SH20-1965-0

Installed User Program

Display Editing System for CMS Users Guide

Program Number: 5796-PJP

This guide contains usage and reference information for Display Editing System for CMS, a full-screen editing system that operates under the Conversational Monitor System (CMS) component of IBM Virtual Machine Facility/370 (VM/370).

The Display Editing System Installed User Program is a full-screen editing system for the Virtual Machine Facility/370 Conversational Monitor System (VM/370 CMS) designed to utilize the features of both local and remote terminals. In addition to a comprehensive array of cursor and command oriented data manipulation functions, the IUP has direct usability in a wide range of user applications. Display terminals supported are the IBM 3277 Model 2 and the IBM 3278 Model 2, including both the TEXT PROCESSING and APL keyboards.

The Display Editing System accommodates CMS files of all types, including program source material, documentation, and data. It permits users to change portions of the display without program intervention. By placing the cursor on the position(s) requiring change, the user may re-enter the correct character(s). The Display Editing System checks for modifications made to the display image and makes permanent copies of the changes to the file.

IBM

PROGRAM SERVICES

Central Service will be provided until otherwise notified. Users will be given a minimum of six months notice prior to the discontinuance of Central Service.

During the Central Service period, IBM through the program sponsor(s) will, without additional charge, respond to an error in the current unaltered release of the program by issuing known error correction information to the customer reporting the problem and/or issuing corrected code or notice of availability of corrected code. However, IBM does not guarantee service results or represent or warrant that all errors will be corrected.

Any on-site program service or assistance will be provided at a charge.

WARRANTY

EACH LICENSED PROGRAM IS DISTRIBUTED ON AN 'AS IS' BASIS WITHOUT WARRANTY OF ANY KIND EITHER EXPRESS OR IMPLIED.

First Edition (September 1977)

A form for readers' comments has been provided at the back of this publication. If this form has been removed, address comments to: IBM Corporation, 1133 Westchester Avenue, White Plains, New York 10604. Attention: R. F. Walsh.

Contents

Illustrations

Use of this Manual

Although this is primarily a reference manual, it does provide an introduction to some of the concepts necessary for effective use of the editing system. New users of the DISPLAY EDITING SYSTEM for CMS should acquaint themselves with the overall system flow, a few of the more basic commands, and the facilities of the IBM 3270 Information Display System (see Operator's Guide, GA27-2742). It is important not to attempt to learn all the unusual commands until the need arises. Becoming aware of the various "types" of functions available and referring to the manual later for specific commands as they are needed would be the most effective approach to familiarization. The effective use of the "Advanced Editing Facilities" requires an understanding of the CMS console stack and the basic facilities of the DISPLAY EDITING SYSTEM for CMS.

I. Abstract

The DISPLAY EDITING SYSTEM for CMS, is a virtual storage, full-screen editing system for VM/CMS files, designed to use the full range of 3270 terminal features on either a local or remote basis. In addition to a comprehensive array of cursor and command oriented data manipulation features, the following points are of special interest:

- Editing records of varying lengths and formats.
- Nested editing levels.
- Multiple independent split screen control.
- Multi-level Change Data Set Facility.
- Minimum data transmission (changed data only).
- Minimum CPU activity (multiple functions per interrupt).
- Full-function execution of the editor from EXEC's.
- Invoking EXEC's from within edit sessions.
- User controlled data display - EBCDIC, HEX, or both.
- User defined Program Function Key functions.
- Simulation of 3270 operations.
- User written editing commands.
- User defined Command Synonyms.

II. Introduction

The DISPLAY EDITING SYSTEM FOR CMS is a general purpose data file editing system for CMS data files of all types, including program source material, documentation, etc. This document, for example, was originally produced on a display terminal using the DISPLAY EDITING SYSTEM FOR CMS.

The system comprises several components:

- EDGAR Edits Data Graphically And Recurrently. This component provides the basic editing system functions and is invoked by the CMS command EDGAR.

- $PROFILE is the filetype of a user modifiable set of specifications which establish the editing environment.

- Preprocessor EXEC's are automatically invoked, based on filetype, to establish default editing parameters.

- ECOMMAND is a CMS command which interfaces directly between EXEC's and the editing functions of EDGAR.

The display facilities of the DISPLAY EDITING SYSTEM were designed to permit maximum use of the display editing capabilities of an IBM 3270 INFORMATION DISPLAY SYSTEM. Some of the resulting features are as follows:

- The user can directly change any portion of the display to meet his requirements. These changes do not require program intervention. By placing the cursor on the position(s) requiring change, the user may re-enter the correct character(s). Upon depression of the "ENTER" key EDGAR checks for modifications made to the display image and makes corresponding changes within the working copy of the file.

- The data to be reviewed and/or edited is sometimes contained in more than one CMS file. EDGAR permits the user to divide his physical screen into logical screens each of which may display a different file for independent editing. Since it is often convenient to review nonconsecutive portions of a single file concurrently, EDGAR provides for division of a screen or logical screen into multiple displays, which can also be edited independently.

- Although the physical screen size of a display terminal is most readily adaptable to files of fixed-length, 80-character records, most users will frequently have need to edit files with other characteristics. EDGAR allows the editing of both large and small, fixed and variable length records, up to 2047 characters in length.

- It is often desirable to view data either in hexadecimal form or in column order other than that of the file items. The VIEW command provides these capabilities.

Various editing commands of the DISPLAY EDITING SYSTEM have been implemented in ways that reduce the number of operations required of the user and thus improve the overall efficiency of the editing system. For example:

- Repetitive editing operations may be predefined and invoked as user written EXECS or commands.

- The CHANGE command, which allows the user to change one character string within an item to another, also can be made to specify a horizontal "zone" within which the change is to occur, as well as both vertical and horizontal continuation counts. Thus, the change may be made on consecutive items or repeatedly within the same item.

- It is frequently necessary to apply a change made by cursor positioning on the display screen to several consecutive lines. The REPEAT command has been installed for this purpose.

- A facility called Screen Operation Simulation, SOS, has been included to allow the user to simulate the operation of the display terminal. Through the use of SOS commands and the CMS console stack, the user can preprogram the operation of the terminal to simplify repetitive or complex actions.

Above all, the DISPLAY EDITING SYSTEM is flexible and has been designed to allow the user to tailor his editing environment. Several components are included in the system to facilitate this. For example:

- Program Function Keys may be assigned by the user to perform any function, or combination of functions, desired.

- Optionally, at edit level initiation, a "default" EXEC can be automatically invoked to establish a user - specified editing environment.

- An SVC interface to the EDGAR command processor allows the user to write his own editing commands and invoke them during an active edit session.

III. System Overview

System Flow

As shown in figure 1, the DISPLAY EDITING SYSTEM editor, EDGAR, is comprised of three major components:

I/O Processor
Display Processor
Editing Command Processor

The I/O Processor

The I/O Processor maintains a storage copy of an I/O virtual screen. The I/O virtual screen has the same physical characteristics as the user's 3270 (if any) and may be updated as a result of activity on either of two sources; the physical display device, or the Screen Operation Simulation (SOS) facility. (The operation of SOS is described in Section V, Advanced Editing Facilities). The operation of the I/O Processor is as follows:

1. For each edit level currently selected for display, map the corresponding edit level virtual screen to the appropriate areas of the I/O virtual screen.

2. Allow the SOS facility to update the I/O virtual screen.

3. If ENTER depressed by SOS, go to step 6.

4. Replicate the contents of the I/O virtual screen on the physical display device and wait for action by the terminal user. When user depresses a PF key or ENTER, reflect changes made on the physical display device to the I/O virtual screen.

5. If ENTER not depressed by user, return to step 2.

6. Reflect changes made to the I/O virtual screen back to the edit level virtual screen(s).

7. Pass control to the Display Processor once for each currently displayed edit level virtual screen. Process the edit level virtual screens in the order, top down, that they appeared on the I/O virtual screen. Then return to Step 1.

The Display Processor

Upon receiving control from the I/O Processor the Display Processor analyzes the edit level virtual screen. The data and command areas are passed through the CMS input translate table. For information on the use of the CMS input/output translate tables, see IBM Virtual Machine Facility/370: CMS Command and Macro Reference GC20-1818. The data file records are, then, updated according to the changes in the data areas, and the commands are passed, one at a time, to the Editing Command Processor. Each command is terminated by either a X'01' or the end of the command area on the edit level virtual screen. Thus, through the use of the CMS input translate table, a user may assign any character to be a logical command separator. Also, since all data is passed through the CMS input translate table a user may enter any character from the keyboard (or via SOS) and have it appear as a different character in the data file.

After processing the edit level virtual screen, the Display Processor builds a new display based on the updated information. The data, starting with the "current line", is taken from the file, passed through the CMS output translate table, and placed in the data areas of the screen. Any messages that were received from the Edit Command Processor are placed on the command line. Control is then returned to the I/O Processor.

Editing Command Processor

The Editing Command Processor receives control from either the Display Processor or, via an SVC, from a user program. (The SVC interface is explained in Section V, Advanced Editing Facilities). In either case, the command is analyzed, processed if a recognized editing command, passed to the CMS SUBSET if not, and control is returned to the caller.

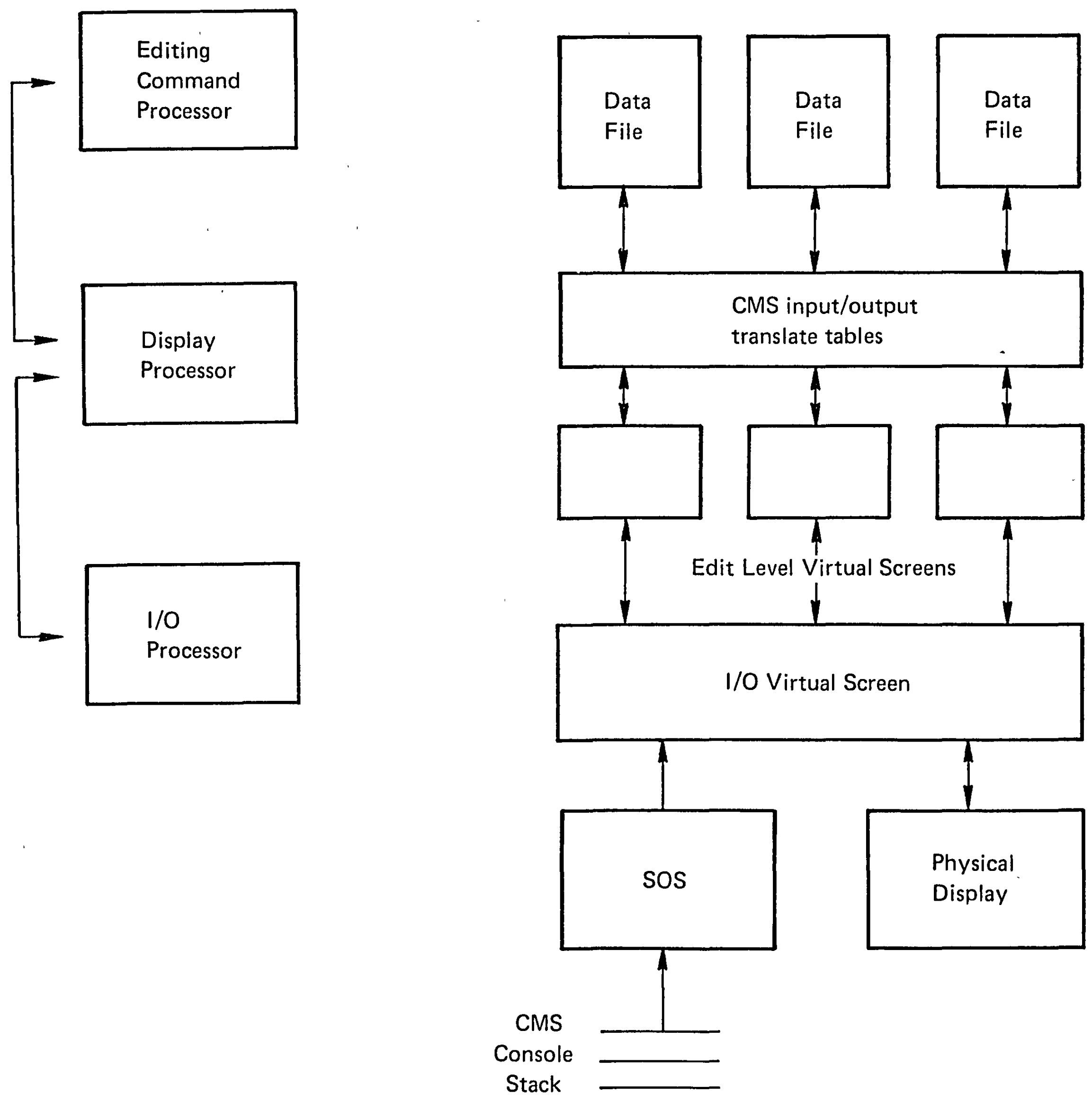


Figure 1. Basic system flow

System Initialization and Termination

The DISPLAY EDITING SYSTEM for CMS runs as a command under CMS and is invoked by entering the command:

```
EDGAR Fn Ft      <Fm>  <Lrecl nn>
                       <Seq8 | NOSeq8>
                       <Ctl xxxxxxxx | NOCtl>
                       <Update | NOUpdate>
                       <DEFault yyyyyyyy | NODEF>
                       <CLEAR | NOCLEAR>
```

(For an explanation of these operands and options refer to the ENTER command in Section IV of this manual). Upon initialization, and prior to reading the first edit level file, EDGAR reads a control file named 'EDGAR $PROFILE' and, based on the information contained therein, establishes the overall editing system environment. A standard version of this file is maintained on the system disk. By changing the information in 'EDGAR $PROFILE' the user can tailor the editing system to his needs.

'EDGAR $PROFILE' may be either a fixed or variable format file with a maximum record length of 130 bytes. It may contain six (6) different types of control statements. The first positional parameter in each control statement indicates the record type. The valid control statement types are as follows:

FILE — This is the default if no other valid control type is found as the first parameter. FILE allows the user to specify default settings, by filetype, for the operands of the ENTER (or EDGAR) command. The defaults established by this control statement may be overridden when the command, itself, is entered.

```
<FILE>  filetype <filemode>   <Lrecl nn>
                              <Default xxxxxxxx>
                              <Update>
                              <Ctl  nnnnnnnn>
                              <Seq8>
                              <Noclear>
                              <Synonym yyyyyyyy <m>>
```

The presence of any of the keyword operands except SYNONYM establishes the corresponding default for the ENTER (or EDGAR) command with the matching filetype.

For example, if the FILE statement read

```
FILE ASSEMBLE * LRECL 80 DEFAULT ASMDEF U
```

and the user entered

```
EDGAR TEST ASSEMBLE
```

the following statement would be generated

```
EDGAR TEST ASSEMBLE * LRECL 80 DEFAULT ASMDEF UPDATE
```

The presence of the SYNONYM keyword establishes a synonym filetype, yyyyyyy, by which a file may be called. In addition, the filetype may be abbreviated to the minimum number of characters indicated by 'm'. For example, if the FILE statement read

```
FILE ASSEMBLE * LRECL 80 SYN TEST 1
```

and the user entered

```
EDGAR SOME T
```

the following statement would be generated

```
EDGAR SOME ASSEMBLE * LRECL 80
```

A filetype of asterisk (*) not starting in position one (1) of the statement establishes the default ENTER settings for all filetypes for which there is no FILE control statement.

*	An asterisk in column one indicates a comment card and has no effect on the editing system environment.
PFxx	The PF control statement allows the user to establish the meaning of the depression of a PF Key. The data which appears on the rest of the PF control statement becomes associated with the key and is placed, LIFO, in the CMS console stack when the PF key is depressed during an editing session. The data is then available to the SOS facility and may, as a result, cause a simulated screen operation or become part of the data file. (See Section V, Advanced Editing Facilities for a discussion of SOS and its use of the CMS console stack).
TABCHR	The TABCHR control statement defines a logical tab character. When this character is encountered in the data as the I/O virtual screen is being updated (see Section III, System Flow), a tab operation is performed based on the current "tab settings" as defined by the editing command TAB. Use of the TABCHR is functionally equivalent to use of the "FIELD MARK" key. (See Section III, Keyboard Functions).

