A string mode command with an empty string specification has exactly the same effect as a line mode command.

3. Edit control commands are not addressed to specific lines or strings of text. In general, they perform such necessary functions as search-pointer control, format control, large-scale file manipulations, and exit from the Text Editor program.

SYSTEM OPERATION 2

GENERAL

This section presents a general description of terminal operation in a KRONOS time-sharing system. It is oriented to the inexperienced KRONOS user, emphasizing those commands and formats most relevant to EDIT usage.

Its purpose is to provide the inexperienced user with enough information to gain access to the KRONOS system, perform the necessary file operations, build and edit files of text, and finally, to end the editing session. In general, an experienced KRONOS user can omit this section.

The outlined techniques are not necessarily the only ways to accomplish a given result, nor necessarily the most efficient.

For more detailed information on KRONOS time-sharing operation, refer to the KRONOS Time-Sharing User's Reference Manual. Also, if operating under BATCH subsystem, refer to the KRONOS Reference Manual.

TERMINAL EQUIPMENT

The time-sharing user communicates with a central site computer. The remote terminal can be any of several types of low-speed device; it always has a typewriter-like keyboard by which the user enters information and an output medium that allows the user to observe the information entered as well as the responses by the system. The output medium can be either hard-copy printing (as on a Teletype® unit or an IBM 2741 terminal) or a raster image on a CRT display device.

The operating procedures that follow are based on the assumption that a TTY-33/35 terminal is being used. There is no recognized standard KRONOS terminal although the TTY-33/35 may be considered typical.†

Throughout the manual, the symbol (CR) is used to denote the RETURN key (or its equivalent).

† Complete operating information for every type of terminal that can interface with KRONOS is beyond the scope of this manual; for this information, refer to documentation for the particular hardware, or consult with the CDC field representative.

DEFINITIONS OF TERMS

This part defines several terms pertaining mostly to the KRONOS file system. .These terms appear throughout the manual. The definitions are neither rigorous nor detailed; rather, they are intended as clarification for the inexperienced user who lacks a general understanding of KRONOS files as they relate to Text Editor usage. For additional information regarding the types of files and their relevant KRONOS commands, refer to the KRONOS Time-Sharing User's Reference Manual.

PERMANENT FILE

A permanent file is a file situated in mass storage so that it can be removed only by a PURGE command. Two types of permanent file exist, direct access and indirect access.

DIRECT ACCESS FILE

A direct access file is a permanent file accessed only by an ATTACH command. No copy of the permanent file is created for user access; hence, all editing and other alterations must be performed directly on the direct access file. Only one user can write on a direct access file at any one time.

INDIRECT ACCESS FILE

An indirect access file is a permanent file created by a SAVE command and accessed by means of an OLD, LIB, or GET command. Any of these three KRONOS commands causes a copy of the indicated permanent file to be generated as a working file.

WORKING FILE

A working file is a mass storage file, available to the user only through the duration of the terminal session. (The terminal session ends when the user logs off, or in some cases, if an accidental disconnect occurs.) A working file is obtained either as a new file (created by a NEW command) or as a working copy of an indirect access file (accessed by an OLD, LIB, or GET command).

PRIMARY WORKING FILE (OR PRIMARY FILE)

A primary working file is a working file with special handling characteristics (as explained in this section. The user can have only one primary file active at a given time.

EDIT FILE

An edit file is a working file under Text Editor control. When the Text Editor is active, one and only one edit file is active.

NOTE

If the edit file is a direct access file, manipulations are performed on the only existing copy of the information.

GAINING ACCESS

To gain access to the KRONOS system from a TTY-33/35 terminal, the user must perform the following log-in procedure.

1. If the terminal is not on-line, turn the terminal on (by pressing the ORIG button) and dial the system's telephone number.

2. If the connection is successful, the system prints the date, the time, and the system identification.

3. The system types the request

   ```
   USER NUMBER:
   ```

   Enter the assigned seven-character user number on the same line and press the RETURN key. For example,

   ```
   USER NUMBER:  USER123 (CR)
   ```

4. The system types the request

   ```
   PASSWORD
   ```

   Enter the password in the blacked-out area and press the RETURN key. If no password has been established, simply press the RETURN key.

5. The system prints

   ```
   TERMINAL:  nnnn, iii          or        TERMINAL:  nnnn, iii
   RECOVER/CHARGE:                         RECOVER/SYSTEM:
   ```

   nnnn is the 1- to 4-digit terminal number used by the system as part of the unique identifier for each active user. iii identifies the type of terminal being used (TTY in the case of a Teletype unit).

   After RECOVER/SYSTEM: enter the name of the KRONOS subsystem under which you intend to operate, such as BASIC or BATCH.

   However, it is possible to operate the Text Editor without specifying a subsystem; to do this, either type NULL or enter some other KRONOS command.

   After RECOVER/CHARGE: type the command CHARGE, followed by the assigned charge number and project number on the same line. For example,

   ```
   RECOVER/CHARGE: CHARGE, CH456, PN789 (CR)
   ```

   The system responds by typing

   ```
   READY.
   ```

   Enter the name of the KRONOS subsystem to be used (as is done if RECOVER/SYSTEM: had been printed).

6. The system prints

   OLD, NEW, OR LIB FILE:

   If you intend to build a new working file at this time, enter NEW on the same line. If you intend to edit an existing file, enter OLD or LIB (LIB specifies a library file). Press the RETURN key. For example,

   OLD, NEW, OR LIB FILE: NEW (CR)

7. The system requests

   FILE NAME:

   Enter the name of the file on the same line and press the RETURN key. The file whose name is entered is called the primary file. Each subsequent execution of an OLD, NEW, or LIB command establishes a different primary file and causes the previous primary file to be dropped.

   NOTE

   It is legal to enter OLD or NEW and the primary file parameters on the same line with the subsystem specification. If this is done, the FILE NAME request does not appear. For example: SYSTEM: BASIC, NEW, TEXT (CR)

8. If the system now replies

   READY.

   the file name specification is valid. If OLD is followed with a file name that does not exist in the set of permanent files, an appropriate error message is printed. Correct this error by typing in a valid file name.

The initial log-in procedure is now complete. A typical log-in procedure might appear as follows:

```
 72/12/06. 08.51.55.
KRØNØS TIME SHARING SYSTEM - VER. 2.1.
USER NUMBER: CRAIGGE
PASSWØRD
[redacted]
TERMINAL:     40, TTY
RECØVERY/SYSTEM: BASIC
ØLD, NEW, ØR LIB FILE:  NEW,ADDEM1
READY.

EDIT
 BEGIN TEXT EDITING.
```

ENTERING TEXT EDITOR

After log-in is complete, you are ready to enter Text Editor. This is done by typing the command

```
    EDIT (CR)
      or
    EDIT,lfn (CR)
```

lfn is the name of the file that you intend to edit. If lfn is not specified, the primary file is assumed.

The system replies

```
    BEGIN TEXT EDITING
    ?
```

This message indicates that you are under EDIT control. Note that you are now using a program designed to process only the Text Editor commands discussed in section 4 of this document. Thus, the regular KRONOS time-sharing commands are illegal until you exit the Text Editor. If may sometimes be necessary to enter and exit Text Editor several times during an editing session to use features not available under EDIT control.

It is possible to enter Text Editor without an edit file and develop the edit file at the beginning of the edit session. Refer to Adding and Building Text in section 4.

EXITING TEXT EDITOR

To terminate an editing session, type the command

```
    END (CR)
```

The system replies

```
    END TEXT EDITING
```

Edit operations can be resumed at any time by typing the system EDIT command.

WORKING FILES

All files associated with a KRONOS user are called working files (also known as local files or logical files in other manuals, and represented as lfn in command format parameters). One type of working file, the primary file, has special significance in certain KRONOS commands.

A user can have one primary file active at any given time. It can be a copy of a permanent file call by an OLD or LIB command or a file that is being initiated by a NEW command. It is possible to have no primary file if no OLD, NEW, or LIB command is issued during the session.

Whenever an OLD or NEW command is given, the file named in the command becomes the next primary file. The former primary file is released; it can be accessed again with a OLD or GET command only if the file was made permanent prior to its release.

KRONOS commands are performed using the primary file unless a different working file is specified in the command. For example, the command

```
SAVE (CR)
```

causes the current primary file to be stored in the permanent file system (assuming that a permanent file does not already exist under the same name). But the command

```
SAVE, F505 (CR)
```

causes the file F505 (which can be the primary file) to be stored in the permanent file system. Following any KRONOS operation on a primary file, the primary file is automatically positioned at its beginning. The user is responsible for the position of all working files, however. For detailed information on the KRONOS file system, refer to the KRONOS 2.1 Timing-Sharing User's Reference Manual.

FILE HANDLING PROCEDURES

This part discusses briefly the KRONOS file handling procedures and commands commonly used in conjunction with the Text Editor. Its purpose is to provide simple procedures that work; it does not reflect the full range of KRONOS file-handling capability. For a detailed discussion of KRONOS file handling concepts and the file handling commands, refer to the KRONOS 2.1 Time-Sharing User's Reference Manual.

CREATING A NEW FILE

To create a new file, enter a NEW command.

```
NEW (CR)
```

The system responds

```
FILE NAME:
```

and a valid file name should be entered. The name should be different from all permanent file names currently stored under the user number if the file is to be made permanent. When the system accepts the file name, it responds

```
READY.
```

The two steps can be combined into one by entering

```
NEW, lfn (CR)
```

where lfn is the name of the file.

Several methods of building a file are available. EDIT can be entered immediately and the file can be built using Text Edit or commands, as explained in the Adding and Building Text paragraphs. Also, various features of the KRONOS time-sharing language can be used. For information on the latter, refer to the following sections of the KRONOS 2.1 Time-Sharing User's Reference Manual.

Section 3: File Sorting

Section 4: Terminal Control Commands (AUTO, NORMAL); Time-Sharing Job Commands (NOSORT, PACK, SORT, TEXT).

SAVING A NEW FILE IN THE PERMANENT FILE SYSTEM

If the file is to be retained in the KRONOS permanent file system for subsequent use, type the command

```
    SAVE (CR)
```

to store the primary file, or

```
    SAVE,lfn (CR)
```

to store the working file lfn (lfn can be the name of the primary file). If the system responds

```
    lfn ALREADY PERMANENT
```

a permanent file named lfn already exists. To save the new working file without destroying the existing permanent file, enter the command

```
    SAVE, lfn = pfn (CR)
```

This causes the working file lfn to be saved permanently under the permanent file name pfn.

When the system responds

```
    READY.
```

the working file is established as a permanent file.

CAUTION

Newly created files should generally be saved prior to editing as a precautionary measure. Some Text Editor commands are powerful and can ruin a working file if the user makes a mistake. Also, it may be desirable to exit the Text Editor at various stages of the edit procedure to save the edit file.

ACCESSING A PERMANENT FILE

There are two modes of permanent file access available to the user, direct and indirect.

Direct access files are accessed by means of the KRONOS command ATTACH, as explained in the KRONOS Time-Sharing User's Reference Manual, Permanent File Processing Commands.

CAUTION

> When you are editing a file that has been accessed by means of an ATTACH command, you are working directly on the permanent file, not a copy thereof.

Indirect access files are accessed by using a working copy of the permanent file. The working copy is obtained through a GET or OLD command. After the file has been altered, the new version can be made permanent by a REPLACE command (or a SAVE, lfn=pfn command.

If the command

```
GET, pfn (CR)
    or
GET, lfn=pfn (CR)
```

is entered, the permanent file pfn is copied as a working file, and the primary file is unchanged. If the second form is used, reference the working file by specifying the file name lfn, not the permanent file name pfn.

If the command

```
OLD, pfn (CR)
```

is entered, a copy of the permanent file pfn becomes the primary file. To give the copy of pfn a different file name lfn, use the command

```
OLD, lfn=pfn (CR)
```

In either a GET or an OLD command, the choice of using the single file name (pfn) or the double file name specification (lfn=pfn) depends on what you intend to do with the working file. If you plan to store the working copy (with or without alternations) as a permanent file without disturbing the current version of file pfn, it is necessary to use an lfn=pfn specification or a RENAME command (discussed later in this section) at some point in the processing of the file (although not necessarily when the file is first accessed).

REPLACING A PERMANENT FILE

After a copy of a permanent file is obtained and alterations are performed, replace the former version with the altered version. If the working copy is obtained with the command

```
OLD, pfn (CR)
```

the subsequent command

```
REPLACE (CR)
```

causes the permanent file pfn to be purged and replaced with the working copy, including alterations.

The REPLACE command without parameters acts as a SAVE if the primary file does not have the same name as a permanent file. The command

```
    REPLACE,lfn (CR)
```

causes working file lfn to replace a permanent file of the same name if such a file exists; otherwise, it is equivalent to the command

```
    SAVE,lfn (CR)
```

The command

```
    REPLACE,lfn=pfn (CR)
```

causes working file lfn to replace the current contents of permanent file pfn.

It is important to use caution in the use of the REPLACE command or any other command that alters a permanent file. In a long session at the terminal, it is not uncommon for an operator to forget what the primary file is, and inadvertently specify a file replacement not intended. For example,

```
    OLD,BLAB
        .
        .
        .
    GET,GAB=BLAB
```

Two copies of the permanent file BLAB now exist as working files. If alterations are done on GAB, and you give the command

```
    REPLACE (CR)
```

the permanent file BLAB does not receive the changed version.

LISTING A FILE

Any working file can be listed with the command

```
    LIST,F=lfn (CR)
          or
    LNH,F=lfn (CR)  (List with no header)
```

The F=lfn parameter may be omitted if the primary file is being listed.

