MUSIC/SP User's Reference Guide

Chapter 7. Using The Editor

Chapter 7. Using The Editor

Overview of the Editor

This chapter describes the Editor component of the MUSIC system and is divided into the following main sections:

- This initial section describes Editor concepts including modes of operation.
- Starting and ending an edit session and description of the EDIT command.
- Editor full-screen mode and program function key support.
- Creating a new file using input mode of the Editor.
- Common editing functions.
- Using the prefix area.
- Editor commands, arranged alphabetically. Included is information on command syntax.
- Advanced features: starting column suffixes, logical commands, defining function keys, hexadecimal input and output, and other topics.
- Editor macro facility in REXX.
- Editor restart facility.

Editor Concepts

The Editor is a general purpose interactive utility for creating, modifying, viewing, and storing files.

At the beginning of the edit, the Editor copies the specified file to a temporary area. All requested changes during the edit session are made to this temporary copy. The changes are not permanent until a FILE, SAVE, or EXEC command is used, which creates a new original file from the current temporary copy.

The Editor is used to create, look at, or modify files. A file consists of lines of information arranged in sequential order. These lines are called records. The Editor can handle records up to 8192 bytes (characters) long.

The Editor is not subject to the user's execution time limit. This enables a userid which has a small time limit to do lengthy edit operations.

What Can the Editor Do?

- search for lines based on their contents.
- make global changes throughout the file.

- move and copy text and lines.
- add and delete text and lines.
- replace the original file or create another file.
- merge part or all of another file into the current file, and vice-versa (store part or all of current file into another).
- The Editor can be tailored to suit your own needs by defining your own function keys.
- you can create your own editor commands and macros.
- you can program the Editor to be used with your own programs.
- you can execute MUSIC commands and programs from the Editor.

Workstation Modes for the Editor

The Editor can display the lines being edited in various ways, depending on the type of workstation used. A workstation can be either a terminal or a personal computer. IBM 3270-type terminals are commonly used for mainframe computers, and personal computers accessing mainframes will often emulate the characteristics of 3270-type terminals. 3270-type workstations can support full-screen applications such as the Editor.

Full-Screen Mode — IBM 3270-type workstations use *full-screen* mode for the Editor. Full-screen mode displays an entire section of the file. It lets you make changes by typing directly over the displayed text, and lets you use function keys for many editing operations. Full-Screen mode is described later in this chapter.

Screen Mode — Another mode for editing is called *screen* mode. It is intended mainly for ASCII (i.e. non-3270) screen workstations that are not emulating 3270s. It displays a section of the file but does not allow direct modification of text on the screen or use of function keys.

Line Mode — If neither full-screen nor screen mode is used, the Editor works with only one line at a time. Hard-copy terminals, those without screens, would used this method.

Editor Modes

The Editor operates in one of two modes: command mode and input mode. The Editor starts off in command mode unless the file you are editing is empty (then you start in input mode). In command mode, the Editor accepts commands and performs them for you. To change from command to input mode, use F11 or the INPUT command. In INPUT mode, you can type multiple lines without interaction with the Editor.

Starting and Ending the Editor

This section describes the procedures for invoking the Editor and ending the edit session. Once invoked, the Editor operates in full-screen mode for most workstations. See the section entitled "Editor Full-Screen Mode" for details. Information about typing in data is covered in the section "Creating a New File". For information about making changes to a file, refer to the section "Common Editing Functions".

The diagram below is an overview of the necessary steps for starting and ending an edit session.

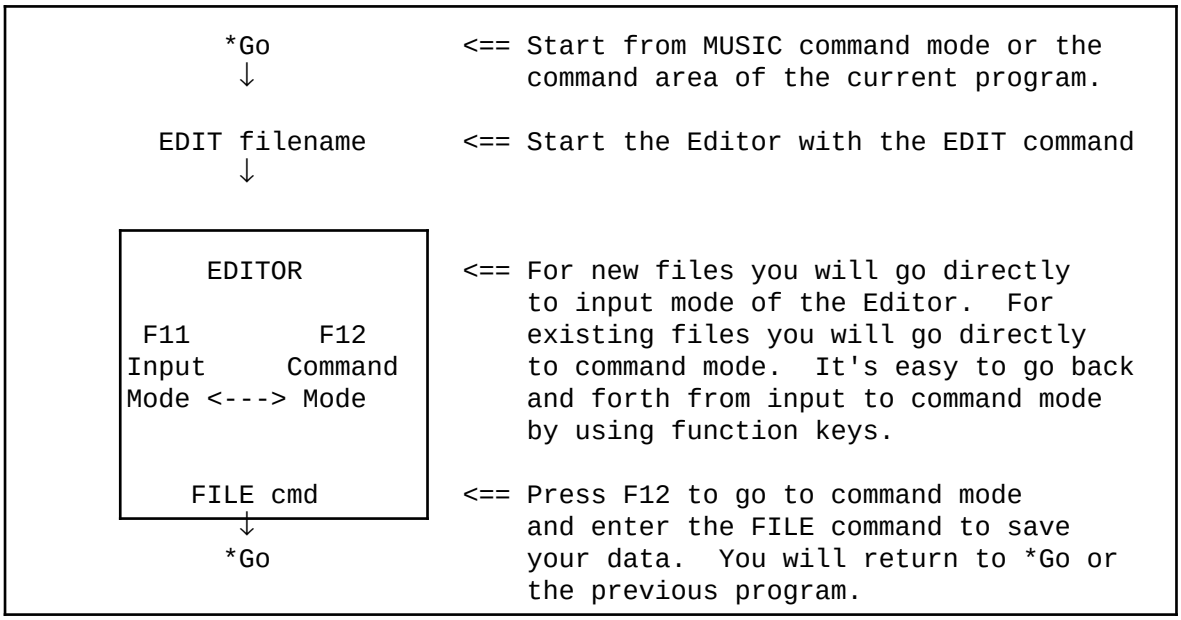


Figure 7.1 - Steps for Using the Editor

Starting the Editor

The Editor can be started in a variety of ways. It can be invoked from *Go, from the command area of a program, or as a selection from a menu facility such as FSI (Full Screen Interface) or TODO (Time, Office, and Documentation Organizer). Also, some MUSIC programs invoke the Editor automatically for collecting data. For example, the MAIL program invokes the Editor when you are ready to type in the text of your message.

There are two MUSIC commands that can be used to start the Editor: EDIT and BROWSE.

EDIT — The EDIT command (abbreviation E) is used most often to start the Editor. The syntax for this command is described below.

BROWSE — Use the BROWSE command (abbreviation B) to edit a file which you do not intend to change. BROWSE is equivalent to the EDIT command with the RO (read-only) option.

EDIT Command Syntax

```
EDIT [name  ] [NEW] [LRECL(n)] [F ] [RO] [NOLOG] [UIO] [MAX(n)]
E    [NEW   ] [OLD] [LR(n)   ] [FC]
     [/INPUT]       [nn      ] [V ]
     [/HOLD ]                  [VC]
     [/REC  ]
```

```
Examples:    EDIT MYFILE
             E FILE25 NEW
             E FILE30;L xx
```

Parameters:

name — This specifies the name of the file to be edited. If the file name NEW is specified, the Editor goes into Input mode and a completely new file may be created. If the file name /INPUT is specified or the file name is omitted completely, the Input file will be edited. If the file name is omitted, the other parameters must be omitted also. The file name /HOLD can be specified, to edit the output holding file. The special name /REC edits the recording file created by the RECORD command.

Note: to edit a file whose actual name is NEW, specify the userid of the file's owner in front of the name. For example, EDIT ABCD:NEW. The OLD option can also be used, as in EDIT NEW OLD.

NEW
OLD — If NEW is specified, the Editor assumes that the named file is a new file and goes into Input mode automatically. If the named file actually does exist, then when a FILE, SAVE, or EXEC command is issued during the edit session the Editor will ask the user whether the user wants to replace the file or not. If OLD is specified, the Editor will look for the file in your library and go into command mode. OLD is the default.

LRECL(n)
LR(n)
nn — This specifies the record length to be used during the edit. A number from 1 to 8192 can be specified. The record length option should be entered as LRECL(n) or LR(n). If the number *n* contains at least 2 digits (as in 25 or 03), the LRECL keyword can be omitted, and the option entered as simply *n*. The record length will be assigned to the new file when a SAVE, FILE or EXEC command is performed. If not specified, the original record length of the named file (if the file is an existing file), or a record length of 80 (if the file is a new file) will be used. Note that if *n* is less than the record length of the original file, lines may be truncated as they are read by the Editor. For variable length records, *n* represents a maximum record length.

F
FC
V
VC — This specifies the record format to be attributed to the file being edited. This record format will be assigned to the new file when a SAVE, FILE or EXEC command is performed. Either F (fixed format), FC (fixed compressed), V (variable length), or VC (variable compressed) can be specified. Refer to *Chapter 4. File System and I/O Interface* for a description of the record formats. If not specified, the original record format of the named file (if the file is old), or FC (if the file is new) will be used.

RO — Causes the Editor to use Read Only mode. This is equivalent to the BROWSE MUSIC command. In full-screen mode, the text on the screen is protected (non-modifiable). RO implies the NOLOG option and the Editor command "SUBSET 2". If you decide you want to make changes, use the "SUBSET 0" Editor command to revert to normal editing.

NOLOG — Suppresses the restart feature of the Editor. This prevents the Editor from looking for a log file or creating a new one. Refer to the discussion on the restart facility later in this chapter.

UIO — Causes the Editor to read the file by 512-byte blocks, using MFIO UIO requests. Each block becomes a record for the edit. This lets you edit (but not change) record format U files, or files that cannot be read sequentially. The record length for the edit is automatically set to 512. UIO implies RO and NOLOG. Do not use the FILE or SAVE command to write to the file being edited, since UIO is not used for the writes.

MAX(n) Specifies the maximum number of records the Editor is to read from the file being edited. This option implies RO and NOLOG. It is useful for looking at the first part of a large file. Example: EDIT BIGFILE MAX(500).

The parameters can be specified after the file name on the EDIT command in any order. A blank or a comma must be used between parameters.

Editor commands can also be specified at the end of the EDIT command. A semicolon (;) is used to separate the commands. For example, EDIT SAMPLE;L XY;C/XY/123/ instructs the Editor to edit the file named SAMPLE, locate the first line which has the character string XY in it, and then change the string XY to 123.

Ending the Editor

The function key F3 (Quit) is used in the Editor to exit the Editor program **without** saving any changes. The following commands are needed to save your changes and update the file stored in your library.

```
FILE [newname]
```

When this command is entered, the Editor saves any changes done during the current edit session, and ends the session.

The original file is updated, unless you changed the file name during the edit session (see NAME command later). Another way to leave the original file alone is to specify a different file name on the FILE command.

Examples:

```
1.  file                    2.  file program2
```

Example 1 saves the file with the name displayed at the top of the Editor screen. Example 2 uses another name and the name displayed at the top of the screen is ignored.

```
SAVE [newname]
SA
```

Same as FILE except the edit session continues.

```
EXECUTE [newname]
EX
```

Same as the FILE command above except the file is executed also.

Editor Full-Screen Mode

Introduction to Full-Screen Mode

Users of 3270-type workstations take advantage of the special features of these workstations in *full-screen* mode of operation (default). (For information about other modes of operation see the topic "Workstation Modes for the Editor" earlier in this chapter.)

In full-screen mode, you can modify data directly on the screen, using the local editing keys, as well as enter commands in an area near the bottom of the screen. Also, various editing operations can be done by pressing program function (PF or F) keys to enter Editor commands.

Screen Format for Full-Screen Mode

The screen display for full-screen editing is divided into the following areas:

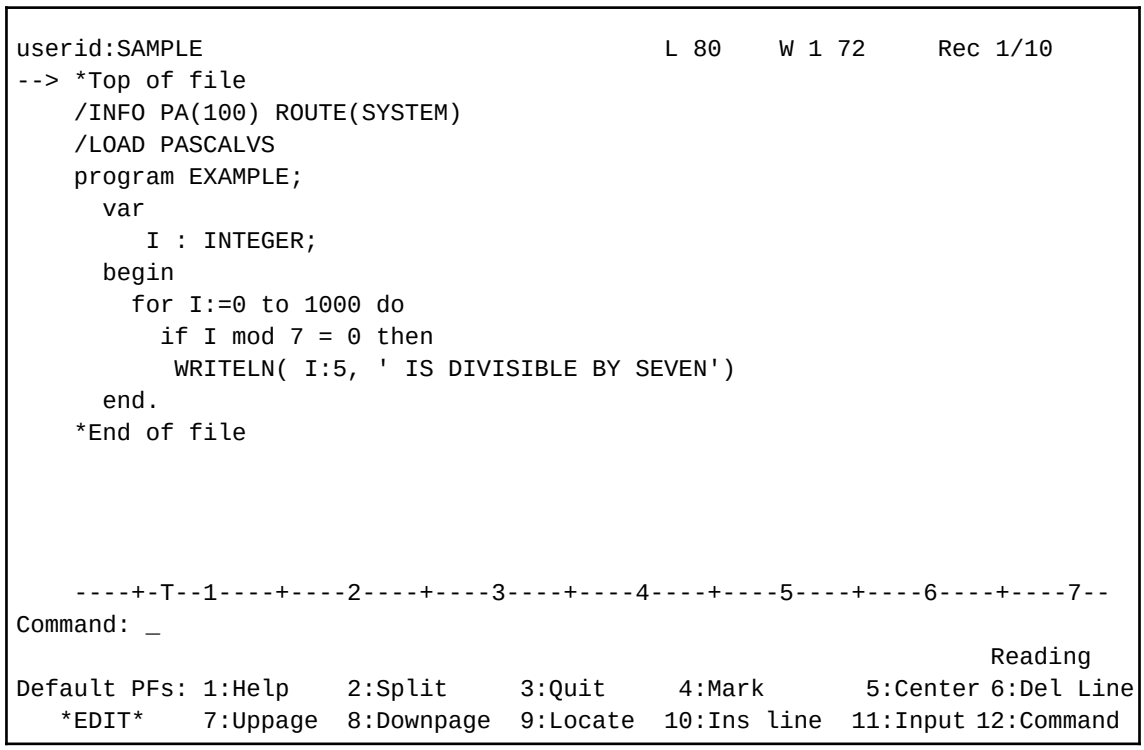


Figure 7.2 - Screen Display in Full-Screen Mode

1. **Title line:** this is the first line on the screen, and contains the name of the file being edited, the file's logical record length (L), the starting and ending window columns (W), and the current line number with the total number of lines (Rec) in the file. This field is not modifiable on the screen, although Editor commands can be used to change the file name and window setting.

2. **Lines of the file:** this is the main body of the screen, immediately following the title line. It may be up to 20 lines on a 24-line screen, or 39 lines on a 43-line screen. Each screen line displays one record of

the file, and the text is directly modifiable on the screen. Only the window portion of each record (see the WINDOW command) is displayed, to a maximum of 72 characters (124 on wide-screen terminals). The current line is indicated by an arrow pointer in the left margin and is also displayed in high intensity (red on a color terminal). If line numbering is in effect (the NUMBER command), line numbers are displayed in the left margin. If the PREFIX command is in effect, then a 4-character modifiable area indicated by "====" is in the left margin. The beginning of the file, if displayed, is indicated by `*Top of file.` Similarly the end of the file is indicated by `*End of file.` In Input mode, several empty lines are displayed following the current line, thus allowing you to add new lines to the file by typing them on the screen.

3. **Tab line:** this shows column numbers and input tab positions. The tab positions (displayed as T's) correspond to the column numbers specified on the TABIN command. The tab positions are relative to the starting window column, rather than to column 1 of the file's records. See the topic "Defining a TAB Key" later in this chapter.

4. **Command area:** this is used for entering Editor commands. Several commands may be entered by separating them by the command delimiter character, normally semicolon (;). If the Editor is in input mode rather than command mode, the command area is replaced by the centered message `* Input Mode *.`

   Note: REXX Editor macros and MUSIC commands can also be entered in the command area.

5. **Message area:** this line of the screen is reserved for messages from the Editor. Up to 3 messages can be packed into this area. The messages are displayed in high intensity (white on a color terminal). If the messages do not fit, or if considerable output is generated by a command such as SCAN, the output is displayed on a new screen in normal MUSIC format, with the status message `More...` in the bottom right corner. Pressing the ENTER key allows you to continue. Editor commands may not be entered until the `More...` condition has been cleared. While output is being displayed in normal MUSIC format, the PA1 key may be used to go to attention mode (**Attn**) and skip output (by the /SKIP command).

6. **Status indicator:** the current status of the workstation and the Editor is shown in the bottom right-hand corner of the screen. Possible status words are `Reading, Working, **Attn**,` and `More...`

7. **SHOW area:** if a "SHOW filename" command has been used, lines of the specified file are displayed at the very bottom of the screen. By default, the area is used to display the definitions of the function keys. Refer to the SHOW command for more information.

General Notes

1. Since at most 72 characters are displayed from each line of the file, the largest possible window width (as set by the WINDOW command) is 72. (For a wide-screen terminal such as the 3270 model 5, the maximum width is 124 rather than 72.) When full-screen mode is initially turned on and when the WINDOW command is used, the horizontal window setting is automatically adjusted if necessary. Also, the zone (refer to the ZONE command) is modified at this time to match the window setting. The resulting zone is from column 1 to the end of the window. This zone may be changed by a subsequent ZONE command.

2. If a line contains unprintable or nonstandard characters, avoid using direct screen changes to modify the line. This is because any change to the line causes unprintable characters in the line to be replaced by blanks. Use the CHANGE command instead.

3. A logical input tab character may be used when typing lines on the screen. The tab characters are expanded to the appropriate numbers of blanks when the next action key is pressed. When the command TEDIT is used (rather than EDIT), or when the Editor command TEXT SCRIPT is used, input tab characters are not expanded.

Editing Keys

The Editor uses a full screen technique to let you change data on the screen and manipulate the file by using 3270 *local editing* keys and *action* keys in addition to Editor commands.

Local Editing Keys

The workstation stores in its own memory the lines displayed on the screen. Changes can be made to these lines, without interacting with the host computer, by using local editing keys. For example, the DELETE and INSERT keys are useful for removing or adding characters to a field. The ERASE EOF key deletes the characters from the cursor position to the end of the field (in this case to the end of the screen line or command area).

Characters can be replaced by first positioning the cursor at their location and then typing over them. The cursor position is changed by using the various arrow keys (cursor control keys) on the keyboard. The NEW LINE key (with a *down and to the left* arrow on it) and TAB key are useful for moving the cursor to the start of the next line. This key and the other arrow keys repeat when held down. For NET3270, the NEW LINE key is Ctrl-Enter, or the * on the right-hand keypad.

Changes made with local editing keys are only transferred to the Editor's temporary copy of the file when an action key (a function key or the ENTER key) is pressed (see below).

If you inadvertently make unwanted changes when using local editing keys, the CLEAR key cancels the effect of all screen changes made since the last action key was pressed. The ERASE INPUT key (a local key), which clears all unprotected fields, should not be used.

Note: If the BROWSE command was used to start the edit, or the RO (read-only) option was used on the EDIT or TEDIT command, then the file's text can not be changed on the screen. If you try to type over text on the screen, the workstation will go into the *input inhibited* condition. Press the RESET key to clear this condition.

Action Keys

The keys which cause an interaction with the system are referred to as *action keys*. They are ENTER, the program function (PF or F) keys F1 to F24, CLEAR, PA1, PA2, and SYS RQ (or TEST REQ). The following describes the default definitions of the action keys.

ENTER — This key transmits screen changes, input lines, and commands to the system. It can also be used to advance to the next screen when `More...` appears in the bottom right corner. If no other keys were pressed since the last action key, ENTER moves the cursor to the current line.

PA1 — Not normally used. Refer to the description of the screen message area (above) and the S parameter on the SCAN and CHANGE commands (below) for more information.

PA2 — This key is used for multi-session control. It adds or deletes a MUSIC/SP session, or switches to the previous or next session.

Note: Make sure you have pressed ENTER or one of the function keys before pressing PA2. Otherwise any screen changes you have entered since the last action key are lost. If you press PA2 by mistake and do not wish to go to another session, press the ENTER key.

CLEAR Cancels the effect of any screen changes made since the last action key was pressed, redisplays the current screen, and places the cursor in the command area. Input mode is terminated if it was in effect.

SYS RQ
TEST REQ This key is normally not used. It has the same effect as the CLEAR key, except that the work-station is placed into MUSIC attention mode. You may then enter an attention mode command (such as /TIME) or a blank line. The Editor eventually redisplays the original screen (you may need to press the ENTER key). Use of the TEST REQ key may be prohibited or restricted by the system environment.

Function Keys (F1 - F12) for the Editor

Function keys have been defined to perform certain EDITOR commands, eliminating the need to type them. They provide a fast and easy way to edit files. Function keys can be used both in command and input modes of the Editor.

It is possible for your private EDITOR file or the system *COM:EDITOR file to change the definitions of the function keys. Enter the command SHOW PF to see the actual function key definitions in effect for your edit session. If you want to change the default definition of function keys, see the topics "Defining Function Keys and the X command" and "Creating your own Editor" later in this chapter.

F1, F13 (HELP) provides information about how to use the Editor. You are presented with a list of topics and are asked to enter the number(s) of the topic(s) you want more information about.

F2, F14 (SPLIT) splits a line into two lines. The character at which the cursor is positioned becomes the first character of the second line.

In Browse: (PRINT) Prints the file. Pressing F2 places "PRINT *CUR" into the command area. You can fill in a printer name if you wish, by adding " r(name)" onto the end of the PRINT command. Then press ENTER to complete the print operation, or press CLEAR to cancel it.

F3, F15 (QUIT) terminates the Editor session without performing any save operation. If you have made changes to the file but have not issued a SAVE command to make the changes permanent, you will be prompted for permission to end the edit session. Enter YES or Y to end the session, or NO or N to cancel the QUIT operation and continue editing.