TABDEF

The TABDEF control statement defines the "tab fill" character string. This character string is inserted on the I/O virtual screen from the point where the logical tab character was depressed to the start of the data at the next "tab stop". For example, if the TABDEF and TABCHR read:

```
TABDEF '

TABCHR @
```

and the data appearing on the screen reads as follows:

```
abc  def       ghi
```

with the 'A', the 'D' and the 'G' appearing in the "tab stop" columns, and one entered:

```
x@klm@y
```

the result would be:

```
x      klm       yhi
```

Setting the TABDEF to nothing means "don't fill" and will leave existing data on the I/O virtual screen. (The function of the "DUP" key is useful when using logical tabs. See Section III, Keyboard Functions.)

CMDSYN

The CMDSYN control statement allows the user to create command synonyms. Through the use of this control statement, the user may provide a synonym for any EDGAR, CMS, or CP command enterable from the EDGAR command line. The synonym definition may include the re-ordering and/or reformatting of command operands.

The CMDSYN control word identifies a synonym definition statement describing the synonym name, minimum abbreviation, and the order and format of its operands. This control statement must be immediately followed by a synonym model as shown below.

```
CMDSYN xxxxx < n > < &  & /  & .  & * >
ABC  & 1  & 2  & 3
```

The synonym definition statement contains the name of the synonym, xxxxx, the minimum abbreviation count, "n", and a set of descriptive codes identifying the position and attributes of each of the possible operands. An ' & ' signifies a blank delimited operand. An ' & /' signifies a string delimited operand such as /abc/. An ' & .' indicates a dual string delimited operand. This form is used when two operands utilize a common string delimiter as in /abc/xyz/ such that the embedded delimiter, '/', can be reused for the second operand. In this case, the format of the first string would be signified by an ' & .' and the format of the second by an

'& /'. An '& *' signifies "all remaining command data, as entered".

Immediately following the synonym definition statement is the synonym model. This statement is a model of the command which is to be generated from the command synonym. The model statement includes all literal data to be generated. The variable data (operands supplied in the source command) are indicated by & 1, & 2, etc. In this case, '& 1' refers to the first operand encountered under control of the synonym model. For example, if the synonym HEXC were defined as follows:

```
CMDSYN HEXC 1 &. & / & *
C /X' & 1'/X' & 2'/ & 3
```

and the following command was entered:

```
h /1d/ef/ * * ver
```

the command generated (and processed by the Editing Command Processor) would be:

```
C /X'1D'/X'EF'/ * * VER
```

Note that the "string-delimited" operands (X' & 1' and X' & 2') include only the data within the delimiters; i.e., the delimiters in the source command are removed prior to substitution. Thus the user may provide alternate delimiters if desired.

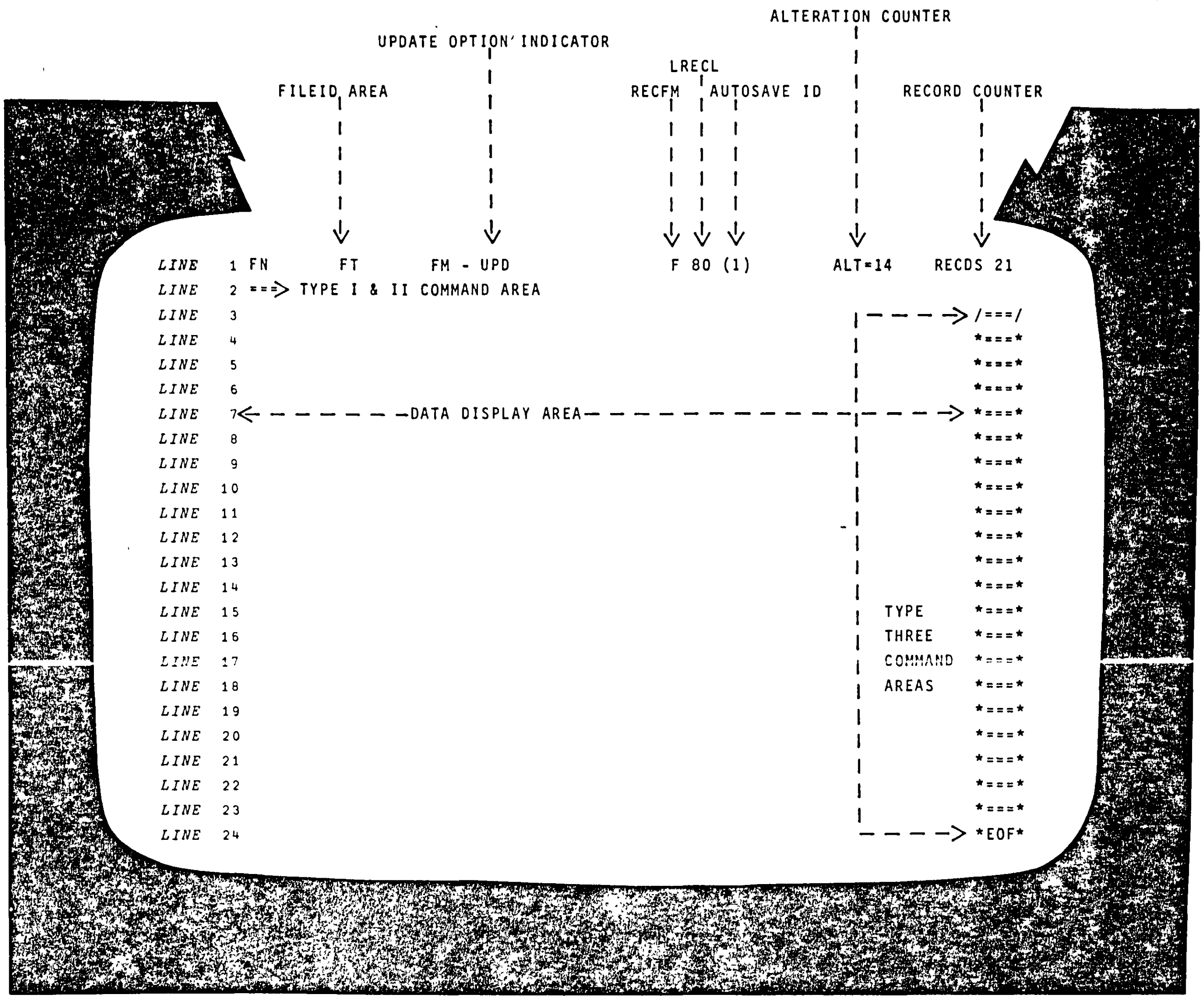
The synonym command table is always examined before analysis of the command thereby permitting synonyms for any EDGAR, CMS, or CP command enterable from the command line.

A synonym definition cannot contain a logical command separator. Since the synonym is processed by the Editing Command Processor, facilities which are provided by the I/O and Display Processors are not available to data generated through use of the synonyms. For example, the resultant command is not translated before processing nor can you define a synonym to execute an SOS command.

After the processing of EDGAR $PROFILE, the file to be edited is read into virtual storage and closed. The file is written from storage-to-disk when a FILE or SAVE command is issued or via the AUTOSAVE facility (see Section IV). The method used in each of these cases is to write the file to a utility file ('EDIT CMSUT1') on the specified disk. When the entire file has been successfully written, the original file is erased and the utility file is 'RENAMED' to the original file-id. Should any errors occur while writing the utility file, the original is always maintained. Furthermore, any AUTOSAVE files for the given level are not erased until the utility file has been successfully written and renamed. If the file 'EDIT CMSUT1' exists, EDGAR cannot be initialized. This reduces inadvertent deletion of important data. If this condition exists, the user should verify the information in the utility file and 'ERASE' it if not needed or 'RENAME' the file to the desired fileid if required.

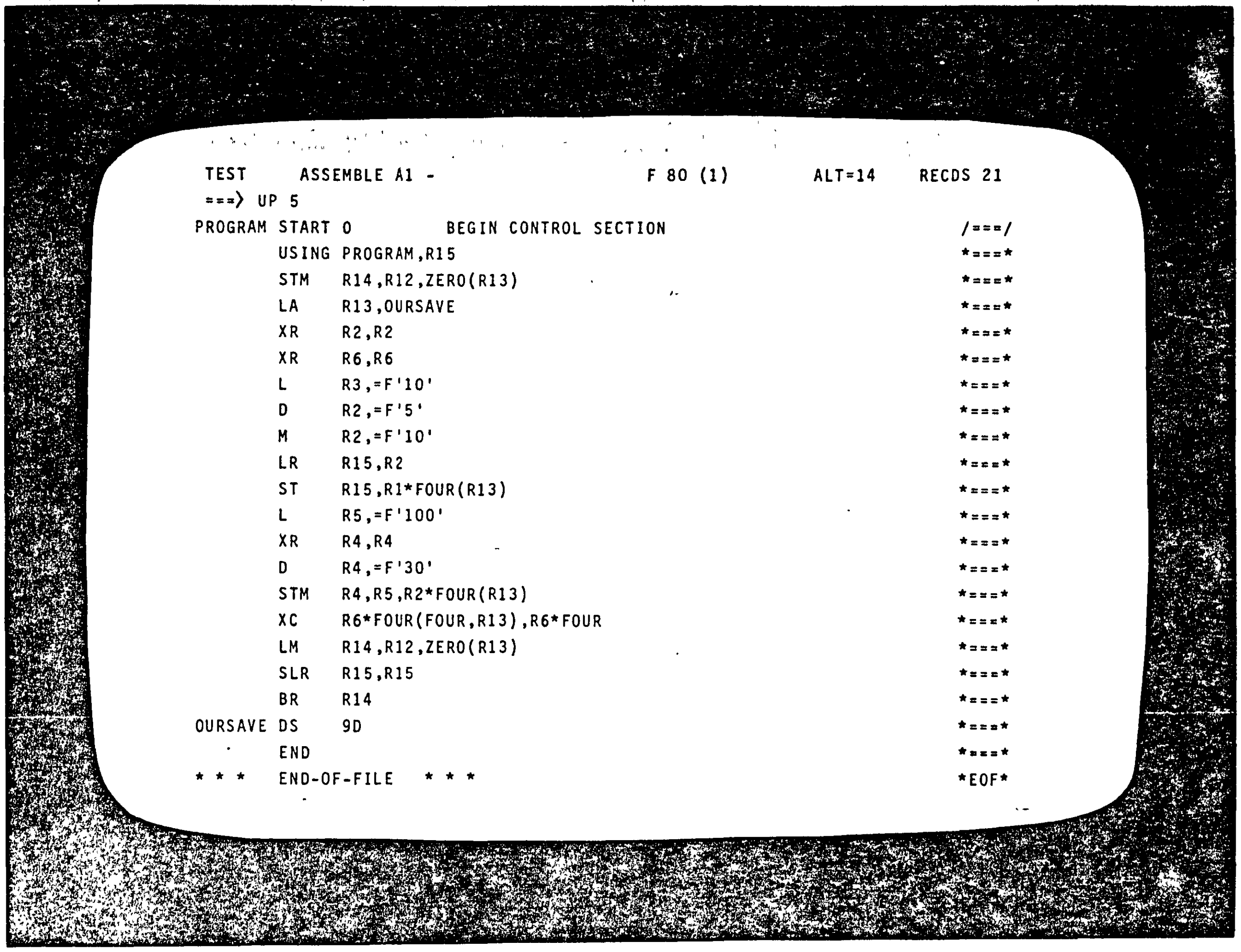
Screen Structure and Formatting

Figure 2 below describes the basic screen layout and names of the various screen fields. The user should attempt to become familiar with these names because they will be used throughout this manual and will be referenced in error messages generated by EDGAR.



```
LINE  1 FN       FT       FM - UPD              F 80 (1)      ALT=14     RECDS 21
LINE  2 ==> TYPE I & II COMMAND AREA
LINE  3                                                                   /===/
LINE  4                                                                   *===*
LINE  5                                                                   *===*
LINE  6                                                                   *===*
LINE  7                                                                   *===*
LINE  8                                                                   *===*
LINE  9                                                                   *===*
LINE 10                                                                   *===*
LINE 11                                                                   *===*
LINE 12                                                                   *===*
LINE 13                                                                   *===*
LINE 14                                                                   *===*
LINE 15                                                                   *===*
LINE 16                                                                   *===*
LINE 17                                                                   *===*
LINE 18                                                                   *===*
LINE 19                                                                   *===*
LINE 20                                                                   *===*
LINE 21                                                                   *===*
LINE 22                                                                   *===*
LINE 23                                                                   *===*
LINE 24                                                                   *EOF*
```

Figure 2. Basic screen layout and definition



```
 TEST     ASSEMBLE A1 -                    F 80 (1)        ALT=14    RECDS 21
 ===> UP 5
PROGRAM START 0          BEGIN CONTROL SECTION                            /===/
        USING PROGRAM,R15                                                 *===*
        STM   R14,R12,ZERO(R13)                                           *===*
        LA    R13,OURSAVE                                                 *===*
        XR    R2,R2                                                       *===*
        XR    R6,R6                                                       *===*
        L     R3,=F'10'                                                   *===*
        D     R2,=F'5'                                                    *===*
        M     R2,=F'10'                                                   *===*
        LR    R15,R2                                                      *===*
        ST    R15,R1*FOUR(R13)                                            *===*
        L     R5,=F'100'                                                  *===*
        XR    R4,R4                                                       *===*
        D     R4,=F'30'                                                   *===*
        STM   R4,R5,R2*FOUR(R13)                                          *===*
        XC    R6*FOUR(FOUR,R13),R6*FOUR                                   *===*
        LM    R14,R12,ZERO(R13)                                           *===*
        SLR   R15,R15                                                     *===*
        BR    R14                                                         *===*
OURSAVE DS    9D                                                          *===*
        END                                                               *===*
* * *   END-OF-FILE   * * *                                               *EOF*
```

Figure 3. Sample screen with one logical display

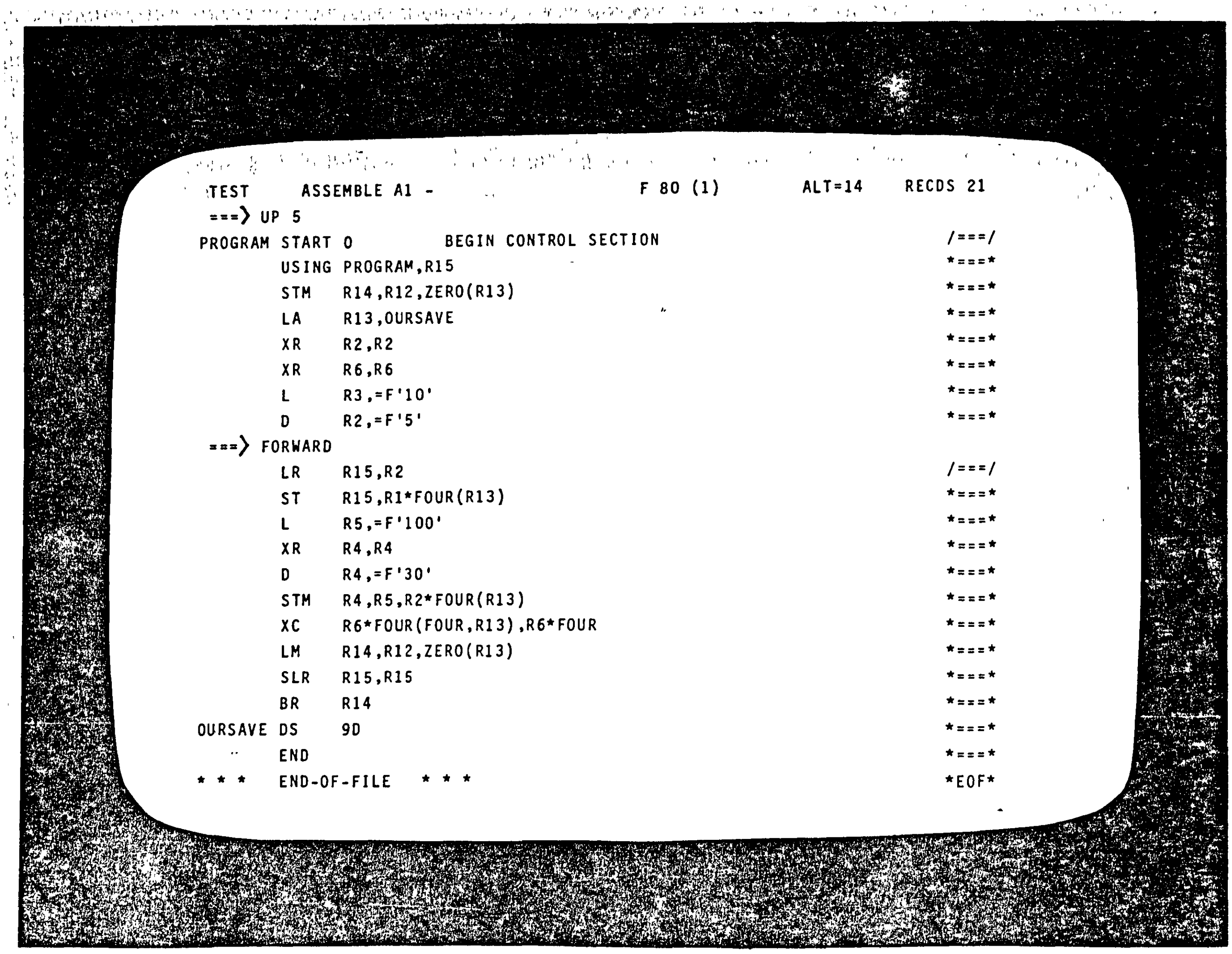
The basic screen contains a system information area on line 1. Lines 2 through 24 are used for a series of one or more "logical displays". The number of "logical displays" and format of each is established by the user via the Type I command FORMAT (see FORMAT command in Section IV). Each logical display contains a command line and several data-display areas. Optionally, each data-display area may be accompanied by a Type III command area pertaining only to the individual data-display area.