If the file has line numbers, begin the listing at line n with the command

```
    LIST,n (CR)
          or
    LNH,n (CR)
```

The n parameter and F=lfn cannot both be used in the same command.

RENAMING A WORKING FILE

The command

RENAME, lfn_1=lfn_2 $\textcircled{\scriptsize CR}$

changes the name of working file lfn_2 to lfn_1. If another working file has the name lfn_1, that file is released.

Example:

```
OLD, TXTFILE
  .
  .
  .
RENAME, TXTFL1=TXTFILE
SAVE
```

The working file is saved as TXTFL1 without affecting the permanent file TXTFILE.

ANNOTATED SAMPLE TERMINAL SESSION

The following sample terminal session illustrates the use of KRONOS file handling commands described earlier in this section. Several other commands, such as TEXT and RUN, are used in the example; although not directly related to Text Editor usage, they tend to be used frequently in a typical terminal session.

It is suggested that the reader try first to follow the session without reference to the numbered comments.

```
 73/02/20. 08.59.00.
KRØNØS TIME SHARING SYSTEM - VER. 2.1.  ┐
USER NUMBER: CRAIGGE                    │
PASSWØRD                                │
[redacted]                              ├ (1)
TERMINAL:      40, TTY                  │
RECØVER/ SYSTEM:BASIC                   │
ØLD, NEW, ØR LIB FILE:  NEW,FILDEM      │
READY.                                  ┘

TEXT  ←── (2)
READY.

THIS FILE IS BEING CREATED IN TEXT MØDE.  IT DØES NØT
REQUIRE LINE NUMBERS, AND IS BEING USED TØ DEMØNSTRATE
VARIØUS FILE HANDLING ØPTIØNS UNDER KRØNØS TIME SHARING.

EXIT TEXT MØDE. ←── (3)
LIST  ←──────────── (4)

 73/02/20. 09.05.54.
PRØGRAM    FILDEM

THIS FILE IS BEING CREATED IN TEXT MØDE.  IT DØES NØT
REQUIRE LINE NUMBERS, AND IS BEING USED TØ DEMØNSTRATE
VARIØUS FILE HANDLING ØPTIØNS UNDER KRØNØS TIME SHARING.
READY.

PACK ←── (5)


CP        0.008 SECS.
READY.

SAVE  ←── (6)
READY.

ØLD,RN2                     ┐
READY.                      │
                            │
LNH                         │
                            │
100 INPUT A1,A2,A3          │
110 FØR N=1 TØ A1           ├ (7)
120 LET C=N                 │
130 LET D=RND(A2)           │
140 LET E=INT(A3*D)         │
150 PRINT C,D,E             │
160 NEXT N                  │
170 END                     │
READY.                      ┘
```

Figure 2-1. Sample Terminal Session

```
RNH ←── (8)

? 55 *DEL* ←── (9)
5,-5,10
 1              .558929          5
 2              .664355          6
 3              5.96941E-3       0
 4              .128846          1
 5              .528141          5

CP         0.057 SECS.

RUN CØMPLETE.

NEW,FILDEM ←── (10)
READY.

00100 REM THIS GENERATES RANDØM NUMBER INTEGERS
00200 INPUT N
00300 FØR A=1 TØ N
00400LET B=INT(10*RND(-N))
00500 PRINT B
00600NEXT A
00700 END
RUN

 73/02/20. 09.36.55.
PRØGRAM    FILDEM

? 10
 7
 0
 2
 1
 0
 1
 8
 5
 8
 1


CP         0.032 SECS.

RUN CØMPLETE.                      ┐
                                   │
SAVE                               │
FILDEM ALREADY PERMANENT.          ├ (11)
                                   │
SAVE,FILDEM=FILDEM1                │
READY.                             ┘
```

Figure 2-1. Sample Terminal Session (Cont'd)

```
ØLD,FILDEM      ┐
READY.          │
                ├ ⑫
LNH             ┘

THIS FILE IS BEING CREATED IN TEXT MØDE.  IT DØES NØT
REQUIRE LINE NUMBERS, AND IS BEING USED TØ DEMØNSTRATE
VARIØUS FILE HANDLING ØPTIØNS UNDER KRØNØS TIME SHARING.
READY.

GET,FILDEM1     ┐
READY.          │
                ├ ⑬
LNH,F=FILDEM1   ┘

00100 REM THIS GENERATES RANDØM NUMBER INTEGERS.
00200 INPUT N
00300 FØR A=1 TØ N
00400LET B=INT(10*RND(-N))
00500 PRINT B
00600NEXT A
00700 END

CP        0.001 SECS.
READY.

TEXT                                                          ┐
READY.                                                        │
                                                              │
THIS SHØULD BE ADDED TØ THE END ØF THE FILE FILDEM,           │
NØT FILDEM1.                                                  │
                                                              │
EXIT TEXT MØDE.                                               │
PACK                                                          │
                                                              │
                                                              │
CP        0.003 SECS.                                         ├ ⑭
READY.                                                        │
                                                              │
LNH                                                           │
                                                              │
THIS FILE IS BEING CREATED IN TEXT MØDE.  IT DØES NØT         │
REQUIRE LINE NUMBERS, AND IS BEING USED TØ DEMØNSTRATE        │
VARIØUS FILE HANDLING ØPTIØNS UNDER KRØNØS TIME SHARING.      │
THIS SHØULD BE ADDED TØ THE END ØF THE FILE FILDEM,           │
NØT FILDEM1.                                                  │
READY.                                                        ┘
```

Figure 2-1. Sample Terminal Session (Cont'd)

```
REPLACE  <-----  (15)
READY.

RENAME,FILDEM1=FILDEM2         ┐
                               │
FILE NØT FØUND.                │
READY.                         │
                               │
RENAME,FILDEM2=FILDEM1         > (16)
                               │
                               │
                               │
CP        0.002 SECS.          │
READY.                         │
                               │
SAVE,FILDEM2                   │
READY.                         ┘

ØLD,FILDEM2
READY.

LIST,200  <-----  (17)

 73/02/20. 09.53.35.
PRØGRAM    FILDEM2

00200 INPUT N
00300 FØR A=1 TØ N
00400LET B=INT(10*RND(-N))
00500 PRINT B
00600NEXT A
00700 END
READY.
```

Figure 2-1. Sample Terminal Session (Cont'd)

(1.) This log-in sequence establishes user CRAIGGE under the BASIC subsystem. The user intends to build a new working file named FILDEM.

(2.) TEXT allows the building of a file without line numbers. All further information entered is treated as part of the file (that is, further KRONOS commands are impossible) until the user exits text mode.

(3.) Although it does not show on the listing, exit from text mode is accomplished by pressing the RETURN key, and then pressing the CNTL (or ATTN on some terminals) and C simultaneously. When the message

```
EXIT TEXT MODE.
```

appears, nontext mode is resumed.

(4) LIST with no parameters causes the primary file to be listed. The header information, consisting of the date, time, and file name, is not part of the file.

(5) PACK causes text mode files to be stored as a single logical record.

(6) The primary file is stored as a permanent file.

(7) OLD,RNZ establishes a copy of permanent file RN2 as the new primary file. The command LNH (list no header) is the same as a LIST, except no header is included in the listing.

(8) The listing of the file RNZ showed RNZ to be a short program written in BASIC. Now the command RNH (run no header) causes the program to be executed.

(9) The INPUT statement in the program causes the ? to be printed, requesting data. The user made a mistake and pressed ESC to delete the line, after which the intended input data was entered.

(10) The NEW statement causes the old primary file to be dropped. The fact that a permanent file named FILDEM already exists is of no concern at this time.

(11) The user attempted to save the newly constructed BASIC program as FILDEM, but a permanent file already exists under that name. Thus, the user stores the newly created file under a different name, FILDEM1.

(12) The permanent file FILDEM is copied as the new primary file. The user lists it without a header to ascertain that he has the proper file.

(13) Another working file, FILDEM1, now exists in addition to the primary file. Because it is not the primary file, the user must use the F=filename parameter.

(14) This sequence adds information to the end of the primary file FILDEM; to ascertain this, the file is listed.

(15) The new version of FILDEM replaces the old version in the permanent file system. If this were not done, the changes would be lost when the working file FILDEM is dropped for some reason.

(16) This sequence causes a copy of FILDEM1 to be placed in the permanent file system. Thus, two copies of the new BASIC program exist as permanent files.

(17) The command LIST, 200 causes the primary file to be listed starting at line 200; a header precedes the listing.

TEXT EDITING CONCEPTS 3

This section describes the fundamental concepts and terms associated with the KRONOS Text Editor as a preparation for the discussion of the edit commands. Included are such subjects as general command syntax, search logic, and the use of the string buffer.

EDIT FILE

The Text Editor operates on one and only one edit file at any given time. When the editing session is completed (that is, when the END command is given to the Text Editor), the edit file replaces the working file.

The Text Editor is entered with the KRONOS command

```
EDIT,lfn
```

The working file lfn becomes the edit file. If the file name lfn is omitted, the primary file becomes the edit file.

If the lfn parameter is included in the EDIT command, it is not necessary that lfn be an existing working file. In other words, EDIT,lfn can be used to create a new working file that is empty when the Text Editor is entered.

SEARCH POINTER

The search pointer is a place marker that indicates a particular line of the edit file. Unless command parameters indicate otherwise, the operation implied by the command word is performed on the line indicated by the search pointer. In any case, all action on a file begins relative to the search pointer.

The search pointer is set at the beginning of the edit file when EDIT is initiated. The SET, FIND, RESET, and LENGTH commands are used to change its value, and are the only commands capable of doing so.

A command that operates on more than one line of the edit file always begins operation at the line indicated by the search pointer (or relative to that line).

EDIT COMMAND (GENERAL FORMAT)

Each editing operation on the edit file is specified by an edit command. An edit command consists of the following four elements.

1. Command word
2. String specification, consisting of zero, one, or two string fields
3. n parameter, consisting of an unsigned integer
4. Comment

The string specification and/or the n parameter are not used with certain commands, and commentary is optional with all commands.

The general form of an edit command is

⟨cmwd⟩ ⟨strdef⟩ ⟨n⟩ ⟨comment⟩

where:

⟨cmwd⟩	Any EDIT command word, or short form thereof, as listed in Command Words
⟨strdef⟩	Any of the following: : ⟨string⟩ , ⟨string⟩ : ⟨string⟩ omitted ⟨string⟩ consists of any number of alphanumeric characters, bounded on each end of a nonblank character (called a delimiter). In most commands the string identifies the part of the edit file being sought, although in several commands the string has a special purpose. The delimiters on each end must be the same character and must not be the character $.
⟨n⟩	Either of the following: ; n omitted n is an integer or an asterisk. The integer is unsigned, except that a negative n is permitted if ⟨cmwd⟩ is SET. An asterisk implies an n value equal to the number of text lines in the edit file.
⟨comment⟩	A comment, if included, consists of a dollar sign ($), followed by any sequence of characters ending with the (CR) . It has no effect on the operation of the command.

The entire command, including comment, must be on one line. Pressing the carriage return signifies the end of the command. Only one command is permitted on a single line.

LINE MODE AND STRING MODE

Some edit commands have two modes of operation, line mode and string mode. In a line mode command, all operations are performed with a line of the edit file as the basic unit of operation. In a string mode command, all operations are performed with a character string as the basic unit of operation. The string may be a portion of a line or may extend over several lines.

NOTE

It is important not to confuse string mode with the search string used in both line mode and string mode edit commands. The search string specifies the point or area of the edit file to which the command operation is directed. The string mode refers to the nature of the command operation.

A string mode command with an empty search string specification has the same action as the corresponding line mode command.

COMMAND WORDS

The command word determines the operation to be performed. The EDIT command words are listed with their corresponding short forms (if any) shown in parentheses.

Line Command Words		String Command Words	
ADD	(A)	ADDS	(AS)
BLANK	(B)	BLANKS	(BS)
CHANGE	(C)	CHANGES	(CS)
DELETE	(D)	DELETES	(DS)
EXTRACT	(E)	EXTRACTS	(ES)
FIND	(F)	FINDS	(FS)
LIST	(L)	INSERTS	(IS)
NUMBER	(N)	LISTS	(LS)
		NUMBERS	(NS)
		REPLACES	(RS)

Note that all string command words, and only string command words, end with an S. Also note that INSERTS and REPLACES are string commands without corresponding line command format.

Control Command Words	
ALIGN	(AL)
CLEAR	(CL)
DEFTAB	(DT)
END	
LENGTH	
LINE	(LN)
LISTAB	(LT)
MERGE	(M)
RESET	(R)
SET	(S)
TAB	(T)
WIDTH	(W)

Thus, there are 22 basic command words in the Text Editor language, eight of which have forms for both line mode and string mode. Also, 28 of the 30 command word forms have an equivalent short form.

STRINGS AND DELIMITERS

A string is a sequence of alphanumeric characters that may include blanks and special characters. Strings are used in two ways.

1. In the ⟨strdef⟩ field of a Text Editor command
2. In response to an ENTER TEXT request

The two ends of the string must be explicitly defined by a pair of matching characters called delimiters. A delimiter is any nonblank character except a dollar sign, and is chosen by the user. The delimiter character can be used within the string, except at the end of a line, because EDIT tests for the closing delimiter only after a carriage return.

(In this manual the character / (slash or virgule) is used to denote a delimiter in the presentation of command formats.)

<u>Correct String Definition</u>	
/ABCDE/	
/THE FORMAT OF/	
BALWAYS IS B	
? INT(R*TAN(2*M))?	
<u>Incorrect String Definition</u>	
/THIS STATEMENT WILL	(no closing delimiter)
(HOWEVER)	(different delimiter characters)
ANY COMMAND TERMINATED BY/	(unintended beginning delimiter)
$THIS LOOKS LIKE A COMMENT$	(illegal delimiter character)

CAUTION

Improper or unintended string definitions are common errors, and because of the powerful nature of some Text Editor commands, are potentially destructive to a file.