F4, F16 (MARK) is used to designate (mark) a line or group of lines. To mark a group of lines, mark the first and last line of the group. The marked group can be used by commands such as MOVE., COPY., DELETE., and STORE. Marked lines are displayed with a vertical bar character in the left margin of the screen.

In Browse: (TOP) Goes to the top of the file.

F5, F17 (CENTER) causes the screen to be redisplayed so that the current line (or the one containing the cursor when the function key is pressed) appears in the middle of the screen display. The screen display is shifted the appropriate number of lines up or down in the file in order to center the current line. The UPWINDOW and DOWNWINDOW commands perform a similar function.

F6, F18 (DELETE) removes the current line from the file. Note that if the cursor points to a line on the screen when the function key is pressed, that line is the one which is deleted. This is because the current line pointer is moved to the cursor before the function key request is done. The same applies to the other function keys. To delete a line, position the cursor on it and press F6. This function key should not be used if prefix area commands are entered on the same screen.

In Browse: (LAST) Goes to the bottom of the file.

F7, F19 (UPPAGE) displays the previous page (screen) in the file (towards the beginning of the file).

F8, F20 (DOWNPAGE) displays the next page (screen) in the file (towards the end of the file).

F9, F21 (LOCATE) locates the next occurrence of the character string used on the last LOCATE, SEARCH, FIND, HUNT, UPLOCATE, or UPFIND command. The search starts at the line following the current line. If the CURSOR LOCATE command is in effect, the cursor will be positioned at the start of the found string when the screen is displayed.

F10, F22 (INSERT) inserts a blank (null) line after the current line. This is useful for adding a single line to the file. You can type the line over the inserted blanks.

In Browse: (WINDOW LEFT) Shifts the display window up to 72 columns to the left.

F11, F23 (INPUT FLIP) puts the Editor into Input mode. This is useful for entering several lines after the current line. In Input mode, the screen displays the current line and a few lines above it, followed by null lines in the remainder of the screen. The user types over these lines in order to enter the new lines. The NEW LINE local key is used to go to the next line. If more space is needed, press the ENTER key. If the Editor is already in input mode, F11 terminates Input mode. In this way, F11 *flips* back and forth between Input and command mode.

In Browse: (WINDOW RIGHT) Shifts the display window up to 72 columns to the right.

F12, F24 (CMDPFK) moves the cursor to the command area. This is necessary for entering Editor commands that are not provided for by function keys. The arrow keys may also be used to place the cursor in the command area prior to entering commands. F12 also terminates Input mode. The Editor command corresponding to this function key operation is CMDPFK.

Notes:

1. Other action keys should not be used.

2. The ERASE INPUT key should not be used. It cancels all screen changes made since the last action key was pressed and erases the screen display. If you hit the ERASE INPUT key by mistake, press CLEAR to redisplay the screen.

3. After pressing an action key, the INPUT INHIBITED indicator is on briefly while the system processes the request. On 3277 terminals, INPUT INHIBITED is indicated by a bright square at the right of the screen. On newer models of terminals, it is indicated by a large X at the bottom of the screen. If INPUT INHIBITED comes on before an action key is pressed, it means that you have tried to modify a protected part of the screen; press the RESET key to continue.

Order of Operations

It is possible to combine several operations at the workstation before pressing an action key. Operations are done by the system in the following order:

1. Changes made to text on the screen.
2. Movement of the current line pointer to the line indicated by the cursor, if no prefix area commands are entered.
3. Prefix area commands.
4. Command-area commands.
5. Program function key operation.

Note that the cursor must be placed into the command area before typing an Editor command. The cursor can be positioned by using any of the arrow keys. It can also be put into the command area by using the F12 key.

If the Editor is in Input mode when a function key is used (standard or user-defined), Input mode is terminated before the function key request is done.

Suggestions for Efficient use of Full-Screen Mode

- If your 3270-type workstation has more than 12 function keys, set up a private EDITOR file to define extra function key operations in addition to (or in place of) the standard ones. Using function keys is faster and easier than typing commands. Good candidates for function key operations are the commands: MOVE., COPY., DELETE, DELETE., WINDOW FLIP, JOIN, TOP, BOTTOM, DUP, UPLOCATE, MINSERT, MDELETE, UPWINDOW, DOWNWINDOW, FILE, ECHO, CURSOR END, CURSOR n TEMP, NULLS FLIP, NUMBER FLIP, SUBMIT.

- Don't use the ENTER key in input mode to end each line. Use the NEW LINE key or TAB key. The NEW LINE key has a *down and to the left* arrow on it, and is located on the right-hand side of the alphabetic keyboard. This gives much better response in Input mode, since no system interaction is required (NEW LINE and TAB are local keys). Once the screen is full or nearly full with input lines, press the ENTER key to get a fresh screen. Outside of Input mode, the NEW LINE or TAB key is often quicker than the command function key (F12) for positioning the cursor in the command area. For NET3270, the NEW LINE key is Ctrl-Enter, or the * on the right-hand keypad.

- Use the INPUT function key (F11) or the command function key (F12) to terminate Input mode, rather than pressing ENTER twice. This requires only one interaction with the system. The input function key (F11) places the cursor at the last input line; the command function key (F12) places the cursor in the command area.

- When you know that you will be typing a command in the command area (rather than making a change on the screen), end the previous action by using the command function key (F12) instead of the ENTER key. This ensures that the cursor will be positioned in the command area, and may eliminate one interaction with the system.

- Several Editor commands can be entered in the command area by separating them by a semicolon (;). This technique can really speed up editing, but use it carefully, since an error on one of the commands does not stop execution of the remaining commands.

- To retrieve the previous command line entered in the command area, place the cursor at the tab line before pressing an action key. This redisplays the last non-blank text entered in the command area. You may then modify and re-issue the command, or the command may be re-issued (as is) by simply pressing the ENTER key.

- The screen cursor may be used to move the current line pointer. Simply use the arrow keys to place the cursor at the desired new current line. That line becomes the new current line before any command area commands or function key operations are done. Of course, a command or function key operation may further move the pointer. Positioning the cursor at the `*Top of file` line is equivalent to a TOP command. Positioning it at the `*End of file` line is equivalent to a BOTTOM command.

Creating a New File

The first step to creating a new file is to invoke the Editor. If you are choosing an edit selection from one MUSIC's menu facilities (like FSI) then you will be prompted for the necessary information. Otherwise, you need to use the MUSIC command "EDIT" with a file name and the NEW parameter. For example,

```
EDIT myfile new
```

starts the Editor with an empty file named "MYFILE". (For full details about the EDIT command, see the earlier topic in this chapter called "Starting the Editor".)

The figure below shows the Editor's screen display for input mode. The Editor starts automatically in input mode for new files.

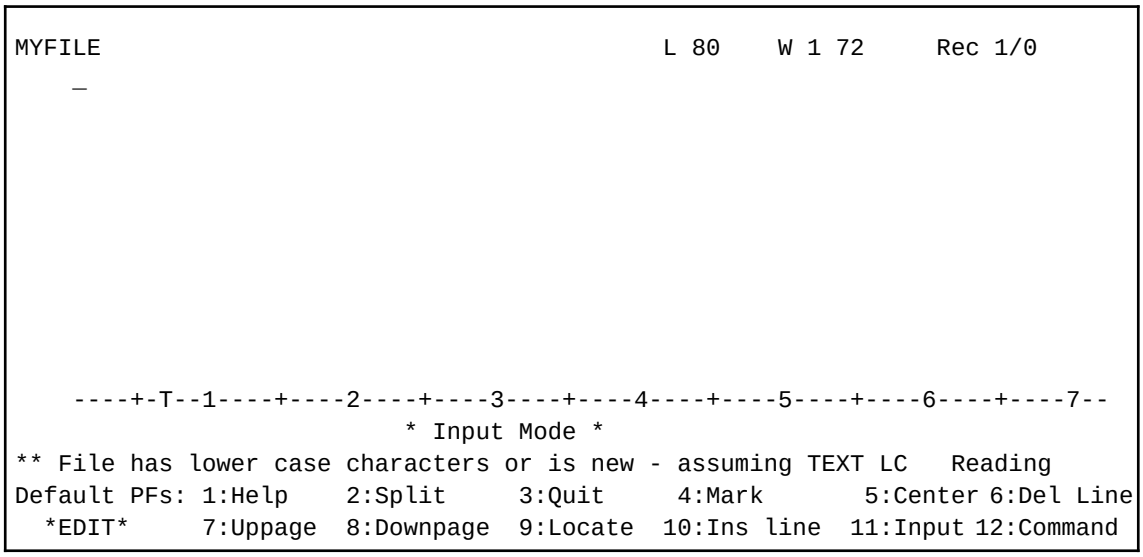


Figure 7.3 - Screen Display for Input Mode

In the figure above your cursor is positioned on the first input line on the screen. You are ready to type in your data. Remember to use the TAB key to end each line. When you are finished and want to save the file, press F12 to go to the command area, type FILE, and press ENTER.

Input of Data for the Editor

There are five important rules to know for typing in data. Take the time now to understand and learn the correct procedures.

1. You can correct typing mistakes on the line you are currently typing by using the BACKSPACE key or <-- key. Just backspace over the incorrect characters and retype.

2. When in input mode, use the TAB key to skip to the next line, rather than the ENTER key. The TAB key shows an arrow pointing to a vertical line (-->|). (The TAB key is a local key and does not interact with the system.) Once the screen is full or nearly full with input lines, press the ENTER key to get a fresh screen.

Note: You do not have to wait until you get to the bottom of the screen before pressing ENTER. However, if you press ENTER after typing only a single line, the Editor will automatically bring you to command mode. You will need to press F11 to return to input mode.

3. It is important to stay inside the boundaries (in the input area) on the screen. You can go out of bounds by moving the cursor with some of the arrow keys (-->, <--, etc.). If your cursor is accidentally moved out of the input area and you try to type, the keyboard becomes LOCKED. Press the RESET key to unlock the keyboard, and use the TAB or arrow keys to move the cursor back to the input area.

4. When adding blank spaces between words, use the SPACEBAR and not the right arrow key (-->). Arrow keys are for moving the cursor around the screen. In the example below, the line was typed using the incorrect --> key for spaces. Notice the results after the ENTER key is pressed.

Before	After
`10  FORMAT(' THE ROOT IS')`	`10FORMAT('THEROOTIS')`

5. When the Editor is invoked, a message appears at the bottom of the screen indicating whether the command TEXT LC or TEXT UC is in effect. TEXT LC means that lower case letters are retained and are not automatically converted to upper case (default for new files or existing files already containing lower case). TEXT UC means that all letters are automatically converted to upper case when an action key is pressed (default for existing files that contain only upper case letters.) To change the default setting, issue the appropriate TEXT command.

Common Editing Functions

This topic covers the most common editing functions. When you need to do more complicated editing, like moving text, or making global changes, refer to the section in this chapter entitled "Editor Commands".

Local editing keys and function keys are used to perform most editing tasks, the exception being the FILE command, which is used to end your edit session. The following describes the most common editing tasks.

Making Changes on the Current Screen

Make changes to the text displayed on the current screen by moving the cursor to that position and typing over the existing text. Some areas of the screen, such as the title line, are protected and cannot be changed. If you attempt to type in one of these areas, the keyboard will lock. Press RESET to unlock the keyboard, so you can move the cursor to a valid input area.

Moving the Cursor

There are several arrow keys for moving the cursor on the screen.

↑ up	↓ down	← left	→ right	→| tab	|← tab

Be careful not to move out of the input area with the UP, DOWN, LEFT, and RIGHT arrow keys. Use the TAB key to skip to the next line. (Remember to use the SPACEBAR when you want to add blanks and text to the end of a line. The arrow keys only appear to leave spaces. As soon as an action key is pressed, the blanks disappear.)

Paging Up and Down

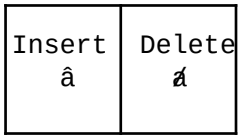
To see the previous screen use the F7 key (UPPAGE). To see the next screen use the F8 key (DOWNPAGE). When you need to see part of the previous or next screen, position your cursor on a line and press F5 to center that line on the screen.

F7 uppage	F8 downpg	F5 center

Commands such as TOP, BOTTOM, and LOCATE can also be used to move up and down in your file. They are described later in the topic "Editor Commands".

Inserting and Deleting Characters

Two important keys for editing are the INSERT and DELETE keys. They are local editing keys and should not be confused with the function keys F10 key (INSERT LINE) and F6 key (DELETE LINE).



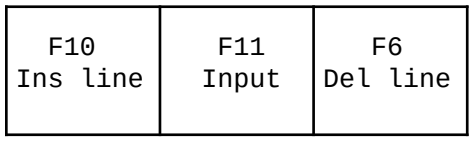
Insert Key — Position the cursor where you want to insert new characters and press the INSERT key. Type in the desired character(s). The new characters are placed in front of the old text. If you fill up the entire line the keyboard will lock. If you need more room, position the cursor where you want to end the line and press F2 to split the line in two.

Either the word INSERT or a symbol appears at the bottom of the screen indicating that INSERT mode is in effect. To cancel INSERT mode, press the RESET key or the INSERT key a second time, depending on your workstation.

Delete Key — Position the cursor on the characters to be deleted. Press the DELETE key to remove each unwanted character.

Inserting and Deleting Lines

There are two function keys for inserting lines and one function key for deleting lines. To use these keys, position the cursor on a line and press the appropriate key. If the cursor is in the command area, the line pointer (-->) indicates where to insert or delete.



Ins Line — Position the cursor on a line and press F10 to insert 1 blank line after the line the cursor is on.

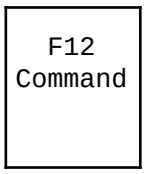
Input — Position the cursor on a line and press F11 to go to input mode. When you have several lines to add, input mode is more efficient than pressing the F10 (Ins line) key for each new line.

Press F12 (Command) to return to command mode. (F11 works as a toggle key, it also returns to command mode, but the cursor is left on the current line.)

Del line — Use F6 to delete the line the cursor is on.

FILE and QUIT Commands

When you are ready to save your file and exit the Editor, do the following:

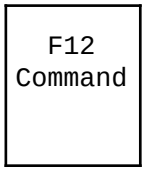


- press F12 to go to the command area
- type the command: FILE
- press ENTER

If you do not want to save any changes, press F3 (Quit).

Getting Help for the Editor

Press F1 to get help for the Editor. A list of topics is displayed for you to choose from. For help on a particular command, do the following:



- press F12 to go to the command area
- type the command: HELP with a command
- press ENTER

Common Editor Commands

Editor commands are issued in the command area on the screen. In command mode you can move into the command area using the TAB key or F12 key. In input mode, press F12 to go to the command area.

Commands can be typed in upper or lower case letters. After entering a command, you must press ENTER to have the Editor process it. An example of an Editor command is "FILE".

Note: MUSIC commands are also allowed from the Editor's command area. Use a slash (/) to distinguish MUSIC commands from Editor commands. You can use the EDIT command to edit another file while remaining in the first edit session. More information about MUSIC commands is in *Chapter 5 - MUSIC Commands*.

The following Editor commands are grouped into task-similar categories. Here is a table of the most common commands:

Moving in the File	Changing the File	Working with Marked Groups	
FIND	CHANGE	MARK	UNMARK
LOCATE	MERGE	COPY.	MOVE.
TOP		DELETE.	CHANGE.
BOTTOM		STORE	

Ending the Edit Session	Miscellaneous
FILE	HELP
SAVE	DUP
EXECUTE	NAME
QUIT (F3)	PURGE

There are many Editor commands available for novice and advanced users. For complete documentation on all Editor commands, refer to the section in this chapter entitled "Editor Commands" or or type "HELP COMMANDS" in the Editor's command area.

Marking a Group of Lines

When editing a file, the need often arises to define a set of consecutive lines within the file, and then perform some operation on those lines. Examples of this are moving or copying a group of lines from one part of the file to another, restricting a global change to a section of the file, converting a set of lines to upper case, and deleting a group of lines.

The MARK command and the *dot* forms of commands such as MOVE, COPY and CHANGE, provide this capability in the MUSIC Editor. Use of marked groups greatly reduces the need for line numbers while editing.

To define a group of lines, the user moves the line pointer to the first line of the group and issues the MARK command, then moves the pointer to the last line of the group and issues the MARK command again. Once the group has been defined, it can be referred to in various commands by using a period (.) as a parameter. For example, the command CHANGE/ABC/DEF/.G changes every occurrence of ABC to DEF within the group. The commands MOVE. and COPY. move and copy the group. These and other "." commands are described below.

Although the MARK command and the various . commands can be used on any type of workstation, they are best suited to full-screen mode on a 3270-type workstation. In full-screen mode, each line of a marked group is identified by a vertical bar in the left margin on the screen. The user's private EDITOR file would define a program function key as MARK, and probably two other keys as MOVE. and COPY..

Some Notes on the MARK Command

Only one set of consecutive lines can be marked at a time. The marked group is maintained until a new group is defined, or until an UNMARK command is issued, or until all the lines of the group are deleted. As new lines are added to or deleted from the file, the marks are adjusted accordingly. The command MARK ? reports which lines are currently marked. The blank between MARK and ? may be omitted. Abbreviations for the MARK and UNMARK commands are MA and UNMA.

When defining a group, the first and last lines may be marked in either order. The group may consist of only

one line, in which case the second MARK command is not needed. Once a multi-line group has been marked, or a 1-line group has been marked and used in a . command, a subsequent MARK command starts a new group.

Commands for Operating on a Marked Group

In these commands, the . character is a command parameter, and the blank between it and the command name may be omitted. Also, the command name may be abbreviated, as in MO., CO., and DEL..

= . Moves the current line pointer to the first line of the group. This is handy for returning to a particular point in the file. The command is usually entered as =. to save typing.

MOVE . Moves the marked group of lines to after the current line. The group remains marked. For example, if lines 51 through 75 are marked and the current line is 20, then MOVE. is equivalent to MOVE 20 51 75

COPY . Copies the marked group of lines to after the current line. The original group remains marked.

DELETE . Deletes the marked group. Note that the MDELETE command is another (sometimes faster) way of deleting a block of lines. The operation may be qualified by a logical expression as the second parameter, as in DEL.,(NOT/ABC/), which deletes all lines of the group which do not contain ABC.

CHANGE A period (.) can be used in place of the *n* (line count) parameter on a CHANGE or logical CHANGE (CHANGEL) command. This causes the change to apply to each line of the marked group (subject to the logical expression, if CHANGEL is used). The Editor automatically goes to the first line of the group before starting the change. Example: C/ABC/DEF/.F

REPEAT . Causes the next BLANK or OVERLAY command to apply to each line of the marked group. It also moves the current pointer to the first line of the group, and places the screen cursor into the command area.

TOUC .
TOLC . Converts the marked group to upper or lower case.

COPYCOL n1 n2 n3 .
When . is used as the 4th parameter, instead of a line count, the COPYCOL operation is applied to each line of the marked group.

SEQ LINES=. When the LINES= parameter specifies a period rather than a line count, the sequencing operation is applied to each line of the marked group. Example: SEQ 1 1 COL=15 LEN=3 LINES=.

SORT . This Editor macro uses the MUSIC SORT command to sort the marked group.

STORE name Writes the marked lines to the specified file. "A" for append can be added after the file name to have the text added to the end of an existing file (i.e. STORE abc a).

PRINT . Prints the marked group.

Prefix Area

The prefix area, turned on by the command PREFIX ON, is a 4-character modifiable field at the left of each displayed line of the file. It contains ====, or the last 4 digits of the line number if NUM ON is in effect. You can enter various special commands in the prefix area, to do editing operations such as inserting or deleting lines, moving and copying lines, etc.

The following diagram illustrates the screen display after the PREFIX ON command is entered:

```
userid:SAMPLE                                L 80    W 1 72     Rec 1/10
> ==== *Top of file
  ====  /INFO PA(100) ROUTE(SYSTEM)
  ====   /LOAD PASCALVS
  ====  program EXAMPLE;
  ====  var
  ====       I : INTEGER;
  ====    begin
  ====      for I:=0 to 1000 do
  ====        if I mod 7 = 0 then
  ====         WRITELN( I:5, ' IS DIVISIBLE BY SEVEN')
  ====    end.
  ====  *End of file



    ----+-T--1----+----2----+----3----+----4----+----5----+----6----+----7--
Command: _
                                                                    Reading
Default PFs: 1:Help    2:Split     3:Quit     4:Mark       5:Center 6:Del Line
  *EDIT*     7:Uppage  8:Downpage  9:Locate  10:Ins line  11:Input 12:Command
```

Figure 7.4 - Prefix Area of the Editor

The prefix area is controlled by the commands:

```
PREFIX ON                              NOPREFIX
PRE    OFF
       FLip
       CLear
```

Parameters:

ON is assumed if there is no parameter

OFF turns off prefix area

FLIP reverses the previous setting

CLEAR clears any pending prefix operations

The following prefix commands are automatically defined. Command names can be entered in upper or lower case. "n" is an optional number, which usually defaults to 1 if omitted.