Figure 3 illustrates the simplest screen format. At the top of the 3270 screen (Fig. 3, Line 1) is a system-required area containing the file-identifiers and file status information. In this example, the "fileid" is "TEST ASSEMBLE A1". The file-id may be modified by the user at any time during the editing session. The editor will verify any new fileid entered. If any errors are encountered in the new fileid, the fileid will not be changed, the old fileid will be re-displayed and the 3270 audible alarm will sound (no error message will be produced). When a FILE or SAVE request is issued, the fileid at that point becomes the name of the file written to disk (unless over-ridden by the FILE or SAVE command). The specified fileid must contain all three identifiers; that is, filename, filetype, and filemode. Each of these must conform to the specifications required by CMS for standard file identifiers. Furthermore, the filename, filetype, and first character of the filemode (disk identification letter) must not match that of any existing edit level.

The file status area, which makes up the rest of line one (1), cannot be modified and contains pertinent information relative to the current edit level. Following the RECFM and LRECL indicators within the status area is the "AUTOSAVE-ID" field (in parentheses); the next field is the current file alteration count. In Figure 3, the "AUTOSAVE-ID" is 1 and the file alteration count is 14. Further information on these fields can be found in Section IV, under the discussion of the SET command. The last field in the file status area is the current record count. This count is always updated to reflect the number of items currently in the working copy of the file.

Line 2 in figure 3 is the start of the first "logical display". Each logical display begins with a pointer (===>). This pointer identifies the start of the Type I & II command area (known as the 'command line') for that "logical display". Immediately following the command line is the first data-display area (Figure 3, Line 3). Each data-display area is used to display one (1) item of the file. Using the Type I command, FORMAT, the user may specify the number of data-display areas for each logical display as well as the number of physical screen lines to be allocated for each data-display area. There must be at least one data-display area for each "logical display". In Figure 3, only one physical screen line has been allotted to each data-display; thus, the user may view one file item on each successive line of the screen.

The screen format of Figure 3 is the "default" arrangement (that which requires no user action).



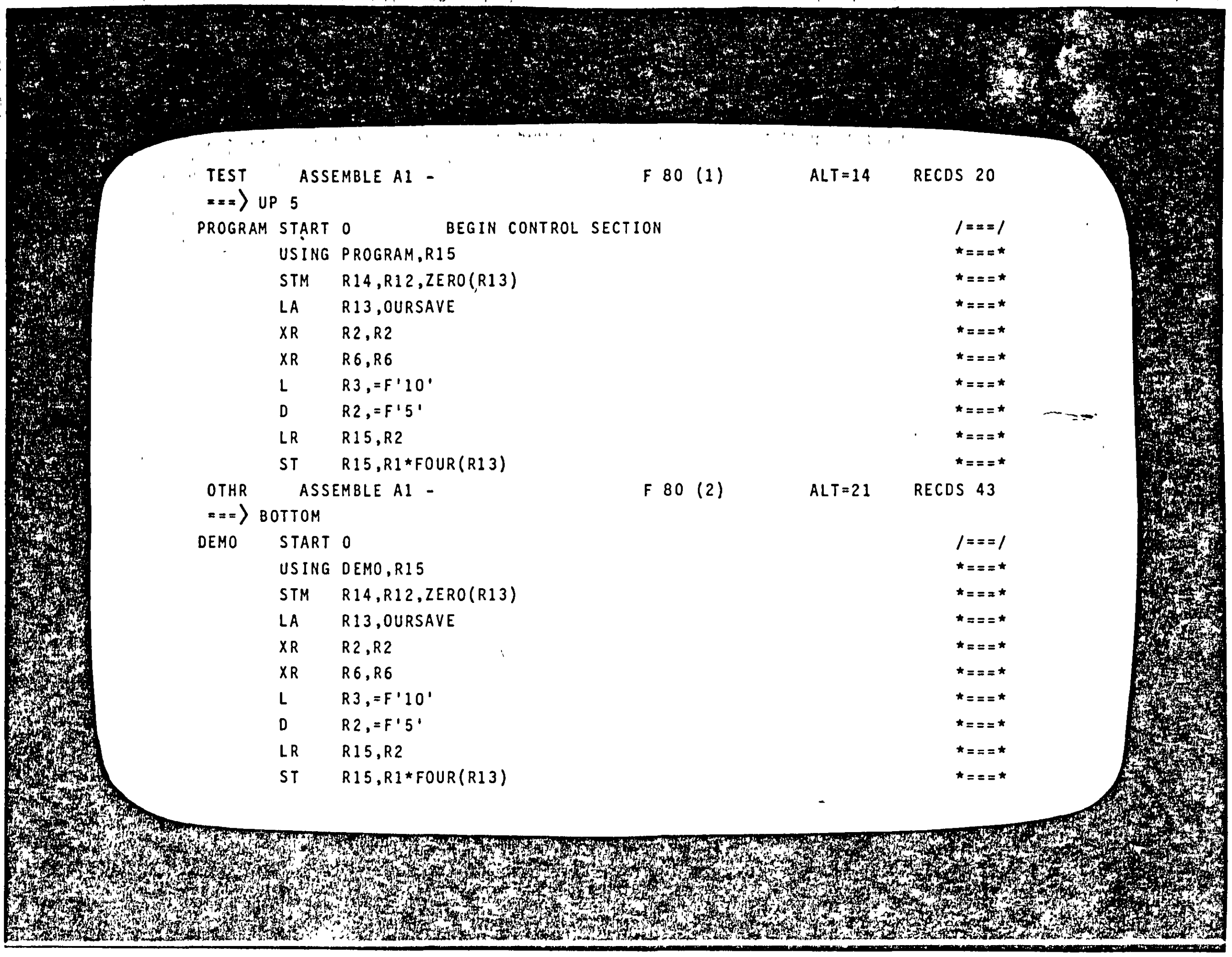
```
 TEST      ASSEMBLE A1 -                 F 80 (1)        ALT=14     RECDS 21
 ===> UP 5
PROGRAM START 0         BEGIN CONTROL SECTION                          /===/
        USING PROGRAM,R15                                              *===*
        STM   R14,R12,ZERO(R13)                                        *===*
        LA    R13,OURSAVE                                              *===*
        XR    R2,R2                                                    *===*
        XR    R6,R6                                                    *===*
        L     R3,=F'10'                                                *===*
        D     R2,=F'5'                                                 *===*
 ===> FORWARD
        LR    R15,R2                                                   /===/
        ST    R15,R1*FOUR(R13)                                         *===*
        L     R5,=F'100'                                               *===*
        XR    R4,R4                                                    *===*
        D     R4,=F'30'                                                *===*
        STM   R4,R5,R2*FOUR(R13)                                       *===*
        XC    R6*FOUR(FOUR,R13),R6*FOUR                                *===*
        LM    R14,R12,ZERO(R13)                                        *===*
        SLR   R15,R15                                                  *===*
        BR    R14                                                      *===*
OURSAVE DS    9D                                                       *===*
        END                                                            *===*
* * *   END-OF-FILE   * * *                                            *EOF*
```

Figure 4. Sample screen with two logical displays

By permitting the user to format a screen into multiple logical displays, EDGAR allows the user to view and edit multiple portions of a given file on one screen simultaneously.

Figure 4 shows a sample screen layout with two "logical displays". Note the second command line (Fig. 4, Line 11). Any Type I or II command may be entered on either command line. Line 12 in Fig. 4 is the first data-display area for the second logical display. This line is the "current line" for the second logical display as indicated by the '/' at the start of the Type III command area for this line.

The screen format of Figure 4 would result from the command:
FORMAT 1-8 1-13 (see command discussion in Section IV).



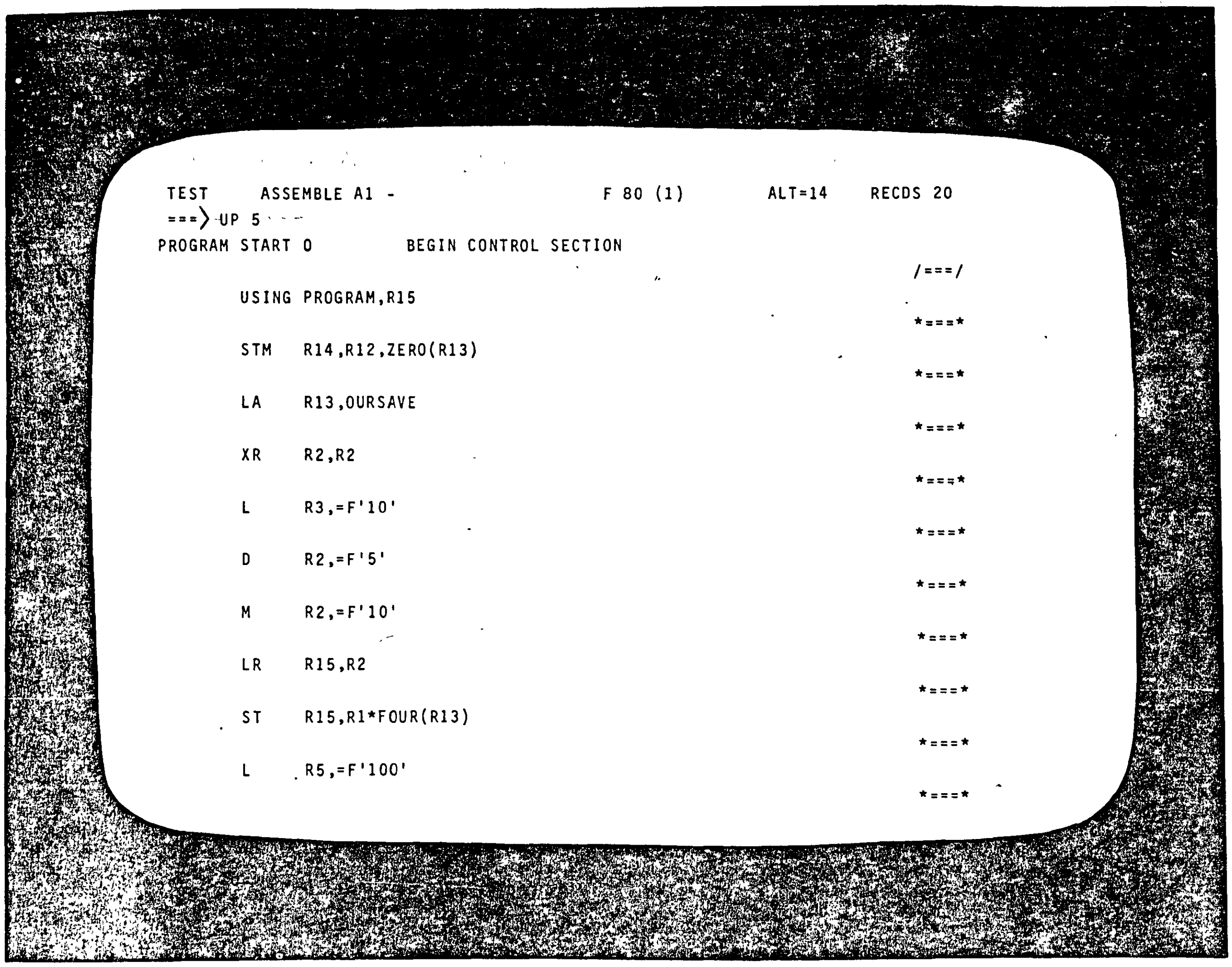
```
 TEST     ASSEMBLE A1 -                    F 80 (1)       ALT=14    RECDS 20
===> UP 5
PROGRAM START 0         BEGIN CONTROL SECTION                          /===/
        USING PROGRAM,R15                                              *===*
        STM   R14,R12,ZERO(R13)                                        *===*
        LA    R13,OURSAVE                                              *===*
        XR    R2,R2                                                    *===*
        XR    R6,R6                                                    *===*
        L     R3,=F'10'                                                *===*
        D     R2,=F'5'                                                 *===*
        LR    R15,R2                                                   *===*
        ST    R15,R1*FOUR(R13)                                         *===*
 OTHR     ASSEMBLE A1 -                    F 80 (2)       ALT=21    RECDS 43
===> BOTTOM
DEMO    START 0                                                        /===/
        USING DEMO,R15                                                 *===*
        STM   R14,R12,ZERO(R13)                                        *===*
        LA    R13,OURSAVE                                              *===*
        XR    R2,R2                                                    *===*
        XR    R6,R6                                                    *===*
        L     R3,=F'10'                                                *===*
        D     R2,=F'5'                                                 *===*
        LR    R15,R2                                                   *===*
        ST    R15,R1*FOUR(R13)                                         *===*
```

Figure 5. Sample screen with two logical screens.

A second form of screen subdivision permits the user to format a screen into multiple logical screens by use of the SCREEN command (see Section IV).

Figure 5 shows a sample screen layout with two logical screens. Note the second file status area (Fig. 5, Line 13). The significant feature of this arrangement is that two or more different files can be edited and viewed at the same time. When two or more logical screens are defined, EDGAR uses them when possible in response to the ENTER command.

The screen format of Figure 5 would result from the command:
SCREEN 12 12



```
TEST     ASSEMBLE A1 -                     F 80 (1)        ALT=14    RECDS 20
===> UP 5
PROGRAM START 0          BEGIN CONTROL SECTION
                                                                      /===/
          USING PROGRAM,R15
                                                                      *===*
          STM    R14,R12,ZERO(R13)
                                                                      *===*
          LA     R13,OURSAVE
                                                                      *===*
          XR     R2,R2
                                                                      *===*
          L      R3,=F'10'
                                                                      *===*
          D      R2,=F'5'
                                                                      *===*
          M      R2,=F'10'
                                                                      *===*
          LR     R15,R2
                                                                      *===*
          ST     R15,R1*FOUR(R13)
                                                                      *===*
          L      R5,=F'100'
                                                                      *===*
```

Figure 6. Sample screen layout with multiple-line data display area

Sometimes one physical item is wider than the screen line. One approach to this problem is to use the Type II commands, LEFT, and RIGHT to allow viewing of the appropriate portions of the data item. As an alternative, a display may be formatted in which two or more successive screen lines are allotted to each data-display area. By "formatting" the screen in this manner, and defining the "viewed" data columns to be wider than one screen line, data-wraparound (from one screen line to the next) can be effected.

The screen format of Figure 6 would result from the command:
FORMAT 2-11

The last eight positions of each data-display area contain the Type III command line for that data-display. Type III commands are used to perform "edit-in-place" functions for each individual data-display (see Type III Editing Commands in Section IV).

Optionally, the user may specify (using the FORMAT command) that the given logical display is not to contain Type III command areas, permitting full screen-width viewing. However, users should avoid, whenever possible, eliminating the Type III areas since these are used as physical line separation fields on the 3270. Elimination of these fields will cause substantially increased communication overhead (especially on remote devices) and seriously complicate cursor movement and positioning.