SEARCH STRING PARAMETER

The search string parameter of an EDIT command indicates to the Text Editor where the operation is to be performed. If no search string is given in a command, the operational location depends solely on the setting of the search pointer. If a search string is given, the operation specified is performed with respect to the first occurrence of the string after the beginning of the line indicated by the search pointer.

If the specified string does not occur after the beginning of the line indicated by the search pointer, the following message is printed.

```
PHRASE NOT FOUND
```

The search string must be specified to identify uniquely the string being sought. If too small a string is given, the search may result in operating on an occurrence of the string that was not the intended target.

A search string is given in two forms, a single phrase or an ellipsis.

SINGLE PHRASE SEARCH STRING

In a single phrase search string, the entire string of consecutive characters is placed between a pair of delimiters. The string can include as many characters as required (subject to the requirement that the entire command be on a single line), and the search is satisfied only when an identical string is found within a single line of the edit file.

ELLIPSIS SEARCH STRING

An ellipsis search string specification consists of two delimited bracket strings, separated by a comma. The search process attempts to locate a string of consecutive characters that begins with the first phrase and ends with the second phrase. The string implied by an ellipsis search string may appear in the file over more than one line.

Example:

The ellipsis search string

```
:/FORM/,/LONG/
```

is satisfied by the string underlined.

THE ELLIPSIS IS A <u>FORM OF SHORTHAND FOR LONG</u> OR MULTILINE STRINGS.

One frequent source of error in using ellipsis search strings is a tendency to make the bracket strings too short. Consider the following text.

AS ANOTHER EXAMPLE, ASSUME THAT <u>THE TARGET STRING EXTENDS OVER SEVERAL LINES LIKE THIS ONE.</u>

If the underlined string is to be referenced, a command with the following string specification might be entered.

```
:/THE/,/ONE/
```

This does not reference the string desired, however, because the first occurrence of THE is in the word ANOTHER. The string specification

```
:/THE T/,/ONE/
```

identifies the underlined string properly.

SPECIAL STRING FIELDS

A special string has a format similar to that of a search string. Its interpretation depends on the command word with which it appears. The following are the three types of special string fields and the statements with which they are used.

- Tab stop sequence in a TAB command
- Tab character defined in a DEFTAB command
- Merge file name in a MERGE command

n PARAMETER

The n parameter is an integer whose meaning depends on the context in which it appears; its use adds flexibility to EDIT commands. The following are possible interpretations.

- The number of lines on which a command is to be performed
- The number of lines the search pointer is to be moved forward or backward
- The length of a file in lines or the maximum width of the lines in character columns
- The point in a file where new data is to be inserted

When omitted, n is assumed to equal 1 if applicable. The n parameter is not applicable for the commands RESET, LINE, LISTAB, CLEAR, NUMBER(S), DEFTAB, and END. Negative values of n are allowed only in a SET command.

An asterisk (*) instead of a number in the n parameter indicates that the operation is performed at or until the end of the edit file. Refer to the description of the particular command of interest for specific details.

DOCUMENTARY COMMENTS

To annotate the editing session (possibly for review purposes), append a dollar sign to any or all commands and follow the dollar sign with commentary information. The comment is ignored by the editor.

STRING BUFFER

The string buffer is a temporary storage area for information that is to be moved within the edit file.

Information is copied from the edit file into the string buffer using the EXTRACT command. This information may then be inserted elsewhere in the file, using the ADD or CHANGE command.

After the ADD or CHANGE command is entered, the system responds

```
    ENTER TEXT
    ?
```

If the user responds by typing

```
    $ (CR)
```

on the same line, the contents of the string buffer are inserted into the edit file at the point or points indicated by the ADD or CHANGE command.

The CLEAR command, and only the CLEAR command, erases the contents of the string buffer. CLEAR is used whenever the contents of the string buffer is no longer needed. Until a CLEAR command is issued, repeated EXTRACT operations cause extracted strings to appear cumulatively in the string buffer, concatenated in the order of their extraction. Thus, for separate and unrelated extractions, failure to use a CLEAR command may produce unintended and rather startling results.

ENTER TEXT REQUEST

The Text Editor issues an ENTER TEXT request in response to an ADD command and in response to a CHANGE command.

After the ENTER TEXT request, type an opening delimiter, followed by the body of text to be entered, and then followed by a closing delimiter. The delimiters do not become part of the actual file.

The delimiter character is the first nonblank character entered in response to the ENTER TEXT request. The closing delimiter is the first recurrence of the delimiter character that is followed immediately by a carriage return.

This delimiter character is allowed in the body of the text; however, this should be avoided because the character at the end of a line may be inadvertently typed before the typing of the intended entry is complete.

The Text Editor types a question mark at the beginning of each line until the closing delimiter appears. The system then responds

```
    READY.
?
```

The READY message indicates that the next line entered is treated as a Text Editor command.

EDIT COMMANDS 4

This section describes the allowable formats for each Text Editor command and rules governing their use. The commands are grouped by general category of function; for example, the removal of information category includes the DELETE and BLANK commands.

A group of contextual examples is included at the end of each category. These examples are designed to illustrate the effect of the various formats, and in particular, to clarify the differences between similar commands. An arrow in the right-hand margin of an example indicates a special or nontrivial point of interest.

ENTERING COMMANDS

All Text Editor commands are entered at the time-sharing terminal according to the general format described in section 3 of this manual. After an EDIT command is typed and the RETURN key is pressed, the Text Editor either executes the command immediately or requests additional information. Appendix B contains a summary of all Text Editor messages and requests.

Edit operations are performed in response to specific commands entered at the terminal.

TEXT LISTING AND SEARCH POINTER CONTROL

LIST COMMAND

The LIST command allows the operator to print all or selected portions of the edit file. The printout can include a string of characters, a single line, a set of lines each including a common character string, or a set of contiguous lines.

If an asterisk is specified in the n parameter or if the value of the n parameter extends beyond the end of the edit file, all remaining lines are printed, followed by

```
    -END OF FILE-
```

If an ellipsis string is specified, a line mode command causes all lines to be printed that contain any portion of the ellipsis string. A string mode command prints only the string implied by the ellipsis.

LINE MODE FORMATS (LIST OR L)

Command	Explanation
LIST	Prints the line of text specified by the search pointer
LIST;n	Prints n lines of contiguous text, beginning at the search pointer. (If n equals *, all lines to the end of the edit file are printed.)
LIST:/string/	Prints the line containing the specified string (the phrase must be contained in a single line)
LIST:/string/;n	Prints the first n lines containing the string (n can equal *, in which case all lines in the edit file that contain the string are printed)
LIST:/string1/, /string2/	Prints the line or group of lines containing the ellipsis string1 through string2
LIST:/string1/, /string2/;n	Prints the first n occurrences of lines or groups of lines containing the ellipsis string1 through string2

STRING MODE FORMATS (LISTS OR LS)

Command	Explanation
LISTS	Same as LIST
LISTS;n	Same as LIST;n
LISTS:/string/	Prints the specified string, if present in the edit file
LISTS:/string/;n	Prints the first n occurrences of the string
LISTS:/string1/, /string2/	Prints the string of characters specified by the ellipsis /string1/, /string2/
LISTS:/string1/, /string2/;n	Prints the first n occurrences of the string of characters specified by the ellipsis /string1/, /string2/

FIND COMMAND

The FIND command scans the edit file, beginning at the line indicated by the search pointer. When a line (or string) is encountered that fulfills the combined requirements of the search string and/or the n parameter, the Text Editor lists that line or string and sets the search pointer accordingly (as explained in the discussion of the FIND formats).

If n is omitted, n=1 is assumed.

If the end of the edit file is reached before the nth occurrence is found, the search pointer is set to the first line of the last string found.

LINE MODE FORMATS (FIND OR F)

Command	Explanation
FIND;n	Advances the search pointer n lines and lists the line indicated by the new value of the search pointer
FIND:/string/;n	Advances the search pointer to the nth line that contains at least one occurrence of /string/, and lists the line
FIND:/string1/,/string2/;n	Advances the search pointer from its current position to the nth line that contains the beginning of the ellipsis search string. If the search string is multiline, all lines containing some part of the nth occurrence of /string1/,/string2/ are listed, and the search pointer is set to the line in which the nth occurrence begins.

STRING MODE FORMATS (FINDS OR FS)

Command	Explanation
FINDS;n	Same as FIND;n
FINDS:/string/;n	Advances the search pointer to the line containing the nth occurrence of /string/, and lists the string
FINDS:/string1/,/string2/;n	Advances the search pointer to the line containing the beginning of the nth occurrence of /string1/,/string2/. The string is listed.

SEARCH POINTER CONTROL (SET AND RESET)

EDIT initially locates the search pointer at the first line of the edit file. With the SET command, the search pointer can be moved to a particular line in the edit file without listing it. The RESET command sets the search pointer to the first line of the edit file, regardless of its former position. Activity on the edit file always begins at the current search pointer setting.

SET COMMAND (SET OR S)

The following are the three forms of the SET command.

Command	Explanation
SET;n or SET;-n	Advances (or sets back) the search pointer n lines relative to its current setting. If the SET instruction results in a negative search pointer (the pointer being set back past the beginning of the file), the pointer is set to the first line. (If n equals * or extends beyond the end of the file, the pointer is set to the end of the edit file.)
SET:/string/	Moves the search pointer to the next line containing the string, relative to the current setting of the search pointer
SET:/string/;n	Moves the search pointer forward from its current setting to the beginning of the line containing the nth occurrence of the search string

The SET command requires locational information. If no search string is present, the use of an n parameter is implied. If the command contains neither a search string nor an n parameter, an n parameter value of 1 is assumed.

Only single-phrase search strings are allowed. Ellipsis search strings are not allowed.

Using a search string without an n parameter moves the search pointer from its current setting forward to the line containing the first occurrence of the search string.

RESET COMMAND (RESET OR R)

The RESET command brings the search pointer to the beginning of the edit file. Its format is

```
RESET
```

Operand fields are not used with the RESET command.

LINE COMMAND (LN)

The LINE command causes a message to be printed that gives the current setting of the search pointer.

The format is

```
LINE
```

The message is

```
FILE AT LINE NUMBER        n
```

where n indicates the line of the edit file to which the search pointer is currently pointing. If n is the last line of the file, the words

```
-END OF FILE-
```

are included in the message.

```
EDIT

 BEGIN TEXT EDITING.
? LIST;*      $THIS LISTS THE ENTIRE FILE                       ←
     THIS IS THE FIRST SENTENCE ØF A PARAGRAPH.  THIS IS THE
SECØND SENTENCE ØF PARAGRAPH ØNE.  SENTENCE THREE IS HERE, AND
THIS CLAUSE IS PART ØF SENTENCE THREE.  THIS ENTIRE FILE MAKES
LITTLE SENSE, BUT IS CØNSTRUCTED FØR THE PURPØSE ØF
DEMØNSTRATING TEXT EDITØR CØMMANDS.
     THIS SENTENCE BEGINS THE SECØND PARAGRAPH ØF THE FILE.
THIS SENTENCE, ØN THE ØTHER HAND, ENDS IT.
     THE THIRD PARAGRAPH ØF THIS EXCEEDINGLY NØNSENSICAL
TEXT SEQUENCE HEADS UP A LIST ØF EQUALLY NØNSENSICAL ITEMS:
             1.  THIS IS ITEM ØNE
             2.  THIS ITEM IS THE SECØND
             3.  AND THIS IS THE THIRD AND LAST ITEM
THIS SENTENCE CØNTINUES THE PARAGRAPH FØLLØWING THE LIST, UNLESS,
ØF CØURSE, WE DECIDE TØ REMØVE IT ENTIRELY ØR IN PART WITH
TEXT EDITING CØMMANDS.
     AND HERE, THANKFULLY, IS THE FØURTH AND FINAL PARAGRAPH,
ALTHØUGH NØT NECESSARILY THE FINAL SENTENCE.  THIS SAMPLE
FILE WILL APPEAR IN VARIØUS PARTS ØF THE TEXT EDITØR MANUAL
TØ ILLUSTRATE THE ACTIØN ØF VARIØUS EDITING CØMMANDS.
 -END ØF FILE-
? SET;8                                                         ←
? LIST;4
TEXT SEQUENCE HEADS UP A LIST ØF EQUALLY NØNSENSICAL ITEMS:
             1.  THIS IS ITEM ØNE
             2.  THIS ITEM IS THE SECØND
             3.  AND THIS IS THE THIRD AND LAST ITEM
? SET;-3        $MØVE SEARCH PTR BACK 3 LINES                   ←
? L             $L IS A VALID ABBREVIATIØN FØR LIST
     THIS SENTENCE BEGINS THE SECØND PARAGRAPH ØF THE FILE.
? SET:7CØNTINUES7                                               ←
? LISTS:/EXCEEDINGLY NØNSENSICAL/
 -END ØF FILE-
? FIND:8EXCEEDINGLY NØNSENSICAL8
 PHRASE NØT FØUND.
? L
THIS SENTENCE CØNTINUES THE PARAGRAPH FØLLØWING THE LIST, UNLESS,
? LISTS:/UNL/,/THANKF/                                          ←
                                                          UNLESS,
ØF CØURSE, WE DECIDE TØ REMØVE IT ENTIRELY ØR IN PART WITH
TEXT EDITING CØMMANDS.
     AND HERE, THANKF
```

Figure 4-1. Examples of LIST, FIND, SET, RESET, and LINE Usage

```
? RESET
? SET:6PARAG6;4
? LIST
     THE THIRD PARAGRAPH ØF THIS EXCEEDINGLY NØNSENSICAL
? FIND
TEXT SEQUENCE HEADS UP A LIST ØF EQUALLY NØNSENSICAL ITEMS:
? FIND
           1.  THIS IS ITEM ØNE
? FINDS
           2.  THIS ITEM IS THE SECØND
? FINDS:/FINAL/,/FILE/
                                                FINAL PARAGRAPH,
ALTHØUGH NØT NECESSARILY THE FINAL SENTENCE.  THIS SAMPLE
FILE
? FIND:8FINAL8
     AND HERE, THANKFULLY, IS THE FØURTH AND FINAL PARAGRAPH,
? FIND:7FINAL7
     AND HERE, THANKFULLY, IS THE FØURTH AND FINAL PARAGRAPH,
? RESET
? FIND:/FINAL/;*
ALTHØUGH NØT NECESSARILY THE FINAL SENTENCE.  THIS SAMPLE
       2  ØCCURANCES ØF PHRASE FØUND.
? RESET
? LISTS:/SENTENCE/;*
                        SENTENCE
       SENTENCE                             SENTENCE
                        SENTENCE
           SENTENCE
     SENTENCE
     SENTENCE
                                            SENTENCE

 -END ØF FILE-
? FINDS:/SENTENCE/;*
                                            SENTENCE
       8  ØCCURANCES ØF PHRASE FØUND.
? RESET
? FINDS:/NØNSEN/;2
                                                  NØNSEN
? LINE
 FILE AT LINE NUMBER       9.
? SET;-3
? LINE
 FILE AT LINE NUMBER       6.
? SET;37
? LINE
 FILE AT LINE NUMBER      20     -END ØF FILE-
? END
 END TEXT EDITING.
```

Figure 4-1. Examples of LIST, FIND, SET, RESET, and LINE Usage (Cont'd)

ADDING AND BUILDING TEXT

The ADD and INSERTS commands cause new information to be included in the edit file at a place specified by the user.