In	Inserts *n* lines after this line.
An	(Add) Same as In. Inserts *n* lines.
Dn	Deletes *n* lines, starting at this line.
D*	Deletes to end of file.
DD	Indicates first and last line of a group of lines to be deleted.
Cn	Copies *n* lines.
CC	Indicates first and last line of a group of lines to be copied.
Mn	Moves *n* lines.
MM	Indicates first and last line of a group of lines to be moved.
F	(Following) indicates a target for an operation such as move or copy. The lines are moved/copied to after this line.
P	(Preceding) indicates a target for an operation such as move or copy. The lines are moved/copied to before this line.
"n	Duplicates the line *n* times.
""n	Duplicates a group of lines *n* times.
.xxx	Assigns label xxx to the line (as in the POINT command).
.	Does a MARK command for this line.
<n	Shifts the line's text *n* columns to the left.
>n	Shifts the line's text *n* columns to the right.
<<n	Shifts a group of lines left *n* columns.
>>n	Shifts a group of lines right *n* columns.
INP	Starts Input Mode after this line.
POW	Starts Power Input mode after this line. See the command POWERINP.
/n	Makes this line the current line and places the cursor in the nth position (column *n)* of the record. If column *n* is outside the current window display, the window is shifted left or right as needed. If *n* is not specified, the window is not changed, the cursor is placed at the beginning of the window, and the specified line becomes the new current line. This operation is done after other prefix operations.
/E	Makes this line the current line and places the cursor after the last nonblank in the text field. This operation is done after other prefix operations.
/END	Same as /E
CE	Centers the display around this line.
J	Joins the next line to the end of this line (JOIN command).
Ln	Converts *n* lines to lower case.
Un	Converts *n* lines to upper case.
LL	Converts a group of lines to lower case.
UU	Converts a group of lines to upper case.
FC	(Flip Case) Reverses the case of each letter in the line.
FF	Formats a group of lines, using Script. The left and right margin columns can be specified by the MARGINS macro: "margins left right" The default margins are 1 65. Type FF in the prefix area of the first and last lines of the group of lines to be formatted. Other associated macros: FF, UNFF, and FORMAT.
WW	Formats a group of lines, without using Script. It formats the text by invoking the Rexx macro $FLIN. All 3 parameters of the MARGINS command are honoured. The default MARGINS values are 1 65 0. Type WW in the prefix area of the first and last lines of the group of lines to be formatted. A blank line causes a formatting break, and can be used to separate paragraphs. Unlike FF, Script control words like .SP are not honoured. WW is faster than FF for small sections of text. WARNING: There is no undo for WW.

The optional numeric parameter n can be omitted, in which case the value 1 is used. n can be entered before

or after the command name, e.g. nI or In. For paired commands (such as <<), n can be entered on either the first or second command. If entered on both, the parameter entered on the line nearer the end of the file is used. Similarly, if mXn is entered, n is used and m is ignored.

Care must be taken when line numbering is in effect and numeric values are entered in the prefix area, since the "background" digits of the line number become part of the entered text, and are indistinguishable from the digits actually typed. The editor processes the text by ignoring any leading or trailing digits that are the same as the original "background" digits. For example, if the line number is 1234 and you type i13 over "123", then the editor sees the entered text as i134, ignores the trailing 34, and the prefix command is processed as i1. If i13 is meant, you should type i13b (b=blank).

Some prefix operations do their work by MARKing a group of lines, so they UNMARK any previously marked lines. These are D (in FLAG mode only), DD (in flag mode only), "", <, >, <<, >>, L, U, LL, UU, and FF.

Prefix operations are done in the order they were entered, except that operations entered on the same screen (by the same 3270 action key) are done from top to bottom of the screen. An exception to this rule is that the / prefix command (/n and /E) are done last. Also, an operation that requires more than one entry (such as DD or CC) is not done until it has been completely specified. Until then, it is "pending", and a pending message appears on the tab line.

Input Mode is turned off before a prefix operation is done. This means that the INP prefix command has no effect when combined with other prefix operations, unless it is done last.

Each prefix operation is done by an editor macro, so REXX ON (default) is required. You can define your own prefix operations, by using the DEFINE PREFIX command and writing the appropriate macro. See the DEFINE command description for more information.

Editor Commands

Notation

The notation *string* indicates a sequence of blanks, letters, etc. Leading blanks in character strings are considered part of the string starting with the second position after the command name. In the command descriptions, square brackets are used to indicate optional items.

Command Syntax

Parameters of commands may be separated by one or more commas or blanks. A blank is not required between the command name and the parameter unless the first character of the parameter is a letter (A to Z). A blank is not required following a flip character or column number suffix. The following commands require a blank in any case: SAVE, FILE, EXEC, RUN, LIST, MERGE, PURGE, INPUT, NAME, STORE. Commands may not be preceded by a slash (/).

In general, any abbreviation between the shortest abbreviation and the full command name may be used. For example, the following may be used for the CHANGE command: C, CH, CHA, CHAN, CHANG, CHANGE. An exception is the PRINT command, which allows P but not PR (PR is the PROMPT command). Another exception is the INSERT command, which allows I and INS but not IN, to avoid confusion with the INPUT command. In the following command descriptions, the shortest abbreviations are shown under the full form of the command.

The command delimiter character (normally semi-colon), the flip character, and all string delimiters must be special characters. They may not be letters, digits or blanks.

In logical expressions, the string modifiers F and Cn (where n is a column number) may be separated by blanks or commas, or the separator character may be omitted entirely. For example, (/ABC/(FC10)). The same applies to options on the CHANGE and SPLIT commands. Examples: C/ABC/DEF/*GV, SPLIT/XXX/C5IC10.

String Separator

Commands such as CHANGE require that strings be separated by a string separator character. The usual character used is a slash (/), but any non-alphanumeric character except the command delimiter character may be used. The string separator character is self-defining. For example, the first nonblank character after the command CHANGE is automatically defined as the string separator for that command.

Functional Summary of Commands

The following Editor commands are grouped into several categories. More information about any of these commands can be found after this summary under the heading "Editor Command Descriptions", or by typing "HELP xxx", where xxx is the command name. Some of the commands below are actually Editor macros and the REXX ON command (default setting for the Editor) must be in effect to use them.

Moving in the File:

=n	move to line number n
BOTTOM	move after the last line of the file
FIND	same as LOCATE, except the string must start in column 1
HUNT	same as FIND, except start at the top of the file
LAST	move to the last line of the file
LOCATE	locate the next line containing a specified character string
LOCATEL	LOC the next line that tests true against logical expression
NEXT	move towards the end of the file
SEARCH	same as LOCATE, except start at the top of the file
SEARCHL	same as LOCATEL, except start at the top of the file
TOP	move to the first line of the file
UFIND	(Upward FIND) same as ULOCATE, but string in column 1
ULOCATE	(Upward LOCATE) move towards the top, looking for a string
UP	move towards the beginning of the file
XF	(macro) extended find
XL	(macro) extended locate
XUF	(macro) extended up find
XUL	(macro) extended up locate

Displaying lines:

CENTER	center the current line in the middle of the screen
DOWNPAGE	shift the screen display 1 screen towards the end of file
DOWNWINDOW	shift the screen display towards the end of the file
LEFT	shift the window display left
LIST	display the file being edited, or an external file
NONUMBER	turn line numbering off
NUMBER	turn line numbering on
PRINT	display a number of lines, starting with the current line
RIGHT	shift the window display right
SCAN	display all lines containing a specified character string
UPPAGE	shift the screen display 1 screen towards the top of file
UPWINDOW	shift the screen display towards the beginning of the file
WINDOW	define starting and ending columns to be displayed

Making changes to the file (see also: Formatting):

ADD	add text to the end of a line
BLANK	set specified characters of a line to blanks
CHANGE	make a change to a line or group of lines
CHANGEL	make a logical change to a line or group of lines
COPY	copy lines from one place to another within the file
COPYCOL	copy text from one part of a line (or lines) to another part
DELETE	delete a line or group of lines from the file
DELETEL	logically delete a line or group of lines from the file
DUP	duplicate the current line a no. of times
INPUT	begin INPUT mode, to add lines to the file
INSERT	add a new line following the current line
JOIN	join the next line to the end of the current line
MDELETE	delete MINSERT unused lines or a group of lines
MERGE	bring a copy of an external file into the file being edited
MINSERT	add a group of new lines following the current line

MOVE	move lines from one place to another within the file
OVERLAY	overlay part of a line with new characters
POWERINP	begin Power Input mode
REPEAT	specify the no. of times to repeat the next BLANK or OVERLAY
REPLACE	replace an entire line
SHIFT	(macro) shift marked lines left or right
SORT	(macro) sort a file or part of a file
SPELL	(macro) invokes SPELL program
SPLIT	break the current line into two lines
TOLC	change characters to lower case
TOUC	change characters to upper case

Getting information:

=	get the line number of the current line
ATTRIB	get information (attributes) about a file
ECHO	cause information to be displayed in the command area
GETV	(macro) displays the value of a SETV name
HELP	request information about the editor or an editor command
NAME	display or set the file name for the edit session
NOSHOW	remove text displayed by SHOW command
SHOW	display function key or X cmd definitions; from a file
SIZE	get the total number of lines in the file
SPACE	get library space information for your userid
TAG	display or set the tag string of the file being edited
TIME	display time of day, date, cpu time, and number of users
USERS	same as TIME

Ending the edit:

END	same as QUIT
EXECUTE	store the file (as in FILE command) and then execute it
FILE	store the changed file in place of the original file
OFF	same as QUIT, but also terminates the MUSIC session
QQUIT	(quick quit) terminates without any messages or prompting
QUIT	terminate the edit without saving the file
RUN	same as EXECUTE, except use the MUSIC input file (/INPUT)
SAVE	similar to FILE, but do not terminate the edit
SETRC	set a job return code (exit code) for the edit

Full-Screen Mode:

AUTOSKIP	cause automatic cursor skip at end of text fields
CMDPFK	cause the cursor to be put into the command area
COLOR	define the color of various parts of the screen
CURSOR	control the placement of the cursor on the screen
DEFINE	define a program function key or the X command
FILL	put blanks at the end of each screen field
FS	start full-screen mode (for 3270-type workstations)
NOFILL	put nulls at the end of each screen field (the default)
NOFS	end full-screen mode
NUMBER	turn line numbering on or off
PREFIX	turn on or off the modifiable prefix area

WINDOW	control which columns of the records are displayed

Formatting:

FF	(macro) format marked lines or entire file using SCRIPT
FORMAT	(macro) format the file using SCRIPT
MARGINS	(macro) adjust margins before using FF
UNFF	(macro) restore file to original form before FF was used
UNFORMAT	(macro) restore file to original form before FORMAT was used

Working with Marked Lines:

COPY.	copy marked lines to another place in the file
DELETE.	delete marked lines
FF .	(macro) format marked lines using SCRIPT
MARK	define a group of lines to be operated on
MOVE.	move marked lines from one place to another
SHIFT	(macro) shifts marked lines left or right
SORT .	(macro) sorts marked lines
STORE	store marked lines in another file
TOLC.	change marked lines to lower case
TOUC.	change marked lines to upper case
UNMARK	remove the marking for a group of lines
=.	go to a marked group in the file

Other commands:

AIN	terminate hexadecimal input mode
ALPHA	terminate hexadecimal output mode
ARROW	use arrow pointer form of screen mode display
ASMFIX	align assembler source statements to specified columns
BEEP	beep the speaker the next time the screen is displayed
BOTH	obtain output in both hexadecimal and character form
BR	(macro) uses show area to display a file
BRIEF	stop automatic verification (displaying) of file changes
CALC	(macro) Perform a calculation, e.g. "calc 1+2"
CASE	ignore or respect case differences during string searches
CD	changes directories
CMDS	enable or disable MUSIC (*Go) commands
CREP	enable or disable command name replacement
DBCS	turn the double-byte character set on or off
DEFINE	define a program function key or the X command
DELCHAR	define the delete char. to be used by MDELETE & MINSERT cmd.
DELIM	change or remove the command separator character (normally ;)
ENQ	(macro) prevents more than 1 simultaneous edit of a file
FLAG	begin automatic flagging of changed lines
FLIP	define a "flip" character, used to control verification
HEX	obtain output in hexadecimal form
KEYS	(macro) Display and/or change function key definitions
LANGUAGE	turns on the language of your choice
LOG	control frequency of writing to log file (restart feature)
MD	makes a new directory
MSG	display a message line at the workstation

MSGS	suppress or enable all messages
NOARROW	turn off the arrow pointer form of screen mode display
NOCHANGE	indicate that there are no unsaved changes
NOFLAG	stop automatic flagging of changed lines
NOSCREEN	turn screen mode off
NOTRAN	turn output character translation off
OKREPL	suppress FILE/SAVE verification prompt for a file name
POINT	assign a 1-8 character label to a line
PRINT	print a file on a specified line printer
PROMPT	define or remove the prompting character
PURGE	remove (delete) a file (alias: ERASE)
RENAME	change the name of a file
REXX	enable or disable the use of Rexx procedures as Editor cmds
SCREEN	turn screen mode on
SEQ	put sequence numbers into each line of the file
SET	(macro) compatible command with CMS editor
SETV	assign a character string yyy to a name xxx
SUBMIT	submit a job to MUSIC batch or other batch processors
SUBSET	define which subset of editor commands is to be allowed
TABIN	specify input tab positions
TABOUT	specify output tab positions
TEXT	specify handling of input lower case and tab characters
TRAN	turn output character translation on
TREE	(macro) shows graphical display of directories
UNDELETE	restore a deleted line
VERIFY	control verification (displaying) of changes to the file
X	execute a predefined string of commands (see DEFINE)
XIN	begin hexadecimal input mode
ZONE	define the ending column for some commands such as CHANGE
*	signify a comment line

Editor Command Descriptions

```
ADD  string
A
```

This command adds *string* after the last non-blank character of the current line.

Spacing: One blank character must occur between the keyword and the first character of *string.* Any additional blanks are considered part of *string.*

Example: Before: **SAMPLE LINE**
Command: **ADD 1234**
After: **SAMPLE LINE1234**

```
AIN
```

This command can be used to restore alphabetic input format after an XIN command has been used.

```
ALPHA
ALP
ALFA
ALF
```

This command can be used to restore alphabetic output format after a HEX or BOTH command has been used.

```
ARROW
AR
```

(This command is not needed with full-screen editing.) The ARROW command can be used to get a slightly different display format for the old *screen* mode, i.e. the mode set by the SCREEN command. When ARROW is used, the current line is indicated by an arrow in the left margin, rather than by "=====" lines above and below the current line. This is useful on low speed ASCII screens, since the number of characters transmitted is less. Also, the number of lines displayed can be decreased by specifying a number on the SCREEN command. For example: SCREEN 9;ARROW.

```
ASMFIX  [n1,n2,n3]
```

This command is useful when editing an assembler source file. If no parameters are used on this command then it will attempt to align the operation code field to column 10, the parameter field to column 16, and the comments field to column 35. The 3 optional parameters on this command can be used to give other column numbers. All lines of the file are aligned, except assembler comments, object module statements, and lines starting with / or =.

Pointer: Set to the top of the file.

```
ATTRIB  filename
AT
```

The ATTRIB command displays some of the attributes of a specified file, namely its logical record length (LRECL), record format (RECFM), number of lines (or 0 if not known), and the file's allocated size in K (1K = 1024 bytes).

Record formats are:	F	Fixed-length records, uncompressed.
	FC	Fixed-length compressed records.
	V	Variable-length records, uncompressed.
	VC	Variable-length compressed records.
	U	Undefined record format.

Note: The number of lines is especially useful if you wish to merge some lines from the end of the file into the file being edited.

```
AUTOSKIP  ON
AUTOSK    OFF
          INPUT
          NOINPUT
          COMMAND
          NOCOMMAND
```

The AUTOSKIP command controls whether fields on the 3270 screen end with an automatic skip to the next modifiable field. Auto skip is OFF by default, but can be requested for displayed lines of the file: for command-mode lines (COMMAND or C) or Input Mode null lines (INPUT or I) or both (ON). The NOCOMMAND or NOC option turns off auto skip for command-mode lines. The NOINPUT or NOI option turns off auto skip for Input Mode null lines. The OFF option turns off both. The last displayed line of the file never ends with an auto skip. For example, AUTOSKIP INPUT is used by the POWERINPUT command to enable continuous typing in Input Mode.

If no parameter is specified, ON is assumed.

If more than one parameter is specified on the command, each is processed in sequence, as if given on separate commands. For example, "AUTOSK OFF I" is equivalent to "AUTOSK I NOC".

"AUTOSKIP COMMAND" should not normally be used when a PREFIX area is displayed, since typing would continue into the prefix area of the next line.

When NET3270 is used for 3270 emulation, Document Mode (Alt-E toggle) must be off in order for auto skip to work.

BEEP

This command causes the speaker at the workstation to beep the next time the FS screen is displayed. It is mainly used for programming purposes.

BLANK string
BL

This command replaces characters in the current line with blanks. The replacement is position dependent. Characters that appear in the command string replace characters in the same position on the line with blanks. Any character can be used to indicate where you want blanks. *String* may contain up to 78 characters, each of which will cause one blank to appear in the current line. This command is the complement of the OVERLAY command.

The BLANK command is useful in combination with the REPEAT command. If there is a preceding REPEAT command, the BLANK is repeated for the specified number of lines, beginning with the current line.

Spacing: One blank should be present between the keyword and *string*. All other blanks are part of *string*.

Example:

Before:	**SAMPLE LINE**		
Command:	**BL XX**	**or**	**BLC2 XX**
After:	**S PLE LINE**		

Specifying "BLC2 XX" would have the same results as "BL XX" above. For more information about using a starting column suffix, see the topic "Advanced Features" later in this chapter.

BOTH

This command causes lines to be displayed such that each printable character is shown above the left hand portion of the hexadecimal representation. It also temporarily turns off screen mode if it is on at the time the command is issued. The command HEX or ALPHA can be used to tell the Editor to use another output

representation.

```
BOTTOM
B
```

The BOTTOM command moves the pointer to one line after the last line in the file.

Example:

```
Before:   SAMPLE LINE          <--Pointer
          ANOTHER LINE
          THE LAST LINE
Command:  B
After:    SAMPLE LINE
          ANOTHER LINE
          THE LAST LINE
                               <--Pointer
```

```
BR filename
```

The BR command uses the SHOW area (the bottom part of the screen) to browse a file. This lets you see 2 files at the same time.

Once browse has started, some extra function keys are defined, as indicated in the display. For example, F19 goes down 1 page in the browsed file, and F15 ends the browse. (On most PC's, you get F(n+12) by pressing Shift-Fn. For example, F15 is Shift-F3.)

Also, while in BR browse, you can enter commands that apply to the browsed file:

```
BR TOP                    BR BOTTOM
BR UP n                   BR DOWN n
BR LEFT n                 BR RIGHT n
BR LOCATE string          BR FIND string
BR =n
```

The normal abbreviations can be used, e.g. BR T, BR B

```
BRIEF
```

This command puts the Editor into BRIEF status, which bypasses display of the new current line after the following commands: ADD, BLANK, CHANGE, CHANGEL, DELETE, DELETEL, FIND, HUNT, LAST, LOCATE, LOCATEL, NEXT, OVERLAY, SEARCH, SEARCHL, SPLIT, UP. Display of the current line does not resume until a VERIFY command is given. (VERIFY is assumed when the Editor begins.)

```
CALC valid-REXX-expression
```

This Editor macro allows you to evaluate a REXX expression, and have the result displayed in the message area. This is useful for performing arithmetic calculations without exiting the Editor or starting another session; sometimes you want the result and the current screen to be displayed together. For example, CALC 11+7 displays the following in the message area:

```
** 11+7 = 18
```

Note: Note that a "valid REXX expression" can consist of a simple arithmetic calculation, or a number of REXX function calls.

Examples:

```
calc 5*7 + 22/7
calc 2**16              (2 to the power 16)
calc (3.5-1.9)*1.09
calc d2x(26)            (convert decimal to hex)
calc x2d(1a)            (convert hex to decimal)
calc date()             (current date)
calc userid()
```

```
CASE [IGNORE   ]
CA   [I        ]
     [RESPECT  ]
     [R        ]
     [MIXED    ]
     [M        ]
     [UPPERCASE]
     [U        ]
```

The CASE command specifies whether or not differences between upper and lower case characters should be ignored when the editor is searching for a string of characters (IGNORE/RESPECT), and/or whether text entered by the user should be left as is (MIXED) or converted to upper case (UPPERCASE).

The IGNORE/RESPECT setting affects all commands which involve string searches, such as LOCATE, LOCATEL, SCAN, CHANGE, etc. It does not affect changes typed over screen text in full-screen mode, or *string2* of the CHANGE command, or commands such as ADD or INSERT.

With CASE RESPECT, upper and lower case characters are considered different when searching.

With CASE IGNORE (the initial default), upper and lower case characters are considered the same, for the purposes of searching for a specified string of characters. This is as if all characters were in upper case. The file *COM:EDITOR normally contains the command CASE IGNORE, so that setting is the default for most edits.

When CASE is specified without a parameter, the current IGNORE/RESECT setting is shown. To show the

current UPPER/MIXED setting, use the TEXT command without any parameter.

The command CASE MIXED is equivalent to the command TEXT LC. CASE UPPERCASE is equivalent to the command TEXT UC.

More than one parameter can be used on a CASE command. For example, CASE UPPERCASE RESPECT. This can be abbreviated to CA U R.

Example:

```
CA I  The command "LOCATE abc" finds "ABC" and "aBc" as well as "abc". The
      command "CHANGE/mcgill/McGill/*" changes "Mcgill", "MCGILL", "MCGill",
      etc. to "McGill" (assuming TEXT LC is in effect).
```

CD dirname
CD..
CD

This command changes directories for tree-structured file system. Same as the MUSIC command CD.

CENTER
CE

The CENTER command moves the full-screen display up or down in the file, so that the current line will be in the center of the screen display. By default, F5 is defined as CENTER. To shift the display up or down in the file, place the cursor at a line and press F5 to center that line on the screen.

CHANGE [/string1/string2/][n][G][D][F][Cn][V][P][S]
C

The CHANGE command replaces the first occurrence of a character string with another in a number of lines (normally 1), beginning with the current line. Within the range of lines, only the first occurrence of the string in each line is replaced. The number of lines searched depends on the parameters specified in the command. The last line searched becomes the current line. If the line pointer is at EOF (end of file), an automatic TOP command is done before the change.

When a CHANGE command is entered with no parameters, the editor's CHANGE Panel is displayed. The CHANGE Panel lets you do a change operation by filling in fields on the panel.