Each data-display area normally contains the "viewed" columns of one data-item in the file (see the VIEW command in Section IV). However, a line may also contain the "TOP-OF-FILE" or "END-OF-FILE" indicators (example on line 24 in figure 3). These lines do not become part of the file on disk and any changes to these lines or their corresponding Type III areas are ignored.

Screen Order of Processing

The order of processing for the screen is of prime importance because it determines the net effect of entering multiple commands and various changes to the display. First the "FILEID" area of the screen is examined for change. Then each logical display is processed, in order, from the top of the screen to the bottom. The processing of each logical display consists of: 1) scanning each data-display area and recording any modifications made, therein; 2) processing the Type III command for the given data-display area (if Type III commands exist) immediately after recording any changes to that display; 3) processing the Type I & II command area; and 4) performing any "post-processing" (see Section IV) necessitated by Type I commands.

When all display processing is complete, the "alteration count" is inspected to see whether an "AUTOSAVE" is necessary (if that option is in effect). If required, the file is checkpointed before the "set-up" of the new display begins. When all of the above processing has been completed, a new display is constructed and the logical screen is returned to the I/O Processor.

Keyboard Functions

This section describes the functions of the 3270 keyboard which are unique to DISPLAY EDITING SYSTEM for CMS. Users should also acquaint themselves with the 'OPERATOR'S GUIDE for IBM 3270 INFORMATION DISPLAY SYSTEM' (GA27-2742).

•PA1 Depression of this key will cause 'CP' mode to be entered and a "CP READ" status will be displayed. While in this environment, any 'CP' commands may be issued. To return to EDGAR from the 'CP' environment, issue the 'CP' command, "BEGIN".

•PA2 Depression of PA 2 will cause the CMS SUBSET environment to be entered and a "RUNNING" status to be displayed. While in the CMS SUBSET environment, any CMS commands which run in the transient area may be issued. For example:

ACCESS	LISTFILE	RENAME
CP	PRINT	RETURN
DISK	PUNCH	SET
ERASE	QUERY	STATE
EXEC	READCARD	TYPE

To return to EDGAR from the CMS SUBSET environment, issue the CMS SUBSET command, "RETURN".

•PA3 Depression of the PA 3 key (available only on "data-entry" type keyboards) is considered invalid by EDGAR and will cause the "input inhibited" indicator to appear, the keyboard to "lock," and the audible alarm to sound (if this feature is installed).

NOTE: Depression of any PA key does not cause transmission of data to the computer. Hence, alterations made to the screen before depressing a PA key and after depressing ENTER or a Program Function key, will be lost.

•FIELD MARK Depression of this key inserts a logical tab character into the data stream. This is functionally equivalent to entering the character defined by the TABCHR control statement. (See Section III, System Overview, Initialization and Termination). Logical tabs are useful as a means of shifting data to a specified data column without causing an I/O interrupt for each "tab" operation. When using logical tabs the data on the screen is not rewritten and will not appear to be shifted until an "ENTER" key is depressed. At that time, the data file is updated, the I/O virtual screen is rebuilt and, when it is displayed will reflect the changes to the data.

•DUP The DUP key is useful in conjuction with a logical tab operation. It indicates that the rest of the data read as part of this screen field is to be ignored. When used in conjuction with the overlay facility of logical tab it allows the user to indicate the end of the new "typed-in" data. For example if the data appears as

ABC DEF GHI

and the user overlays that with

;bc*DEF GHI

where ';' is a FIELD MARK, '*' is a DUP, the upper case is residual data and the lower case that which the user types in; the result will be

ABC BCF GHI

This example assumes TABDEF is set to a "null string" and that the 'A', the 'D', and the 'G' are in the "tab stop" columns.

- **ENTER** Depressing the ENTER key indicates that the I/O virtual screen is to be passed to the Display Processor (see Section III, System Overview, System Flow). It is not until the ENTER key has been depressed, either at the physical display device itself or simulated via SOS, that the screen is analyzed and any resultant action taken.

- **CLEAR** When depressed on the physical display device once, the CLEAR key causes the physical screen to be rebuilt as it appeared the last time an interrupt causing key (PF keys or ENTER key) was depressed. In all other cases, the CLEAR key causes the I/O virtual screen to be rebuilt as it appeared after the ENTER key was last depressed, and the CMS console stack to be purged.

IV. Editing Functions

EDGAR commands are divided into three (3) categories. Type I and II commands may either be entered on the Type I & II Command Area of the screen (Fig. 2, Line 2), or passed to the Editing Command Processor via the SVC interface. Type III commands generally pertain to individual Data Display Areas and may be entered only in the Type III Command Areas.

Type I commands differ from Type II commands in that they require "edit level post-processing." This means that the immediate effect of a command is to flag the requested action for execution at the appropriate time. Edit level post-processing actually occurs after all logical displays on the screen have been processed and all data has been analyzed. Post-processing involves edit level disposition, edit level storage, and edit level re-formatting.

The Type I & II command line entry may be preceded by the character " & " causing EDGAR to leave the associated command(s) on the screen if no errors are encountered during processing. This reduces the need for re-entering commands which are used repeatedly. For example, if " & F" is entered on the command line, the FORWARD command will remain on the screen, allowing the user to "scroll" through the file without re-entering the FORWARD command.

Multiple commands (both Type I and II) may be entered on the command line if they are separated by a X'01'. The commands will be processed, in order, from left to right, with the exception of any post-processing activity, which will occur only once, when all other command processing is complete.

Type I and II commands which are not recognized by the Editing Command Processor are passed to the CMS subset.

Messages generated during edit level post processing, such as errors in writing to disk or errors in re-formatting are not issued on the command line. Instead, these messages are produced on the standard VM or CMS screen format (with the command line and the "RUNNING" status at the bottom). After displaying these messages, the screen will enter "MORE" status at which point the user may depress PA 2 or CLEAR to return to EDGAR. If post-processing errors occur, the necessary action must be taken (including re-issuing of commands) to renew the post-processing request.

Type I Commands

The Type I commands and their formats are described below. A Type I command is entered on the Type I & II Command Area of the screen (Fig. 2, Line 2). Each command described below is shown with its shortest form. Optional arguments are enclosed in brackets (<>) and the "OR" symbol (|) is used to designate operands of which one and only one must be selected from the list.

CANCEL

CANCEL

This command causes immediate termination of the editor. It is functionally equivalent to entering QUIT for each active edit level. (For information on edit levels, see the TYPE I command, ENTER).

ENTER

```
E  <Fn <ft <fm>>>   <Lrecl nn>
                    <Seq8 | NOSeq8>
                    <Ctl xxxxxxxx | NOCtl>
                    <Update | NOUpdate>
                    <DEFault yyyyyyyy | NODEF>
                    <CLEAR | NOCLEAR>
```

The ENTER command provides the capability to edit on several different "edit levels". Each edit level pertains to a different file and each is edited independently. If the ENTER command is issued with no arguments, another level is entered. If no other level exists, no apparent change occurs.

Optionally, a filename, filetype, and filemode may be specified in the ENTER command. If the filename and/or filetype is specified as "*", the corresponding attribute of the current fileid will be substituted. (Note: the use of "*" for the filename or filetype is not permitted when entering the first level, i.e. from the CMS command level). The filemode must be specified as a one- or two-character filemode or "*". If only the disk letter is specified, the mode number defaults to "1". When a fileid is specified in this manner, the ENTER command causes EDGAR to search all existing edit levels for a file name, type and mode matching that requested. If the mode is specified as "*", EDGAR will scan for matching filename and filetype, only.

If no existing edit level can be found with a fileid matching that requested, a new edit level is created using the requested fileid. In this case, the requested disk is searched for a file with the same filename and filetype as that specified. If the filemode is specified as "*", the standard order of disk search (A, B, C, D, E, F, G, S, Y, Z) is used to locate the file. Once found, the file is read from disk and editing initialized on the new level.

If no previous edit level exists with the specified filename and type, and no file can be found on disk with a matching fileid, EDGAR assumes that a new file is to be created. If the mode was specified as "*", the new level will be created with a default mode of A1.

The remaining parameters on the ENTER command are used only when creating a new edit level. These parameters will be ignored if an existing edit level is found with a matching fileid. Each option permits specification of file characteristics for the level being created. Only those characteristics which cannot be altered after the level has been initialized are used in the ENTER command.

LRECL - this option sets the maximum logical record length for the newly created edit level. As with other options, LRECL may be specified explicitly on the command, or implicitly through the EDGAR $PROFILE. If this option is not specified and the file exists on disk, the maximum logical record length in the existing file is used. If the option *is* specified and the file exists on disk, the maximum record length for the newly-created edit level will be equal to the specified length or the maximum record length in the file, whichever is greater. If the file does not exist, the specified LRECL is used. If none has been specified, LRECL will default to 80.

DEFAULT - The DEFAULT option specifies the name of the EXEC, yyyyyyyy, that is to be called when this edit level is initialized. The EXEC is passed the following parameters:

& 1 - filename
& 2 - filetype
& 3 - filemode
& 4 - OLD or NEW
& 5 - UPDATE (exists for UPDATE FILES ONLY)

It is in this "pre-processing" EXEC that the user has the opportunity to tailor the editing environment to his needs. The DEFAULT option has a corresponding no-op option, NODEF, which may be used to override the ENTER default parameters as specified in EDGAR $PROFILE.

CLEAR - The CLEAR option will cause the CP formatted screen to be "cleared" with no intervention required by the user. This option is permitted only when the first edit level is being initialized. If unspecified, CLEAR is assumed. The CLEAR option has a corresponding no-op option, NOCLEAR, which may be used to override the ENTER default parameters as specified in EDGAR $PROFILE.

CTL, UPDATE and SEQ8 - these options are used when an "update" file is to be created as a result of changes made during editing. Each has a corresponding no-op operand, NOCTL, NOUPDATE and NOSEQ8, for use in overriding EDGAR $PROFILE defaults.

The UPDATE facility of EDGAR is intended primarily for programming applications where "update" streams are frequently generated. This facility allows the user to request that changes made to a program source file while editing be written to disk in the form of a CMS UPDATE file. Thus, the source file remains unaltered and the update file may be applied to the source using the CMS UPDATE command to re-create the source as it appeared after editing. Furthermore, when editing of the source file is initiated using this facility, the existing update file is automatically applied to the source during initialization. Thus, the updated version of the source becomes available for editing and changes made during this editing process are "merged" with the initial updates. The resultant update file replaces the original update file.

To request the automatic update inclusion and generation, the UPDATE option of the ENTER command must be specified. This option is ignored if the file requested has been previously "entered" (using multiple editing levels). When the ENTER command with the UPDATE option is issued, the requested source file is initialized for editing. The source file must exist on disk and must contain fixed-length, 80-character records; otherwise, an error message is generated and ENTER will not be performed. When the source file is read, all source records are checked for valid sequencing in columns 76-80. If the 'SEQ8' option was also specified in the ENTER command, columns 73-80 are checked for valid sequencing. Once the source file has been initialized, the update file is applied to the source. The file name of the update file must be the same as that of the source and the file type must be "UPDATE". The update file must exist on disk and must follow the rules required for a CMS update file (see CMS "UPDATE" command in VM/370 CMS Command and

Macro Reference, GC20-1818). If the update file does not conform to these requirements, an error message will be generated and ENTER will not be performed.

Once the requested editing level has been successfully initiated, editing may proceed in the normal fashion. There are no restrictions on editing commands used. Data placed in the sequencing columns (73-80) will be lost when the update file is created as sequence numbers are automatically generated by EDGAR. EDGAR will maintain a record of changes made in dynamic fashion and indicate collection of "change" data by placing the characters 'UPD' at the start of the file status field (Figure 2, Line 1). In addition, the fileid area will reflect the name of the update file - not the original source.

When the UPDATE option is in effect for a given editing level and a "FILE" or "SAVE" (Type I) command is issued, the update file only is written to disk. The source file, therefore, remains unaltered. The fileid of the update file will be determined in the standard fashion for a "FILE" or "SAVE" command.

The update stream generated by EDGAR will attempt to maximize the use of the './ R ' update function. Thus if "inserted" records are found to be adjacent to "altered" or "deleted" records in the file (or vice versa), a single './ R ' card will be generated followed by the necessary input cards. All data cards written to the update file are sequenced by EDGAR regardless of the current setting of the SERIAL command. "Altered" cards are given the same sequence numbers as the source records they replace. "Inserted" cards are sequenced with the highest increment divisible by ten which will not introduce sequence errors in the resultant file. For example, if nine records are inserted into a file between items XXX00010 and XXX00020 they will be sequenced XXX00011 through XXX00019. If more records are inserted than will fit in the increment, the following records in the file are automatically "replaced" to provide extra sequence numbers. To illustrate this using the above example, ten records inserted between the items indicated above, would be sequenced as XXX00010 through XXX00020 and the original item XXX00020 would be "replaced" as item XXX00021.

EDGAR, also supports the multi-level update facility of CMS/370. A multi-level update is controlled through the use of an update "CNTRL" file. The structure of this file and the associated auxiliary files is described in some detail in "IBM Virtual Machine Facility/370: CMS Command and Macro Reference" (GC20-1818) under the discussion of the CMS "UPDATE" command. Users considering this facility for program development and maintenance should evaluate its features based on this discussion. All features of the 'CNTRL', 'AUX', and 'PTF' files described for the "UPDATE" command are supported by EDGAR.

The multi-level update function of EDGAR is invoked by specifying the "CTL" option followed by the filename of the desired control file on the "ENTER" command. Use of this option implies the "UPDATE" option. As with "UPDATE" and "SEQ8", the "CTL" option is ignored if the edit level being entered already exists.

When initiating an edit level with the "CTL" option, the following steps are performed by EDGAR:

1. The source file is located and read.

2. The multi-level "CNTRL" file is read and each record is validated for syntax. For each update level to be applied, the update file is read and applied to the original source as described in the previous discussion of the UPDATE facility.

3. As each update level is applied, all previous update levels are "forgotten"; that is, they become an integral part of the working copy of the file and their original identity lost. The edit level then assumes the filetype of the newly applied update file. Thus, the "last" update level fileid implied by the CTL (and AUX) file structure becomes the fileid of the edit level being initiated. If no file with this identifier was found, the edit level is assumed to be creating a new update file and an appropriate message will be generated. Note that another edit level of the same name must not already exist.

The normal process, therefore, for developing a "new" update level to a base source file would be to:

1. Edit the source with the "CTL" option applying any existing update levels.

2. Make the changes which will constitute the "new" update level and "FILE".

FILE

FILE <Fn <Ft <Fm>>>

The FILE command causes the file at the current edit level to be re-written to disk and the current level to be released. Optionally, an alternate file-id may be specified in the FILE command to override the "current" file identifiers. An "*" in place of any of the three identifiers specifies that the corresponding identifier in the "current" file is to be used. After the file has been successfully written to disk, any "checkpoint" files (see the 'AUTOSAVE' option of the 'SET' command in Section IV, TYPE II commands) having the "current" 'AUTO-ID' are erased (on any disk) and an exit is taken from the current to the "next" level. If no other level exists, the editor terminates normally, and control is returned to CMS.

FORMAT

FORM <>x-y x-y .... x <-y>*

The FORMAT command allows the user to set-up the logical display(s) for the current file edit level. Each pair of numbers separated by a hyphen (-), provides the format specifications for one logical display. Each logical display consists of a command line, and a series of one or more "data-display" areas. Each "data-display" area may be used to display one "item" from the file. As many "screen" lines as are required may be specified for each "data-display" area, thereby allowing "data-wraparound" from one line to the next. The number 'x' in each pair of "format" specifications designates the number of screen lines to be allotted for each "data-display" area in the corresponding logical display. The number 'y' in each pair of "format" specifications designates the number of "data-display" areas for the display (and thus the maximum number of file items which may be displayed at once within the logical display). For example:

FORMAT 1-22

requests the construction of one logical display containing (22) "data-display" areas (allowing simultaneous viewing of up to 22 items within the file) and the allotment of one (1) screen line for each file item. (This is the default screen format). Figure 3 shows this format.

FORMAT 1-8 1-13

will create two logical displays each of which will allot one screen line for each data-display area. The first logical display will contain eight (8) data-display areas and the second will contain thirteen (13). Figure 4 illustrates this format.