ADD COMMAND

An ADD operation requires two sets of information, the location where the text is added (supplied in the command) and the actual new information to be inserted in the edit file (supplied by the user in response to the ENTER TEXT request).

After the command is entered, the system types

```
ENTER TEXT
?
```

Respond to this request in one of three ways.

1. Type the actual information to be added (including carriage returns and line numbers if required), bracketed with delimiters.

2. Type the dollar sign ($) character with no delimiters or other characters. This causes the current contents of the string buffer to be added. (Information is placed in the string buffer by one or more EXTRACT statements.)

3. Type CR only. This causes the data entered in response to the most recent previous ENTER TEXT request to be added.

Only single phrase search strings are allowed with this command. Ellipsis search string specifications are illegal.

With no search string specification in force, the n parameter indicates where the insertion shall be made relative to the search pointer.

LINE MODE FORMATS (ADD OR A)

Command	Explanation
ADD	Inserts text after the line of the edit file specified by the search pointer
ADD;n	Inserts text after the nth line (counting forward from the search pointer) of the edit file
ADD:/string/	Inserts text after the line containing the specified string
ADD:/string/;n	Inserts text after each of the first n lines containing the specified string

STRING MODE FORMATS (ADDS OR AS)

Command	Explanation
ADDS	Same as ADD
ADDS;n	Same as ADD;n
ADDS:/string/	Inserts text immediately following the specified string
ADDS:/string/;n	Inserts text immediately following each of n occurrences of the specified string

Line mode ADD commands cause the addition of text following the end of a particular line, whereas string mode ADD commands cause text to be added following a particular string of characters. A string mode command without a string specification is equivalent to a line mode command.

INSERTS COMMAND (INSERTS OR IS)

The INSERTS command is similar in purpose to the ADDS command, except that the text to be inserted is embedded within the command, thus speeding the interaction.

The command has the following format.

```
INSERTS:/string1/,/string2/;n
```

If the n parameter is omitted, 1 is assumed.

The character string denoted by string2 is inserted immediately after each of n occurrences of string1, beginning at the search pointer. Note that /string1/,/string2/ specification is not an ellipsis search string in this command.

```
 BEGIN TEXT EDITING.
? ADD
 ENTER TEXT.
? /      THIS FILE IS BEING BUILT TØ DEMØNSTRATE THE ADD.
? DURING THIS SESSIØN VARIØUS FØRMS ØF THE ADD WILL BE USED,
? BØTH LINE ADD AND STRING ADD, AS WELL AS THE INSERT
? CØMMAND./
 READY.
? ADD
 ENTER TEXT.
? /     ***THE PLACEMENT ØF THIS LINE MAY SURPRIZE YØU.*/
 READY.
? LIST;*
     THIS FILE IS BEING BUILT TØ DEMØNSTRATE THE ADD.
    ***THE PLACEMENT ØF THIS LINE MAY SURPRIZE YØU.*
DURING THIS SESSIØN VARIØUS FØRMS ØF THE ADD WILL BE USED,
BØTH LINE ADD AND STRING ADD, AS WELL AS THE INSERT
CØMMAND.
 -END ØF FILE-
? ADD;*
 ENTER TEXT.
? /      EVERY CØMMAND HAS A WØRKING N PARAMETER.  IF IT
? IS NØT GIVEN EXPLICITLY IN THE CØMMAND, AN N PARAMETER
? VALUE ØF 1 IS AUTØMATCLY ASSUMED.
?       IN THIS CASE AN ASTERISK (*) IN THE N PARAMETER
? CAUSES THE ADDITIØN ØF TEXT TØ TAKE PLACE AT THE END
? ØF THE FILE./
 READY.
? ADD;5
 ENTER TEXT.
? 8      THIS PARAGRAPH, ØR SET ØF LINES, IS PLACED AFTER
? THE FIFTH LINE ØF THE FILE.8
 READY.
? ADD:9ØF 19
 ENTER TEXT.
? /   A SET ØF LINES IS NØW BEING PLACED FØLLØWING THE
? LINE THAT CØNTAINS THE FIRST ØCCURRENCE ØF THE STRING "ØF 1"./
 READY.
? ADD:/*/;*
 ENTER TEXT.
? /LINES*HAVING*AT*LEAST*ØNE ASTERISK*ARE*FØLLØWED*BY*THIS*LINE/
 READY.
      2  ØCCURANCES ØF PHRASE FØUND.
? LIST;*
     THIS FILE IS BEING BUILT TØ DEMØNSTRATE THE ADD.
    ***THE PLACEMENT ØF THIS LINE MAY SURPRIZE YØU.*
LINES*HAVING*AT*LEAST*ØNE ASTERISK*ARE*FØLLØWED*BY*THIS*LINE
DURING THIS SESSIØN VARIØUS FØRMS ØF THE ADD WILL BE USED,
BØTH LINE ADD AND STRING ADD, AS WELL AS THE INSERT
CØMMAND.
     THIS PARAGRAPH, ØR SET ØF LINES, IS PLACED AFTER
THE FIFTH LINE ØF THE FILE.
     EVERY CØMMAND HAS A WØRKING N PARAMETER.  IF IT
IS NØT GIVEN EXPLICITLY IN THE CØMMAND, AN N PARAMETER
VALUE ØF 1 IS AUTØMATCLY ASSUMED.
     A SET ØF LINES IS NØW BEING PLACED FØLLØWING THE
LINE THAT CØNTAINS THE FIRST ØCCURRENCE ØF THE STRING "ØF 1".
     IN THIS CASE AN ASTERISK (*) IN THE N PARAMETER
LINES*HAVING*AT*LEAST*ØNE ASTERISK*ARE*FØLLØWED*BY*THIS*LINE
CAUSES THE ADDITIØN ØF TEXT TØ TAKE PLACE AT THE
END ØF THE FILE.
 -END ØF FILE-
```

Figure 4-2. Examples of ADD and INSERTS Usage

```
? ADDS:5INSERT5
 ENTER TEXT.
? 9S9
 READY.
? ADDS:/ADD/;*    $THIS SHØWS IMPØRTANCE ØF CAREFUL PARAM SELECTIØN
 ENTER TEXT.
? 7 CØMMAND7
 READY.
      5  ØCCURANCES ØF PHRASE FØUND.
? LIST:/ADD/;5
     THIS FILE IS BEING BUILT TØ DEMØNSTRATE THE ADD CØMMAND.
DURING THIS SESSIØN VARIØUS FØRMS ØF THE ADD CØMMAND WILL BE USED,
BØTH LINE ADD CØMMAND AND STRING ADD CØMMAND, AS WELL AS THE INSERTS
CAUSES THE ADD CØMMANDITIØN ØF TEXT TØ TAKE PLACE AT THE
 -END ØF FILE-
? INSERTS:/  EVERY/, / ADD/
INSERTS   SYNTAX ERRØR.
? INSERTS:/  EVERY/,/ ADD/
? INSERTS:/AUTØMAT/,/I/
? INSERTS:/N PARAMETER/,/ FIELD/;3
? LIST;*
     THIS FILE IS BEING BUILT TØ DEMØNSTRATE THE ADD CØMMAND.
    ***THE PLACEMENT ØF THIS LINE MAY SURPRIZE YØU.*
LINES*HAVING*AT*LEAST*ØNE ASTERISK*ARE*FØLLØWED*BY*THIS*LINE
DURING THIS SESSIØN VARIØUS FØRMS ØF THE ADD CØMMAND WILL BE USED,
BØTH LINE ADD CØMMAND AND STRING ADD CØMMAND, AS WELL AS THE INSERTS
CØMMAND.
     THIS PARAGRAPH, ØR SET ØF LINES, IS PLACED AFTER
THE FIFTH LINE ØF THE FILE.
     EVERY ADD CØMMAND HAS A WØRKING N PARAMETER FIELD.  IF IT
IS NØT GIVEN EXPLICITLY IN THE CØMMAND, AN N PARAMETER FIELD
VALUE ØF 1 IS AUTØMATICLY ASSUMED.
     A SET ØF LINES IS NØW BEING PLACED FØLLØWING THE
LINE THAT CØNTAINS THE FIRST ØCCURRENCE ØF THE STRING "ØF 1".
     IN THIS CASE AN ASTERISK (*) IN THE N PARAMETER FIELD
LINES*HAVING*AT*LEAST*ØNE ASTERISK*ARE*FØLLØWED*BY*THIS*LINE
CAUSES THE ADD CØMMANDITIØN ØF TEXT TØ TAKE PLACE AT THE
END ØF THE FILE.
 -END ØF FILE-
```

Figure 4-2. Examples of ADD and INSERTS Usage (Cont'd)

```
? INSERTS:/LINE ADD CØMMAND/,/S (ADD)/                          ←
? ADDS:/STRING ADD CØMMAND/
 ENTER TEXT.
? /S
? (ADDS)/
 READY.
? ADDS:/INSERTS CØMMAND/                                        ←
 PHRASE NØT FØUND.
? SET;6
? LIST
CØMMAND.
? ADDS:/ND./
 ENTER TEXT.
? /  THE INSERTS CØMMAND IS A FAST WAY TØ MAKE TEXT INSERTIØNS
? THAT DØ NØT EXTEND BEYØND THE END ØF A LINE.  THE ADDS
? CØMMAND IS SLØWER BUT ALLØWS CARRIAGE RETURNS.  THE
? SEARCH STRING ØF AN ADD, ADDS, ØR INSERTS STATEMENT CANNØT
? EXTEND ØVER MØRE THAN ØNE LINE, AS WAS SHØWN ABØVE./
 READY.
? RESET
? LIST;*   $FINAL RUNØFF
     THIS FILE IS BEING BUILT TØ DEMØNSTRATE THE ADD CØMMAND.
    ***THE PLACEMENT ØF THIS LINE MAY SURPRIZE YØU.*
LINES*HAVING*AT*LEAST*ØNE ASTERISK*ARE*FØLLØWED*BY*THIS*LINE
DURING THIS SESSIØN VARIØUS FØRMS ØF THE ADD CØMMAND WILL BE USED,
BØTH LINE ADD CØMMANDS (ADD) AND STRING ADD CØMMANDS
(ADDS), AS WELL AS THE INSERTS
CØMMAND.  THE INSERTS CØMMAND IS A FAST WAY TØ MAKE TEXT INSERTIØNS
THAT DØ NØT EXTEND BEYØND THE END ØF A LINE.  THE ADDS
CØMMAND IS SLØWER BUT ALLØWS CARRIAGE RETURNS.  THE
SEARCH STRING ØF AN ADD, ADDS, ØR INSERTS STATEMENT CANNØT
EXTEND ØVER MØRE THAN ØNE LINE, AS WAS SHØWN ABØVE.
     THIS PARAGRAPH, ØR SET ØF LINES, IS PLACED AFTER
THE FIFTH LINE ØF THE FILE.
     EVERY ADD CØMMAND HAS A WØRKING N PARAMETER FIELD.  IF IT
IS NØT GIVEN EXPLICITLY IN THE CØMMAND, AN N PARAMETER FIELD
VALUE ØF 1 IS AUTØMATICLY ASSUMED.
     A SET ØF LINES IS NØW BEING PLACED FØLLØWING THE
LINE THAT CØNTAINS THE FIRST ØCCURRENCE ØF THE STRING "ØF 1".
     IN THIS CASE AN ASTERISK (*) IN THE N PARAMETER FIELD
LINES*HAVING*AT*LEAST*ØNE ASTERISK*ARE*FØLLØWED*BY*THIS*LINE
CAUSES THE ADD CØMMANDITIØN ØF TEXT TØ TAKE PLACE AT THE
END ØF THE FILE.
 -END ØF FILE-
? END
 END TEXT EDITING.
READY.
```

Figure 4-2. Examples of ADD and INSERTS Usage (Cont'd)

REMOVAL OF INFORMATION

Two types of operation are available for removing information from the edit file, DELETE and BLANK.