The search for the string is affected by whether or not CASE IGNORE is in effect. CASE IGNORE ignores differences between upper and lower case characters. Refer to the CASE command.

The parameters n, G, D, F, Cn, V, P, S are not required. If used, they can be specified in any order. Commas are not required to separate the parameters.

Parameters:

string1 Character string which is to be replaced.
string2 Replacing character string. If omitted, *string1* is deleted. The final string delimiter (/) may be omitted if no parameters are used.
n Number of lines searched. If not specified, 1 is assumed. If * is specified, the search will be applied to the rest of the file, starting with the current line.
G If specified, the change will be applied to ALL occurrences of *string1* in as many lines as specified by the *n* parameter.
D If specified, the remainder of the line (or zone) following the changed string is replaced by blanks. ("D" stands for "delete".)
F If specified, the change is made only if "string1" begins in column 1 of the line (or in column n if Cn is used). ("F" stands for "first" or "FIND mode".)
Cn If specified, the change is made only if "string1" occurs starting in column n or later.
V If specified, all lines which are changed are displayed regardless of whether the editor is in the BRIEF status or the VERIFY status. If V is not specified and the range is more than one line, the changed lines are not displayed. ("V" stands for "verify".)
P If specified, the editor prompts for the user's permission to make a change. The user can respond Y (for yes), N (for no), S (to stop the execution of the CHANGE command), G (to continue command execution without further prompts), or = (to find the current line number).
S If specified, it causes the screen to be displayed for each line found (as in 3270 "screen mode"). On a 3270 terminal, the PA2 key must be pressed between screens.

Examples:

`C/ABC/DEFG/`	Changes the first occurrence of ABC to DEFG in the current line. The line pointer is not changed.
`C/OLD/`	Removes the characters OLD from the current line.
`C/OLD//3,G`	Removes all occurrences of OLD from the next 3 lines, beginning with the current line. The line pointer is moved down 2 lines.
`CH /XXX/YY/ 20,F,C11`	In the next 20 lines, changes XXX to YY wherever XXX occurs starting in column 11.
`C/ /*/G`	Changes every blank to * in the current line.
`C//AAA/`	Adds AAA to the beginning of the line.
`CHANGE/E/ES/*GV`	This will change all occurrences of E to ES in the rest of the file, starting with the current line. Changed lines are displayed.
`CHANGE$/$.$`	This changes "/" to "." in the current line. The dollar sign ($) acts as the string delimiter in this case.

CHANGEL (logical expression)/string1/string2/[parameters]
CL

This command is similar to the CHANGE command except that, for each line searched, the change will only occur if the logical expression is true for that line.

The *n*, G, D, F, Cn, V and P parameters are also permitted on this command as they are on the CHANGE command.

Example:

If it is desired to change all occurrences of the string WRONG to the string RIGHT in all lines which also contain the string MAYBE, but not if the line contains one and only one of the strings HOT and CHA, the commands would be:

```
TOP
CL(/MAYBE/AND.NOT/(/HOT/XOR/CHA/))/WRONG/RIGHT/*GV
```

CMDPFK
CMDPF

This command is used only in 3270 full-screen mode, and is intended primarily for use in function (PF) key definitions. It causes the cursor to be put into the command area the next time the screen is displayed. The standard definition of F12/24 is equivalent to a CMDPFK command.

CMDS ON
OFF
NONE

The CMDS command controls execution of MUSIC (*Go) commands and programs entered as commands in the editor. Normally, when command xxx is entered in the Editor and xxx is not an Editor command or abbreviation, the system searches for macro xxx (normally file xxx.MAC). If no macro is found, xxx is executed as a MUSIC command (*Go command or the name of a program to be executed).

CMDS ON is the initial setting (no restrictions).

With CMDS OFF, a / must be entered before the MUSIC command or program name. This does not affect execution of macros. CMDS OFF could be used to prevent command typing errors from executing files or *Go commands by mistake.

With CMDS NONE, no MUSIC commands or programs can be executed from within the editor, even with a / (slash).

Note: REXX OFF does not prevent execution of explicit MUSIC commands (/xxx), provided CMDS NONE is not in effect.

CNTINFO

The CNTINFO macro shows the file usage and creation date of the current file. You can reset a file's count to 0 and set the creation date to today's date. Type "HELP COUNT" for more information.

```
COLOR
COLOR fieldname=color fieldname=color...
COLOR BASE
COLOR DEFAULTS
```

The COLOR (COLOUR) command redefines the color of various parts of the editor screen. This assumes that the workstation supports 3270 extended data streams (i.e. the Start Field Extended order). Not all workstations support all the possible colors. See the topic "Changing Screen Colors" in the section "Advanced Features" later in this chapter for details.

```
COPY  i,j,k
CO
COPY.
CO.
```

The COPY command copies a group of lines from one part of the file to another by reference to the line numbers of the lines. The original group of lines will not be deleted. The last line of the group of lines inserted will become the new current line. The = command is useful for finding the line number of a line.

If the MARK command has been used to define a group of lines, the "COPY." command can be used to copy the lines to after the current line (or to the top of the file if the last command was TOP). For more information about marking a group of lines for copying, see the topic "Marking a Group of Lines" earlier in this chapter.

COPY differs from MOVE in that the MOVE command deletes the source lines, while the COPY command does not.

Parameters:

i Line number of the line after which a group of lines is to be copied. If i is 0, the lines are copied to the beginning of the file.
j Line number of the first line of the group of lines to be copied.
k Line number of the last line of the group of lines to be copied.

Line numbers may be specified in any of the following forms: n, *+n, *-n, LAST, LAST-n, where n is a number, * means the current line, and LAST means the last line of the file.

Examples:

```
COPY 10 25 30          copies lines 25 to 30 to after line 10.
COPY 42 35 35          copies line 35 to after line 42.
COPY 0 18 20           copies lines 18 to 20 to the start of the file.
COPY * LAST-2 LAST     copies the last 3 lines of the file to after the current line.
```

```
COPYCOL  c1 c2 c3 [n] [NOPROMPT]
COPYC             [.] [N       ]
CC
```

The COPYCOL command copies text from one part of a line to another part, in each of one or more lines starting with the current line. The copied text replaces existing text in the destination field of each line. The last line modified becomes the new current line. This command is not affected by the zone setting.

Parameter *c1* is the starting column number of the text to be copied. *c2* is the ending column number of the text to be copied. *c3* is the starting column number of the destination field. *n* is the number of lines to be modified (the default is 1 line). A dot (.) specifies a marked group of lines. If NOPROMPT or N is specified as a final parameter, the normal verification prompt for this command is bypassed.

Examples:

```
COPYCOL 11 15 21 10
```

Copies columns 11 through 15 to columns 21 through 25, in 10 lines starting with the current line. The original contents of columns 21-25 is replaced.

```
CC 40 59 7
```

Copies columns 40-59 to 7-26 in the current line.

```
CC 10 10 20 100 N
```

Copies column 10 to column 20 in 100 lines. No prompt is done.

```
CREP ON
     OFF
```

The CREP command is used to enable or suppress automatic command name replacement. Command name replacement is requested by the DEFINE command in the form:

```
DEFINE CMD xxx yyy
```

The above command would cause the command name XXX (entered by the user) to be processed as YYY. CREP OFF disables command replacement. The definitions are still retained, and can be later re-enabled by CREP ON.

```
CURSOR LOCATE
CU     NOLOCATE
       n
       n TEMP
       END
       PREFIX
       PREFIX TEMP
```

The CURSOR command controls output cursor positioning during full-screen mode.

When the Editor displays the screen, it normally puts the cursor at the first position of the current line or the command area. However, commands such as CURSOR, LOCATE and SPLIT can cause the cursor to be displayed at a different position in the current line.

Parameters:

LOCATE — places the cursor at the start of the found string after any of the commands: LOCATE, UPLOCATE, FIND, UPFIND, SEARCH, HUNT. You can put a CURSOR LOCATE command into your private EDITOR file to make this option the default for all your edits. Abbreviation is LOC.

NOLOCATE — cancels the LOCATE option. Abbreviation is NOLOC.

n — is a number from 1 to 72. It is a column position number relative to the start of the screen field for the current line. 1 refers to the first character of the field, 2 to the second character, etc. *n* places the cursor at the specified column position whenever the cursor is not put out in the command area, provided a command such as LOCATE or SPLIT or CURSOR END or CURSOR n TEMP does not result in a different placement. This stays in effect for the remainder of the edit. To cancel the effect of this command, use CURSOR 1.

n TEMP — places the cursor at position *n* in the screen field for the current line, but only for the next screen display. Abbreviation T may be used for TEMP, as in CU 15 T. This command is intended mainly for function key definitions. For example, DEFINE F1 INSERT ABC---XYZ;CURSOR 4 TEMP.

END — is similar to CURSOR n TEMP, except that the cursor is placed after the last nonblank character in the field.

PREFIX — Causes the editor to place the cursor at the first column of the prefix area, rather than at the beginning of the record. This persists until reset by the command CURSOR 1 or CURSOR n. PREFIX may be abbreviated PRE.

PREFIX TEMP — Places the cursor at the first column of the prefix area, but for the next screen display only. TEMP may be abbreviated T, as in CURSOR PRE T.

Examples:

```
CURSOR 41
CU 15 T
CUR LOC
```

```
DBCS ON
     OFF
```

The DBCS command is used to turn Double-Byte Character Set mode on or off for the Editor. DBCS mode is needed to handle the special 2-byte characters (representing pictographic symbols) used in some Asian languages such as Japanese (Kanji), Chinese, and Korean.

Notes:

1. In order to set DBCS mode on, you must be using a workstation that supports DBCS display and input.

2. When the Editor starts, DBCS mode is automatically set on if a double-byte national language (such as Kanji) is in effect. Also, DBCS mode is set when the Editor LANGUAGE command is used to set a DBCS language, e.g. LANGUAGE KANJI implies DBCS ON. Setting a non-DBCS language turns off DBCS mode, e.g. LANGUAGE ENGLISH implies DBCS OFF.

3. Using the Editor DBCS command affects only the current edit session, and does not change the language setting.

4. Internally, a string of 0 or more double-byte characters is preceded by a Shift-Out (hex 0E) byte and followed by a Shift-In (hex 0F) byte. Normal text (single-byte) and double-byte text can be mixed in the same record.

5. Since case (upper or lower) is not defined for double-byte characters, case is always significant within double-byte text, and double-byte text is not affected by the commands TOUC, TOLC.

6. Character strings in commands such as LOCATE and CHANGE can be double-byte or mixed. However, the command name and syntax items such as delimiters and options must be single-byte text.

7. The Editor SPLIT command in DBCS mode operates on the entire record, rather than on just the ZONE part of the record.

See also: LANGUAGE.

```
DEFINE  Fn     commands
DEF     PFn    <TAB>
        PFPn   <RETRIEVE>
        PFAn
        X

DEFINE  CMD commandname commands
DEF     CMD 0
        PREFIX pppp tt mmmmmmmm
        ENTER commands
        INPUT commands
```

n is a function key number (1 to 24). *commands* is a string of one or more editor commands (or macros),

separated by the command delimiter character (normally semicolon ";").

Abbreviations: DEF (for DEFINE), <RETR> (for <RETRIEVE>), PRE or PREF (for PREFIX), ENT (for ENTER), INP (for INPUT).

You can customize the operation of the program function keys by using the DEFINE command, which defines a function key as equivalent to any string of Editor commands. Any function keys defined in this way override the standard default function key definitions.

The DEFINE command can also change the definition of the X command, redefine command and macro names (by the CMD option), define prefix area commands (by the PREFIX option), specify additional commands to be done whenever the Enter key is pressed, and specify a macro to process text entered during Input Mode.

Defining Function Keys and the X Command:

n in the Fn or PFn parameter is an actual program function key number (1 to 24). Fn and PFn mean the same thing.

n in the PFPn (primary PF key) or PFAn (alternate PF key) parameter is a program function key number (1 to 12), which maps to either PF1-12 or PF13-24 on your actual keyboard. (The parameters PFPn and PFAn are rarely used, and should be avoided for most newer types of workstations.)

commands is a string of 1 or more Editor commands, separated by the command delimiter character (normally semicolon, (;)).

The DEFINE command must be the only command on the input line. An X command string must not contain an X command. Function keys may not be available on some non-3270-type workstations, but the DEFINE command can make the X command equivalent to a string of Editor commands. Then typing the command X causes the specified sequence of commands to be executed. Also, the command Xn (where n is 1 to 24) can be used to execute the command string defined for PFn, on any type of workstation.

If the command string is omitted, the function key or X command is made undefined.

You can define a function key to retrieve previous command lines with the <RETRIEVE> (abbr. <RETR>) parameter. Example: define pf24 <retrieve>. Up to 10 command lines are saved. It is different from other function keys in the editor, in that the command area is ignored. Press the key repeatedly to retrieve successively older commands.

You can define a function key as a tab operation (moves the cursor to the next tab position, as defined by the TABIN command), by using the <TAB> parameter. Example: define pf4 <tab>.

Redefining Command Names: DEFINE CMD xxx yyy

The form DEFINE CMD xxx yyy, where xxx is a command name and yyy is any character string, tells the editor to change the command "xxx tttt" (entered by the user) to "yyy tttt" before processing it. The original command name xxx is replaced by yyy.

Example:

```
define cmd list print
```

Then "list abc" (entered by the user) is processed as "print abc".

This replacement is done before searching the editor's command table, and before passing the command to

Rexx. If yyy is omitted, then xxx is considered to be undefined to the editor (and the command will be passed to Rexx). These DEFINE commands are cumulative. A table of keyword replacements is built in a 2K area. DEFINE CMD 0 clears the table.

xxx can be specified as fff-ggg, where fffggg is the full command name and fff is the minimum abbreviation.

Notes on command replacement:

- If a command name appears twice in the replacement table, the last matching entry (latest definition) is used.
- xxx should not start with % or /, since replacement is not done on commands starting % or /.
- If yyy is longer than xxx, long parameters may be truncated.
- Command replacement is not done on commands from Rexx macros.
- Command CREP OFF turns off replacement.
- Be careful when using macro or program names that contain digits or special characters, since the first non-letter is taken by the editor as the end of the command name. The leading letters may be subject to command name replacement. (See the next note.)
- If the last character of yyy is ! (exclamation point), it is changed to a blank when replacement is done. This can act as a separator between the macro name and the parameters when the original editor command does not require a blank after the command name (e.g. INSERT).

  Example:

```
def cmd i-insert Xi!
```

  causes i* to become xi * (where XI is a macro).

Defining Prefix Area Commands (PREFIX):

The form DEFINE PREFIX pppp tt mmmmmmmm defines a prefix area command name pppp, of type tt, to be processed by macro name mmmmmmmm. pppp (1 to 4 characters) is the name of the prefix command. tt (1 or 2 characters) defines the type of prefix command (see below). mmmmmmmm (1 to 8 characters) is the name of the macro that performs the prefix operation. The macro name should not be the same as the name of a normal editor command. The keyword PREFIX may be abbreviated PREF or PRE.

The standard prefix commands (I, D, C, MM, etc.) are predefined. You would use the DEFINE PREFIX command only to add your own prefix operations, or to undefine or rename existing ones. When parameters tt and mmmmmmmm are omitted, the prefix command pppp is made undefined. Refer to topic "Prefix Area" earlier in this chapter for more information on the prefix area and prefix commands.

The types tt for DEFINE PREFIX are:

1 Non-paired command, with no target required (e.g. In, Dn)
1T Non-paired, requiring a target (e.g. Cn, Mn)
1L Same as 1, except the operation is done last, after other prefix operations (e.g. /).
2 Paired, no target (e.g. DD, >>)
2T Paired, requiring a target (e.g. CC, MM)
2L Same as 2, except the operation is done last, after other prefix operations.
F Target, of type "following" (e.g. F)
P Target, of type "preceding" (e.g. P)

For example, suppose you wish to use prefix command XX to print a group of lines. XX is entered in the prefix area of the first and last lines of the group. You would define it by:

```
define prefix xx 2 myxxmac
```

"myxxmac" is the name of the macro to be invoked and "2" is the type of prefix operation (in this case, a "paired" command). You would have to write the editor macro in Rexx, and store it as file MYXXMAC.MAC. The macro would do whatever is needed to print the lines. It would be called by the editor as "MYXXMAC n1 n2 0 0 XX op", where n1 and n2 are the line numbers of the first and last lines of the group; XX is the prefix command name; op is the parameter (if any).

Defining Commands for the ENTER Key

The command DEFINE ENTER xxx, causes the command string xxx to be done whenever the Enter key is pressed in full-screen (FS) mode. This is in addition to the normal effects of the Enter key. The commands xxx are done after any prefix operations or command-area commands. The command DEFINE ENTER (with xxx omitted) cancels this.

Defining a Macro to Process Input Mode Text

The command DEFINE INPUT xxx, where xxx is normally the name of an editor macro, causes xxx to be invoked to process lines added during full-screen (FS) Input Mode. The command DEFINE INPUT (with xxx omitted) cancels this. For example, Power Input Mode (the POWERINP macro) is supported by using DEFINE INPUT $FINP. See the comments in the $FINP macro (file $FINP.MAC).

The macro xxx is invoked with the following parameters:

xxx NOTINP	Just before screen display for Command Mode.
xxx INPSTART	Just before screen display for Input Mode.
xxx n1 n2	After lines n1 thru n2 have been added to the file by Input Mode. This is done when Enter or a function key is pressed in Input Mode, if 1 or more lines were entered. The macro is invoked before any prefix operations are done. Input Mode is turned off before the macro is called; the macro should turn Input Mode back on if it wants Input Mode to continue.

For more information see the topic "Defining Prefix Macros" later in this chapter.

Examples:

```
DEFINE F13 TOP
```
Define the F13 key as equivalent to a TOP command.

```
DEFINE X LOCATE ABC;CHANGE/ABC/1234/
```
Define the X command as equivalent to 2 successive commands: LOCATE ABC and CHANGE/ABC/1234/.

```
DEF CMD AT-TRIB /ATTRIB
```
Any ATTRIB command (minimum abbreviation AT) entered by the user is passed to Rexx as MUSIC command /ATTRIB. It is not processed as the normal editor ATTRIB command.

```
DEF CMD LIST
```
Makes the LIST command name undefined to the editor. It will be passed to Rexx. (Note: abbreviation LI is still defined as the editor LIST command.)

```
DEFINE CMD TR-OUVER LOC
```
Defines TROUVER (minimum abbreviation TR) as an alias for the LOCATE command.

("Trouver" is French for "find").

Refer to the topic "Advanced Features" for examples of the DEFINE command and how to make definitions automatic whenever you use the Editor.

```
DELCHAR [x  ]
DELC    [OFF]
```

This command defines the delete character for use with the MINSERT and MDELETE commands. *x* is the delete character to be used. It must be a special character, not a letter or a digit. If *OFF* is specified, the delete character is made undefined. If the command is used without a parameter, the current delete character is displayed. The default delete character is ":" (the colon).

When a file is saved by an Editor SAVE, FILE or EXEC command, the Editor looks for lines with the delete character in column 1 (followed by a blank or another delete character). If such a line is found, this probably means that you forgot to use MDELETE. The Editor issues a warning message: `LINE WITH DELETE CHARACTER FOUND. CONTINUE SAVE?` Answer Y or YES if you wish the file to be saved as is. Answer N or NO if you wish the save not to be done. You can then use MDELETE and redo the save operation. (If editor messages are currently suppressed by the MSGS OFF command, this verification prompt is not done.)

```
Form 1:  DELETE [n][,(logical expression)]
         DEL
Form 2:  DELETE /string/[,(logical expression)]
         DEL
Form 3:  DELETE.
         DEL.
```

The DELETE command removes lines from the file, beginning with the current line. (If you want to delete an entire file see the PURGE command.)

With form 1, it deletes *n* lines from the file, beginning with the current line. With form 2, it deletes lines down to, but not including, the line containing *string*, starting with the current line. Logical expressions are optional. With form 3 (DELETE.) the dot at the end of the command signifies that a group of lines previously marked with the MARK command are to be deleted.

Parameters:

n — It specifies the number of lines that are to be deleted starting with the current line. If * is specified, the deletion will apply to the remainder of the file. The default value of *n* is 1.

(logical expression) — If it is specified, only the lines for which the logical expression is true will be deleted.

string — Character string which signifies the end of deletion. The current line and all lines after the current line down to, but not including, the line containing *string* are deleted.

Examples:

```
DEL                 Removes the current line from the file.
DEL10               Deletes 10 lines from the file starting with the current line.
DELETE /STOP/       This deletes the current line and all lines after it down to, but not
                    including, the line containing the string STOP.
DELETE *,(/TEST/(F,C73))

                    This deletes all lines in the file with the character string TEST in
                    columns 73 to 76, from the current line to the end of the file.
```

See also DELETEL, MDELETE, and UNDELETE.

DELETEL (logical expression1)[,(logical expression2)]
DELL

The current line and all following lines are deleted, down to but not including the next line which tests true against the *(logical expression1).* If no such line exists, an error message is displayed and the file is not changed.

If the *(logical expression2)* is specified, only those lines for which the *(logical expression2)* is true will be deleted. Parentheses are required around both logical expressions.

DELIM [x]
OFF
ON

The DELIM command changes the command delimiter character to the character specified by *x.* The delimiter character *x* may be any non-alphanumeric character. It separates editor commands entered on the same line.

If the DELIM command is not used, the command delimiter character is assumed to be a semicolon (;) by default. If the command DELIM is entered with no parameter, then no character is the delimiter character, and only one command may be entered on a line.

The command DELIM OFF disables the delimiter character, and only one command can subsequently be entered on a line. However, the original delimiter character (if any) is remembered, and can be enabled later by the command DELIM ON. This makes it possible to temporarily undefine the command delimiter (for example, in order to use the INSERT command to insert text that may contain the delimiter), without knowing what the original delimiter character is. If DELIM ON is used after a DELIM command with no parameter, nothing is changed.