FORMAT 2-11

will construct a screen as shown in Figure 6 in which two screen lines have been allocated for each of eleven (11) data-display areas. Note that the Type III command line for each data-display area is located at the end of the entire data-display area (end of the second screen line). In each data-display area, "displayed data" may wrap-around from the first screen line to the second.

Optionally, the number of "data-display" areas may be left-out of the last logical display specification indicating that as many consecutive "data-display" areas, as the screen size will permit, are to be created. Thus, for the first example above, 'FORM 1' would indicate that one logical display is to be built, containing as many "data-display" areas each occupying one (1) screen line as the screen size will permit. Similarly, 'FORM 1-7 1' will create two (2) logical displays; the first will contain seven (7) "data-display" areas, each occupying one (1) screen line, and the second will contain as many "data-display" areas, each occupying one (1) screen line, as the screen size will permit (fourteen in this example). Correspondingly, 'FORMAT 2' will construct a display as illustrated in Figure 5.

Placing an asterisk ("*") immediately preceding (no intervening blanks) any logical display specification pair indicates that the corresponding logical display is not to contain Type III command areas. Because of the nature of 3270 display "fields", this mode of operation can effect a significant degradation in data transmission time (especially for remote 3270 systems). Furthermore, the "Local Alteration" features of the 3277 lose significance. Therefore, operation without Type III command areas should be limited to cases where viewing a full display width is an absolute requirement and the screen is being used for display only (no data editing is to occur).

If the 'FORMAT' command is issued with no arguments, the current format settings will be displayed on the command line.

QUIT

QUIT

This command may be used to cause an exit from any given editing level without re-writing the file to disk. When the QUIT command is entered, all storage for the given level is released, and any autosave files for this level (on any READ/WRITE disk) are erased.

SAVE

SAVE <Fn <Ft <Fm>>>

The SAVE command causes the file at the current edit level to be written to disk without releasing the level. After issuing the SAVE command, the user may continue editing the current file. This command is the same as FILE except that no new level is entered after command completion (see FILE command for a description of the valid arguments).

SCREEN

SCR <n1> <n2> ....

The SCREEN command is used to define logical screens by depth. The default setting is "24". If the SCREEN command is entered with no parameters, the current setting will be displayed. The operands to the SCREEN command each represent one logical screen and indicate the number of screen lines of an edit level virtual screen which can be displayed in the corresponding logical screen. The first line displayed in each logical screen is always line one of an edit level virtual screen. When an edit level is ENTERed, it automatically becomes eligible for the "next available" logical screen (if any). Figure 5 illustrates a display with two logical screens.

Type II Commands

Most Type II commands pertain to the "current line". The "current line" is normally the displayed line immediately following the command line. In any case, the "current line" may be identified (if Type III command areas exist on the screen) by a "/" at the start of the corresponding Type III area instead of the normal "*". A display normally consists of the "current line" followed by as many succeeding items as possible within the dimensions of the current logical display. Movement or "scrolling" through the file, therefore, is the adjustment of the "current line".

The Type II commands, their shortest forms, and valid arguments are listed below:

ADD

A <n>

The ADD command is used to add blank lines to the file immediately following the "current" line. These blank lines may be used for new data whenever desired. "n" may be specified to indicate the number of lines to be "added". If "n" is omitted, one blank line is added. In some files, particularly those destined for certain compilers, blank lines cannot be allowed to remain in the file when re-written to disk. To eliminate the need to delete blank lines before a FILE or SAVE request, use the Type II function: "SET DBLANK ON". This will cause wholly blank lines to be eliminated from the file written to disk. These lines, however, will remain embedded within the in-storage (working copy) of the file to allow use of the "added" lines later in the editing session.

BACKWARD

B <n>

To allow "backward" scrolling through the file, the BACKWARD command may be used to move the "current" line back (toward the top of the file) "n" pages. A "page" is defined as the depth (number of items) last displayed on the I/O virtual screen. In the example in figure 2 the depth is 22. Thus, the BACKWARD command will cause the "current line" to be moved "back" (toward the top of the file) 22 lines.

BOTTOM

BOT

BOTTOM sets the current line to the last line in the file. If there are no lines in the file, the current line becomes "EOF".

CASE

CAS < M | U >

CASE is used to display or alter the upper case/lower case translation setting. "CASE U" specifies that all characters entered in the corresponding logical display are to be translated to upper case automatically. "CASE M" indicates that no translation is to be performed. If no parameters are specified on the CASE command, the current setting is displayed on the Type II command line.

CHANGE *C /string 1/string 2/ < n | .XXX | * < n | * <VER>>>*

The CHANGE command is used to "change" a character string to another string within defined limits and within defined columns. As indicated in the example above, the standard string delimiting character is the "/". However, any character may be used as the string delimiter so long as it is unique (does not appear within either "string 1" or "string 2"). The "string" specifications are followed by the vertical and horizontal change counts. These may be used to specify the number of times the change is to be be repeated vertically and horizontally. The CHANGE command operates only within the previously specified "ZONE" settings. (See the Type II command: ZONE, in this section). Thus, if the "start" zone is currently set to 10 and the "end" zone is set to 50, only columns 10 through 50 will be scanned for "string 1". If "string 1" is located within these columns, "string 2" will be substituted for it and data which previously followed "string 1" will be moved after "string 2". If this movement causes any non-blank data to be shifted past the "end" zone, data truncation will occur at the "end" zone column.

The vertical and horizontal "change" counts may be used to indicate the number of successive lines to be scanned for "string 1" and the maximum number of occurrences per line to be changed, respectively. The vertical change count may be specified as: "n" where "n" is a positive decimal integer not greater than the total number of items in the file; ".XXX" to indicate a "named" line to be the last line scanned for the "change"; "*" indicating that every line through end-of-file is to be scanned for the change. The "VER" option causes a display of only those lines on which a change has occurred. The CHANGE command does not cause the "current line" to be moved unless the VER option is used. In this case, the "current" line at the completion of the command is set to the first line immediately following the last line on which a change occurred, or to "EOF" if the command ended at "EOF". The first line to be scanned for the change is always the "current" line.

The horizontal change count may be specified as: "n" where "n" is a positive, decimal integer indicating the maximum number of occurrences of "string 1" per line, to be changed; or "*" to indicate that all occurrences (within the specified zones) are to be changed. If either the vertical or horizontal change count is not specified, a default value of "one" is used.

NOTES:

1. If vertical and horizontal change counts are not specified, it is not necessary to include the final delimiter character.

2. If "string 1" is "null" (i.e., two successive delimiters found with no intervening data) "string 2" will be inserted starting at the "start" zone column.

3. If "string 2" is "null" (or missing), "string 1" will simply be deleted.

4. By using the "command-hold" (&) capability (see Section IV - Type I Commands) with the CHANGE command and using the VER option, the user may view each successive change made (in a "global change"). Thus, if the user wishes to abort the "global" change, he may do so having changed only as many lines as the current "logical display" depth setting.

CMS

CMS

The CMS command will call the CMS SUBSET and will pass to it the rest of the command data, if any.

COPY

COPY n | .XXX | TOF | / < n | .XXX | / < n | .XXX | / > >

The COPY command is used to "copy" data from one portion of a file to another portion of the same file. The command requires at least one and no more than three parameters. The three parameters are the "TARGET", the "LOWER LIMIT", and the "UPPER LIMIT", respectively. The "target" item is the one after which the new data is to be copied. The target may be specified as: "n" - a positive decimal integer (not greater than the total number of records in the file) indicating the line number after which data is to be copied; ".XXX" where ".XXX" is a previously defined "named" line; "TOF" indicating that the target is the top of the file (prior to item number one); or "/" indicating that the "current line" is to be the target. The "target" must be specified. The second parameter (if included) is the "lower limit" which is the first item to be copied. The "LOWER LIMIT" may be specified as: "n" - a positive decimal integer indicating the line number to be the lower limit; ".XXX" indicating that the lower limit is the previously defined "named" line, ".XXX"; or "/" which signifies that the "current line" is to be the lower limit. If no second parameter is specified, the lower limit will be "/" (the "current line"). The third and final parameter for COPY is the "upper limit". This will be the last line to be "copied". The upper limit may be specified as: "n" - a positive, decimal integer (not greater than the total number of records in the file) signifying the line number to be the upper limit; ".XXX" where ".XXX" is the previously defined "named" line to be the upper limit; or "/" indicating that the current line is to be the upper limit. The upper limit and lower limit may refer to the same item in the file in which case only that one item is copied. If the upper limit is not specified, it defaults to "/" (the current line). The "UPPER" limit must be equal to or follow the "LOWER" limit within the file. The "TARGET" item may not be between the "UPPER" and "LOWER" limits of the "range" nor equal to either the "UPPER" or "LOWER" limit.

NOTE: The COPY function may be used to duplicate several successive line items. However, since the target and the upper limit cannot be the same, the target should specify the item immediately prior to the lower limit; in effect, COPY the data *prior* to rather than *after* itself.

CP

CP

The CP command will call CP and pass to it the rest of the command data, if any.

DELETE

*DEL < n | . XXX | * >*

The DELETE command is used to delete items from the file. If no arguments are specified, only the current line is deleted. Valid arguments are: "n" where "n" is any positive decimal integer (not greater than the total number of items in the file) indicating the number of consecutive items to be deleted; ".XXX" where ".XXX" is a previously defined "named" line to be the last deleted line (this line must lie between the "current line" and "EOF"); or "*" indicating that all lines between the current line and "EOF" are to be deleted. Note that deletions always terminate if "EOF" is encountered.

DOWN

D <n>

DOWN causes the display to appear to move "down" "n" lines by moving the "current line" toward the top of the file by "n" lines. If "n" is not specified it will default to one. If specified, "n" must be a positive decimal integer not greater than the current total items in the file. Thus, "DOWN 2" causes the "current line" to move 2 lines toward the top of the file.

DSPC

DSPC

The DSPC command causes a display on the command line in the form * LL CC of the current cursor position. (LL = Line Number, CC = Column Number). Having entered the DSPC command on the command line the user must move the cursor to the position of interest before depressing the "ENTER" key.

DSPF

DSPF

The DSPF command causes a display of the current settings of the Program Function Keys. To return from this display the user must depress PA2.

DUP

DUP <n>

DUP causes the current line to be duplicated (in-line) "n" times where "n" is any positive, decimal integer. If "n" is not specified, it will default to one. Note that the current line, itself, is not included in "n". Thus "DUP 3" would cause the current line to be duplicated three times yielding a total of four duplicate lines (including the "current line"). Following a DUP command the cursor is positioned at the first new line.

FIND

FI string

The FIND command causes a search for the specified string starting only at the beginning of the current zone. (See ZONE command in this section).

FORWARD

F <n>

To allow scrolling through the file, the FORWARD command may be used to move the "current" line forward (toward the end of the file) "n" pages. A "page" is defined as the number of items last displayed on the I/O virtual screen. In the example in figure 2 the depth is 22. The FORWARD command in that case would cause the "current line" to be moved "forward" (toward the end of the file) 22 lines.

GETFILE

GET < Fn |. < Ft | * < Fm | * < n1 < n2 | * <APPEND>>>>>>*

GETFILE is used to imbed one CMS file or a portion thereof within another. The file name and type may be specified explicitly or with an "*" indicating that the corresponding identification in the current level fileid is to be used. The file mode may be specified explicitly or as "*" (the default) indicating that the standard order of disk search (A,B,C,D,E,F,G,S,Y,Z) is to be used to locate the file. "n1" may be used to specify the starting record number at which to begin reading. "n2" specifies the last item number to be read; if specified as "*" reading will continue to "end-of-file". Normally, data read is imbedded after the "current line" and the line pointer remains unchanged. However, if the "APPEND" option is specified (this may be abbreviated to "A"), data will be "appended" at the end of the working copy of the current file and the line pointer will be set to the first item read.

INPUT

INP <c>

INPUT allows the user to input new data items after the current line. After the INPUT command is entered, the indicated logical display is set to allow data inputting. The current line is placed at the top of the display and the remainder of the logical display is completed with blank unused "input" areas. These lines differ from those created using the ADD command in that they are not actually placed into the file until used. The Type III command area pertaining to each "input" line contains the the designator, "*INPUT". Type III commands may be entered on INPUT Type III command areas. Because INPUT requires the "current line" to be displayed at the top of the logical display, at least two data-display areas must be defined for that display prior to issuing the INPUT command.

Optionally, a 1-8 character string may be entered as a parameter to the INPUT command. This character string will be placed at the start of each "input" area. This is especially useful if there are no Type III areas on the display to indicate where each data-display area ends.

As previously mentioned, only those "input" data areas which have been modified are actually added to the file. The criterion used to determine whether a given "input" data area has been modified depends on whether Type III areas are present. If Type III areas exist, a line is considered modified if any change has been made to the line via cursor movement. Simply moving the cursor over a line without changing any data therein does not constitute a modification to the line. However, placing any character on the line (including a blank) does constitute a modification (even if the data is subsequently erased). If a modification is inadvertently made, the Type III "D" (DELETE) command may be used to prevent the line from being entered in the file. If Type III areas are not present on the screen, a line is considered modified if it "appears" different than when originally written. Thus, if data is placed on a line and subsequently erased, that line will not be considered modified.

After new data has been "inputted", the current line will be set to the last line entered. Therefore, if the "command-hold" feature is being used, continuous inputting of lines can be effected.

INSERT

INS <c>

The INSERT command is functionally identical to INPUT except that at the conclusion of the command the current line will remain at its present position (will not be set to the last line entered).

LEFT

LE <n>

Frequently, it is necessary to "view" columns of data which are not currently on the screen. The selective column "viewing" facility may be used to accomplish this (see Type II command: VIEW) or the user may simply adjust the data left or right as desired. By specifying LEFT the columns currently in view will be shifted one position to the left. No data loss occurs and the user may simply enter the RIGHT command to cause data to be shifted back into its original position. "n" specifies the number of columns to be shifted to the left (or "right") and defaults to one if not specified. The data will not be shifted if the requested shift would cause the total offset (total shift magnitude) to exceed the maximum record length of the current level.

LOCATE

*L /string1/ <z1 | * <z2 | *>> < /string2/ > ...*

The LOCATE command may be used if the location of a desired line is not known. A specified string of characters or multiple strings of characters are entered as parameters to the LOCATE command. This will cause a search, starting at the line immediately following the current line, for one containing all of the requested character strings within the currently defined "zones" (see Type II command: ZONE). The search ends when such a line is "located", and the line found then becomes the current line. Optionally, over-riding zones may be specified ("z1" and "z2"). These zones function in exactly the same fashion as those defined by the ZONE command but are in effect only for the immediately preceeding character string. If an over-riding zone is specified as "*", the corresponding permanent zone will be used. If no overriding zones are specified for the last character string, and the delimiting character is "/", the second delimiter may be eliminated. Similarly, if no overriding zones are supplied for the last character string, it is not necessary to enter any delimiter. If only the starting delimiter is entered, in this case, it will be considered a delimiter (and not part of the data string) only if it is a "/"

negative LOCATE

*L ¬string1¬ <z1 | * <z2 | *>> < ¬string2¬ > ...*

The negative locate is a feature of the basic LOCATE command and may be used to locate a line in which a specified string of characters does "not" occur. Specified strings of characters are entered as in the basic LOCATE command. However, by using the logical "not" symbol (¬) as a character string delimiter, a search for a line in which the delimited string does "not" occur is initiated. "Negative" and "normal" strings may be combined in one issuance of the locate command. Only a line in which ALL conditions (positive and negative) of the LOCATE command are met, will cause termination of the search. The selected line then becomes the "current line".

All of the characteristics which apply to the slash ("/") symbol regarding LOCATE, pertain as well to the logical not (¬) symbol.

MOVE

MOVE n | .XXX | TOF / < n | .XXX | / < n | .XXX | / >>

The MOVE command parameter list is identical to that of the COPY command. The difference between MOVE and COPY is that data "moved" is deleted from its old position whereas data "copied" is not deleted. (For an explanation of the MOVE command parameter list refer to the Type II command: COPY). Note that when "moving" lines, no additional virtual storage is required. Thus lines may be moved even when a "virtual storage full" condition has already been encountered.

PFn

PFn string1

The PFn command establishes 'string1' as the meaning of Program Function Key (n) for the current logical display. Specifying the (PFn) command with no operand causes the PF Key meaning to be reset. (See also the DSPF command in this section).