DELETE COMMAND

A DELETE operation erases one or more occurrences of a particular string of characters or one or more lines containing a particular string of characters. The text is realigned, leaving no excess blanks.

LINE MODE FORMATS (DELETE OR D)

Command	Explanation
DELETE	Erases the line of the edit file specified by the search pointer
DELETE;n	Erases the first n lines of the edit file beginning at the search pointer
DELETE:/string/	Erases the line containing the string
DELETE:/string/;n	Erases the first n lines containing the string
DELETE:/string1/, /string2/	Erases the line or group of lines containing string1 and string2
DELETE:/string1/, /string2/;n	Erases the first n occurrences of the line or group of lines containing string1 and string2

STRING MODE FORMATS (DELETES OR DS)

Command	Explanation
DELETES	Same as DELETE
DELETES;n	Same as DELETE;n
DELETES:/string/	Erases the specified string
DELETES:/string/;n	Erases the first n occurrences of the specified string
DELETES:/string1/, /string2/	Erases the string of characters specified by the ellipsis /string1/, /string2/
DELETES:/string1/, /string2/;n	Erases the first n occurrences of the string of characters specified by the ellipsis /string1/, /string2/

BLANK COMMAND

The BLANK command replaces a specified string, line, or set of lines with blank characters. Unlike the DELETE command, BLANK does not relocate text.

LINE MODE FORMATS (BLANK OR B)

Command	Explanation
BLANK	Replaces with blanks the line of the edit file specified by the search pointer
BLANK;n	Replaces with blanks the first n lines of the edit file, beginning at the search pointer
BLANK:/string/	Replaces with blanks the line containing the string
BLANK:/string/;n	Replaces with blanks the first n lines containing the string
BLANK:/string1/,/string2/	Replaces with blanks the first line or group of lines containing string1 and string2
BLANK:/string1/,/string2/;n	Replaces with blanks the first n occurrences of the line or group of lines containing string1 and string2

STRING MODE FORMATS (BLANKS OR BS)

Command	Explanation
BLANKS	Same as BLANK
BLANKS;n	Same as BLANK;n
BLANKS:/string/	Replaces with blanks the specified phrase
BLANKS:/string/;n	Replaces with blanks the first n occurrences of the specified phrase
BLANKS:/string1/,/string2/	Replaces with blanks the string defined by the ellipsis /string1/,/string2/
BLANKS:/string1/,/string2/;n	Replaces with blanks the first n occurrences of the string defined by the ellipsis /string1/,/string2/

```
 BEGIN TEXT EDITING.
? L;*
THIS DEMØNSTRATES VARIØUS FØRMS ØF THE DELETE AND BLANK
CØMMANDS.  THESE CØMMANDS ARE VERY SIMILAR, EXCEPT THAT
THE DELETE ADJUSTS ØR RELØCATES THE TEXT,WHEREAS THE BLANK
LEAVES ØPEN SPACE IN THE AREAS FRØM WHICH TEXT WAS REMØVED.
THE INFØRMATIØN BELØW IS THE MATERIAL ØN WHICH WE WILL
BE WØRKING PRIMARILY.

THIS IS A SAMPLE FILE TØ DEMØNSTRATE THE ADD CØMMAND.
YØU MAY BE SURPRIZEDTHØRØUGHLY AT WHERE THIS LINE ØF TEXT IS INSERTED.
THIS TIME THE INSERTIØN SHØULD WØRK.SHØULD FINALLY WØRK PRØPERLY.
WATCH CAREFULLY WHERE THIS LINE ØF TEXT IS INSERTED.
THIS LINE ØF TEXT IS INSERTED AFTER THE LINE THAT CØNTAINS "SURPRIZED."
THIS TIME THE INSERTIØN SHØULD WØRK.
WATCH CAREFULLY WHERE THIS LINE ØF TEXT IS INSERTED.
THIS LINE ØF TEXT IS INSERTED AFTER THE SECØND EXISTING LINE.
THIS SENTENCE IS BEING ENTERED BY THE SIMPLE ADD LINE
CØMMAND; IT HAS NØ SEARCH STRING ØR N PARAMETER.
THIS IS AT THE END ØF THE FILE.
 -END ØF FILE-
? SET;7
? L
THIS IS A SAMPLE FILE TØ DEMØNSTRATE THE ADD CØMMAND.
? DELETE:/"SURPRIZED/                                          ←
? L;6
THIS IS A SAMPLE FILE TØ DEMØNSTRATE THE ADD CØMMAND.
YØU MAY BE SURPRIZEDTHØRØUGHLY AT WHERE THIS LINE ØF TEXT IS INSERTED.
THIS TIME THE INSERTIØN SHØULD WØRK.SHØULD FINALLY WØRK PRØPERLY.
WATCH CAREFULLY WHERE THIS LINE ØF TEXT IS INSERTED.
THIS TIME THE INSERTIØN SHØULD WØRK.                           ←
WATCH CAREFULLY WHERE THIS LINE ØF TEXT IS INSERTED.
? D:/SIMPLE ADD/,/PARAMETER/                                   ←
? SET;4
? L
THIS TIME THE INSERTIØN SHØULD WØRK.
? BLANK                                                        ←
? SET;-4
? DELETES:/THØRØUGHLY/
? L;*
THIS IS A SAMPLE FILE TØ DEMØNSTRATE THE ADD CØMMAND.
YØU MAY BE SURPRIZED AT WHERE THIS LINE ØF TEXT IS INSERTED.
THIS TIME THE INSERTIØN SHØULD WØRK.SHØULD FINALLY WØRK PRØPERLY.
WATCH CAREFULLY WHERE THIS LINE ØF TEXT IS INSERTED.

WATCH CAREFULLY WHERE THIS LINE ØF TEXT IS INSERTED.
THIS LINE ØF TEXT IS INSERTED AFTER THE SECØND EXISTING LINE.
THIS IS AT THE END ØF THE FILE.
 -END ØF FILE-
```

Figure 4-3. Examples of DELETE and BLANK Usage

```
? DELETE:/WATCH/;2                                        ←
? BLANKS:/WØRK.SHØULD/                                    ←
? DELETES:/INSERTED AFT/,/IS/                             ←
? L;*
THIS IS A SAMPLE FILE TØ DEMØNSTRATE THE ADD CØMMAND.
YØU MAY BE SURPRIZED AT WHERE THIS LINE ØF TEXT IS INSERTED.
THIS TIME THE INSERTIØN SHØULD              FINALLY WØRK PRØPERLY.

THIS LINE ØF TEXT IS TING LINE.
THIS IS AT THE END ØF THE FILE.
 -END ØF FILE-
? DS:/TING L/,/THIS IS/                                   ←
? L;*
THIS IS A SAMPLE FILE TØ DEMØNSTRATE THE ADD CØMMAND.
YØU MAY BE SURPRIZED AT WHERE THIS LINE ØF TEXT IS INSERTED.
THIS TIME THE INSERTIØN SHØULD              FINALLY WØRK PRØPERLY. ←

THIS LINE ØF TEXT IS
 AT THE END ØF THE FILE.
 -END ØF FILE-


? DS:/TØ DEM/,/CØMMAND/                                   ←
? RESET
? L;*
THIS DEMØNSTRATES VARIØUS FØRMS ØF THE DELETE AND BLANK
CØMMANDS. THESE CØMMANDS ARE VERY SIMILAR, EXCEPT THAT
THE DELETE ADJUSTS ØR RELØCATES THE TEXT,WHEREAS THE BLANK
LEAVES ØPEN SPACE IN THE AREAS FRØM WHICH TEXT WAS REMØVED.
THE INFØRMATIØN BELØW IS THE MATERIAL ØN WHICH WE WILL
BE WØRKING PRIMARILY.

THIS IS A SAMPLE FILE .
YØU MAY BE SURPRIZED AT WHERE THIS LINE ØF TEXT IS INSERTED.
THIS TIME THE INSERTIØN SHØULD              FINALLY WØRK PRØPERLY.

THIS LINE ØF TEXT IS
 AT THE END ØF THE FILE.
 -END ØF FILE-
```

Figure 4-3. Examples of DELETE and BLANK Usage (Cont'd)

SUBSTITUTION OF INFORMATION

The CHANGE and REPLACES commands each cause a specified set of text information to replace text already present in the edit file. The length of the new information is independent of the length of the replaced text.

CHANGE COMMAND

In effect, the CHANGE command combines a DELETE operation with an ADD operation. A complete CHANGE operation requires two sets of information, a definition of the area to be changed (which is supplied in the CHANGE command) and the information that is to be inserted into that area (which is supplied by the user in response to the ENTER TEXT request).

After the command is entered, the system types

```
  ENTER TEXT
?
```

Respond to this request in one of three ways.

1. Type the actual change information (including carriage returns and line numbers if required), bracketed with delimiters.
2. Type the dollar sign ($) character with no delimiters or other characters. This causes the current contents of the string buffer to be used as the change information. (Information is placed in the string buffer by one or more EXTRACT statements.)
3. Type CR only. This causes the data entered in response to the most recent previous ENTER TEXT request to be used as the change information.

LINE MODE FORMATS (CHANGE OR C)

Command	Explanation
CHANGE	Replaces the line specified by the search pointer with the text that follows
CHANGE;n	Replaces the first n lines of the edit file beginning at the search pointer
CHANGE:/string/	Replaces the line containing the specified string
CHANGE:/string/;n	Replaces the first n lines containing the string
CHANGE:/string1/, /string2/	Replaces the line or group of lines containing string1 and string2
CHANGE:/string1/, /string2/;n	Replaces the first n occurrences of the line or group of lines containing string1 and string2

STRING MODE FORMATS (CHANGES OR CS)

Command	Explanation
CHANGES	Same as CHANGE
CHANGES;n	Same as CHANGE;n
CHANGES:/string/	Replaces the specified string
CHANGES:/string/;n	Replaces the first n occurrences of the specified string
CHANGES:/string1/, /string2/	Replaces the string of characters specified by the ellipsis /string1/,/string2/
CHANGES:/string1/, /string2/;n	Replaces the first n occurrences of a string of characters specified by the ellipsis /string1/, /string2/

REPLACES COMMAND (REPLACES OR RS)

The REPLACES command is similar to the CHANGE command, except that it performs only string replacements and the replacement text is embedded in the command, thus speeding the interaction. Also, the structure of the REPLACES command does not allow ellipsis string specifications.

There are two valid formats.

Command	Explanation
REPLACES:/string/;n	Equivalent to DELETES:/string/;n
REPLACES:/string1/, /string2/;n	Each of n occurrences of string 1 is replaced with string2, beginning at the search pointer

```
? L;*
     THIS FILE SHALL BE USED FØR DEMINSSRAITING THE CHANGE AND
REPLACES STATEMENTS.  TØ INSURE THAT WE HAVE PLENTY ØF THINGS
TØ CHANGE, WE WILL CØNSTRUCT IT Ø INITAILY CØNTAIN ANUMBER ØF
TYPØ AND GRAMATICICAL ERRØRS, LIKE YØU CAN PLANELY SEE WITH
INCREDIBLY LITTLE EFFØRT.

 THE CHANGE STATEMENT IS QUITE FLEXIBLE, IN THAT IT CØMBINES
THE ADD AND DELETE FUNCTIØNS DESCRIBED EARLIER IN THIS MANUAL.
INFACT, ANY ØPERATIØN THAT CAN BE DØNE BY AN ADD, DELETE, ØR
BLANK STATEMENT CAN ALSØ BE DØNE BY A CHANGE STATEMENT, ALTHØUGH
IT MAY TAKE SØME EXTRA EFFØRT.
THE MØST ØBVIØUS, AND MØST CØMMØN USE ØF THE CHANGE
STATEMENT IS TØ CØRRECT STRING ERRØRS ØN A ØNE-ERRØR ØNE-
STATEMENT BASIS, SØ WE'LL DEMØNSTRAIT A FEW ØF THØSE FIRST.
AFTER THAT, WE'LL TRY SØME LINE-ØRIENTED AND SØME MULTIPLE
MANIPULATIØNS.
 -END ØF FILE-
? CHANGES:/ THE CHANGE ST/    $WHAT HAPPENS IF ST ØMITTED?   ←
 ENTER TEXT.
? /     THE CHANGE ST/
 READY.
? CS:/THE MØST ØBV/                                          ←
 ENTER TEXT.
? 9     THE MØST ØBV9
 READY.
? REPLACES:/INFACT/,/IN FACT/                                ←
? CHANGES:/INSURE/,7CHANGE,7
 ENTER TEXT.
? /ENSURE ØURSELVES ØF AMPLE CHANGE                          ←
? PØSSIBILITIES,/
 READY.
? LIST;*
     THIS FILE SHALL BE USED FØR DEMINSSRAITING THE CHANGE AND
REPLACES STATEMENTS.  TØ ENSURE ØURSELVES ØF AMPLE CHANGE
PØSSIBILITIES, WE WILL CØNSTRUCT IT Ø INITAILY CØNTAIN ANUMBER ØF
TYPØ AND GRAMATICICAL ERRØRS, LIKE YØU CAN PLANELY SEE WITH
INCREDIBLY LITTLE EFFØRT.

     THE CHANGE STATEMENT IS QUITE FLEXIBLE, IN THAT IT CØMBINES
THE ADD AND DELETE FUNCTIØNS DESCRIBED EARLIER IN THIS MANUAL.
IN FACT, ANY ØPERATIØN THAT CAN BE DØNE BY AN ADD, DELETE, ØR
BLANK STATEMENT CAN ALSØ BE DØNE BY A CHANGE STATEMENT, ALTHØUGH
IT MAY TAKE SØME EXTRA EFFØRT.
     THE MØST ØBVIØUS, AND MØST CØMMØN USE ØF THE CHANGE
STATEMENT IS TØ CØRRECT STRING ERRØRS ØN A ØNE-ERRØR ØNE-
STATEMENT BASIS, SØ WE'LL DEMØNSTRAIT A FEW ØF THØSE FIRST.
AFTER THAT, WE'LL TRY SØME LINE-ØRIENTED AND SØME MULTIPLE
MANIPULATIØNS.
 -END ØF FILE-
? CHANGES:/DEM/,/RAIT/;*                                     ←
 ENTER TEXT.
? 8DEMØNSTRATE8
 READY.
       2  ØCCURANCES ØF PHRASE FØUND.                        ←
```