Example:

```
delim $
l xaz$c/a/y/$l abc
```

```
delim off
insert The $ in this line is not a delimiter
delim on
last$insert xyz
```

DOWN [n]
DN

The DOWN command is the same as the NEXT command. Refer to the description of the NEXT command.

DOWNPAGE
DOWNP
DNP

This command is used only in 3270 full-screen mode. It shifts the screen display to the next screen (page) in the file (towards the end of the file). The first line displayed on the new screen will be the line after the last line on the current screen. This command corresponds to F8 by default.

DOWNWINDOW [n]
DOWNW
DNW

The DOWNWINDOW command is used only in 3270 full-screen mode. It shifts the screen display (window) towards the end of the file. The parameter *n* is the number of lines by which the screen is to be shifted. If *n* is omitted, 6 lines is assumed.

DUP [n]

This command causes the current line to be duplicated *n* times. If *n* is not specified, it is assumed to be 1.

```
ECHO  [string]
      [NAME  ]
      [MSG   ]
```

This command causes the current line or a specified text string to be displayed in the command area of the screen if the user is in the full-screen mode. The file name or the screen message area can also be displayed. This command is mainly used in conjunction with function keys. The Editor will ignore this command if the user is not in the full-screen mode.

If no parameter is specified, the current line will be displayed from column 1 to the rightmost end of the window setting in the command area. If *string* is specified, *string* will be displayed in the command area. If *NAME* is specified, the file name from the title line of the last full-screen display will be displayed in the command area. *(NAME* can be abbreviated to *NAM* or *NA.)* If *MSG* is specified, the contents of the message area at the bottom of the last full-screen display will be displayed in the command area. In any case, if the length of the line to be displayed is longer than the command area, the line will be truncated to match the length of the command area. The cursor is also placed in the command area. The position of the line pointer is not changed by this command.

Command delimiter characters which appear after the ECHO command are not honoured, provided the ECHO command is not preceded by other commands on the same line. This allows the parameter to be a series of Editor commands.

Spacing: One blank should be present between the command name and *string.* All other blanks are part of *string.*

Example:

Enter the following in the command area:

```
DEF F8 ECHO MERGE F1 001 002;LOC ABC
```

Press the F8 key and the following is displayed in the command area:

```
MERGE F1 001 002;LOC ABC
```

Change the line numbers (001, 002) accordingly and press ENTER key to execute the MERGE and LOC commands.

```
END <n>
```

Same as QUIT command. The END command terminates the Editor session without saving a copy of the changed file.

If you have made changes to the file but have not issued a FILE command to make the changes permanent, you will be prompted for permission to end the edit session. Enter YES or Y to end the session, or NO or N to cancel the END operation and continue editing.

See QUIT for description of *n* parameter.

```
ENQ
```

This macro is used to prevent more than 1 simultaneous edit of a file. Invoke once only at the start of an edit session. Subsequent invocations will be ignored.

Note: Warning: This macro enqueues on only the first 22 chars of the file name (after "userid:" part has been added if not already there). So for long names, it could erroneously report that the file is already being edited.

```
EXECUTE  [name]
EXEC
EX
```

This command saves the changed file in your library under the name *name*, and then automatically requests MUSIC to execute the program contained in that file in a way equivalent to the user typing the MUSIC command "EXEC name". If *name* already exists in the user's library, the Editor will prompt the user whether to replace the existing file or not. If YES (or Y) is replied, the edited file will replace the previously existing file, and the EXEC will be done. Otherwise, no operation will be done. The prompt is not done if replacing an existing file which was read by EDIT at the start of the edit. File attributes can also be specified after the file name, as in the FILE command. If no name is specified on the Editor EXEC command, the name on the original EDIT command or on the last NAME command will be used. If desired, column number limits (as on the FILE command) may be specified before *name*. The parameter *name* may be /INPUT.

If this command is used in the User Data Set version of the Editor, a file name must be specified with the command. The Editor will try to replace the named file by the temporary updated version of the UDS file being edited. If the replacement is successful, a request to execute the program in the named file will be made to MUSIC. The original UDS file is not updated, and the edit is terminated.

```
FF *
FF .
```

The FF macro formats part or all of your file using MUSIC SCRIPT. The lines to be formatted can contain SCRIPT control words such as .SP and .PA, which will be honoured. The margins are from columns 1 to 65 by default, but can be changed by the MARGINS command. There is also a paired prefix area operation called FF. See the topic "Prefix Area" earlier in this chapter. One of the following parameters must be included with the FF command:

* - formats the entire file
. - formats only the marked lines (MARK command)

UNFF undoes the immediately preceding FF command or FF prefix operation.

Note: Another way of formatting text is to use the WW prefix area command. See the topic "Prefix Area" earlier in this chapter.

See also UNFF and MARGINS commands.

```
FILE  [m] [n] [name] [PRIV] [XO] [CNT]
              [*   ] [PUBL]
                    [SHR ]
```

The FILE command is the usual way of terminating an Editor session and saving the updated file. This command saves or replaces the updated file directly into your library.

Parameters:

name	Specifies the name for saving (or replacing) the updated file in your library. If *name* already exists in your library, the Editor will ask whether to replace the existing file or not. If the user replies with YES (or Y), the edited file will replace the previously existing file. No save operation will be done if the user's reply is not YES. Note that if *name* is not specified, the Editor will use the name which was specified on the EDIT command or on the last NAME command, or prompt you to enter a file name if no name is known to the Editor.
m n	These parameters specify the first and last column numbers, respectively, that will be saved for each line. Remaining columns are filled with blanks. If only one number is supplied it is assumed to be n and m will be assumed to be 1. The column numbers must appear before the "name" parameter. If omitted, each line will be saved in full. When you specify column numbers, the editor asks for verification before doing the command. Enter yes (or y) to allow the command to continue. Enter anything else to cancel the command. (If editor messages are currently suppressed by the MSGS OFF command, the verification is not done.)
PRIV	The file will be made private.
PUBL	The file will be publicly readable and in the common index.
SHR	The file will be publicly readable but not in the common index. Other users must specify the owner's userid in order to access the file, as in userid:filename.
XO	The file will be execute-only.
CNT	The system will maintain a usage count for the file (not for UDS files). The count is increased by 1 each time the file is opened. It is displayed by the "ATTRIB filename" command in *GO mode. Note that using the editor to modify the file resets the count to zero, since the editor always recreates the file.

Notes on File Attributes

If an existing file is edited and then saved back by using the SAVE, FILE or EXEC command without specifying a file name or attributes, all the original attributes of the file are preserved. This technique should be used when it is desired to change a file without changing its attributes. In other cases, PRIV is used as the default attribute.

Examples:

`FILE 1,72,FILEAB`
saves columns 1 to 72 inclusive for each line in the updated file, under the name "FILEAB".

`FILE`
replaces the original file with the updated version of the file.

`FILE FILE1,PUBL,XO`
saves the updated file as a public execute-only file in your library under the name "FILE1".

Messages:

`INVALID COMMAND, or INVALID OPERAND`
Something is wrong with the command name or the parameters.

`FILE IS EMPTY`
This is a warning message telling you that the file being edited is empty (has no lines). The FILE command is allowed (not for UDS files).

`INVALID FILE NAME, PLEASE TYPE A NEW NAME`
The file name specified in the command is incorrect. Check that the file name is of the form NAME or userid:NAME, where NAME is 1 to 17 characters long and userid is the owner's userid. Valid characters are letters (A to Z), digits (0 to 9), and the special characters $ # @ _ and period (.). The first character of NAME must not be a digit or period. Case is not significant in file names, since lower case letters are automatically converted to upper case. The NAME part can be preceded by directory names, as in ABCD:TEST\NAME or ABCD:PROGS\TEST\NAME. The special names /INPUT, /HOLD and /REC are also allowed in most cases.

`TYPE NAME OF FILE TO BE REPLACED OR CREATED`
No name is currently associated with the edit session, and you did not specify a file name on the command. Enter the name of the file you wish to replace (if the file already exists) or create (if this is a new file).

`ONLY COLUMNS n TO m WILL BE SAVED. TYPE YES OR NO.`
You have used the column number option on the command (1 or 2 column numbers were specified as the very first parameters). Type yes (or y) if you wish the command to proceed. Otherwise type no (or n). (If editor messages are currently suppressed by the MSGS OFF command, this verification is not done.)

`*EXCESSIVE OUTPUT, JOB TERMINATED`
The editor has attempted to write too much data to the holding file (unit 10). This is usually caused by a command such as FILE /INPUT, RUN or SAVE (10) when the file being edited is too big for unit 10 or the /INPUT file. Specify a file name instead.

`NOT YOUR LIBRARY ...OPERATION NOT DONE`
The file name you have used is the name of a file belonging to another user (a userid different from your userid). You cannot replace that file. Choose a different name.

`NAME ALREADY USED BY SOMEONE ELSE`
The (public) file name you have used is already used by someone else, or is a name which the system will not let you use. Choose some other name.

`UNABLE TO OPEN TEMPORARY FILE`
The most likely cause of this error is that the MUSIC Save Library is full, and there is no room to save your file. Wait a few moments, then retry the save. While still in the editor, you can use the PURGE command to delete any of your files which you no longer need. If the problem persists, contact the User Consultant or the MUSIC Systems Administrator, who will try to make space available. Keep trying the save.

`ERROR WRITING TO TEMPORARY FILE`
An error occurred while writing the file to disk. Try the save again. If the error persists, notify the User Consultant.

`UNABLE TO CLOSE -- FILE NOT SAVED`
The editor was unable to complete the save operation to the specified file. Try the save again. If the

error persists, notify the User Consultant.

`FILE IS TOO BIG - SIZE LIMIT EXCEEDED, or:`
`FILE WOULD CAUSE LIBRARY SPACE LIMIT TO BE EXCEEDED`
The file you are trying to save exceeds one of the space limits associated with your userid, or would cause your total space limit to be exceeded.

`FILE IS TOO BIG - CANNOT GET MORE SPACE`
The file could not be saved because too many extents were required, or there was not enough free space in the Save Library. Try the save again. If the error persists, notify the User Consultant.

`YOUR USERID CANNOT CREATE PUBLIC FILES - FILE NOT SAVED`
Your userid has the "private-only" restriction, meaning that you cannot save a public file (i.e. one that other users can access). Redo the save, without specifying the PUBL or COM option.

`FILE EXCEEDS SIZE OF DATA SET BY n RECORDS`
The user data set is too small to hold all the records you have attempted to SAVE or FILE. Either delete some lines from the file being edited, or increase the size of the data set and redo the edit.

`DATA SET IS READ-ONLY`
You attempted a SAVE or FILE operation to a user data set which was defined as read-only (no writes are allowed to the data set). Redo the edit, specifying OLD on the /FILE statement for the data set.

`I/O ERROR WHILE WRITING TO THE DATA SET`
A disk input/output error occurred during a SAVE or FILE operation to a user data set. Try the command again. If the problem persists, notify the User Consultant or the MUSIC Systems Administrator.

`NO BUFFERS AVAILABLE`
This message indicates an internal problem with the editor. If the problem persists, notify the User Consultant or the MUSIC Systems Administrator.

`LINE WITH DELETE CHARACTER FOUND. CONTINUE SAVE?`
This message indicates that there are lines which start with the delete character followed by blanks, or start with 2 delete characters. It gives you a chance to delete those lines, if you want, before the file is saved. The delete character (normally ":") is defined by the DELCHAR command. Type yes (or y) if you want the file saved as is. Type no (or n) if you want to stop the save operation (this gives you a chance to fix the file and then redo the save). If editor messages are currently suppressed by the MSGS OFF command, this verification prompt is not done, and the file is saved as is.

See also QUIT, SAVE, EXEC, RUN, NAME, STORE, OKREPL.

FILL [FLIP]
[F]

The FILL command applies only to 3270 full-screen mode. It causes blanks to be displayed at the end of each field of the screen, rather than null characters. An alternate name for the FILL command is NONULLS. The opposite of this command is NOFILL, which is the default method of display. Refer to the section on full-screen mode for more information.

The FLIP parameter reverses the current setting for the command; i.e., FILL FLIP causes blanks to be displayed at the end of each field if null characters were previously displayed, or causes null characters to be displayed if blanks were previously displayed.

```
FIND  [string]
F
```

The FIND command searches the file, starting with the line after the current line, for a line beginning with *string.* The search continues down the file until the first match, or until the pointer has been moved past the last line of the file (the message *EOF is displayed). If issued after the pointer is past the end of the file, an automatic TOP is performed before the search begins. If VERIFY status is in effect, the line found is displayed at the workstation.

If *string* is not specified, the same string as specified in the last used FIND, LOCATE, UFIND, ULOCATE, HUNT, or SEARCH command is used. (The UFIND command is similar to the FIND command but searches upwards through the file.)

Spacing: One blank should be present between FIND and *string.* All other blanks are considered to be part of *string*, which extends to and includes the rightmost non-blank in the command line.

Pointer: Set to the found line, or beyond the end of the file if no line is found which begins with *string.* (Message *EOF is displayed in this case).

Example:

Before:
```
SAMPLE LINE          <--Pointer
ANOTHER LINE
LINE THREE
```
Command:
```
F LINE
```
After:
```
SAMPLE LINE
ANOTHER LINE
LINE THREE           <--Pointer
```

```
MUSIC/SCRIPT Form:

FLAG  SCRIPT [COL=n]
FLA
```

This form of the FLAG command is particularly useful when using the Editor to make modifications to a file that is to be used by the MUSIC/SCRIPT program. (The MUSIC/SCRIPT program is described briefly in *Chapter 1. Introduction* of this manual and in more detail in a separate *MUSIC/SP Mail and Office Applications Guide*.) This command will cause all modified lines to be date stamped in columns 73 through 77, with the current date in the form YYDDD. Deletion text to be added to the front of all deleted lines when flag mode is in effect (see the general form of the FLAG command).

This command automatically issues the following Editor commands:

```
TOP
TEXT SCRIPT
ZONE 72
WINDOW 1,72
```

The specification of the COL=n parameter can be used to put the date stamp starting in column number *n*

rather than 73. The generated ZONE and VERIFY commands would then be changed accordingly.

Pointer: The pointer is moved to the top of the file.

General Form:

```
FLAG  [flgtxt] [DEL=deltxt] [COL=n]
FLA
```

This command is the general form of the FLAG command. It puts the Editor into *flag mode*, which causes lines which are subsequently changed, added or deleted, to be identified with a special character string called a *flag*. The flag is normally placed towards the end of the line. This makes it possible to maintain a record of all modifications to a file.

The parameter *flgtxt* is a 1 to 10-character string, optionally enclosed in single quotes, to be used to identify all new, changed or *logically* deleted lines when flag mode is in effect. The default for this parameter is the current date in the format YYDDD (5 characters long) if no previous FLAG command was used, or else whatever was specified on the previous FLAG command.

The parameter *deltxt* is a 1 to 8-character string, optionally enclosed in single quotes, to be added to the front of all "logically" deleted lines when flag mode is in effect.

The parameter *n* is the starting column number where *flgtxt* is to be placed in each line. The default for the first FLAG command is COL=73.

When the flag option is turned on, the Editor automatically issues the following commands:

```
TOP
ZONE n-1
WINDOW 1,n-1
```

The *n* in the above commands is the number given in the COL=n parameter, or 73 if not previously defined.

When flag mode is in effect, any new lines added by INPUT, INSERT, etc., will be flagged. The flagging operation is performed by placing *flgtxt* at the specified column. The CHANGE command only flags a line if a change was actually made.

The DELETE command only *logically* deletes the line, by inserting *deltxt* at the beginning of the line and *flgtxt* at the appropriate column. A line will not be logically deleted if (1) it already begins with the deletion text, in which case it is left as is, or (2) the line has the current flag, in which case the delete is really done. Case (1) takes precedence over (2).

A line is considered to be logically deleted if it has *deltxt* starting in column 1. In flag mode, the logically deleted lines are generally transparent to the Editor. They are not displayed by PRINT n, are skipped by UP and NEXT, and are not seen by commands which search for character strings. However, the commands TOP, LAST, = and SIZE have the same effect with or without flag mode.

Pointer: The pointer is moved to the top of the file.

FLIP [x]
FL

This command defines the first non-blank character following the command as the *flip* character. It must not be a letter or digit. This flip character may then be entered after a command name or abbreviation, and has the effect of reversing the last preceding BRIEF or VERIFY command, for the current command only.

A flip character may be used with the following commands: ADD, BLANK, CHANGE, CHANGEL, DELETE, DELETEL, FIND, HUNT, LAST, LOCATE, UFIND, ULOCATE, LOCATEL, NEXT, OVERLAY, SEARCH, SEARCHL, SPLIT, UP. If a flip character is used in combination with a column number specification Cn, the flip character must follow the last digit of the column number.

If a flip character is used while the Editor is in BRIEF status, the number of characters displayed is as defined by the last WINDOW command issued.

If no non-blank character is specified, the FLIP option is turned off. At the start of the Editor session, the option is off (i.e. there is no flip character defined).

Spacing: Blanks between the command and the first non-blank character are ignored.

Example:

In this example, the pointer would be moved three lines down, as specified by the NEXT command, but typing of the line reached is suppressed.

```
VERIFY
FLIP *
N* 3
```

FORMAT

This macro is used to format all the text in a file while remaining in the edit session. The SCRIPT program is used to perform the formatting with default SCRIPT control words incorporated to fit the text in 72 columns. If you wish, you can include your own control words to override the default settings.

The original text (unformatted) is stored temporarily in a holding file at the time you issue the FORMAT macro. You can restore the file to its original form by using the UNFORMAT macro.

This macro is not recommended for large files as the process is time consuming. It is ideal for formatting mail text when you are using the editor in the MAIL program.

See also FF, MARGINS, and PREFIX.

```
FS  [NOPFK]
    [NOPF ]
    [PF12 ]
    [PF24 ]
    [n    ]
```

The FS command begins full-screen mode, which applies only to 3270-type workstations or ASCII terminals that accept 3270 data streams. Refer to the section on full-screen mode for more details. The opposite of this command is NOFS.

Full-screen mode is automatically assumed at the beginning of the edit, whenever the workstation can support it.

The optional parameter *NOPFK* should be used only if your 3270-type workstation does not have any program function (PF) keys. It tells the Editor to put the screen cursor into the command area whenever the screen is displayed, thus facilitating the entry of commands.

The optional parameter PF12 or PF24 is used to inform the Editor about the actual number of function keys on the workstation being used if the number was not correctly set at sign-on time. This is important for the Editor in order to set the default function key definitions correctly on the workstation being used. PF12 means that the workstation being used has 12 function keys, while PF24 means that there are 24 function keys.

The command FSn begins full-screen mode, and defines the total number of lines on the screen. n can be a number from 7 to the actual screen size (usually 24, 32, or 43). When 3270 full-screen mode is being used from a remote 3270, or from an ASCII terminal simulating a 3270, it can take an appreciable amount of time for the screen to be displayed, especially when connected at 300 or 1200 baud. In these cases it is sometimes useful to define the screen as containing fewer lines than it actually does.

```
GETV name
```

The macro GETV displays the value of a SETV name. Refer to the SETV command.

```
HELP [command-name] [list-of-topics]
HE
```

This command is used to obtain information about a particular Editor command, or about the Editor in general.

If no parameter is specified, the command gives general information about the Editor, such as a one-line description of common commands, Editor concepts, etc.

If *name* is specified, where *name* is the name or abbreviation of an Editor command, information about the command is displayed.

To access MUSIC's general help facility from the Editor, use a slash (/) preceding the HELP command. Enter "/HELP" or "/HELP topicname" from the command area of the Editor. (Without the "/" you will receive help on the Editor and not MUSIC's general help facility.)

Examples:

```
HELP          obtains general information about the editor.
HELP MOVE     explains the use of the MOVE command.
HELP TOPICS   gives a list of all available topic names
/HELP COPY    places you in MUSIC's general help facility and gives information about
              MUSIC's COPY command (not the Editor's COPY command).
```

HEX

This command causes the lines to be displayed in *hexadecimal* format. Hexadecimal format causes each character to be displayed as two hex *digits* 0 to F. For example, a blank is hexadecimal 40 and the number 9 is F9.

The Editor will display 32 characters per line in this mode. This command also temporarily turns off the screen mode if it is on at the time the command is issued; screen mode is resumed when the ALPHA command is used.

The commands BOTH or ALPHA can be used to tell the Editor to use other output representations.

HUNT [string]
H

The HUNT command combines the effect of a TOP command followed by a FIND. It searches the entire file for the first line beginning with *string.* If in VERIFY status, the found line is displayed at the work-station. If there is no line beginning with *string*, the message *EOF is displayed.

If *string* is not specified, the same string as specified in the last used FIND, LOCATE, UFIND, ULOCATE, SEARCH, or HUNT command is used.

Pointer: Set to the found line, or beyond the end of the file if no line is found which begins with *string.* (Message *EOF is displayed in this case.)

Spacing: One blank should be present between the keyword or abbreviation and the first character of *string.* All other blanks are part of *string*, which extends to the last non-blank in the command line.

```
INPUT [END=xx]
INP
```

The INPUT command changes the mode mode of operation from edit to input. All lines typed after the INPUT command are placed, sequentially, after the current line. Input mode can also be started by the INP prefix command.

In 3270 Full-Screen mode, input mode provides a large area of the screen for you to type lines. If you fill up the area and wish to continue typing more lines, press the Enter key. When you are finished typing lines, press the Enter key twice, or press a program function key such as F12, to end input mode.

Outside of 3270 Full-Screen mode, you enter a blank or null line during input mode, to end input mode and return to edit (command) mode. The Editor switches to edit mode with the pointer at the last line entered (not counting the blank line which is not saved).

If the command is given at the very beginning of the Editor session, or immediately after a TOP command, the new lines are placed before the first line of the file.