POINT

P < .XXXX | :XXXX <DELETE>>

POINT is the primary command for the "alphameric pointer system" which may be used while editing. Using this system of line referencing, it is not necessary to remember the "line number" of a particular line in the file. Using the POINT command, a "name" may be assigned to the desired item which, unlike "line numbers", remains assigned to that item throughout the editing session (unless specifically re-assigned) regardless of editing changes. This name may be used in most commands which require referencing a particular item. Using line "names", a line may always be located no matter what its location simply by its name. A line name must always begin with a "point" (.) or a "colon" (:). The name may be any 1-4 characters which can be entered from the 3270 keyboard. If the POINT command is issued with a name, the specified name is assigned to the current line. If the name has already been assigned to a different item, it is re-assigned if the name begins with "point" (.). Names beginning with a "colon" (:) can not be reassigned. If entered with no parameters, the POINT command will cause all names assigned to the current line to be displayed on the command line. If the POINT command is issued with a "name" followed by the keyword "DELETE" (which may be abbreviated to "D"), the specified "name" will be deleted from the index (regardless of the line to which it currently pertains). NOTE: Any given item may have as many names as desired (limited only by the amount of virtual storage available for storage of same). However, when displaying names assigned to the current line only as many names will be displayed as will completely fit on the command line.

PUTFILE

*PUT <Fn | * <Ft | * <Fm | * <n | .XXX | * <REPLACE <APPEND>>>>>>*

The PUTFILE command will cause one or more lines of the current file to be written to disk starting with the current line. These lines will be placed in the file indicated by Fn, Ft, and Fm. The file name, type, and mode may be specified explicitly or as "*" (the default) indicating that the corresponding identifier of the file at the current edit level is to be used. Writing begins at the current line and continues: for "n" lines where "n" is any positive integer not greater than the number of records in the current file; or through ".XXX" where ".XXX" is a previously defined "named" line; or through "END-OF-FILE" if "*" has been specified. If no limit is specified, only the current line will be placed in the specified file. If the requested file already exists, an error message will be displayed and the PUTFILE will not be performed, unless the REPLACE option (this may be abbreviated to "R") has been specified If the keyword "APPEND" (this may be abbreviated to "A") is supplied, the specified data will be "appended" to the requested file.

RECFM

RECFM < F | V >

The RECFM command is used to set the current record format characteristic for the file at the current edit level. "F" indicates that the file is to contain fixed-length records and "V" indicates variable-length records. The record format is significant only when the file is written to disk (on a SAVE, FILE, or PUTFILE request or during an "AUTOSAVE"). At that time, if the record format is "V", trailing blanks are removed from all records written to disk. If the RECFM command is issued without any arguments, the current setting is displayed on the command line.

REPEAT

*REPEAT < n | .XXX | * < c > >*

The REPEAT command is used to "repeat" any changes made to the current line on succeeding lines. If no operand is specified, the change is repeated only once. Using the operand, the change may be repeated: "n" times where "n" is any positive decimal integer not greater than the total number of items in the file; or through ".XXX" where ".XXX" is a previously defined "named" line located between the current line and "END-OF-FILE"; or through "END-OF-FILE" (indicated by "*"). Frequently, it is necessary to cause a character position in the current line to be repeated without being changed. This is considered a "logical overlay". If a "logical overlay" is desired, a logical overlay character ("c") may be provided with the REPEAT command. The original contents of all positions in the current line in which this character has been inserted, will be "overlayed" on the requested succeeding lines. Therefore, the character selected must be unique to all changed data in the current line. Note that the specification of the logical overlay character pertains only to this issuance of the REPEAT command. If the command is invalid or the overlay character is inadvertently not specified, the overlay character is considered a modification to the data, and will remain in the working copy of the file.

REPLACE

*REP < n | .XXX | * >*

The REPLACE command is used to "replace" lines in the current file. It is functionally equivalent to performing a DELETE, followed by a "DOWN 1" followed by an INSERT. Thus, REPLACE causes the lines specified (as in the DELETE command) to be deleted (starting with the current line), after which the current line is backed-up to the line immediately preceding the first deleted line, and INSERT is automatically entered at this point. The arguments for the REPLACE command are equivalent to those used for the DELETE command (see Type II command: DELETE).

RIGHT

RI <n>

The RIGHT command is functionally similar to LEFT. Its purpose is to cause a display shift to the right (the actual data is not shifted). For details on use of the RIGHT and LEFT commands see the Type II command, LEFT, in this section.

SEARCH

*S /string1/ <z1 | * <z2 | *>> < /string2/ > ...*

The SEARCH command is functionally similar to LOCATE. However, if LOCATE does not find the requested line by "EOF", it will terminate and an error message ("EOF REACHED") will be displayed; whereas, if the requested item is not found by SEARCH when "EOF" is encountered, a wrap-around to the top of the file will occur. Thus, using SEARCH, the error message will not be produced. At the conclusion of the SEARCH command, if the requested item has not been found, the line pointer will remain at the current line and the message ("DATA NOT FOUND") will be displayed. The SEARCH command may be issued without the command verb (ie: "/string1/" alone).

SERIAL

SER <ON | OFF | ALL | XXX <10> >

The SERIAL command may be used to specify automatic sequencing of records within a file. Sequencing is significant only when the file at the current edit level is written to disk and then only if the file contains fixed-length, 80-character records.

With "SERIAL ON" the first three characters of the file name are placed in columns 73-75 of all records and columns 76-80 are sequenced according to the increment value. With SERIAL OFF, no sequencing is performed. Optionally, SERIAL XXX may be specified where "XXX" is any three characters to be placed in columns 73-75 of each record before being written to disk. Columns 76-80 are again sequenced according to the increment value. When SERIAL ALL is specified, all eight columns (73-80) are sequenced (no prefix characters are used). If the SERIAL command is issued without any arguments, the current setting is displayed on the command line.

SET

```
SET  < Dblank     ON  | OFF >
     < Nulls      ON  | OFF >
     < Autosave   n   | OFF >
     < NUMbers    ON  | OFF >
     < NOread     ON  | OFF >
     < Convert    ON  | OFF >
```

The SET command is used to vary settings of certain optional functions of EDGAR while editing is in progress. The SET command is followed by a keyword indicating which option is to be set. The keyword must be followed by the new setting. If the SET command is issued with no arguments the valid keywords and their current settings will be displayed. As many options as desired may be set in one command. However, if any errors are encountered while processing any of the options, no setting changes are performed. The keywords and valid settings for each are described below:

NUMBers - This option causes display of the "line numbers" within Type III Command Areas. Normally, this area will contain a symbol ('*===*' or '/===/') over which the user enters the desired Type III command. With the setting, NUMBERS ON, the Type III command areas will, instead, contain an "*" or '/' (if the "current line") followed by the item number of the corresponding data item in the working copy of the file. Since these numbers are continually re-calculated, substantial overhead can be created by requesting display of numbers. The amount of overhead will depend on the size of the file and the position of the "current line" within the file. Therefore, 'SET NUMBERS ON' should be issued only when necessary and when system overhead is non-critical.

DBLAnk - This option causes EDGAR to eliminate any blank records before writing a file to disk. DBLANK may be SET ON or OFF. With DBLANK ON, totally blank records contained within the working copy of the file (in storage) are not written to disk. These records, therefore, will not be serialized (if serialization is in effect) but will remain as blank records in the working copy of the file. The DBLANK option is especially useful, for example, in ASSEMBLE files where lines have been "ADDED" (using the ADD command). Since such records (if they remain in the file) will cause compiler errors, the DBLANK option makes manual deletion of excess "ADDED" records unnecessary.

NULLs - This option causes the I/O Processor to replace any trailing blanks within each 3270 field of the edit level virtual screen with "null" characters before transmitting the data to the physical display. Null characters are useful when adding characters to a data record using the "insert mode" feature of the 3270 system. Since only null characters may be shifted out of a line (using "insert mode"), setting "NULLS ON" allows the user to insert characters until the first non-blank reaches the end of the data-display area.

***NOTE: Users should refrain from setting NULLS ON when Type III command areas are not present in the logical display. In this case, nulls are added only at the end of the last data-display area resulting in a shift of all data-display lines when "insert mode" is used. This will lead to unpredictable editing results.

NOREad - This option determines the effect of the SOS ATTN command. With NOREAD OFF, the ATTN command is functionally equivalent to a "null line" in the stack. With NOREAD ON, the ATTN command is functionally equivalent to SOS ENTER. For a full discussion of this subject, refer to Section V (Advanced Editing Facilities, Screen Operation Simulation).

CONVert - This option permits data description within commands (such as CHANGE, LOCATE, etc) to use hex or binary conversion notation. With CONVert set ON, for example, the command L/X'F0F0F0'/ would have the same meaning as L/000/. Similarly, the command L/B'11110000'/ would have the same meaning as L/0/.

AUTOsave - This option invokes the "Autosave" facility. It may be set "OFF" or an alteration threshold value may be supplied. The threshold value must be a positive decimal integer not greater than 32767 and indicates the alteration threshold at which "AUTOSAVE" will take place. Thus, "SET AUTOSAVE 1" will cause the file to be checkpointed every time at least one alteration is made to the working copy. For details on the "Autosave" facility see Section IV, AUTOSAVE FACILITY.

SHIFT

*SH < Left | Right < n < m | * | .XXX < c > > > >*

The SHIFT command causes data within the currently defined ZONE (See ZONE command in this section) to be shifted left or right a distance of (n) columns. The third operand dictates the number of lines effected starting with the current line. The user may specify a number of lines (m), to End of File (*), or to a named line (.XXX). Optionally, a "fill" character (c) may be specified. In this case, the specified character will be inserted in all columns "from" which data has been shifted. Note that SHIFT operates only on data within the currently defined ZONE (see ZONE command in this section).

STACK

*STACK <LIFO | FIFO> <n | .XXX | * <string 1>>*

The STACK command causes data from file items to be placed in the CMS console stack. Stacking will be FIFO unless LIFO is specified. The number of lines to be treated is specified by "n", ".XXX", or "*" (thru EOF). An additional character string, "string1", will be stacked after each item in the file, if specified. STACK operates only on the currently ZONEd columns.

TABS

T < n1 n2 n3 n4 n5 n6 ...... >

The TABS command is used to establish tabbing columns for all data-display areas in the corresponding logical display. The arguments to the TABS command are the desired tab columns. Each tab position represents one column. The list must be in ascending order and each number must be a positive decimal integer not greater than the maximum record length of the file at the current edit level. A maximum of 25 tab positions may be set. If specified with no arguments the TABS command will display the current tab settings on the command line. Once tab settings have been established the user may use the tabbing feature to achieve cursor movement. With the standard options "PF1" requests a backward tab and "PF2" a forward tab (the display station must be equipped with these keys to utilize the tabbing facility). Effective tabbing with this facility is dependent on system response time since the tabbing mechanism is under software control. Under average to good response conditions, this tabbing feature will be faster than the cursor movement cluster keys on the 3277. However, depression of the "PF" keys for this purpose causes transmission of all "modified" data on the 3277 screen to the CPU. This may be an unacceptable condition on a remotely attached device if several data-display lines (or command lines) have been altered prior to depressing the tab keys, unless high-speed line transmission is utilized.

TOP

TOP

"TOP" positions the current line to the first item in the file. There are no arguments to the TOP command. If no items exist in the file, the current line is set to "TOF".

UFIND

UF string1

The UFIND command is functionally identical to FIND except that the file scan is in an upward direction.

ULOCATE

*UL /string1/ <z1 | * <z2 | *>> < /string2/ > ...*

The ULOCATE command is functionally equivalent to LOCATE except that the file scan is in an upward direction.

UP

U <n>

UP causes the screen to appear to move "up" "n" lines. When UP is issued and the current line is "EOF", no movement takes place. UP allows the user to view lines further toward "EOF" in the file. If no parameters are supplied with the UP command "UP 1" is assumed. The argument specifies the number of lines (items) by which the current line is to be moved (forward in the file). This number must be a positive decimal integer not greater than the total items currently in the file.

USEARCH

*US /string1/ <z1 | * <z2 | *>> < /string2/ > ...*

The USEARCH command is functionally equivalent to SEARCH except that the file scan is in an upward direction.

VIEW

VIEW < <Rn | Ln> <<H>nA1 nA2 <H>nB1 nB2 <H>nC1 nC2...>>

The VIEW command is the primary command for the selective column viewing feature of EDGAR. Using the VIEW command the user may select, for viewing, only those columns which are necessary for the editing to be performed. Thus, pertinent information in records of greater width than the current display may be viewed. VIEW specifications are supplied in pairs of positive decimal integers each of which must not exceed the maximum record length for the file at the current edit level. Each pair designates a "viewing field". The first number of the pair indicates the "starting" column of the desired field, and the second the "ending" column. Within each pair the "ending" column must be equal to or greater than the "starting" column. However, view fields (successive pairs) may be in random order. Furthermore, view fields may overlap or be equal. Each field will be displayed (one following another) in the data-display area beginning at the left margin. Immediately preceding the "starting" column designation with the letter "H" causes that field to be displayed in HEX format. The total width of all "viewed fields" may not exceed the width of the data-display areas. Specification of an offset value is optional. If supplied, the offset must be the first argument to the VIEW command. The offset value is the magnitude (left or right) by which the selected fields are "offset" from their specified "start" columns. This must be designated as "R" (for RIGHT) or "L" (for LEFT) immediately followed (no intervening space) by the offset value. Thus, "R5" indicates that the selected view fields are to be offset from their specified starting columns by 5 postions to the right. This is identical to providing the view fields followed by a "RIGHT 5" command. When the VIEW command is issued with no arguments, the current view fields and the current offset value (if any) are displayed in the Type II command area. Note: it is possible for truncation to occur on such a display, if the column numbers currently in view plus the offset value number exceed the display width of the Type II command line. Use of the display option is valuable if the user does not know the exact current offset value but wishes to restore the display to its normal (no offset) positon. By issuing the VIEW command with no arguments (creating a display) and then erasing the offset value from the command, a zero offset will be re-established.

ZONE

Z < n1 < n2 >>

ZONE is used to establish the current column limits for the STACK, SHIFT, SEARCH, FIND, CHANGE and LOCATE commands. If specified with no parameters, the current zone settings are displayed on the command line. To alter the current settings enter the ZONE command specifying the new "START" and "END" zone columns. Each must be a positive decimal integer not greater than the maximum record length of the file at the current edit level. The "END" zone must be equal to or greater than the "START" zone. Optionally, the "END" zone setting may be omitted in which case the current "END" zone value will be used.

.XXXX
:XXXX

.XXXX
:XXXX

Once a given item in a file has been "named" via the Type II command, POINT, it is possible to find that line regardless of its current location by entering the "POINT" name, as shown above, on the command line. The name must be 1-4 characters in length and must begin with a "point" (.) or a "colon" (:).

$XXXX

$XXXX < parms >

An EXEC can be entered from an EDGAR session by entering the $ command where XXXX is the EXEC name. This command may be followed by whatever PARM's the user wishes to have passed to the EXEC. For further information about EDGAR EXEC's refer to Section V, Advanced Editing Facilities.

nnnn	*nnnnn* Line callout by item number is possible as a Type II command. This number must be the current item number of the desired line in the working copy of the file. This must be a positive decimal integer not greater than the total items in the file.

Type III Commands

Type III commands are the basis for "in-line" editing functions, i.e., functions pertaining to an individual line within a given logical display. Type III commands do not necessarily reference or alter the "current line" pointer. Each data-display area within a given logical display may contain a Type III command area in the last 8 positions (unless deleted by the Type I command: FORMAT). Type III commands are "overlayed" on the Type III command areas. The initial contents of each Type III area depends on the contents of the corresponding data-display area as well as user option settings and may be one of the following:

1. the characters "/===/" or "*===*" as illustrated in figure 2, lines 3 and 4, respectively; either of these indicators will appear in the Type III command areas for data-display areas containing file data when NUMBERS have been set OFF (See the Type II command: SET). Since SET NUMBERS OFF is the default setting, these constants will always appear unless SET NUMBERS ON has been issued. "*===*" is provided for all such data-display areas except those containing the "current line", for which the constant "/===/" is used.