Figure 4-4. Examples of CHANGE and REPLACES Usage

```
? L;5
     THIS FILE SHALL BE USED FØR DEMØNSTRATEING THE CHANGE AND
REPLACES STATEMENTS.  TØ ENSURE ØURSELVES ØF AMPLE CHANGE
PØSSIBILITIES, IT WAS FIRST CØNSTRUCTED WITH SEVERAL ERRØRS
TYPØ AND GRAMATICICAL ERRØRS, LIKE YØU CAN PLANELY SEE WITH
INCREDIBLY LITTLE EFFØRT.
? SET;1
? RS:/CHANGE/,/DEMØNSTRATIØNAL/
? LIST
REPLACES STATEMENTS.  TØ ENSURE ØURSELVES ØF AMPLE DEMØNSTRATIØNAL
? CHANGE:8CØNSTRUCT8
 ENTER TEXT.
? /PØSSIBILITIES, THE FILE WAS CØNSTRUCTED FIRST WITH SEVERAL/
 READY.
? CHANGE:/TYPØ/,/EFFØRT/
 ENTER TEXT.
? /SPELLING, GRAMMATICAL, AND TYPØGRAPHICAL ERRØRS AND LATER
? CØRRECTED BY MEANS ØF A VARIETY ØF CHANGE AND REPLACES
? STATEMENTS./
 READY.
? RESET
? LIST;*
     THIS FILE SHALL BE USED FØR DEMØNSTRATEING THE CHANGE AND
REPLACES STATEMENTS.  TØ ENSURE ØURSELVES ØF AMPLE DEMØNSTRATIØNAL
PØSSIBILITIES, THE FILE WAS CØNSTRUCTED FIRST WITH SEVERAL
SPELLING, GRAMMATICAL, AND TYPØGRAPHICAL ERRØRS AND LATER
CØRRECTED BY MEANS ØF A VARIETY ØF CHANGE AND REPLACES
STATEMENTS.

     THE CHANGE STATEMENT IS QUITE FLEXIBLE, IN THAT IT CØMBINES
THE ADD AND DELETE FUNCTIØNS DESCRIBED EARLIER IN THIS MANUAL.
IN FACT, ANY ØPERATIØN THAT CAN BE DØNE BY AN ADD, DELETE, ØR
BLANK STATEMENT CAN ALSØ BE DØNE BY A CHANGE STATEMENT, ALTHØUGH
IT MAY TAKE SØME EXTRA EFFØRT.
     THE MØST ØBVIØUS, AND MØST CØMMØN USE ØF THE CHANGE
STATEMENT IS TØ CØRRECT STRING ERRØRS ØN A ØNE-ERRØR ØNE-
STATEMENT BASIS, SØ WE'LL DEMØNSTRATE A FEW ØF THØSE FIRST.
AFTER THAT, WE'LL TRY SØME LINE-ØRIENTED AND SØME MULTIPLE
MANIPULATIØNS.
 -END ØF FILE-
? REPLACES:/ATEING/,/ATING/
? REPLACES:/STATEMENT/,/CØMMAND/;*
       7  ØCCURANCES ØF PHRASE FØUND.
? CHANGES:/EFFØRT./,/THE/   $BLANK IN SPACE LINE, ØR WØN'T WØRK
 ENTER TEXT.
? /EFFØRT.
?
?      THE/
 READY.
```

Figure 4-4. Examples of CHANGE and REPLACES Usage (Cont'd)

```
? LIST;*
     THIS FILE SHALL BE USED FØR DEMØNSTRATING THE CHANGE AND
REPLACES CØMMANDS.  TØ ENSURE ØURSELVES ØF AMPLE DEMØNSTRATIØNAL
PØSSIBILITIES, THE FILE WAS CØNSTRUCTED FIRST WITH SEVERAL
SPELLING, GRAMMATICAL, AND TYPØGRAPHICAL ERRØRS AND LATER
CØRRECTED BY MEANS ØF A VARIETY ØF CHANGE AND REPLACES
CØMMANDS.
     THE CHANGE CØMMAND IS QUITE FLEXIBLE, IN THAT IT CØMBINES
THE ADD AND DELETE FUNCTIØNS DESCRIBED EARLIER IN THIS MANUAL.
IN FACT, ANY ØPERATIØN THAT CAN BE DØNE BY AN ADD, DELETE, ØR
BLANK CØMMAND CAN ALSØ BE DØNE BY A CHANGE CØMMAND, ALTHØUGH
IT MAY TAKE SØME EXTRA EFFØRT.
     THE MØST ØBVIØUS, AND MØST CØMMØN USE ØF THE CHANGE
CØMMAND IS TØ CØRRECT STRING ERRØRS ØN A ØNE-ERRØR ØNE-
CØMMAND BASIS, SØ WE'LL DEMØNSTRATE A FEW ØF THØSE FIRST.
AFTER THAT, WE'LL TRY SØME LINE-ØRIENTED AND SØME MULTIPLE
MANIPULATIØNS.
 -END ØF FILE-
? END
 END TEXT EDITING.
```

Figure 4-4. Examples of CHANGE and REPLACES Usage (Cont'd)

LOADING THE STRING BUFFER

The EXTRACT command copies information from the edit file into the string buffer; it does not affect the contents of the edit file in any way. The CLEAR command restores the string buffer to an empty condition.

LINE MODE FORMATS (EXTRACT OR E)

Command	Explanation
EXTRACT	Moves one line beginning at the search pointer
EXTRACT;n	Moves n lines beginning at the search pointer. (If n equals *, all lines to the end of the edit file are moved.)
EXTRACT:/string/	Moves the first line containing the string
EXTRACT:/string/;n	Moves the nth line containing the string
EXTRACT:/string1/,/string2/	Moves the first line or group of lines containing string1 and string2
EXTRACT:/string1/,/string2/;n	Moves the nth occurrence of the line or group of lines containing string1 and string2

STRING MODE FORMATS (EXTRACTS OR ES)

Command	Explanation
EXTRACTS	Same as EXTRACT
EXTRACTS;n	Same as EXTRACT;n
EXTRACTS:/string/	Moves the string specified
EXTRACTS:/string/;n	Moves the nth occurrence of the specified string
EXTRACTS:/string1/,/string2/	Moves the string of characters specified by the ellipsis /string1/,/string2/
EXTRACTS:/string1/,/string2/;n	Moves the nth string of characters specified by the ellipsis /string1/,/string2/

CLEAR STRING BUFFER (CLEAR OR CL)

The string buffer is not cleared automatically after an ADD or CHANGE command. It is the user's responsibility to clear the string buffer; if he does not do so, information from subsequent EXTRACT operations is appended to the information from previous EXTRACT operations.

The format is

```
CLEAR
```

Operand fields are never used with this command.

```
EDIT

 BEGIN TEXT EDITING.
? L;*
THIS FILE DEMØNSTRATES THE EXTRACT CØMMAND.  TØ GIVE
US SØME MATERIAL WITH WHICH TØ WØRK, SØME NØNSENSICAL
BUT ILLUSTRATIVE TEXT IS SHØWN BELØW.
THIS IS LINE 1
THIS IS LINE 2
THIS IS LINE 3, CLAUSE 1, CLAUSE 2.  LINE 3, SENTENCE 2.
THIS IS LINE 4 ØF THE TEST FILE
THIS IS LINE 5.  THIS FILE IS FØR DEMØNSTRATIØN PURPØSES AND
LINE 6, AS WITH ALL LINES CØNTAINS LITTLE THAT MAKES SENSE.
 -END ØF FILE-
? SET;3                                                         ←——
? L
THIS IS LINE 1
? EXTRACT;2                                                     ←——
? ADD;*
 ENTER TEXT.
? $                                                             ←——
? EXTRACT:7DEMØNST7
? ADD
 ENTER TEXT.
? $
? L;*
THIS IS LINE 1
THIS IS LINE 1                                                  ←——
THIS IS LINE 2
THIS IS LINE 5.  THIS FILE IS FØR DEMØNSTRATIØN PURPØSES AND
THIS IS LINE 2
THIS IS LINE 3, CLAUSE 1, CLAUSE 2.  LINE 3, SENTENCE 2.
THIS IS LINE 4 ØF THE TEST FILE
THIS IS LINE 5.  THIS FILE IS FØR DEMØNSTRATIØN PURPØSES AND
LINE 6, AS WITH ALL LINES CØNTAINS LITTLE THAT MAKES SENSE.
THIS IS LINE 1                                                  ←——
THIS IS LINE 2
 -END ØF FILE-
? CLEAR                                                         ←——
? EXTRACTS:/ ØF THE TEST FILE/                                  ←——
? ADDS:/LINE 1/;*
 ENTER TEXT.
? $

     3  ØCCURANCES ØF PHRASE FØUND.
? RESET
? CLEAR
```

Figure 4-5. Examples of EXTRACT and CLEAR Usage

```
? LIST;*
THIS FILE DEMØNSTRATES THE EXTRACT CØMMAND.  TØ GIVE
US SØME MATERIAL WITH WHICH TØ WØRK, SØME NØNSENSICAL
BUT ILLUSTRATIVE TEXT IS SHØWN BELØW.
THIS IS LINE 1 ØF THE TEST FILE                      ←
THIS IS LINE 1 ØF THE TEST FILE
THIS IS LINE 2
THIS IS LINE 5.  THIS FILE IS FØR DEMØNSTRATIØN PURPØSES AND
THIS IS LINE 2
THIS IS LINE 3, CLAUSE 1, CLAUSE 2.  LINE 3, SENTENCE 2.
THIS IS LINE 4 ØF THE TEST FILE
THIS IS LINE 5.  THIS FILE IS FØR DEMØNSTRATIØN PURPØSES AND
LINE 6, AS WITH ALL LINES CØNTAINS LITTLE THAT MAKES SENSE.
THIS IS LINE 1 ØF THE TEST FILE
THIS IS LINE 2
 -END ØF FILE-
? EXTRACTS:/TØ GIVE/,8IS SHØWN BELØW.8               ←
? ADD;*
 ENTER TEXT.
? $
? L;*
THIS FILE DEMØNSTRATES THE EXTRACT CØMMAND.  TØ GIVE
US SØME MATERIAL WITH WHICH TØ WØRK, SØME NØNSENSICAL
BUT ILLUSTRATIVE TEXT IS SHØWN BELØW.
THIS IS LINE 1 ØF THE TEST FILE
THIS IS LINE 1 ØF THE TEST FILE
THIS IS LINE 2
THIS IS LINE 5.  THIS FILE IS FØR DEMØNSTRATIØN PURPØSES AND
THIS IS LINE 2
THIS IS LINE 3, CLAUSE 1, CLAUSE 2.  LINE 3, SENTENCE 2.
THIS IS LINE 4 ØF THE TEST FILE
THIS IS LINE 5.  THIS FILE IS FØR DEMØNSTRATIØN PURPØSES AND
LINE 6, AS WITH ALL LINES CØNTAINS LITTLE THAT MAKES SENSE.
THIS IS LINE 1 ØF THE TEST FILE
THIS IS LINE 2
TØ GIVE                                              ←
US SØME MATERIAL WITH WHICH TØ WØRK, SØME NØNSENSICAL
BUT ILLUSTRATIVE TEXT IS SHØWN BELØW.
 -END ØF FILE-
? END
 END TEXT EDITING.
```

Figure 4-5. Examples of EXTRACT and CLEAR Usage (Cont'd)

EDIT FILE DIMENSIONING COMMANDS

The LENGTH and WIDTH commands are used to respecify the dimensions of the edit file. The ALIGN command removes extraneous blanks for printing purposes.

LENGTH COMMAND (LENGTH)

The LENGTH command limits the number of lines of the edit file on which other edit commands can operate and also resets the search pointer to the first line. Multiple truncations are allowed to a maximum of eight.

The following are valid forms of the command.

Command	Explanation
LENGTH;n	Truncates the edit file at line n. All text information beyond line n is saved in a scratch file SCR3. Information in SCR3 is not affected by editing commands.
LENGTH;*	Restores original processing boundaries of the edit file by appending the contents of scratch file SCR3 to the edit file. This version of the command is meaningful only if a LENGTH;n command has been given previously.

WIDTH COMMAND (WIDTH OR W)

The WIDTH command defines the maximum number of character columns that can be contained in a single line of the edit file.

The format is

```
WIDTH;n
```

where n is the new line length, to a maximum of 136 characters.

Following a WIDTH command, the ALIGN command can be used to remove superfluous blanks and reformat in accordance with the changed right margin.

ALIGN COMMAND (ALIGN OR AL)

The ALIGN command eliminates extraneous blanks from the edit file, while retaining the structural integrity of words, sentences, and paragraphs.

A word is defined as a set of characters between spaces. A sentence is defined as a group of words ending with a period (or question mark). The beginning of a paragraph is defined by an indented sentence.

The following are valid forms of this format control command.