The parameter END=xx, if used, causes the Editor to recognize *xx* as the input mode terminator, rather than a blank line. This option cannot be used in 3270 full-screen mode. *xx* may be one or two characters long, and any nonblank characters may be used. Input mode is terminated by entering a line with *xx* starting in column 1, followed by blanks. This *xx* line is not saved. The *END=* parameter is useful when it is required to enter blank lines while in input mode.

See also MINSERT and POWERINP.

```
INSERT  [string]
I
```

INSERT places a new line, containing *string*, after the current line. Note: If the command is given immediately after TOP, the new line is inserted before the first line of the file. A blank line can be inserted in the file just by typing INSERT.

CAUTION: The use of tab characters with this command may not have the desired effect. For example, the sequence "I tx", where "t" is the tab character, will put "x" in column 8 if the first tab location was defined to be column 10.

Spacing: One blank must be present between the command name or abbreviation and *string.* Any other blanks are part of *string.*

Pointer: Set to the inserted line.

Example:

```
Before:   SAMPLE LINE          <--Pointer
          ANOTHER LINE
Command:  I YET ANOTHER
After:    SAMPLE LINE
          YET ANOTHER          <--Pointer
```

ANOTHER LINE

JOIN [/string/]
JO

The JOIN command may be thought of as the opposite of the SPLIT command. It joins the next line to the end of the current line to form one line.

The second line is added after the last nonblank character in the "ZONE" part of the first line. A specified character string (or a single blank if a string is not specified) is placed between them. Only the "ZONE" parts of the lines participate (refer to the ZONE command). If the joined text is too long for one line, a warning message is displayed and the excess is left on the second line.

A single blank is used if */string/* is not specified.

Example:

```
Before:   line1
          line2          <--- current line
          line3
Command:  join /+++/
After:    line1
          line2+++line3  <--- new current line
```

KEYS

The KEYS Editor macro allows you to change your function key definitions.

LANGUAGE [language]
LANG

The LANGUAGE command specifies the national language to be used for the remainder of this workstation session. It is similar to the MUSIC LANGUAGE command. The national language setting affects the language used in messages from some applications. It also turns Double-Byte Character Set (DBCS) mode on for the current Editor session, if the requested language is a DBCS language, such as KANJI.

language is the language name, or is DEFAULT to request the system default language. In most cases, the first 3 characters of the name can be used as an abbreviation. Not all languages are supported or installed at all sites. Some language names are:

DEFAULT	(the system default language)
ENGLISH	
FRENCH	

KANJI (JAPANESE)
PORTUGUESE
SPANISH

See also: DBCS.

LAST
LA

This command positions the line pointer to the last line of the file.

LEFT [n]
LE

The command LEFT shift the window display left a specified number of columns. This is equivalent to the command WINDOW LEFT n. If *n* is omitted, the window is shifted the maximum number of columns possible.

See also RIGHT, WINDOW and ZONE.

LIST [filename[,m][,n]]
LI

This command is used to display the contents of a file, or part of a file, or the entire file being edited. LIST is often used in conjunction with the MERGE command.

Line numbers *m* through *n* of the file *filename* are listed. If the *m* or *n* parameter are omitted the entire file is listed. If the *n* parameter is omitted or is too big, then the listing stops at the end of the file. The LIST command without any parameters displays the entire file being edited. Line numbers will also be displayed if line numbering is in effect. Line numbering is requested by using the NUMBER command.

The file name may be /INPUT (the Input File) or /HOLD (the Holding File).

"(3)" may be specified instead of *filename.* This causes the UDS file on MUSIC unit number 3 to be listed. The file is rewound both before and after the listing operation.

```
LOCATE  [string]
L
```

LOCATE is used to find the first line after the current line which contains *string* anywhere in the line and, if in VERIFY mode, to display the line on the workstation. The search begins with the line following the current line and continues down the file. If *string* does not exist in the file between the line following the current line and the end of the file, the message *EOF is displayed. If the command is given after an end of file an automatic TOP is performed before the search begins. If the command is given immediately after *EOF (bottom of file) or a TOP command, the first line of the file will also be searched for *string*. The operation of LOCATE may be affected by the ZONE setting (refer to the description of the ZONE command).

If *string* is not specified, the same string as specified in the last used FIND, LOCATE, UFIND, ULOCATE, HUNT, or SEARCH command is used.

(The ULOCATE command is similar to the LOCATE command but searches upwards through the file.)

Pointer: Set to the line in which the *string* is found, or beyond the end of the file, if it is not found.

Spacing: One blank should be present between the keyword or abbreviation and *string*. All other blanks are part of *string*, which extends as far as the rightmost non-blank in the command line.

Example:

```
Before:  SAMPLE LINE          <--Pointer
         ANOTHER LINE
         YET MORE
Command: L ER
After:   SAMPLE LINE
         ANOTHER LINE         <--Pointer
         YET MORE
```

```
LOCATEL  [n,](logical expression)
LL
```

The file is searched for the next line which tests true against the *logical expression*. The first parameter is optional and specifies the number of lines to be searched. If it is omitted, the search terminates at the first line which tests true.

If *n* is specified, *n* lines are searched (starting with the one following the current line), each line satisfying the logical expression is displayed, and the pointer is set to the line following the last one searched. If * is used instead of *n*, the search continues to the end of the file and each line satisfying the logical expression is displayed. For example, TOP followed by LL*,(/ABC/) displays all lines containing ABC. In this way the LOCATEL command may be used as a generalized form of the SCAN command.

If the command immediately follows *EOF or a TOP command, the first line of the file is included in the search.

Example:

If it desired to search for the next line which contains the strings HOW and DO, but does not at the same time contain both THUD and THUNDER.

```
LL (/HOW/AND/DO/AND.NOT(/THUD/AND/THUNDER/))
```

```
LOG  [n    ]
     [END  ]
     [PURGE]
```

This command is used in conjunction with the Editor restart facility. When used without an parameter, the LOG command forces any log lines being held in main storage buffers to be written to the log file. This ensures that all edit activity prior to the LOG command will be available for restart if necessary.

When used with a number *n* as an parameter, the command defines the maximum number of log lines which will be held in main storage. Initially, the Editor assumes a value of 25 lines, i.e. LOG 25, for 3270-type workstations and 20 lines for other workstations.

The number *n* specified on the LOG command is also the maximum number of lines which may be entered in input mode (in non-full-screen mode) before the Editor gets control. This may be noticeable as a slight pause after each group of n lines entered in input mode.

Using LOG without an parameter does not change the value n.

As explained in the section on the restart facility, additional lines may be lost if the Editor is in input mode or full-screen mode.

If the parameter END is specified, the Editor will clear and close the log file. The log file can then be purged by issuing the command "PURGE @ELOG" to save file space. Use "PURGE @ELOG.x" if your userid has a subcode *(x* being your 1-8 character subcode).

If the parameter PURGE is specified, it is equivalent to specifying the parameter END and the PURGE command as mentioned above. That is, the log file will be cleared, closed, and purged all at the same time.

```
MARGINS n1 n2 [n3]
```

Before using FF or WW, you can use the MARGINS macro to redefine the output margins. For example "margins 5 60" will make an indented paragraph.

The first number *n1* is the left margin column. Formatted text will start in this column.

The second number *n2* is the right margin column. Formatted text will not extend beyond this column.

The optional third number *n3* is the paragraph indent, which can be a positive or negative number of columns. This option applies to the WW prefix command, but is ignored by the FF macro. FF always uses

n3=0. It gives the number of characters to indent the start of each paragraph, relative to the left margin of the rest of the paragraph. A negative number gives "hanging" paragraphs. The default for n3 is 0 or the previous value specified.

See also FF, WW, FORMAT, PREFIX, and MARK.

```
MARK  [?]
MA
```

The MARK command is used to identify the first and last lines of a group of lines. The group may then be operated on by the *dot* forms of the command =, MOVE, COPY, DELETE, CHANGE, CHANGEL, FF, REPEAT, TOUC, TOLC, COPYCOL, SEQ, SORT, and STORE.

Refer to the topic "Marking a Group of Lines" earlier in this chapter.

```
MD
```

This command makes a new directory. Same as the MUSIC command MD.

```
MDELETE  [n]
MDEL
```

This multiple delete command has a dual function. It deletes unused lines left over from the MINSERT command (without the *B* parameter), and can also delete a group of lines.

First, lines in the neighbourhood of the current line which have the delete character in column 1, followed by blanks, are deleted. The delete character (":" by default) is as defined by the DELCHAR command described above. Then, if lines were found with the delete character in columns 1 and 2 (i.e., the line begins with "::"), all lines between the first and second such lines, inclusive, are deleted.

Thus, to delete a group of lines, put delete characters into columns 1 and 2 of the first and last lines of the group, then use the MDELETE function key.

The *n* parameter of the command defines the number of lines above and below the current line to be searched for delete characters. A command such as MDEL 99999 will apply the delete to the entire file (the default value for *n* is 40).

The line after the last line deleted will become the current line.

```
MERGE  filename[,n][,m]
ME
```

This command brings a copy of line numbers *n* through *m* of the file *filename* into the file being edited. If the *n* and *m* parameters are omitted the entire file is copied. If the *m* parameter is omitted or is too big, the file is copied until the end of the file.

The new lines are inserted after the current line, and the last line inserted becomes the new current line. If the preceding command was a TOP, or the MERGE command is given at the very beginning of the Editor session, the lines are placed at the beginning of the file.

The records merged from the file are truncated or filled out with trailing blanks, if necessary, to match the record length of the file being edited.

The file name may be /INPUT (the Input File) or /HOLD (the Holding File).

"(3)" may be specified instead of *filnam.* This causes the UDS file on MUSIC unit number 3 to be used as input for the merge. The file is rewound both before and after the merge. Examples: MERGE (3), MERGE (3),15,50.

```
MINSERT  [n] [B]
MI
```

This multiple insert command causes the insertion of a number of lines following the current line. The first line inserted becomes the new current line.

n is the number of lines to be inserted. The default is 10 lines.

If the *B* parameter is specified after the number of lines, then blank lines are inserted. Otherwise, lines with the delete character (as specified by the DELCHAR) in column 1 are inserted.

This command can be used as an alternative to the INPUT mode for entering lines into a file.

```
MOVE i,j,k
MO
MOVE.
MO.
```

This form of the MOVE command allows moving lines by reference to the line numbers of the lines. The block of lines from line *j* to line *k* inclusive is moved to after line number *i.* (Line numbers may be determined by using the "=" command.) The original lines *j* to *k* are deleted. If *i* is specified as 0, the lines are moved to the beginning of the file. If *j=k,* only one line is moved.

Line numbers may also be specified in any of the following forms: *n, *+n, *-n, LAST, LAST-n*, where *n* is a number, * means the current line, and LAST means the last line of the file. Example: MOVE LAST,*,*+3

For the command "MOVE.", used to move a marked group of lines, refer to the topic "Marking a Group of Lines" earlier in this chapter.

Pointer: Set to the last line of the inserted section.

Example:

Before:
```
A1
A2
A3
A4
A5
```
Command:
```
MOVE 4,1,2
```
After:
```
A3
A4
A1
A2
A5
```

```
MSG xmessage
```

The MSG command causes a message line to be displayed on the workstation. The first character of the message is assumed to be a print control character, i.e. blank for single spacing, zero for double spacing, etc.

x is the print control character (usually a blank). Exactly one blank must appear between the command name and the control character. In 3270 full-screen mode, the message is displayed in the message area at the bottom of the screen, if there is room.

```
MSGS ON
     OFF
     NOFILMSG
```

The MSGS command is used to suppress or enable all messages from the Editor. When MSGS OFF is in effect, no messages are sent to the workstation. Also, the output of such commands as LIST and SCAN is suppressed.

MSGS OFF is intended mainly for use in function key definitions, and in Rexx procedures invoked from the Editor (Editor macros). The function key operation or Rexx macro would set MSGS OFF to suppress undesired messages, and set MSGS ON before returning.

The command MSGS NOFILMSG suppresses the file name message and the confirmation message issued by a successful FILE command. This may be desirable when the editor is invoked from a full-screen application, to avoid leaving full-screen mode at the end of the edit.

See also: MSG, REXX

Example:

The following defines F1 as a store operation to file ABC. If the file already exists, it is deleted before the store. MSGS OFF ensures that the PURGE command does not produce any messages.

```
DEFINE F1 MSGS OFF;PURGE ABC;MSGS ON;STORE ABC
```

NAME [name]
NA [0]

This command is used to display or change the file name associated with the edit session. *name* may be a file name or the name /INPUT for the Input file.

If no parameter is used, the name of the file being edited is displayed. For a UDS file edit, the volume name is also given.

If 0 (zero) is used as the parameter, the file name is made undefined. Then if a SAVE or FILE command is done, the user will be prompted to enter the file name to be used.

For the normal files, the file name associated with the edit session may be defined or changed by using the new name as an parameter on the NAME command. A subsequent SAVE command, for example, will use the new name.

MUSIC also accepts FNAME, with abbreviation FN, to mean the same as the NAME command.

Spacing: The file name, if used, must be separated from the command by at least one blank.

NEXT [n]
N

This command moves the pointer *n* lines down the file from the current line. If *n* is omitted, 1 is assumed. The new current line is displayed, unless a BRIEF command has been issued previously.

For compatibility with other systems, MUSIC also accepts DOWN (or DN) to mean the same as the NEXT command.

NOARROW
NOAR

This command removes the effect of the ARROW command.

```
NOCHANGE
```

This command informs the Editor that there are no unsaved changes even though there may have been. For example, if a MERGE has been used within a macro the Editor will allow the QUIT command to be used instead of QQUIT.

```
NOFILL  [FLIP]
        [F   ]
```

This command applies only to 3270 full-screen mode. It is the default setting and is the opposite of the FILL command. NOFILL causes null characters (rather than blanks) to be displayed at the end of each screen field. An alternate name for NOFILL is NULLS.

The FLIP parameter reverses the current setting for the command; i.e., NOFILL FLIP causes blanks to be displayed at the end of each field if null characters were previously displayed, or causes null characters to be displayed if blanks were previously displayed.

NOFILL is the default method of display at the start of an Editor session. The opposite of this command is FILL.

```
NOFLAG
NOFL
```

This command terminates the effect of a previous FLAG command, except that the ZONE and WINDOW settings still apply. However, the flag options are remembered by the Editor, and the original flagging can later be continued by issuing a FLAG command without any parameters.

```
NOFS
```

The NOFS command terminates 3270 full-screen mode. It is the opposite of the FS command. NOFS implies the commands WINDOW* and ZONE*. Refer to the section on full-screen mode for more details.

```
NONULLS  [FLIP]
         [F   ]
```

This command applies only to the 3270 full screen mode. It causes blanks to be displayed at the end of each field on the screen instead of null characters. An alternate name for this command is FILL.

The FLIP parameter reverses the current setting for the command; i.e., NONULLS FLIP causes blanks to be displayed at the end of the field if null characters were previously displayed, or causes null characters to be displayed if blanks were previously displayed.

Null characters are displayed by default at the start of an Editor session. The opposite of this command is NULLS.

```
NONUMBER
NONUM
```

This command is used to suppress the line numbering caused by the NUMBER command. At the start of the Editor session, the line numbering is off.

```
NOSCREEN
NOSCR
```

This command is used to turn off *screen* mode.

```
NOSHOW
NOSH
```

The NOSHOW command removes the text displayed at the bottom of the screen by a previous "SHOW file-name" command, and enlarges the FS screen back to its original size.

```
NOTRAN
```

Normally all output from the Editor is translated to ensure that only printable characters are displayed. The NOTRAN command suppresses this translation.

```
NULLS  [FLIP]
       [F   ]
```

This command applies only to the 3270 full screen mode. It causes null characters to be displayed at the end of each field on the screen. An alternate name for NULLS is NOFILL.

The FLIP parameter reverses the current command; i.e., NULLS FLIP causes blanks to be displayed at the end of each field if null characters were previously displayed, or causes null characters to be displayed if blanks were previously displayed.

At the start of the Editor session, null characters are displayed. The opposite of this command is NONULLS.

```
NUMBER [FLIP]
NUM    [F   ]
       [ON|OFF]
```

The NUMBER command causes line numbers to appear whenever lines of the file are displayed. At the start of the editor session, the line numbering is off. The line pointer is not affected. Either NUM FLIP, NUM OFF or NONUM can be used to turn numbering off.

The records of the file are numbered sequentially, starting at 1.

```
OFF
```

The OFF command is similar to the END and QUIT commands, but also terminates the MUSIC session and disconnects the workstation from MUSIC. It is equivalent to an END or QUIT command, followed by the MUSIC command OFF.

OKREPL [filename]

The OKREPL command indicates that the specified file can be replaced without the editor prompting for permission. This affects the FILE, SAVE, EXEC, RUN, and STORE commands. The file name must be specified exactly (except for case), as it will be on the save command, or as in the current file name if the save command does not specify a file name.

The editor remembers only one such file name at a time. A successful save operation has the same effect as the command "OKREPL xxx", where xxx is the name of the file saved. This may nullify a previous OKREPL command.

If OKREPL is used without a parameter, any previous OKREPL is nullified, and subsequent save operations will do normal prompting.

Example:

```
okrepl myfile
file myfile
```

OVERLAY string
O

This command replaces specified characters in the current line with the corresponding characters in *string.* The replacement is position-dependent. If the nth character of *string* in nonblank, then the nth character of the current line is replaced with the nth character of *string. String* may contain up to 79 characters, and for each nonblank character, one character of the current line will be replaced.

This command is applied to the current line. If preceded by a REPEAT command, it is applied to as many lines, beginning with the current line, as were specified in the REPEAT (see description of REPEAT). This command is the complement of the BLANK command. If in VERIFY status, the last line changed by OVERLAY is displayed on the workstation.

Spacing: One blank should be present between the keyword or abbreviation and *string.* All other blanks are part of *string.*

Example:
Before: **SAMPLE LINE**
Command: **O X**
After: **SAXPLE LINE**

```
POINT  xxxxxxxx [n]
POI             [OFF]
```

This command assigns a 1 to 8 character label (xxxxxxxx) to a line of the file. The label must start with a letter or a digit. Case (upper/lower) is not significant in the label. Periods (.) preceding a label are ignored (e.g. POINT .ABCD 10 is the same as POINT ABCD 10). Later during the edit session, you can go to that line by the command =xxxxxxxx. A label stays with the line, even if the line number changes as a result of inserts, deletes, moves, and copies. A label can start with a digit, but in that case the command to go to it must be =.xxxxxxxx.

Add the parameter *n* to specify a line number, otherwise the current line is labeled. *n* can be 0 to the number of lines in the file +1. OFF undefines the label.

The prefix area command .xxx assigns the label xxx (1 to 3 characters) to the line on which it is entered. See PREFIX for more information.

```
POWERINP
POW
```

The POWERINP command (actually a macro) is similar to the INPUT command, except that you can type continuously without having to press the NEWLINE key or TAB key at the end of each line of input. Power Input Mode is similar to Input Mode, except for these differences:

1. The cursor skips automatically to the beginning of the next line when the cursor reaches the end of a line. This allows you to type continuously without having to watch where the cursor is on the screen.

2. When you press ENTER or an F key while in Power Input Mode, the lines you have just typed are reformatted by the editor. This is called "word wrap". Words that were split at the end of a line are put back together, and text is formatted within the current WINDOW columns. A blank line causes a formatting break (paragraph separator). The formatting is similar to what the WW prefix command would do. See the WW command in the topic "Prefix Area" earlier in this chapter.

3. Pressing ENTER after typing one or more lines of input does not end Power Input. The Enter key causes a formatting break, and Power Input continues. This is useful for entering single lines (like Script control words) that should not be formatted with the rest of the text. To end Power Input, press ENTER twice, or F12 once.

Power Input can also be started by the POW prefix area command. Input starts after the line identified by POW.

If POWERINP is used while not in 3270 full-screen (FS) mode, normal Input Mode is used.

Note: To use Power Input with NET3270, document mode (the Alt-E toggle) must be OFF. NET3270 has its own word-wrap feature (Alt-W while in document mode), which makes the editor's Power Input unnecessary.

Internals:

Power Input is started by the macro POWERINP, which, among other things, issues the commands DEFINE INPUT $FINP and INPUT. The text typed during Power Input is processed by the macro $FINP. Automatic cursor skip is obtained by the command AUTOSKIP INPUT. For more information, see the DEFINE command and the comments in file $FINP.MAC.

```
PREFIX ON
PRE    OFF
       FLip
       CLear
```

The prefix area, turned on by the command PREFIX ON, is a 4-character modifiable field at the left of each displayed line of the file. It contains ====, or the last 4 digits of the line number if NUM ON is in effect. You can enter various special commands in the prefix area, to do editing operations such as inserting or deleting lines, moving and copying lines, etc.

Parameters:

ON is assumed if there is no parameter

OFF reverses the previous setting

CLEAR clears any pending prefix operations

See the topic "Prefix Area" earlier in this chapter for more information.

```
Form 1:  PRINT filename [ROUTE(loc)] [FORMS(x)] [COPIES(n)] [CC  ]
         PRI   *CUR      R            F          C          [NOCC]
               .

                   [PAGELEN(m)]
                    P

Form 2:  PRINT [m] [n]
         P
```

Form 1 of the Editor PRINT command is similar to the MUSIC PRINT command. It schedules the printing of a file to a specified printer. The file name must be the first parameter. Special names *CUR (the data being edited) and . (marked lines) may be used in place of a file name. The other parameters are optional and may appear in any order. Carriage control is added to skip to a new page every sixty lines unless CC is specified or the file's record length is 121 or 133.

Form 2 of the Editor PRINT command displays on the workstation the first *m* characters of the *n* consecutive lines beginning with the current line. Defaults are n=1 and m=record length.