2. the characters "/nnnnn" or "*nnnnn" (where "nnnnn" is the line number of the corresponding item within the file. This format is used only when the user has specified SET NUMBERS ON. Although this facility is convenient when item number referencing is utilized, substantial overhead may be incurred in large files when calculation of the line numbers is required. "*nnnnn" is used for all Type III command areas unless the corresponding data-display area contains the "current line" in which case "/nnnnn" is used.

3. the characters "*INPUT" - indicating that the corresponding data-display area is for "input" and has not yet been incorporated in the file. These constants will appear when a display has been created via the INPUT or INSERT commands.

4. the characters "*TOF*" or "*EOF*" - used when the corresponding data-display area represents the imaginary lines at "top-of-file" and "end-of-file", respectively. Type III commands may not be entered on these lines and changes to these Type III areas will be ignored.

Type III commands are single character commands which may or may not have preceding arguments. If the command is of the type which may have a parameter, the parameter must begin in the first position of the command area and must be immediately followed by the command character. No errors are indicated for Type III commands. The user must verify that the command has been successfully executed and re-issue the command, correctly, if not.

Since Type III commands do not produce error messages, no warning of this situation is provided to the user.

A *<n>A*

The "A" command is similar to the Type II command: ADD. It is used to "add" one or more blank lines after the line associated with this Type III command area. (For information on "added" lines refer to the ADD Type II command.) If no argument is provided, one blank lines is "added" to the file. If a parameter is supplied it must be a 1-5 digit, positive, decimal integer indicating the number of lines to be "added".

D *<n>D*

The "D" command is used to delete (in place) items from the file. Its function is similar to the Type II DELETE command and will cause "n" lines (beginning with the line associated with this Type III command area) to be deleted from the file. If no argument is specified, only the associated line will be deleted. The parameter must begin at the start of the Type III command area and must be immediately followed by the "D" command character.

/ */*

The "/" command is used to set the line pointer to the associated line. The "/" should be entered at the start of the Type III command area and has no associated parameters. Its effect is to set the line pointer to the associated line at that time. Since the order of processing for any logical display includes all data-display areas and their associated Type III command areas before the Type I & II command area, the "/" command may be used to set the line pointer ("current line") to a line other than the normal current line for reference by the Type II command. If "/" is issued to more than one line only the last will become the "current" line.

" *<n>"*

The " (double-quote) command allows "in-line" duplication of the associated file item. It is similar, in function, to the Type II command: DUP. The " command causes the associated file item to be replicated "n" times. If "n" is not specified it will default to one. "n" represents the number of duplications to be performed. Thus, if two duplications are requested, three (3) equivalent lines (the original and two duplicates) will appear in-line in the file.

< *.XXXX<*

The "<" command is similar to the Type II POINT command and allows the user to "name" the line associated with this Type III command area for later alphameric line referencing. The argument to the "<" command is the 2-5 character line "name" which must begin with a "." or a ":". (The "point" or "colon" must begin at the start of the Type III command area). Any 4 characters which may be entered from the 3277 display may be used for the remainder of the line "name" (except the character "<", which is treated as the command). If the given name is valid and not previously defined, it is added to the "name" index. If the name has been previously defined and assigned to another line in the file, the action taken is as follows:

1. If the name begins with a ".", it will be re-defined and assigned to the line corresponding to this Type III command area.

2. If the name begins with a ":", no redefinition will take place.

Autosave Facility

EDGAR provides a facility known as "AUTOSAVE" which will automatically checkpoint the working copy of the file. Checkpointing is triggered by the number of alterations made to the file. Therefore, an alteration counter is maintained by EDGAR throughout any editing session. The value of the counter is always displayed in the status area (Line 1) of the screen after the prefix, 'ALT='.

Whenever a new edit level is entered, an "AUTOSAVE-ID" is automatically generated and assigned to that level. This is a 1-8 digit number which will always be unique to the current editing level. The "AUTOSAVE-ID" for any level is always displayed in the status area (Line 1) of the screen bracketed in parentheses. In the example in figure 2, the "AUTOSAVE-ID" is "1". The "AUTOSAVE-ID" becomes the file name of the checkpoint file created for that particular edit level. The file type is always "AUTOSAVE" and the file mode is (X)1 where X represents the disk on which the checkpoint file is written. The file is always written to the R/W disk with the greatest amount of available space. If no R/W disk is 'ACCESSED' or if there is insufficient space to create an "AUTOSAVE" file, or if errors occur while writing the file to disk, the 'AUTOSAVE' facility is cancelled. The AUTOSAVE file is written to disk in the same manner as a FILE or SAVE request (see the description of writing files to disk in Section I: "INTRODUCTION"). Before renaming the file to the correct AUTOSAVE name, any existing AUTOSAVE files (on any disk) having the same 1-8 digit file name are erased. Thus, at any one time, there will be no more than one copy of an AUTOSAVE file containing the unique AUTOSAVE-ID. In the event of system failure the user need only restart the editor to the check-point file and change the file-id while editing (see information on the "FILE-ID" area in Section III.)

Whenever a normal exit is taken from any edit level, any AUTOSAVE files having the unique AUTOSAVE-ID corresponding to that level are automatically erased (from any disk) to prevent the user's disks from becoming overrun with AUTOSAVE files.

To invoke the AUTOSAVE facility, the user must issue the SET AUTOSAVE Type II command (see SET command in Section IV). Normally AUTOSAVE is set "OFF". However, by setting AUTOSAVE to the desired threshold alteration count, checkpointing of the file will occur whenever the alteration counter reaches the specified threshold. To display the current threshold value, issue the "SET" command with no arguments.

V. Advanced Editing Facilities

Screen Operation Simulation

SOS, Screen Operation Simulation, is an EDGAR facility by which the user can automate the operation of the 3270 keyboard. By executing a set of commands, the user can request that SOS simulate the depression of 3270 keys and cause changes to occur on the I/O virtual screen.

SOS has only one input device, the CMS console stack. (A working knowledge of the CMS console stack is a prerequisite for using SOS. See VM/370 CMS User's Guide, GC20-1819). Data may be placed into the stack in several ways. The most commonly used methods are as follows:

- Depressing a PF Key - Data which is assigned to a PF key is placed LIFO into the stack when the PF key is depressed
- Executing an EXEC - An EXEC or CMS application program which places data into the CMS console stack may be executed during an editing session. EDGAR may be invoked from within an exec which had previously placed data into the stack.
- Executing the STACK Editing Command - The STACK command will allow the user to place data records and, optionally, related data into the CMS console stack. NOTE: When reading data from the console stack, CMS considers a X'15' to be a line end character. Thus if the user desires to put multiple lines into the stack with only one statement, he should imbed X'15's (or characters that will be translated to X'15's via the CMS input translate table) in his data.

When SOS is given control it reads the top entry in the console stack and breaks it up into one or more logical lines which are processed one at a time, the unprocessed logical lines remaining in the stack. The start of logical line is defined as the first position of the stack entry. (This position may, in fact, have resulted from the processing of the previous logical line). A logical line ends either with the termination of a valid SOS command and its operands, with the end of the stack entry, or with a X'15' (the CMS line end designation). Trailing blanks are always deleted as the logical line is processed. In addition, if the last non-blank character is an exclamation point (!), it is also deleted. Thus if a line is placed into the stack as follows (with its start and end indicated by a bar (|)):

```
/sos tabcmd abc       !
|                     |
```

the logical lines are defined as follows:

```
/sos tabcmd
|          |
abc        !
|          |
```

and the resulting data to be processed is:

```
/sos tabcmd
|          |

abc
|          |
```

The SOS facility recognizes only two types of data; SOS commands and data to be typed on the I/O virtual screen. As each logical line is obtained from the CMS console stack, a check is make to determine if it contains a valid SOS command. A valid SOS command is defined as one which starts in position one (1) of the logical line, begins with the characters "/SOS ", and has as its first operand a recognized SOS command. If the logical line does contain a valid SOS command, the rest of the line is returned to the CMS console stack and the command is executed. If the logical line does not contain a valid SOS command, it is considered to be data and is placed on the I/O virtual screen at the current position of the virtual cursor. The cursor position is then incremented once for each character so inserted.

SOS continues to process the CMS console stack until the stack is empty or it reads a "null" entry (an entry with zero length). At that time, it relinquishes control of the I/O virtual screen which is subsequently written to the physical display device.

Note: It is important for the user to recognize that changes made to the I/O virtual screen by the SOS facility have exactly the same effect as those made by the user on his display terminal. In fact, except for the I/O Processor, the editing system has no knowledge of the source of changes to the I/O virtual screen.

SOS Commands

All SOS commands are prefixed by the characters "/SOS ". The description of each command, below, conforms to the following format: uppercase characters indicate the shortest allowable form of the command, while the lowercase letters complete the command.

ALarm	Ring the terminal alarm when, and if, this screen is written to the physical display terminal.
Attn	If the NOREAD setting is on, perform the same function as /SOS ENTER. If the NOREAD setting is off, perform the same function as a "null" entry in the stack (display the virtual screen on the physical display terminal).
BField n	Reposition the cursor as per the back field key. The key is to be depressed n times. The default is one (1).
CLEAR	Clears the CMS console stack and thus will terminate SOS activity and display the virtual screen on the physical terminal.
CONTinue	Implies processing is to continue.
Down n	Move the cursor down n lines on the virtual screen. The default is one (1).
DELete n	Delete n characters at the position of the cursor from the 3270 field. The default value for n is one (1).
ENTER	Depress the ENTER key. This will cause the I/O virtual screen to be mapped to the active level virtual screens and thus processed by the Display Processor. It is not until this key is depressed that any changes can be made to the data file.
ERAseof	Depress the ERASE EOF key. Erase the data until the end of this 3270 screen field.
ERRor Continue	If an error message is found on an edit level virtual screen, ignore it and continue processing
ERRor VERIFY	If an error message is found on an edit level virtual screen, issue the SOS "error encountered" message and terminate SOS activity (do not purge the console stack however).
ERRor ESCAPE	If an error message is found on an edit level virtual screen, issue the SOS "error encountered" message, terminate SOS activity, and purge the CMS console stack.
EXec xxx p r	Execute exec xxx and pass to it the parameters p and r.
INsert	Insert turns on insert mode. Insert mode will be on for the duration of one (1) logical line (the next one). After that an automatic reset is issued. Data may be inserted only within fields for which NULLS have been set on.

Left n	Move the cursor n places to the left. The default is one (1).
NField n	Move the cursor to the next 3270 field n times. The default is one (1).
NLine n	Depress the "new line" key n times. The default is one (1).
NUlls	Reverse the current setting of the "nulls" option for the 3270 field in which the cursor is located.
NUlls On	Change the trailing blanks to "nulls" for the 3270 field in which the cursor is located.
NUlls OFF	Change the trailing "nulls" to blanks for the 3270 field in which the cursor is located.
PA1	Depress the PA1 key. (Call CP).
PA2	Depress the PA2 key. (Call the CMS subset).
PFn	Depress a PF key where n is the key number 1-12. The data which had been assigned to the key is placed LIFO in the CMS console stack.
POP	Remove the top position from the cursor stack and place the cursor there.
PUsh	Save the current position on the I/O virtual screen of the virtual cursor. The position is saved in a LIFO fashion in a special internal stack of EDGAR. This stack has room for five (5) positions before it "forgets" the oldest entry.
Right n	Move the cursor n positions to the right on the virtual screen. The default is one (1).
TABB n	Move the cursor backward to the previous "tab" position as indicated by the TAB editing command. The "tab" operation may be performed n times with one (1) being the default.
TABCmd	Position the cursor at the command line for the logical screen in which it currently resides.
TABCMDB	Move the cursor in a backward direction to the first encountered command line.
TABCMDF	Move the cursor in a forward direction to the first encountered command line.
TABF n	Move the cursor forward to the next "tab" position as indicated by the TAB editing command. The "tab" operation may be performed n times with one (1) being the default.
Up	Move the cursor up n lines. The default value for n is one (1).

User Written Commands

The SVC Interface

In addition to communicating with the Display Processor (see Section III, System Overview), the Editing Command Processor of EDGAR can communicate with a user written program via an SVC. The SVC which has been chosen for this purpose is X'ED'. A user program which desires to use this SVC should point general purpose register one (1) to a 132 byte data area and issue the SVC. The data area will be processed by the Editing Command Processor as a command line. Upon return from the SVC, general purpose register 15 contains a return code as follows:

0	- indicating the command executed successfully
-1	- indicating the command executed successfully and produced some informational data. The data has been placed in the data area pointed to by general purpose register 1.
1	- indicating that the command cannot currently be accepted.
xxx	- a message number corresponding to an EDGAR error message indicating the unsucessful execution of the command. The error message has been placed in the data area pointed to by general purpose register 1.

NOTE: Only programs which have been invoked as a result of issuing a CMS command on an EDGAR command line, or programs invoked from a "DEFAULT" EXEC, may issue the EDGAR SVC. If the program is not invoked in this fashion, a return code of one is returned in GPR 15.

ECOMMAND

ECOMMAND is a CMS problem program which runs in the CMS SUBSET and uses the EDGAR SVC interface. ECOMMAND takes its input either in the form of entry parameters entered on the same line, or from the console. If the input is entered as parameters and the user wishes to avoid the tokenizing normally done by CMS, he may "wrap" the data, which he wishes to remain contiguous, with parentheses and enter it in the normal eight (8) byte groupings. For example

```
ECOMMAND L (/bABC/)
```

ECOMMAND will pass its input to the EDGAR editing command processor which will treat it as an edit command to be executed against the currently active edit level file. The result of the edit command processing will be returned to the user in the form of a return code and, possibly, a message line placed LIFO in the CMS console stack. The user may inhibit the placement of the message line in the CMS console stack by executing ECOMMAND while & CONTROL is set to NOMSG. The possible return codes which can be returned from ECOMMAND are:

0	- indicating the command executed successfully
-1	- indicating the command executed successfully and produced a message which would normally be placed on the command line of the screen and has been placed LIFO in the CMS console stack.
1	- indicating that EDGAR is not active
xxx	- a message number corresponding to an EDGAR error message indicating the unsucessful execution of a command. The error message is also placed LIFO in the CMS console stack if the command was called via SVC.

VI. ERROR MESSAGES

EDIMSG500E	INSUFFICIENT STORAGE FOR INITIALIZATION.
EDIMSG501E	INTERNAL DATA GENERATION ERROR; RESTART EDGAR
EDIMSG502E	VIRTUAL CONSOLE IS NOT A 3277.
EDIMSG503E	RECEIVE FORMAT ERROR '...' FROM DISPLAY STATION
EDIMSG504E	INVALID CHARACTER RECEIVED FOR TERMINAL MODE
EDIMSG505E	INSUFFICIENT STORAGE; CANNOT RE-FORMAT.
EDIMSG508E	INCOMPLETE FILEID SPECIFIED
EDIMSG509E	FILENAME OR FILETYPE EXCEEDS EIGHT (8) CHARACTERS
EDIMSG510E	FILEMODE EXCEEDS TWO (2) CHARACTERS
EDIMSG511E	MISSING OR INVALID RECORD LENGTH SPECIFICATION
EDIMSG512E	RECORD LENGTH '........' EXCEEDS ALLOWABLE MAXIMUM OF 2047.
EDIMSG513E	INVALID CHARACTER IN FILE ID: ".................."
EDIMSG514E	INVALID FILEMODE
EDIMSG515E	FILE '....................' NOT FOUND
EDIMSG516E	DISK '..' NOT ACCESSED
EDIMSG517E	RECORD '.....' IS PAST EOF (.....)
EDIMSG518E	FILE "...................." EXCEEDS EXISTING RECORD LENGTH.
EDIMSG519E	ERROR '.....' READING FILE "...................." FROM DISK
EDIMSG520E	TYPE I POST-PROCESSING FAILED; EDIT LEVEL: "................."
EDIMSG521E	CREATING NEW FILE:
EDIMSG526E	"........" IS NOT A VALID SUBSET COMMAND
EDIMSG527E	SUBSET ALREADY ACTIVE.
EDIMSG528E	STACKED LINES CLEARED ...
EDIMSG529E	RETURN CODE '......' FROM "........"
EDIMSG522E	CURRENT LINE IS 'EOF' OR 'TOF'.
EDIMSG530E	DISK '..' IS READ/ONLY OR NOT 'ACCESSED'
EDIMSG531E	DISK '..' IS FULL
EDIMSG532E	DISK '..' HAS NOT BEEN FORMATTED
EDIMSG533E	ERROR '.....' WRITING TO FILE ".................."
EDIMSG534E	CANNOT APPEND .... TO .... LENGTH RECORDS
EDIMSG535E	FILE "...................." IS OPEN FOR INPUT
EDIMSG536E	FILE "...................." HAS UNEQUAL RECORD LENGTH
EDIMSG540E	STORAGE FULL - FILE ADDITIONS INCOMPLETE.
EDIMSG541E	INVALID "REPEAT" COUNT - ....
EDIMSG542E	NO CHANGES ON "CURRENT" LINE - REPEAT IGNORED.
EDIMSG543E	INVALID LOGICAL OVERLAY CHARACTER.
EDIMSG550E	UNKNOWN COMMAND - "........"
EDIMSG551E	EXCESSIVE PARAMETERS FOR "........" COMMAND
EDIMSG552E	UNKNOWN "........" OPTION - "........"
EDIMSG553E	CURRENT LINE HAS NO NAME.
EDIMSG554E	POINT "....." ALREADY ASSIGNED
EDIMSG555E	ILLEGAL "POINT" NAME.
EDIMSG556E	NAMED POINT EXCEEDS FOUR (4) CHARACTERS.
EDIMSG557E	NAMED POINT '.....' IS UNDEFINED
EDIMSG558E	INSUFFICIENT STORAGE - POINT LIST FULL.
EDIMSG560E	FILE '....................' ALREADY EXISTS
EDIMSG561E	TOO MANY RECORDS; LEVEL CANNOT BE STORED.