Command	Explanation
ALIGN	Removes excess blanks between words in the line of text specified by the search pointer
ALIGN;n	Removes excess blanks between words in n lines of text beginning at the search pointer. As many complete words as possible are placed in a line before starting another line.
ALIGN:/string/	Removes blanks from the line of text containing the specified string
ALIGN:/string/;n	Removes blanks from the first n lines containing the specified string
ALIGN:/string1/, /string2/	Removes blanks from the lines of text specified by string1 and string2
ALIGN:/string1/, /string2/;n	Removes blanks from the first n occurrences of the line or group of lines specified by string1 and string2

```
EDIT
 BEGIN TEXT EDITING.
? LIST;*
THIS IS THE FIRST SENTENCE.
THIS IS SENTENCE 2.
THIS IS SENTENCE3.
THIS IS SENTENCE4.
THIS IS SENTENCE5.
THIS IS SENTENCE6.                                           ←
THIS IS THE EIGHTH SENTENCE.
THIS SENTENCE IS THE NINTH.
     THIS SENTENCE BEGINS A NEW PARAGRAPH.  IT IS,
LIKE THE REST ØF THE FILE, CØNSTRUCTED FØR THE PURPØSE
ØF DEMØNSTRATING CERTAIN TEXT EDITØR CØMMANDS.
     THE REMAINING PART ØF THIS FILE IS A MERGED             ←
CØPY ØF A NØNSENSE FILE THAT YØU HAVE SEEN
ELSEWHERE IN THE MANUAL.
     THIS IS THE FIRST SENTENCE ØF A PARAGRAPH.  THIS IS THE
SECØND SENTENCE ØF PARAGRAPH ØNE.  SENTENCE THREE IS HERE, AND
THIS CLAUSE IS PART ØF SENTENCE THREE.  THIS ENTIRE FILE MAKES
LITTLE SENSE, BUT IS CØNSTRUCTED FØR THE PURPØSE ØF
DEMØNSTRATING TEXT EDITØR CØMMANDS.
     THIS SENTENCE BEGINS THE SECØND PARAGRAPH ØF THE FILE.
THIS SENTENCE, ØN THE ØTHER HAND, ENDS IT.
     THE THIRD PARAGRAPH ØF THIS EXCEEDINGLY NØNSENSICAL
TEXT SEQUENCE HEADS UP A LIST ØF EQUALLY NØNSENSICAL ITEMS:
          1.  THIS IS ITEM ØNE
          2.  THIS ITEM IS THE SECØND
          3.  AND THIS IS THE THIRD AND LAST ITEM            ←
THIS SENTENCE CØNTINUES THE PARAGRAPH FØLLØWING THE LIST, UNLESS,
ØF CØURSE, WE DECIDE TØ REMØVE IT ENTIRELY ØR IN PART WITH
TEXT EDITING CØMMANDS.
     AND HERE, THANKFULLY, IS THE FØURTH AND FINAL PARAGRAPH,
ALTHØUGH NØT NECESSARILY THE FINAL SENTENCE.  THIS SAMPLE
FILE WILL APPEAR IN VARIØUS PARTS ØF THE TEXT EDITØR MANUAL
TØ ILLUSTRATE THE ACTIØN ØF VARIØUS EDITING CØMMANDS.
 -END ØF FILE-
? LENGTH;19                                                  ←
? SET:/MERGED/
? LIST;*
     THE REMAINING PART ØF THIS FILE IS A MERGED
CØPY ØF A NØNSENSE FILE THAT YØU HAVE SEEN
ELSEWHERE IN THE MANUAL.
     THIS IS THE FIRST SENTENCE ØF A PARAGRAPH.  THIS IS THE
SECØND SENTENCE ØF PARAGRAPH ØNE.  SENTENCE THREE IS HERE, AND
THIS CLAUSE IS PART ØF SENTENCE THREE.  THIS ENTIRE FILE MAKES
LITTLE SENSE, BUT IS CØNSTRUCTED FØR THE PURPØSE ØF
DEMØNSTRATING TEXT EDITØR CØMMANDS.                          ←
 -END ØF FILE-
```

Figure 4-6. Examples of LENGTH, WIDTH, and ALIGN Usage

```
? LENGTH;6
? ADD;*
 ENTER TEXT.
? 8THIS IS THE SEVENTH SENTENCE IN THE FILE.8
 READY.
? SET:/CE3/
? CHANGES:/ENCE/;4
 ENTER TEXT.
? /ENCE /
 READY.
? LIST;*
THIS IS SENTENCE 3.
THIS IS SENTENCE 4.
THIS IS SENTENCE 5.
THIS IS SENTENCE 6.
THIS IS THE SEVENTH SENTENCE IN THE FILE.
 -END ØF FILE-
? LENGTH;*
? LIST;*
THIS IS THE FIRST SENTENCE.
THIS IS SENTENCE 2.
THIS IS SENTENCE 3.
THIS IS SENTENCE 4.
THIS IS SENTENCE 5.
THIS IS SENTENCE 6.
THIS IS THE SEVENTH SENTENCE IN THE FILE.
THIS IS THE EIGHTH SENTENCE.
THIS SENTENCE IS THE NINTH.
      THIS SENTENCE BEGINS A NEW PARAGRAPH.  IT IS,
LIKE THE REST ØF THE FILE, CØNSTRUCTED FØR THE PURPØSE
ØF DEMØNSTRATING CERTAIN TEXT EDITØR CØMMANDS.
      THE REMAINING PART ØF THIS FILE IS A MERGED
CØPY ØF A NØNSENSE FILE THAT YØU HAVE SEEN
ELSEWHERE IN THE MANUAL.
      THIS IS THE FIRST SENTENCF ØF A PARAGRAPH.  THIS IS THE
SECØND SENTENCE ØF PARAGRAPH ØNE.  SENTENCE THREE IS HERE, AND
THIS CLAUSE IS PART ØF SENTENCE THREE.  THIS ENTIRE FILE MAKES
LITTLE SENSE, BUT IS CØNSTRUCTED FØR THE PURPØSE ØF
DEMØNSTRATING TEXT EDITØR CØMMANDS.
      THIS SENTENCE BEGINS THE SECØND PARAGRAPH ØF THE FILE.
THIS SENTENCE, ØN THE ØTHER HAND, ENDS IT.
      THE THIRD PARAGRAPH ØF THIS EXCEEDINGLY NØNSENSICAL
TEXT SEQUENCE HEADS UP A LIST ØF EQUALLY NØNSENSICAL ITEMS:
            1.  THIS IS ITEM ØNE
            2.  THIS ITEM IS THE SECØND
            3.  AND THIS IS THE THIRD AND LAST ITEM
THIS SENTENCE CØNTINUES THE PARAGRAPH FØLLØWING THE LIST, UNLESS,
ØF CØURSE, WE DECIDE TØ REMØVE IT ENTIRELY ØR IN PART WITH
TEXT EDITING CØMMANDS.
      AND HERE, THANKFULLY, IS THE FØURTH AND FINAL PARAGRAPH,
ALTHØUGH NØT NECESSARILY THE FINAL SENTENCE.  THIS SAMPLE
FILE WILL APPEAR IN VARIØUS PARTS ØF THE TEXT EDITØR MANUAL
TØ ILLUSTRATE THE ACTIØN ØF VARIØUS EDITING CØMMANDS.
 -END ØF FILE-
```

Figure 4-6. Examples of LENGTH, WIDTH, and ALIGN Usage (Cont'd)

```
? CS:/THIS/
 ENTER TEXT.
? /     THIS/
 READY.
? ALIGN;15
? LIST;15
     THIS IS THE FIRST SENTENCE. THIS IS SENTENCE 2.  THIS IS SENTENCE
3. THIS IS SENTENCE 4. THIS IS SENTENCE 5. THIS IS SENTENCE 6. THIS IS
THE SEVENTH SENTENCE IN THE FILE. THIS IS THE EIGHTH SENTENCE. THIS
SENTENCE IS THE NINTH.
     THIS SENTENCE BEGINS A NEW PARAGRAPH.  IT IS, LIKE THE REST ØF THE
FILE, CØNSTRUCTED FØR THE PURPØSE ØF DEMØNSTRATING CERTAIN TEXT EDITØR
CØMMANDS.
     THE REMAINING PART ØF THIS FILE IS A MERGED CØPY ØF A NØNSENSE
FILE THAT YØU HAVE SEEN ELSEWHERE IN THE MANUAL.
     THIS IS THE FIRST SENTENCE ØF A PARAGRAPH.  THIS IS THE
SECØND SENTENCE ØF PARAGRAPH ØNE.  SENTENCE THREE IS HERE, AND
THIS CLAUSE IS PART ØF SENTENCE THREE.  THIS ENTIRE FILE MAKES
LITTLE SENSE, BUT IS CØNSTRUCTED FØR THE PURPØSE ØF
DEMØNSTRATING TEXT EDITØR CØMMANDS.
     THIS SENTENCE BEGINS THE SECØND PARAGRAPH ØF THE FILE.
? WIDTH;56
? ALIGN          $NØTICE THAT N IS ASSUMED AS 1 WHEN N MISSING
? LIST;3
     THIS IS THE FIRST SENTENCE.  THIS IS SENTENCE 2.
THIS IS SENTENCE
3. THIS IS SENTENCE 4. THIS IS SENTENCE 5. THIS IS SENTENCE 6. THIS IS
? ALIGN;*
```

Figure 4-6. Examples of LENGTH, WIDTH, and ALIGN Usage (Cont'd)

```
? LIST;*
     THIS IS THE FIRST SENTENCE.  THIS IS SENTENCE 2.
THIS IS SENTENCE 3.  THIS IS SENTENCE 4.  THIS IS
SENTENCE 5.  THIS IS SENTENCE 6.  THIS IS THE SEVENTH
SENTENCE IN THE FILE.  THIS IS THE EIGHTH SENTENCE.
THIS SENTENCE IS THE NINTH.
     THIS SENTENCE BEGINS A NEW PARAGRAPH.  IT IS, LIKE
THE REST ØF THE FILE, CØNSTRUCTED FØR THE PURPØSE ØF
DEMØNSTRATING CERTAIN TEXT EDITØR CØMMANDS.
     THE REMAINING PART ØF THIS FILE IS A MERGED CØPY
ØF A NØNSENSE FILE THAT YØU HAVE SEEN ELSEWHERE IN THE
MANUAL.
     THIS IS THE FIRST SENTENCE ØF A PARAGRAPH.  THIS
IS THE SECØND SENTENCE ØF PARAGRAPH ØNE.  SENTENCE
THREE IS HERE, AND THIS CLAUSE IS PART ØF SENTENCE
THREE.  THIS ENTIRE FILE MAKES LITTLE SENSE, BUT IS
CØNSTRUCTED FØR THE PURPØSE ØF DEMØNSTRATING TEXT
EDITØR CØMMANDS.
     THIS SENTENCE BEGINS THE SECØND PARAGRAPH ØF THE
FILE.  THIS SENTENCE, ØN THE ØTHER HAND, ENDS IT.
     THE THIRD PARAGRAPH ØF THIS EXCEEDINGLY
NØNSENSICAL TEXT SEQUENCE HEADS UP A LIST ØF EQUALLY
NØNSENSICAL ITEMS:
     1.  THIS IS ITEM ØNE
     2.  THIS ITEM IS THE SECØND
     3.  AND THIS IS THE THIRD AND LAST ITEM THIS
SENTENCE CØNTINUES THE PARAGRAPH FØLLØWING THE LIST,
UNLESS, ØF CØURSE, WE DECIDE TØ REMØVE IT ENTIRELY ØR
IN PART WITH TEXT EDITING CØMMANDS.
     AND HERE, THANKFULLY, IS THE FØURTH AND FINAL
PARAGRAPH, ALTHØUGH NØT NECESSARILY THE FINAL SENTENCE.
 THIS SAMPLE FILE WILL APPEAR IN VARIØUS PARTS ØF THE
TEXT EDITØR MANUAL TØ ILLUSTRATE THE ACTIØN ØF VARIØUS
EDITING CØMMANDS.
 -END ØF FILE-
? END
 END TEXT EDITING.
```

Figure 4-6. Examples of LENGTH, WIDTH, and ALIGN Usage (Cont'd)

TABULATION COMMANDS

The commands DEFTAB, TAB, and LISTAB allow the user to create structured text using tab settings.

DEFTAB COMMAND (DEFTAB OR DT)

The DEFTAB command defines a single tab character that is later used (when responding to an ENTER TEXT request) to cause blank fill to the next tab stop. The tab character must not be present in the body of text that is to be created. Each typing of the tab character that occurs when entering text is ignored, except for purposes of tab control.

The following are valid forms of the command.

Command	Explanation
DEFTAB	Clears previous tab character definition
DEFTAB:/tabchar/	Defines the character tabchar as a tab character

TAB COMMAND (TAB OR T)

The TAB command sets tab stops at specified print columns. Default column numbers are 11, 18, 30, 40, and 50.

The following are valid forms of the command.

Command	Explanation
TAB	Clears existing tab stops
TAB:/t1,...,tn/	Each t; is a column number. A maximum of seven tab column numbers may be specified

Only one TAB command can be active at one time. Entering a TAB command negates the effect of any prior TAB command.