Parameters (Form 1):

filename — The name of the file to be printed. Under the Editor, the special name *CUR indicates the current contents of the file being edited, and the special name . indicates marked lines.

loc — The name of the printer where the file is to be printed. This may be an actual printer name or a printer location. It is 1 to 8 characters long. Some documentation refers to this as a "route name" or "routing name". The names are assigned by your system administrator.

If you do not specify a printer name, a default name is used. If you have used the command "ROUTE printername" previously in this MUSIC session, that name is the default. Otherwise, the name defined by ROUTE(name) in your User Profile (the PROFILE command) is used, if any. Otherwise, a default name based on your workstation location may be used. If none of the above cases apply, the name SYSTEM is used.

The following names are valid:

SYSTEM — Sends the output to the standard system printer.

MUSIC — Sends the output to the MUSIC Output Queue (the OUTPUT Facility).

DUMMY — Discards the output. Nothing is printed.

rscsname — The name of an RSCS printer. For example: R(PRINTER3) means the output is sent to RSCS and queued for printing on linkid PRINTER3.

prtname — The name of a MUSIC-controlled ASCII or 3270 printer as defined by your installation. Consult your installation for a list of valid names.

PC1 — A printer name such as PC1 may be defined by your installation. It prints the file on your PC printer (using DOS device LPT1), provided your PC is connected to MUSIC via NET3270 or PCWS. If the connection does not support PC printing, the data is sent to the MUSIC Output Queue (the OUTPUT Facility). Similarly, PC2 uses device LPT2 and PC3 uses device LPT3. When printing to a PC printer, some PRINT parameters such as COPIES, FORMS and PAGELEN may be ignored.

n — The number of copies that should be printed. The default is 1 copy.

x — A one to 8 character string indicating which forms are to be used when printing the file.

m — The number of lines per page. The default is 60 lines. This parameter is used only when the NOCC parameter is in effect.

CC — If specified, it indicates that the file to be printed already contains a carriage control character on the first character of each line in the file. The PRINT command attempts to honour these control characters.

NOCC — Indicates that the file does not contain printer control characters. The file is printed single spaced, with a skip to a new page after each m lines of output. NOCC is the default, except when the record length of the file is 121 or 133, in which case CC is assumed.

Parameters (Form 2):

n — This specifies the number of consecutive lines that are to be printed. If * is specified, the rest of the file starting from the current line will be printed. If *n* is not specified, it is assumed to be 1.

m — This specifies the number of characters that are to be printed from each line. If not specified, the whole line will be printed.

Examples:

`P`	Displays the current line.
`P5`	Displays 5 lines.
`PRINT * 72`	This prints the first 72 characters of the rest of the lines in the file, beginning with the current line.
`PRINT PGM R(PRTA)`	This prints the file PGM on the line printer called "PRTA".

See also LIST, NUM, and WINDOW commands.

PROMPT [x]
PR

The character specified is used as a prompt character for workstation dialogue, and is displayed each time the Editor expects the user to type a line. This feature is particularly useful for TTY terminals where the keyboard is always unlocked. If *x* is not specified, Editor prompting is terminated if prompts were being issued, or, if prompts were not being issued, prompting is begun using a question mark character (?). At the start of the Editor session, a question mark (?) is automatically used as the prompt character for teletype (TTY) terminals.

Spacing: Blanks between the command and the character are ignored.

PURGE name
PUR

This command is used to permanently remove (delete) a file from the Save Library, as in the MUSIC command /PURGE. Alias: ERASE.

To delete the Input File, specify file name /INPUT. To delete the Holding File, specify file name /HOLD.

Spacing: The file name must be separated from the command by at least one blank.

```
QQUIT <n>
QQ
```

The QQUIT command immediately terminates the Editor session, without any messages or prompting. The file is not stored.

See QUIT for details and description of *n* parameter.

```
QUIT
Q
```

The QUIT command terminates the Editor session, without performing any save operation.

If you have made changes to the file but have not issued a FILE command to make the changes permanent, you will be prompted for permission to end the edit session. Enter YES or Y to end the session, or NO or N to cancel the QUIT operation and continue editing.

The END, QUIT, and QQUIT commands terminate the edit session without writing the editor's copy of the file to disk. An optional job return code *n* (a number 0 or higher) can be specified on each of these commands. If the return code parameter is omitted, the code specified on the last SETRC command is used, or 0 if no SETRC was done. Please refer to the SETRC command for additional notes.

```
RENAME oldname newname
REN
```

The RENAME command changes the name of an existing file in the Save Library. It performs the same function as the MUSIC command RENAME. The names can include an * (asterisk) to indicate groups of files.

Most attributes of the file, such as public/private, tag, and date last referenced, are unchanged by the command. The only exception is the date last written, which is set to the current date.

```
REPEAT  n
```

This command specifies the number of times the following BLANK or OVERLAY command is to be performed on successive lines.

To cancel the effect of a REPEAT command before BLANK or OVERLAY is used, enter the command REPEAT 1.

Pointer: Set to the last line changed.

Spacing: The number of times to repeat may be anywhere in the command line after the keyword.

Example:
```
Before:   * SAMPLE LINE       <--Pointer
          * ANOTHER LINE
Command:  REPEAT 2
Command:  OVERLAY C
After:    C SAMPLE LINE
          C ANOTHER LINE      <--Pointer
```

```
REPLACE  [string]
R
```

This command replaces the current line with *string*. You can replace the current line with a completely blank one by typing REPLACE without specifying a string.

Spacing: One blank must appear between the keyword and the first character of *string*. All succeeding characters are part of *string*.

Example:
```
Before:   SAMPLE LINE
Command:  R NEW STUFF
After:    NEW STUFF
```

```
REXX ON
REX  OFF
```

The REXX command enables or disables use of Rexx procedures from within the Editor. Rexx procedures are also called Editor macros. Refer to the section "Editor Macro Facility" for more information.

The REXX ON command must be used before Editor Rexx macros can be used. REXX ON could be contained in your private EDITOR file. Also, a large user region must be defined in order to use Rexx procedures: e.g. /SYS REGION=256. If the user region is not large enough to accommodate the Rexx processor and work area, the REXX ON command is ignored.

```
REX ON
```

This command must be done at least once in the edit session, before any Rexx procedures can be executed.

Messages:

`INVALID COMMAND OR REXX PROCEDURE NOT FOUND`
This message, issued when you enter an Editor command, means that Rexx procedures are enabled,

Rexx has been loaded, and the command you entered was not an Editor command, but Rexx could not find any procedure for that command name. (If REXX OFF is in effect, the message `INVALID COMMAND` is issued instead.)

`UNABLE TO LOAD REXX - COMMAND NOT DONE`
This message could be issued when the Editor tries to pass the command you entered to Rexx, since the command is not recognized as an Editor command. However, the Editor could not load the Rexx processor. The Rexx procedure you entered has not been done.

RIGHT [n]
RI

The command RIGHT shift the window display right a specified number of columns. These are equivalent to the command WINDOW RIGHT n. If *n* is omitted, the window is shifted the maximum number of columns possible.

See also LEFT, WINDOW and ZONE.

RUN [name]
RU

Usually the RUN command is used without a *name* parameter. When a name is specified, the RUN command has the same effect as the Editor EXEC command. If desired, column number limits and file attributes (PRIV, PUBL etc.) can be specified before and after *name* respectively (as on the FILE command).

When used without a *name* parameter, RUN saves the changed file in place of the Input file and then automatically requests MUSIC to execute the program from the Input file in a way equivalent to using the MUSIC command /RUN. This command is particularly useful when you want to make temporary changes to a file and do not wish to modify the original.

If this command is used in the User Data Set version of the Editor, either the Input file (if *name* is not specified) or the named file (if *name* is specified) will be replaced by the temporary updated version of the UDS file being edited. If the replacement is successful, a request to execute the program in the Input file or the named file will then be made to MUSIC. The original UDS file is not updated, and the edit is terminated.

SAVE [name]
SA [(u)]
SV

This command causes the current temporary copy of the file to replace the specified file, without terminating the Editor. This is similar to the FILE command except that the edit session will not be terminated. The line

pointer is not changed. Therefore, the user may continue to issue additional edit commands on the same file.

Parameters:

name — If a file name is not specified on the command, the name on the original EDIT command or on the last NAME command will be used, or, in the case of editing a User Data Set (UDS) file, the UDS file being edited is replaced.

u — A unit number in parentheses may be specified in place of a file name, causing output to be done to that unit instead of to a file. For example, SAVE (3). The unit number must be 3, 7 or 10. The unit is not rewound before output is begun.

For a description of the various other parameters which can be used on the SAVE command, refer to the FILE command, which takes the same parameters. These include starting and ending column numbers and file attributes (PRIV, PUBL, etc.)

Examples:

`SAVE NEWFIL`	It writes the updated version of file to a file named NEWFIL. The original file is not updated.
`SAVE 72,(3)`	It writes columns 1 to 72 of each line in the updated version of file to the UDS defined in unit 3. The original file is not updated.

See also FILE and STORE.

SCAN /string/[S]
SC

This command searches the entire file being edited for all occurrences of *string*. Each line containing the string is displayed. The string must not be null, and the final delimiter (/) may be omitted if the S parameter is not given. This command is equivalent to a TOP followed by a "CHANGE /string/string/*,v."

If the parameter S is specified, the Editor will cause the screen to be displayed for each line found. On a 3270-type workstations, the ENTER key must be pressed between screens. The PA1 key can be pressed to stop the scan.

Pointer: Set to the bottom of the file.

SCREEN [n]
SCR

This command is useful for workstations that do **not** have full screen editing capabilities. It is used to turn on *screen* mode. The parameter *n* specifies the approximate number of screen lines to be used for displaying the current line and the lines above and below it. It must have a value from 7 to 17. At the start of the Editor session, *n* has a value of 15. If *n* is omitted, the number of lines is not changed. Refer to the section "Screen

Mode" for more details.

On a 3270-type workstation, full-screen (FS) mode is normally in effect, and overrides screen mode. To get screen mode, enter the commands NOFS and SCREEN, then enter a blank line.

```
SEARCH  [string]
S
```

The SEARCH command combines the effect of a TOP command followed by a LOCATE. It searches the entire file for the first line containing *string.* If in VERIFY status, the found line is displayed at the work-station. If *string* does not exist in the file, the message *EOF is displayed.

If *string* is not specified, the same string as specified in the last used FIND, LOCATE, UFIND, ULOCATE, HUNT, or SEARCH command is used.

Pointer: Set to the line containing *string,* or past the end of the file.

Spacing: One blank should be present between the keyword or abbreviation and the first character of *string.* All other blanks are part of *string,* which extends to the last non-blank in the command line.

```
SEARCHL  [n,](logical expression)
SL
```

The SEARCHL command is exactly equivalent to a TOP command followed by the corresponding LOCATEL command. Refer to the description of the LOCATEL command.

```
SEQ [xxx] [m] [n] [COL=k] [LEN=l] [LINES=s]
```

This command puts sequence numbers and an optional identification field into each line of the file. *xxx* is an optional identification field, up to 8 characters long, not all digits. *m* is the increment value, up to 8 digits. If *m* is omitted, a default value of 10 is used. *n* is the starting value for the sequence numbers, up to 8 digits. If *n* is omitted, the starting value will be the same as the increment.

COL=k and LEN=l may be used to specify the starting column *k* and the length *l* (1 to 8) of the sequence number field. Defaults are COL=73 and LEN=8.

If the parameter LINES=s is used, where *s* is a number, only *s* lines are changed, starting with the current line. The last line sequenced becomes the new current line. Note that if LINES=s is omitted, the entire file is sequenced.

Pointer: Set to the first line of the file (unless LINES=s is used).

Spacing: One blank may optionally appear between the command and the first parameter. The parameters are separated by blanks or commas.

Examples:

Command	Resulting sequence numbers
`SEQ PGMA,1`	`PGMA0001,PGMA0002,PGMA0003 ...`
`SEQ 10,200`	`00000200,00000210,00000220 ...`
`SEQ ASM`	`ASM00010,ASM00020,ASM00030 ...`
`SEQ`	`00000010,00000020,00000030 ...`

The following example shows how a series of 5 similar data lines, differing only by a sequence number field, may be inserted into a file:

```
Before:   xxxxxxxx
          yyyyyyyy       <--- current line
          zzzzzzzz
Command:  i name=abnnn
          dup 4
          up 4
          seq 1 1 col=8 len=3 lines=5
After:    xxxxxxxx
          yyyyyyyy
          name=ab001
          name=ab002
          name=ab003
          name=ab004
          name=ab005     <--- new current line
          zzzzzzzz
```

SET xxxx

The SET macro offers compatibility with the CMS editor: "SET xxxx" usually issues the command xxxx. For example,

```
SET NUM ON
```

As a special case, SET PFn xxx issues DEFINE PFn xxx.

SETRC n

The numeric value *n* (0 or more) specified on the SETRC command will be used as the job return code (exit code) for the edit, when the editor terminates. This includes all normal terminations, such as QUIT, QQUIT, END, OFF, FILE, etc. It does not include system aborts or /CAN used in attention mode.

If several SETRC commands are used, the last one is honoured. If no parameter is used on SETRC, the command is ignored. A return code specified on the QUIT, END or QQUIT command overrides any specified on SETRC.

If no return code is specified by SETRC, QUIT, END, or QQUIT, the edit return code is normally zero. However, a non-zero return code can result from an error during editor start-up (insufficient memory, file to be edited cannot be read, etc.) The return code in these cases is a small positive value. Also, some types of system errors can result in large return code values. Therefore, to avoid conflicts, applications should limit themselves to values for SETRC and QUIT/QQUIT/END of 0 or 200 thru 999.

Type "HELP RETURN" during an edit session to see a list of editor return codes. An editor macro can set a return code by the Rexx "EXIT n" statement. These do not affect the editor job return code.

SETV xxx yyy

The SETV command is mainly used when writing editor macros with REXX. It assigns a character string yyy to a name xxx. xxx is 1 to 8 characters. In a macro, you can retrieve the string yyy by "EXTRACT /$xxx/". This allows macros to remember things between calls. Omitting yyy undefines xxx, and the extract returns a null string. SETV definitions are local to the editor session in which they are made.

```
SETV ABCD This is the value
...
"EXTRACT /$ABCD/"     <-- sets abcd to "This is the value"
```

SHIFT < n
> n
l
r

The SHIFT macro shifts marked lines left (<) or right (>) by a specified number of columns (n). Text which is shifted left beyond column 1, or right beyond column n (where n is defined by the command ZONE n), is lost. Text outside the zone (columns 1 to n) is not affected.

See also MARK, ZONE, and PREFIX.

SHOW PFn
SH PF
X
filename n1 n2

The SHOW command displays the command string currently defined for a specific program function key (F1

to F24), or for the X command. Also, definitions of all 24 function keys may be requested. To get the definition of Xn (where n is 1 to 24), use "SHOW PFn".

The NOSHOW command removes the text displayed at the bottom of the screen by a previous "SHOW filename" command, and enlarges the FS screen back to its original size.

An alternate form of the SHOW command can display up to 12 lines from a specified file at the bottom of the screen in full-screen (FS) mode. The display will remain there until removed by a NOSHOW command or replaced by another SHOW command. The number of screen lines available for full-screen editing (i.e. the "n" in the command "FS n") is reduced by the number of lines displayed by the SHOW command. For example, this form of the SHOW command could be used to permanently display function key definitions or other helpful information.

Text for SHOW filename can contain 3270 Start Field orders (hex character 1D, followed by the field attribute byte). They can be used to highlight parts of the show text, or display it in a different colour. Attribute characters are D (unprotected low - green), H (unprotected high - red), U (protected low - blue, the default), and Y (protected high - white). The 1D character can be entered by using the editor XIN command. Each 2-character Start Field order occupies one blank position on the screen.

Within SHOW text, hex character 01 starts highlighting, and hex character 02 ends highlighting. This is an alternative to using Start Field (hex 1D). 01 and 02 are not followed by a field attribute character.

Parameters:

PFn	(where n is 1 to 24) the Program Function key whose definition is to be displayed.
PF	Requests that the definitions of all function keys be displayed.
X	Requests that the definition of the X command be displayed.
filename	The name of a file whose contents is to be displayed at the bottom of the screen when in FS mode.
n1	Starting line number in the file. The display starts with this line. Default for n1 is 1.
n2	Ending line number in the file. The maximum number of lines that can be displayed is 12. The default for n2 is n1+11.

Examples:

```
SHOW PF8            - Displays the current definition of F8.
SH PF               - Displays the definitions of F1 through F24.
SHOW INFOTEXT       - Displays contents of INFOFILE (max of 12 lines).
SHOW INFOTEXT 11 15 - Displays lines 11 through 15 of INFOFILE.
```

SIZE
SI

The SIZE command causes the Editor to display the total number of lines in the file. The line pointer is not changed.

```
SORT * parameters
     . parameters
     * filename2 parameters
     . filename2 parameters
     filename1 filename2 parameters
```

The SORT editor macro uses the MUSIC SORT command to sort the entire file you are editing ("sort *") or part of the file that you have MARKed ("sort ."). The sorted records replace the original records, unless you specify a target file name as the second parameter.

Parameters can be specified, if needed. They are the same as on the MUSIC SORT command.

Possible parameters are:

m-n	Starting (m) and ending (n) column of the sort key.
-d	Sort in descending order.
-r	Replace the target file (filename2) without prompting.
-deldups	Delete output records which have the same key field as the previous output record (i.e. delete duplicates).
-xx	Sort control field type: -CH (character or binary), -FI (fixed-point), -FL (normalized floating-point), -ZD (zoned decimal), -PD (packed decimal), -DA (7-character date e.g. "02JAN89").

Default is to sort in ascending order, using the entire record (or the first 256 characters if the records are longer than 256) as the sort key.

When the first parameter is * or . , the records to be sorted are stored to file @SORT.TMP, which is then used as filename1 on a MUSIC SORT command. The resulting sorted file is merged back into the current file, replacing the original lines. The UNSORT macro uses the contents of file @SORT.TMP to undo the sort.

```
SPACE
SPA
```

The SPACE command displays the values from the UCR (User Control Record) for the user's userid. These are the user's file space limits and the total space currently used.

The unit is 1K = 1024 bytes. This information is especially useful for userids with limited file space.

```
SPELL
```

This macro invokes the SPELL program to spell check the current file. After you are finished with the SPELL program, you are returned to the original edit session.

Note: Invoking SPELL from the Editor instead of *Go mode means that no exception list is kept for your dictionary.

```
SPLIT  /string/[Cn][ICm]
SP
```

This command causes the current line to be searched for *string* and if it is found, the current line is broken into two lines, the first consisting of columns 1 through the last column before *string* and the second extending from the beginning of *string* to the end of the line. The C*n* parameter specifies the column number in which the search for *string* is to begin. The second line will be adjusted to start in column 1 unless IC*m* is used to specify an indentation column number. The first line becomes the new current line.

In 3270 full-screen mode only, the screen cursor may be used to indicate the point at which the line is to be split. In this case, the SPLIT command is used without any parameters. The character at which the cursor is placed becomes the first character of the second line. The cursor is left at the end of the first line. F2 has the default definition of SPLIT.

Pointer: Set to the first of the two lines.

Spacing: One blank may optionally be present between the command name or abbreviation and the first slash (/). The second string delimiter (/) may be omitted if Cn and ICm are not used and the string does not end in a blank.

Example:

```
Before:   SAMPLE LINE          <--Pointer
Command:  SP /LI/
After:    SAMPLE               <--Pointer
          LINE
```

```
STORE [n] [m] filename [attributes] [APPEND]
STO
```

The STORE command writes the group of marked lines to an external file. The edit continues. The MARK command (or function key operation) must be used to mark one or more lines before the STORE command can be used.

If the optional APPEND keyword (abbreviation: APP or A) is used following the file name, the marked lines are added to the end of the file, if the file already exists. A new file is created if one does not exist.

Note: When lines are appended to an existing file, unused space is not released at the end of the file. This is intentional, in order to prevent an excessive number of extents being created when several stores are done to the same file. Unused space can be freed later by editing and filing (FILE command) the file, or by running the following job:

```
/FILE 1 NAME(filename) OLD
/LOAD IEFBR
```

The following parameters are available:

n	(optional) starting column number.
m	(optional) ending column number.
filename	the name of the target file.
attributes	(optional) file attributes: PUBL, PRIV, SHR, COM, XO, CNT. When appending to an existing file, the *attributes* are ignored.

Example:

`STORE MYFILE`	Writes marked lines to file MYFILE. (If the file already exists you are asked if you wish to replace it.)
`STO 11 20 MYFILE PUBL`	Writes columns 11-20 of marked lines to file MYFILE, and creates the file as public.

SUBMIT fn1 [fn2] [fn3] ... [kw1(value1)] [kw2(value2)] ...
SU

This command submits one or more files as a job to MUSIC batch, or to other batch processors accessible at your installation.

fn1, fn2, fn3, etc. specifies the names of one or more files to be submitted. The special name of *CUR can be used to indicate the current contents of the file being edited. The files are concatenated in the order specified to form a single job.

kw1, kw2, etc. are keyword parameters whose values *value1, value2,* etc. are substituted into the control statements that are submitted with the job. These parameters are dependant on the particular processor to which the job is submitted, and are usually used to specify items such as time and page limits, passwords, output destinations, etc. The special keyword parameter TO(*processor*) can be used to specify the name of the processor (system) to which the job is to be submitted. Consult your installation for a list of valid processors. If any keyword parameters are specified, they must follow the file names specifications mentioned above.

Refer to the description of "Submitting Jobs to MUSIC Batch and Other Operating Systems" in *Chapter 3. Using Batch* of this manual for full details.

SUBSET n

The SUBSET command specifies which subset of the Editor commands is to be allowed during this edit session, or displays the number of the command subset currently in effect.

Parameter *n* is the number of the command subset which is to apply to the remainder of this edit session. The possible subset numbers are:

0 The full set of Editor commands. This is the default.

1 Subset used by the MUSCOM facility.

Note that once in this subset, you cannot get out of it!