EDIMSG562E	EMPTY EDIT LEVEL; LEVEL CANNOT BE STORED.
EDIMSG563E	EXISTING LEVEL - .... SPECIFICATION IGNORED
EDIMSG564E	REQUESTED FORMAT EXCEEDS SCREEN CAPACITY
EDIMSG565E	INVALID "FORMAT" SPECIFICATION.
EDIMSG566E	DISPLAYS ACTIVE - CANNOT RE-FORMAT.
EDIMSG567E	INVALID SERIALIZATION .... SPECIFIED
EDIMSG568E	UNKNOWN "SET" FUNCTION - ....
EDIMSG569E	"........" MUST BE SET "ON" OR "OFF"
EDIMSG570E	ILLEGAL "VIEW" COLUMN SPECIFIED.
EDIMSG571E	START COLUMN '....' EXCEEDS END COLUMN '....'
EDIMSG572E	..... COMUMN "...." EXCEEDS RECORD LENGTH (....)
EDIMSG573E	REQUESTED VIEW FIELDS EXCEED DISPLAY WIDTH (....)
EDIMSG574E	INVALID TOTAL LINES SPECIFIED FOR "........" REQUESTED
EDIMSG575E	ILLEGAL OFFSET VALUE.
EDIMSG576E	TOTAL OFFSET EXCEEDS MAXIMUM RECORD LENGTH.
EDIMSG577E	SHIFT MAGNITUDE EXCEEDS CURRENT ZONE WIDTH (....)
EDIMSG579E	INVALID CHARACTER '..' ENCOUNTERED IN IMPLICIT DATA
EDIMSG580E	.... PREFIX STRING MUST BE 1-8 CHARACTERS
EDIMSG581E	ILLEGAL POINTER MOVEMENT VALUE.
EDIMSG583E	ILLEGAL ITEM REFERENCE VALUE - "........"
EDIMSG584E	EXCESSIVE PARAMETERS ON LOCATIVE COMMAND.
EDIMSG586E	DISPLAY TOO SMALL TO .... - PLEASE RE-FORMAT
EDIMSG587E	SHIFT DIRECTION INCORRECTLY SPECIFED.
EDIMSG588E	CANNOT "REPLACE" AND "APPEND".
EDIMSG589E	POINT "....." NOT FOUND BEFORE 'EOF'
EDIMSG590E	NO SEARCH ARGUMENT SPECIFIED.
EDIMSG591E	ILLEGAL ..... ZONE COLUMN
EDIMSG592E	ZONE ..... COLUMN (........) EXCEEDS RECORD LENGTH (......)
EDIMSG593E	START ZONE (....) EXCEEDS END ZONE (....)
EDIMSG594E	STRING ... EXCEEDS ZONE WIDTH (....)
EDIMSG595E	NOT FOUND - EOF REACHED.
EDIMSG596E	INVALID HORIZONTAL ...... COUNT - "........"
EDIMSG597E	DATA NOT FOUND.
EDIMSG598E	DATA TRUNCATION HAS OCCURRED.
EDIMSG599E	EOF REACHED.
EDIMSG600E	ILLEGAL TAB SPECIFIED - "........"
EDIMSG601E	TAB "........" EXCEEDS MAXIMUM RECORD SIZE
EDIMSG602E	TAB "...." NOT GREATER THAN PREVIOUS TAB (....)
EDIMSG603E	NUMBER OF TABS REQUESTED EXCEEDS MAXIMUM.
EDIMSG604E	INVALID "AUTOSAVE" COUNT.
EDIMSG605E	INSUFFICIENT R/W SPACE FOR CHECKPOINT - "AUTOSAVE" CANCELLED.
EDIMSG606E	CHECKPOINT TAKEN TO FILE "...................."
EDIMSG607E	'RECFM' MUST BE "F" OR "V".
EDIMSG608E	'CASE' MUST BE "M" OR "U".
EDIMSG609E	NOT FOUND - TOF REACHED.
EDIMSG610E	RANGE UPPER LIMIT PRECEEDS LOWER LIMIT.
EDIMSG611E	TARGET RECORD IS WITHIN RANGE OF LIMITS.
EDIMSG612E	TARGET NOT SPECIFIED.
EDIMSG613E	INVALID .... ITEM REFERENCE NUMBER
EDIMSG614E	END RECORD PRECEEDS START RECORD.
EDIMSG617E	UPPER LIMIT FOR "PUT" NOT FOUND BEFORE 'EOF'.
EDIMSG619E	INVALID LOGICAL DISPLAY SPECIFIED - ....

EDIMSG620E	LOGICAL DISPLAY "........" DOES NOT EXIST
EDIMSG621E	PLEASE SPECIFY LOGICAL DISPLAY NUMBERS TO BE COMPARED
EDIMSG622E	LOGICAL DISPLAY ..... IS ......
EDIMSG623E	EOF REACHED ON DISPLAY .....
EDIMSG624E	COMPARISON FAILED ON DISPLAY ....
EDIMSG625E	FILE "...................." IS NOT FIXED, 80-CHARACTER RECORDS
EDIMSG626E	SEQ. ERROR INCURRED - SEQUENCE NUMBER IN ERROR: ....
EDIMSG627E	INVALID UPDATE CONTROL CARD IN FILE "..............
EDIMSG628E	UPDATE CONTROL CARD SEQ. ERROR IN FILE "........"
EDIMSG629E	"UPDATE" OPTION INVALID - SOURCE FILE MISSING.
EDIMSG630E	CANNOT SEQUENCE OUTPUT....EXCESSIVE RECORDS.
EDIMSG631E	CREATING NEW UPDATE FILE:
EDIMSG632E	ERROR IN .... FILE: "................."
EDIMSG640E	PF NUMBER IMPROPERLY SPECIFIED
EDIMSG641E	INSUFFICIENT STORAGE TO PERFORM PF ASSIGNMENT.
EDIMSG700E	PROFILE ERROR; INSUFFICIENT STORAGE FOR PROFILE
EDIMSG701E	FILE "...................." EXCEEDS MAX. 'PROFILE' 'LRECL'
EDIMSG702E	PROFILE ERROR; LINE ..... UNKNOWN PROFILE TYPE - ....
EDIMSG703E	PROFILE ERROR; LINE ..... - IMPROPER 'PF' ASSIGNMENT
EDIMSG704E	PROFILE ERROR; LINE ..... - FILETYPE NOT SPECIFIED
EDIMSG705E	PROFILE ERROR; LINE ..... - INVALID FILETYPE
EDIMSG706E	PROFILE ERROR; LINE ..... - INVALID FILEMODE
EDIMSG707E	PROFILE ERROR; LINE ..... - UNKNOWN FILETYPE OPTION ....
EDIMSG708E	PROFILE ERROR; LINE ..... - INVALID RECORD LENGTH
EDIMSG709E	PROFILE ERROR; LINE ..... - INVALID 'CTL' SPECIFICATION
EDIMSG710E	PROFILE ERROR; LINE ..... - IMPROPER 'DEFAULT' SPECIFICATION
EDIMSG711E	PROFILE ERROR; LINE ..... - IMPROPER 'SYNONYM' SPECIFICATION
EDIMSG712E	PROFILE ERROR; LINE ..... IMPROPER TAB CHARACTER ..
EDIMSG715E	ALLOCATION OF SCREEN LINES IMPROPERLY SPECIFIED: "........"
EDIMSG716E	MAXIMUM PHYSICAL SCREEN CAPACITY EXCEEDED.
EDIMSG720E	CURSOR IS NOT WITHIN THIS LOGICAL DISPLAY.
EDIMSG721E	CURSOR IS NOT ON A VALID DATA COLUMN.
EDIMSG725E	EMPTY EDIT LEVEL; CANNOT "RENUM".
EDIMSG726E	'RENUM' FUNCTION NOT INSTALLED.
EDIMSG727E	INVALID 'RENUM' .... SPECIFIED
EDIMSG728E	....'
EDIMSG729E	'RENUM' NOT PERMITTED WHILE UPDATE OPTION IN EFFECT

VII. Installation Procedures

The DISPLAY EDITING SYSTEM installation tape for CMS contains three (3) files in CMS 'TAPE LOAD' format. There are no tape labels and each file is followed by one tape mark.

File 1 includes all material necessary for installation of the DISPLAY EDITING SYSTEM. Included are the ECOMMAND and EDGAR load modules, a set of sample "DEFAULT" EXECs, a sample "EDGAR $PROFILE" and the EXECs and TEXT files required to perform the installation procedure.

File 2 contains the assembly language source code and object form of the system modules and the macro libraries required to assemble these modules. It is not necessary to load this file for installation or maintenance purposes.

File 3 contains the MACRO and COPY files which comprise the macro libraries included in file 2.

Assembly of the DISPLAY EDITING SYSTEM modules requires the VS H-LEVEL ASSEMBLY LANGUAGE PROGRAM PRODUCT (5734-AS1) which in turn requires the ASSEMBLER H/CMS INTERFACE IUP (5796-PEJ) for installation under CMS. Assembly language modifications to the system are discouraged as normal maintenance of the product may include substantial revisions in source code.

Prior to installation the user must assure that the Extended Full Screen 3270 Console Interface for VM/370 (PRPQ # 5799-AWP) or equivalent has been included in the VM/370 system on which the DISPLAY EDITING SYSTEM is to be installed. Furthermore, if remote 3270 systems are to be utilized by the DISPLAY EDITING SYSTEM, the VM/370 control program must include the fix for APAR VM07289. In addition, the 3271 control units which will be utilized must include the following EC's (Engineering Changes): 717956, 739540, and 739278.

The DISPLAY EDITING SYSTEM may be installed to run either always in the CMS user program area, or in a Discontiguous Shared Segment (DCSS) if available. The two methods of installation will be discussed separately.

Installing For Use In The User Program Area Only

1. Generate a VM/370 system which will support the DISPLAY EDITING SYSTEM environment. This includes installing the PRPQ 5799-AWP. Access the disk on which the Display Editing System is to reside as your 'A' disk. Be sure the disk contains at least 280 CMS data blocks.

2. Attach the installation tape to your virtual machine as 181.

3. Enter the CMS command "TAPE LOAD".

The DISPLAY EDITING SYSTEM is now installed and ready to be used. If the disk on which it was installed was the CMS sysres (190), resave the CMS system (SAVESYS CMS).

Installing For Use In Either A DCSS or The User Program Area

1. Generate a VM/370 system which will support the DISPLAY EDITING SYSTEM environment. This includes installing the PRPQ 5799-AWP, and generating an entry in the DMKSNT system name table as follows for the EDGAR DCSS.

```
LABEL NAMESYS    SYSNAME = xxxxxxxx,   (name of EDGAR DCSS)
                 SYSPGCT = 16,
                 SYSPGNM = (272-287),
                 SYSHRSG = (17),
                 SYSSIZE = 1024k,          any value is valid
                 VSYSRES = IGNORE,
                 VSYSADR = IGNORE,
                 SYSVOL = volid,
                 SYSSTRT = (mm,nn)
```

The SYSVOL and SYSSTRT information pertain to the location of the cylinders on which this DCSS is to be saved. These must be supplied by the system programmer doing the installation. (See VM/370 Planning and System Generation Guide, Defining Your VM/370 System, GC20-1801). The SYSNAME parameter may be assigned by the system programmer. The same name must be entered as part of the IUP installation procedure INSTED. The SYSPGNM and SYSHRSG load the shared segment at location x'110000'. These values may be changed to fit the installation's needs.

2. Logon to a virtual machine with "E" class privilege and a virtual machine size at least X'12000' bytes larger than the value represented by the SYSPGNM parameter above.

3. Access the disk on which the DISPLAY EDITING SYSTEM is to reside as your 'A' disk. Be sure the disk contains at least 280 CMS data blocks.

4. Attach the installation tape to your virtual machine as 181.

5. Enter the CMS command "TAPE LOAD".

6. Enter the command INSTED xxxxxx, where xxxxxx is the name of the EDGAR DCSS assigned in step one (1) above. You will be prompted for any additional information or errors encountered. NOTE: The userid from which you invoke INSTED will require 'E' privilege class for successful operation.

The EDGAR component of the DISPLAY EDITING SYSTEM may be regenerated by using the INSTED exec. The INSTED exec when entered with one (1) entry parameter will generate the version of EDGAR which is capable of running in the DCSS. If EDGAR is generated in this manner and the DCSS cannot be loaded, EDGAR will be executed from the user program area (X'20000'). The INSTED exec when entered with no parameters will generate the version of EDGAR which is capable of running only from the user program area. This form should be used if the DCSS will never be loaded.

Index

SH20-1965-0

Display Editing System for CMS Users Guide Printed in U.S.A. SH20-1965-0

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, N.Y. 10604

IBM World Trade Americas/Far East Corporation
Town of Mount Pleasant, Route 9, North Tarrytown, N.Y., U.S.A. 10591

IBM World Trade Europe/Middle East/Africa Corporation
360 Hamilton Avenue, White Plains, N.Y., U.S.A. 10601

Display Editing System for CMS
Users Guide
SH20-1965-0

READER'S COMMENT FORM

This manual is part of a library that serves as a reference source for systems analysts, programmers, and operators of IBM systems. This form may be used to communicate your views about this publication. They will be sent to the author's department for whatever review and action, if any, is deemed appropriate. Comments may be written in your own language; use of English is not required.

IBM shall have the nonexclusive right, in its discretion, to use and distribute all submitted information, in any form, for any and all purposes, without obligation of any kind to the submitter. Your interest is appreciated.

Note: *Copies of IBM publications are not stocked at the location to which this form is addressed. Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.*

Possible topics for comment are:

Clarity   Accuracy   Completeness   Organization   Coding   Retrieval   Legibility

If you wish a reply, give your name and mailing address:

____

____

____

Note: Staples can cause problems with automated mail sorting equipment.
Please use pressure sensitive or other gummed tape to seal this form.

What is your occupation? ______________________________

Number of latest Newsletter associated with this publication: ________________

Thank you for your cooperation. No postage stamp necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments or you may mail directly to the address in the Edition Notice on the back of the title page.)

SH20-1965-0

Reader's Comment Form

Fold and tape | Please Do Not Staple | Fold and tape

First Class
Permit 40
Armonk
New York

Business Reply Mail
No postage stamp necessary if mailed in the U.S.A.

Postage will be paid by:

International Business Machines Corporation
Department 825
1133 Westchester Avenue
White Plains, New York 10604

Fold and tape | Please Do Not Staple | Fold and tape

IBM®

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, N.Y. 10604

IBM World Trade Americas/Far East Corporation
Town of Mount Pleasant, Route 9, North Tarrytown, N.Y., U.S.A. 10591

IBM World Trade Europe/Middle East/Africa Corporation
360 Hamilton Avenue, White Plains, N.Y., U.S.A. 10601

Cut or Fold Along Line

Display Editing System for CMS Users Guide Printed in U.S.A. SH20-1965-0