LISTAB COMMAND (LISTAB OR LT)

The LISTAB command causes a listing of the tab stops as specified in the most recent TAB command.

```
? L;*
THE DEFTAB AND TAB CØMMANDS ARE EFFECTIVE ØNLY
IF GIVEN PRIØR TØ THE "ENTER TEXT" CØMMAND. THUS,
BECAUSE NØ TAB ØR DEFTAB CØMMAND IS ACTIVE AT THE
TIME THIS INFØRMATIØN IS BEING TYPED INTØ THE
FILE, THE FØLLØWING CANNØT BE TABULATED.
INTEGER N#SUM N INTEGERS#FACTØRIAL#SQUARE#CUBE
1#1#1#1#1
2#3#2#4#8
NØW WE WILL DEFINE A TAB CHARACTER AND A SET ØF STØPS.
 -END ØF FILE-
? DEFTAB:/#/
? TAB:/12,24,36,48/
? ADD;*
 ENTER TEXT.
? /INTEGER N#SUM N INT#FACTØRIAL#SQUARE#CUBE
? 1#1#1#1#1
? 2#3#2#4#8
? 3#6#6#9#27
? 4#10#24#16#64/
 READY.
? LIST;*
THE DEFTAB AND TAB CØMMANDS ARE EFFECTIVE ØNLY
IF GIVEN PRIØR TØ THE "ENTER TEXT" CØMMAND. THUS,
BECAUSE NØ TAB ØR DEFTAB CØMMAND IS ACTIVE AT THE
TIME THIS INFØRMATIØN IS BEING TYPED INTØ THE
FILE, THE FØLLØWING CANNØT BE TABULATED.
INTEGER N#SUM N INTEGERS#FACTØRIAL#SQUARE#CUBE
1#1#1#1#1
2#3#2#4#8
NØW WE WILL DEFINE A TAB CHARACTER AND A SET ØF STØPS.
INTEGER N   SUM N INT   FACTØRIAL   SQUARE      CUBE
1           1           1           1           1
2           3           2           4           8
3           6           6           9           27
4           10          24          16          64
 -END ØF FILE-
? LISTAB
 TAB STØPS    12   24   36   48
? DEFTAB
? LISTAB
 TAB STØPS    12   24   36   48
? TAB
? LISTAB
 TAB STØPS NØNE.
? END
 END TEXT EDITING.
```

Figure 4-7. Examples of TAB, DEFTAB, and LISTAB Usage

EXTERNAL FILE MERGE

MERGE COMMAND (MERGE OR M)

The MERGE command causes the contents of a specified file (working or permanent) to be merged into the edit file.

The following are valid forms of the command.

Command	Explanation
MERGE:/lfn/;n	The contents of file lfn are inserted into the edit file. Merging takes place after the nth line of the edit file, relative to the search pointer.
MERGE:/lfn/, /string/;n	The contents of file lfn are inserted into the edit file. Merging takes place after the nth line that contains /string/, and only if n lines containing /string/ are found.

MERGE is the only Text Editor command that can reference a working file other than the edit file. It is also the only Text Editor command that can reference a permanent file.

```
LNH,F=PARANØN
     THIS IS THE FIRST SENTENCE ØF A PARAGRAPH.  THIS IS THE
SECØND SENTENCE ØF PARAGRAPH ØNE.  SENTENCE THREE IS HERE, AND
THIS CLAUSE IS PART ØF SENTENCE THREE.  THIS ENTIRE FILE MAKES
LITTLE SENSE, BUT IS CØNSTRUCTED FØR THE PURPØSE ØF
DEMØNSTRATING TEXT EDITØR CØMMANDS.
READY.

LNH,F=INSFL
****THIS IS A VERY SHØRT FILE USED TØ
****DEMØNSTRATE THE MERGE CØMMAND.
READY.

NEW,MRGTST
READY.

EDIT
 BEGIN TEXT EDITING.
? ADD
 ENTER TEXT.
? /    THIS FILE IS BEING ØRIGINATED FØR THE PURPØSE
? ØF DEMØNSTRATING THE MERGE CØMMAND.  THE REST ØF THIS
? FILE WILL BE BUILT BY MEANS ØF MERGE CØMMANDS DIRECTED
? TØ THE FILES LISTED ABØVE, PARANØN AND INSFL; NØTE
? ALSØ THAT THESE ARE NØT WØRKING FILES, SINCE THEY WERE
? DRØPPED FØLLØWING THE NEW CØMMAND THAT ØRIGINATED THIS FILE./
READY.
? LIST;*
     THIS FILE IS BEING ØRIGINATED FØR THE PURPØSE
ØF DEMØNSTRATING THE MERGE CØMMAND.  THE REST ØF THIS
FILE WILL BE BUILT BY MEANS ØF MERGE CØMMANDS DIRECTED
TØ THE FILES LISTED ABØVE, PARANØN AND INSFL; NØTE
ALSØ THAT THESE ARE NØT WØRKING FILES, SINCE THEY WERE
DRØPPED FØLLØWING THE NEW CØMMAND THAT ØRIGINATED THIS FILE.
 -END ØF FILE-
? MERGE:7PARANØN7;*                                                ←
? MERGE:/INSFL/,/DEMØNSTR/                                         ←
? L;*
     THIS FILE IS BEING ØRIGINATED FØR THE PURPØSE
ØF DEMØNSTRATING THE MERGE CØMMAND.  THE REST ØF THIS
****THIS IS A VERY SHØRT FILE USED TØ                              ←
****DEMØNSTRATE THE MERGE CØMMAND.
FILE WILL BE BUILT BY MEANS ØF MERGE CØMMANDS DIRECTED
TØ THE FILES LISTED ABØVE, PARANØN AND INSFL; NØTE
ALSØ THAT THESE ARE NØT WØRKING FILES, SINCE THEY WERE
DRØPPED FØLLØWING THE NEW CØMMAND THAT ØRIGINATED THIS FILE.
     THIS IS THE FIRST SENTENCE ØF A PARAGRAPH.  THIS IS THE      ←
SECØND SENTENCE ØF PARAGRAPH ØNE.  SENTENCE THREE IS HERE, AND
THIS CLAUSE IS PART ØF SENTENCE THREE.  THIS ENTIRE FILE MAKES
LITTLE SENSE, BUT IS CØNSTRUCTED FØR THE PURPØSE ØF
DEMØNSTRATING TEXT EDITØR CØMMANDS.
 -END ØF FILE-
```

Figure 4-8. Examples of MERGE Usage

```
? MERGE:/INSFL/,/COMMANDS/;5
        2  ØCCURANCES ØF PHRASE FØUND.
? MERGE:/INSFL/,/COMMAND/;4
? LIST;*
     THIS FILE IS BEING ØRIGINATED FØR THE PURPØSE
ØF DEMØNSTRATING THE MERGE CØMMAND.  THE REST ØF THIS
****THIS IS A VERY SHØRT FILE USED TØ
****DEMØNSTRATE THE MERGE CØMMAND.
FILE WILL BE BUILT BY MEANS ØF MERGE CØMMANDS DIRECTED
TØ THE FILES LISTED ABØVE, PARANØN AND INSFL; NØTE
ALSØ THAT THESE ARE NØT WØRKING FILES, SINCE THEY WERE
DRØPPED FØLLØWING THE NEW CØMMAND THAT ØRIGINATED THIS FILE.
****THIS IS A VERY SHØRT FILE USED TØ
****DEMØNSTRATE THE MERGE CØMMAND.
     THIS IS THE FIRST SENTENCE ØF A PARAGRAPH.  THIS IS THE
SECØND SENTENCE ØF PARAGRAPH ØNE.  SENTENCE THREE IS HERE, AND
THIS CLAUSE IS PART ØF SENTENCE THREE.  THIS ENTIRE FILE MAKES
LITTLE SENSE, BUT IS CØNSTRUCTED FØR THE PURPØSE ØF
DEMØNSTRATING TEXT EDITØR CØMMANDS.
 -END ØF FILE-
? END
 END TEXT EDITING.
```

Figure 4-8. Examples of MERGE Usage (Cont'd)

STRING INCIDENCE COUNTING

NUMBER COMMAND

The NUMBER command provides a count of lines in a file or a count dependent on the presence of a specified string of characters. The count always begins relative to the search pointer.

LINE MODE FORMATS (NUMBER OR N)

Command	Explanation
NUMBER	Returns a line count from current search pointer value to end-of-file
NUMBER:/string/ or NUMBER:/string1/, /string2/	Returns a count of the number of lines in the edit file that each contain the entire specified string

STRING MODE FORMATS (NUMBERS OR NS)

Command	Explanation
NUMBERS	Same as NUMBER
NUMBERS:/string/ or NUMBERS:/string1/, /string2/	Returns a count of the number of occurrences of the specified string. Note that the string can be either single phrase or ellipsis.

```
EDIT
 BEGIN TEXT EDITING.
? L;*
     THIS IS THE FIRST SENTENCE.  THIS IS SENTENCE 2.
THIS IS SENTENCE 3.  THIS IS SENTENCE 4.  THIS IS
SENTENCE 5.  THIS IS SENTENCE 6.  THIS IS THE SEVENTH
SENTENCE IN THE FILE.  THIS IS THE EIGHTH SENTENCE.
THIS SENTENCE IS THE NINTH.
     THIS SENTENCE BEGINS A NEW PARAGRAPH.  IT IS, LIKE
THE REST ØF THE FILE, CØNSTRUCTED FØR THE PURPØSE ØF
DEMØNSTRATING CERTAIN TEXT EDITØR CØMMANDS.
     THE REMAINING PART ØF THIS FILE IS A MERGED CØPY
ØF A NØNSENSE FILE THAT YØU HAVE SEEN ELSEWHERE IN THE
MANUAL.
     THIS IS THE FIRST SENTENCE ØF A PARAGRAPH.  THIS
IS THE SECØND SENTENCE ØF PARAGRAPH ØNE.  SENTENCE
THREE IS HERE, AND THIS CLAUSE IS PART ØF SENTENCE
THREE.  THIS ENTIRE FILE MAKES LITTLE SENSE, BUT IS
CØNSTRUCTED FØR THE PURPØSE ØF DEMØNSTRATING TEXT
EDITØR CØMMANDS.
 -END ØF FILE-
? NUMBER
        17 LINES TØ EØF.
? NUMBER:/SENTENCE/
      9  ØCCURANCES ØF PHRASE FØUND.
? NUMBER:/THIS/,/SENT/
      8  ØCCURANCES ØF PHRASE FØUND.
? NUMBERS:8SENT8
     14  ØCCURANCES ØF PHRASE FØUND.
? NUMBERS:/SENTENCE /
     10  ØCCURANCES ØF PHRASE FØUND.
? NUMBERS:/THIS/,/SENT/
     13  ØCCURANCES ØF PHRASE FØUND.
? NUMBERS:/PARA/;*
      3  ØCCURANCES ØF PHRASE FØUND.
? NUMBERS:/PARA/
      3  ØCCURANCES ØF PHRASE FØUND.
? NUMBERS:/IS/
     31  ØCCURANCES ØF PHRASE FØUND.
? NUMBERS:/ IS/
     15  ØCCURANCES ØF PHRASE FØUND.
? NS:1THIS IS SENT1
      4  ØCCURANCES ØF PHRASE FØUND.
? LISTS:5THIS IS SENT5;*
                                   THIS IS SENT
THIS IS SENT         THIS IS SENT
             THIS IS SENT
 -END ØF FILE-
? END
 END TEXT EDITING.
READY.
```

Figure 4-9. Examples of NUMBER Usage

TERMINATING EDIT SESSION

END COMMAND (END)

The END command terminates text editing (that is, exits from EDIT program control) and returns control to the subsystem control language.

The command format is

```
    END
```

The system responds

```
    END TEXT EDITING
    RUN COMPLETE
```

It is necessary to terminate text editing whenever it is necessary or desirable to do a file operation (such as SAVE or REPLACE).

EDIT MESSAGES A

EDIT ERROR MESSAGES

These messages indicate a condition that prevents processing of the command.

PHRASE NOT FOUND

The search string specified in /string/ was not found in the edit file.

ILLEGAL COMMAND.

The command word is invalid.

cmd SYNTAX ERROR.

String and/or n parameter is illegal with command cmd.

ILLEGAL FILE NAME.

The file name passed with MERGE command is illegal.

MERGE ERROR, SECONDARY FILE EMPTY.

The file to be merged with edit file is empty.

RESERVED FILE NAME.

The file name passed with MERGE command or when invoking Text Editor is reserved for use by Text Editor. Reserved file names are:

INPUT
OUTPUT
SCR
SCR1
SCR2
SCR3
SCR4
SCR5

CONTROL CARD ERROR.

More than one parameter was passed when calling the Text Editor.

REQUESTS AND INFORMATIVE MESSAGES

These messages are issued in the course of normal EDIT operation.

BEGIN TEXT EDITING.

This message is issued when initialization of editor is complete and awaiting the first command.

ENTER TEXT FILE NAME

This message is issued when text file name is not passed with Text Editor call.

ENTER TEXT

New or replacement text is required to process ADD (ADDS) or CHANGE (CHANGES) commands.

m OCCURRENCES OF PHRASE FOUND.

End of file was encountered before number of iterations specified in n parameter were completed.

m LINES TO EOF

This message is a line count message issued by NUMBER command processor.

-END OF FILE-

The search pointer is currently set at end of file, or end of file encountered during execution of a LIST command.

END TEXT EDITING.

This message is issued following execution of the END command, indicating a return to subsystem mode.

READY.

The response to an ENTER TEXT request is completed; that is, the last carriage return was preceded immediately by the closing delimiter.

TAB STOPS t_1, t_2, ..., t_n

This message is a list of tab stops issued by LISTAB command processor.

FILE AT LINE NUMBER m

This message is a current search pointer value, issued by LINE command processor.

INDEX

CONTROL DATA
CORPORATION

CORPORATE HEADQUARTERS, 8100 34th AVE. SO., MINNEAPOLIS, MINN. 55440
SALES OFFICES AND SERVICE CENTERS IN MAJOR CITIES THROUGHOUT THE WORLD

LITHO IN U.S.A.