2 Commands for read-only edit (BROWSE). No commands are allowed that would change the file being viewed. Screen text in 3270 full-screen mode is protected. SUBSET 2 is the default for the BROWSE command. You can return to the full command subset by entering SUBSET 0. However, logging (for edit restart) is not started if you used the BROWSE command or the RO option on the EDIT command.

If *n* is not specified, the number of the current command subset is displayed.

Examples:

```
SUBSET      Displays the current command subset number.
SUBSET2     Puts the Editor into read-only (BROWSE) mode.
SUBSET0     Returns to the full set of commands.
```

Messages:

`COMMAND NOT ALLOWED`
Means that the command you entered is not allowed in the current subset.

`OPERANDS ARE NOT ALLOWED ON THIS COMMAND`
Means that you may not use any parameters on this command in the current command subset. The command may be allowed without parameters.

TABIN [c1] [c2] [c3] [c4] [c5]...
TABI
TABSET
TAB

This command provides within the Editor the function of the MUSIC /TABIN command. The command is used to specify to the system the user's input tab settings. When the user then transmits a tab character, the system treats the input line as if the user had spaced to the next tab position. Workstation tabs need not be physically set to correspond with the columns specified in the TABIN command, although it is desirable to do so. (Note that the TABIN command may be used even for a workstation that does not have physical

tabs.) If this command is not used, tab settings remain as specified prior to the Editor session. When the Editor TABIN command is used, tab settings remain in effect after the Editor terminates. See /TABIN in *Appendix C. Seldom Used MUSIC Commands*. When TEXT SCRIPT is in effect tab characters are not expanded. Refer to the TEXT command.

Spacing: Column numbers specifying tab positions must be separated from each other by commas or blanks.

```
TABOUT  [c1] [c2] [c3] [c4] [c5]...
TABO
```

This command provides the function of the MUSIC /TABOUT command when you are using the Editor. The command is used to specify to the system the user's output horizontal tab settings. The system will then automatically take advantage of these tabs whenever possible to save typing time. The command permits the setting of up to eleven (11) output tab positions. These positions should be chosen to optimize printout speed, depending on the format of the output data. The first column is called tab position 1. Invalid tab numbers will be ignored. Tab settings remain in effect after the Editor terminates.

This command should not be used if the workstation is not equipped with horizontal tab capability.

Spacing: Column numbers specifying tab positions must be separated from each other by commas or blanks.

```
TAG  [string]
```

The TAG command displays or sets the tag string associated with the file being edited. The tag is up to 64 characters long, and is stored with the file when a SAVE, FILE, or EXEC command is used. Often the tag is used as a one-line description of the file's contents.

If *string* is specified on the TAG command, it replaces the current tag. To remove the tag, use the command TAG ' '. If no parameter is present, the current tag (if any) is displayed.

```
TEXT  [LC      ]
TEX   [UC      ]
      [SCRIPT  ]
      [NOSCRIPT]
```

The TEXT command is used to specify whether lower case input is to be translated to upper case or not. The command remains in effect until the end of the edit session, or until another TEXT command is used. When TEXT is entered without a parameter, the current setting is displayed.

Parameters:

UC	specifies that lower case input is to be translated into upper case automatically. This is the default when the EDIT command starts the editor with a file that contains all upper case characters.
LC	specifies that lower case input is to be left as is. This is the default for new files and when starting the Editor with files that already contain lower case characters.
SCRIPT	indicates that lower case input is to be left as is, and that input tab characters are not to be translated to the appropriate number of blanks. Tab characters in the file are displayed as "#" characters. TEXT SCRIPT is the default if /TEDIT was used to start the edit.
NOSCRIPT	removes the effect of a previous TEXT SCRIPT command, except that the UC/LC setting is not changed.

See also the CASE command.

```
TIME
TI
```

The TIME and USERS commands cause a line of status information to be displayed. The information includes time of day (in 24-hour clock), date, the amount of processing time (in service units) used since the start of the edit, and the number of MUSIC users on the system.

```
TOLC  [n]
```

This command changes characters to lower case in a line or group of lines, starting with the current line. The last line converted becomes the new current line. Only alphabetic letters (A-Z) in the *zone* portion (as defined by the ZONE command) of each line are changed.

n is the number of lines (1 or more). If *n* is omitted, only the current line is changed. The dot form of this command (TOLC.) can be used to change a marked group of lines.

```
TOP
T
```

The TOP command moves the pointer to the first line of the file.

Pointer: Set to the first line of the file. If the command INSERT, MERGE or INPUT is given immediately after the TOP command, the new lines are placed before the first line of the file. Also, a FIND, LOCATE or

SEARCH command used immediately after a TOP command will include the first line of the file in the search. At the start of the Editor session, the pointer is at the top of the file.

```
TOUC  [n]
```

This command changes characters to upper case in a line or group of lines, starting with the current line. The last line converted becomes the new current line. Only alphabetic letters (a-z) in the *zone* portion (as defined by the ZONE command) of each line are changed.

n is the number of lines (1 or more). If *n* is omitted, only the current line is changed. The dot form of this command (TOUC.) can be used to change a marked group of lines.

```
TRAN
```

The TRAN command restores the output character translation that was suppressed by a NOTRAN command.

```
TREE
```

This macro shows a graphical display of your directories. Same as the MUSIC command TREE.

```
UFIND  [string]
UF
```

The UFIND command searches the file, starting with the line preceding the current line, for a line beginning with *string*. The search works towards the beginning (top) of the file until the first match, or until the pointer is at the top of the file. In the latter case, the message `TEXT NOT FOUND` will result which means that *string* cannot be located between the line preceding the current line and the first line of the file. If VERIFY status is in effect, the line found is displayed at the workstation.

If *string* is not specified, the same string as specified in the last used FIND, LOCATE, UFIND, ULOCATE, HUNT, or SEARCH command is used.

The command can also be specified as UPFIND or UPF.

Spacing: One blank should be present between the keyword or abbreviation and *string*. It is ignored if present. All other blanks are considered to be part of *string*, which extends to and includes the rightmost

non-blank in the command line.

Pointer: Set to the found line, or to the top of the file if no line is found which begins with *string.*

Example:

```
Before:   SAMPLE LINE
          ANOTHER LINE
          YET ANOTHER          <--Pointer
Command:  UF SA
After:    SAMPLE LINE          <--Pointer
          ANOTHER LINE
          YET ANOTHER
```

```
ULOCATE  [string]
UL
```

ULOCATE is used to find the first line before the current line which contains *string* anywhere in the line and, if in VERIFY mode, to display the line on the workstation. The search begins with the line preceding the current line and works towards the beginning of the file. If *string* does not exist in the file between the line preceding the current line and the first line of the file, the message `TEXT NOT FOUND` is displayed. The operation of ULOCATE may be affected by the ZONE setting (refer to the description of the ZONE command).

If *string* is not specified, the same string as specified in the last used FIND, LOCATE, UFIND, ULOCATE, HUNT, or SEARCH command is used.

The command can also be specified as UPLOCATE or UPL.

Pointer: Set to the line in which the *string* is found, or to the top of the file, if it is not found.

Spacing: One blank should be present between the keyword or abbreviation and *string.* It is ignored if present. All other blanks are part of *string,* which extends as far as the rightmost non-blank in the command line.

Example:

```
Before:   SAMPLE LINE
          ANOTHER LINE
          YET MORE             <--Pointer
Command:  UL ER
After:    SAMPLE LINE
          ANOTHER LINE         <--Pointer
          YET MORE
```

```
UNDELETE
UNDEL
```

This command restores the line deleted by the previous DELETE command, provided it was a simple delete of 1 line. The deleted line is inserted before the current line.

```
UNFF
```

This macro undoes the immediately preceding FF macro or PREFIX command.

```
UNFORMAT
```

This macro is used to undo the effect of the FORMAT macro. It restores the formatted file back to its unformatted form that was previous to using the FORMAT macro.

```
UNMARK
UNMA
```

This command unmarks any lines that were previously marked by the MARK command. Refer to the topic "Marking a Group of Lines" earlier in this chapter.

```
UNSORT *
       .
```

The UNSORT editor macro can be used to undo the effects of the immediately preceding SORT macro.

You should specify the same parameter (* or .) as you used on the sort command.

UNSORT uses the contents of file @SORT.TMP, which is created by the SORT command.

```
UP  [n]
U
```

This command moves the pointer *n* lines toward the beginning of the file. If *n* is omitted, 1 is assumed.

If the number of lines specified is large enough to cause the line pointer to be moved beyond the first line of the file, the UP command has the same effect as a TOP command. For example, an immediately following INSERT command would insert a line before the first line.

```
UPPAGE
UPP
```

This command is used only in 3270 full-screen mode. It shifts the screen display one screen (page) towards the beginning of the file. Refer to the section on full-screen mode for more details. UPPAGE corresponds to F7 by default.

```
UPWINDOW  [n]
UPW
```

This command applies only to 3270 full-screen mode. It shifts the screen display (window) towards the beginning of the file. The parameter *n* is the number of lines by which the window is to be shifted. If *n* is omitted, 6 lines is assumed.

```
USERS
US
```

The TIME and USERS commands causes a line of status information to be displayed. The information includes time of day (in 24-hour clock), date, the amount of processing time (in service units) used since the start of the edit, and the number of MUSIC users on the system.

```
VERIFY  [n  ]
V       [ON ]
        [OFF]
```

VERIFY specifies that found or changed lines are to be displayed at the workstation after the following commands: ADD, BLANK, CHANGE, CHANGEL, DELETE, DELETEL, FIND, HUNT, LAST, LOCATE, LOCATEL, NEXT, OVERLAY, SEARCH, SEARCHL, SPLIT, UP. The command VERIFY OFF turns off verify mode (this is equivalent to the BRIEF command). The command VERIFY ON turns on verify mode (same effect as VERIFY without an parameter). If a number *n* is specified, it defines the window as columns 1 to *n* (refer to the WINDOW command), meaning that only columns 1 to *n* of each line are displayed. If an asterisk (*) is specified for *n*, *n* will have a value which is the record length of the file being edited. If *n* is omitted, the window setting is not changed.

```
WINDOW  [n      ]
W       [m n    ]
        [OFF    ]
        [RIGHT k]
        [LEFT k ]
        [FLIP   ]
```

This command specifies the starting and ending columns to be displayed by the Editor whenever a line of the file is displayed. This is particularly useful when long records are being edited. The listing of an external file, by the LIST command, is not affected.

m is the starting column number. *n* is the ending column number, and may be specified by an asterisk (*), which means the record length of the file being edited. If only one number is specified, the number is assumed to be *n*, and *m* is assumed to have a value of 1.

If OFF is specified instead of column numbers, the Editor resets the *window* to be the whole line, as it is when the Editor starts.

The parameter RIGHT or LEFT may be used on the WINDOW command to shift the window towards the right or left hand side of the records being edited. The width of the window is not changed. The parameter *k* is the number of columns by which the window is to be shifted. If *k* is omitted, the window is shifted by the maximum amount. RIGHT and LEFT may be abbreviated R and L.

The FLIP parameter causes the window to be shifted to the extreme left or right, whichever is further from the existing window setting. FLIP can be abbreviated FL or F.

When a WINDOW command is used in 3270 full-screen mode, the zone is automatically set to be from column 1 to the end of the window (refer to the ZONE command).

If no parameters are specified, the current window setting is displayed.

The window setting can also be changed by specifying a number *n* on the VERIFY command. In this case, the window setting will become columns 1 to *n*.

When the WINDOW command is used, the portion of a line displayed by the PRINT command defaults to the window setting. The second parameter on the PRINT command may still be used to display more or fewer characters per line starting from column 1.

Spacing: The parameters must be separated from each other by commas or blanks. If OFF, RIGHT or LEFT is specified, it must be separated from the command by at least one blank.

See also LEFT and RIGHT.

```
X  [n]
```

The X command causes a predefined set of Editor commands to be executed. The commands are defined by the DEFINE command. For example, if "DEFINE X LOCATE ABC;INSERT **" is used, then entering the

command X causes the commands "LOCATE ABC" and "INSERT **" to be performed. An X command string may not contain an X command. The X command may be used on all types of workstations.

If parameter *n* is used (a number from 1 to 24), the command string defined for program function key *n* (PFn) is executed. In this way, function key operations can be defined and used on any type of workstation.

```
XIN
```

This command causes the Editor to go into hexadecimal input format. This form of input is discussed at the beginning of this section. The command AIN is used to return to the regular input format.

```
XL string
XUL string
XF string
XUF string
```

The XL macro locates a character string that you specify. It issues a LOCATE command for the specified character string. However, if the search is unsuccessful, XL does not alter the current line pointer. This has the effect of not altering the current screen when the specified string is not found. If the specified character string is not located, then the current line number is not changed, and a message is displayed indicating that the search failed.

All of the considerations applicable to the LOCATE command apply to the XL macro.

XUL This is the same as the XL macro except that ULOCATE command is used.

XF This is the same as the XL macro except that FIND command is used.

XUF This is the same as the XF macro except that UFIND command is used.

```
ZONE  [n]
Z     [FIXED]
```

The ZONE command is used to display the current zone setting, or to change the zone setting. The zone setting defines the ending column for CHANGE and ADD commands, and for commands which may scan for character strings (such as DELETE, FIND, HUNT, UFIND, ULOCATE, LOCATE, SCAN, SPLIT, and SEARCH), including logical expressions. Columns following the ZONE specification are not changed by CHANGE or ADD commands and are not examined when looking for a character string. ZONE does not affect BLANK, OVERLAY and PRINT commands.

When the SPLIT command is used to break a line into two parts, the second part extends only as far as the

zone setting. Characters following the zone remain on the first line.

n must be a number from 1 to the record length. * is equivalent to specifying the record length, which is the value assumed when the Editor begins execution. If no parameter is specified, the current zone specification is displayed.

The option FIXED (abbr. F) keeps the zone setting unchanged by later WINDOW, LEFT, RIGHT or FS commands (e.g. ZONE * F). The command ZONE F sets the "fixed zone" option without changing the current zone setting. See also the WINDOW command.

```
* comment
```

Any command line starting with an asterisk (*) is treated as a comment, and is ignored by the Editor.

Also, if an Editor command line has an asterisk in column 1, the line is not searched for command delimiter characters (normally semicolon, ";"). But if the * command is not the first one on the line, then delimiter characters are honoured. For example, TOP;*XXX;BOTTOM does a TOP then a BOTTOM, while *XXX;TOP is entirely a comment (no TOP is done).

```
=
=n
=n/string1/string2/[options]
=.
=label
```

This command, without any parameters, causes the Editor to display the line number of the current line. Lines of the file are numbered consecutively starting at 1. When you see the message *EOF (pointer beyond the end of the file), the number of lines in the file, plus 1, will be displayed.

If a line number *n* is specified with the command, the line pointer moves to the line numbered *n* and the line becomes the new current line. The line is also displayed. For example, =30 makes the line numbered 30 the new current line. *n* can be 0 to go to the top of the file or a number greater than the file size to go to the bottom of the file.

If "=." is specified the Editor moves to the first line in a marked group.

If "=label" is specified the Editor moves to the line that was previously labeled with the POINT command.

A change request, with or without options, as in the CHANGE command, can be specified following the line number. Refer to the description of the CHANGE command. For example, =30/ABC/XYZ/ causes the line pointer to move to line 30 and the string ABC is changed to XYZ. The changed line is displayed.

Examples:

```
=                   Displays the line number of the current line.
=300                This sets the line pointer to the line number 300.
=300/NEGATIVE/POSITIVE/

                    This sets the line pointer to the line number 300 and changes the
                    "NEGATIVE" to "POSITIVE" in that line.
=here               This goes to the line that was previously labeled "here" with the
                    POINT command.
```

Advanced Features

The Editor has several advanced features that can be of great assistance in some cases. It is recommended that the user become familiar with the basic functions before studying these advanced features.

Starting Column Suffix -- CN

A number of edit commands can be modified so that their effect will start at a specified column number. For example, you can use the command FC7 WRITE to do a find operation using column 7 rather than 1 -- a great help for FORTRAN programs!

The following is a list of the various Editor commands that take the column number modifier. The notation *Cn* is used where the *n* is the column number from 1 to 8192.

FINDCn string (abbrev. FCn)	a FIND in column n instead of column 1 is performed.
UFINDCn string (abbrev. UFCn)	a UFIND in column n instead of column 1 is performed.
HUNTCn (abbrev. HCn)	a TOP followed by a FINDCn.
LOCATECn string (abbrev. LCn)	a LOCATE starting in column n is performed.
ULOCATECn string (abbrev. ULCn)	a ULOCATE starting in column n is performed.
SEARCHCn string (abbrev. SCn)	a TOP followed by a LOCATECn.
OVERLAYCn string (abbrev. OCn)	*string* is overlaid starting in column *n.*
BLANKCn string (abbrev. BLCn)	a BLANK operation starting in column *n* is undertaken.
INSERTCn string (abbrev. ICn)	a new line is created, composed of *n-1* blanks followed by *string.*
REPLACECn string (abbrev. RCn)	same as INSERTCn, except that new line replaces current line.

For example, to place an X in column 72 of the current line, one might type OC72 X. To remove this X, one might then type BLC72 X. If it is desired to use a *flip* character with any of these modified commands, the flip character must be placed immediately after the last digit in the column number. (The flip character is described under the FLIP command.)

Logical Commands

Several Editor commands have versions which permit the use of *logical expressions*. A logical expression is one or more delimited strings, each of which is considered to have a value of *true* or *false* depending on whether it appears or does not appear in a line. The entire expression must be enclosed in parentheses and may contain any number of nested parenthetical expressions. The string delimiter used is normally a slash (/), but may be any non-alphanumeric character not appearing in the string, except the *not* ¬ symbol. The following connective operators may be used:

OR or |	inclusive "or"
AND or &	"and"
XOR or X	exclusive "or"

The following forms of the *not* modifier may be used:

NOT or .NOT or ¬

Any *not* operation is performed before any *and* and *or* operations at the same level of parentheses, but *and* and *or* operations at the same level are performed from left to right, and *and* does not necessarily have priority over *or*. For this reason, it is recommended that parentheses be used freely to define the order of evaluation of the logical expression. For example, use (/a/or(/b/and/c/)) rather than (/a/or/b/and/c/).

The manner in which the string is searched for may be varied by the use of a string modifier, which is a parenthetical series of parameters placed immediately after the second delimiter. The following parameters apply:

F string must start in the column defined by the Cn parameter, or in column 1 if a Cn parameter is not used in the modifier.

Cn specifies n as the column to begin the search for string.

The following are examples of logical expressions:

```
(/xxx/and/yyy/)

(/xxx/(fc25))

(/xxx/(f)and/yyy/and.not(/jjj/or/kkk/or(/e/and/t/)))

(/aaa/|¬/bbb/(c3)|¬(/rrr/&/ttt/(c16f)))

(not/abcdef/)
```

Defining Function Keys and the X Command

Users editing in full-screen mode on 3270-type terminals may customize the operation of the function keys. This is done by the DEFINE Editor command, which defines a function key as equivalent to any string of Editor commands. Any function keys defined in this way override the standard default function key definitions.

Note: It is possible that your installation has changed the standard default function key assignments, in which case the function key definitions described above may not be accurate. You can check this by entering the command SHOW PF.

For terminals that do not support function keys, the DEFINE command can make the X command equivalent to a string of Editor commands. Then typing the command X causes the specified sequence of commands to be executed. Also, the Xn command (where *n* is a number from 1 to 24) can be used on any type of terminal to execute the command string defined for Fn.

Format:

```
DEFINE Fn  commands
       X
```

n is the program function key number (1 to 24). *commands* are a string of 1 or more Editor commands, separated by the command delimiter character (normally semicolon, ";"). The DEFINE command must be the only command on the input line. An X command string must not contain an X command.

If the command string is omitted, the function key or X command is made undefined.

The abbreviation DEF may be used for DEFINE.

Creating your own Editor

To have certain definitions in effect whenever you use the Editor, create a file named "EDITOR" consisting of /INCLUDE *COM:EDITOR followed by the desired DEFINE commands (plus any other Editor commands).

As an example, suppose you wanted your delete character to be ">" rather than ":", F23 to insert 8 blank lines, F21 to be undefined, F13 and 14 to be MOVE., COPY. and F18 to be the SAVE command followed by SUBMIT *CUR. You could do this by creating a file named "EDITOR" containing the following lines:

```
/INCLUDE *COM:EDITOR
DELCHAR >
DEFINE F23 MINSERT 8
DEFINE F21
DEFINE F13 MOVE.
DEFINE F14 COPY.
DEFINE F18 SAVE;SUBMIT *CUR
SHOW MYPFK
TOP
```

The command SHOW MYPFK permanently displays the contents of file MYPFK at the bottom of the screen. This file could indicate your program function key definitions, as a reminder during the edit session.

The SHOW command is used to display the current definition of program function keys or the X command: SHOW PFn, SHOW PF, or SHOW X.

The ECHO command is specially designed to be used on function keys. Its main function is to cause a string of Editor commands(s) to be displayed in the command area of the screen when a certain function key is pressed. You can then make minor modifications to the command string and subsequently execute it by pressing the ENTER key. As an example, assume that you define the F10 key with the command

```
DEF F10 ECHO MERGE ABC 1 2;LOC XYZ
```

When the F10 key is pressed, the string

```
MERGE ABC 1 2;LOC XYZ
```

is displayed in the command area. You may then modify the line numbers of the MERGE command and press the ENTER key to execute the MERGE and LOCATE commands.

Editing a Large File

The Editor has a work file large enough to edit a file of about 4500 80-byte records (or the equivalent). If you must edit a larger file, or if you get the message `WORK FILE TOO SMALL -- nnnn RECORDS LOST AT END` when you try to edit a file, execute the following job instead of using the EDIT command:

```
/PARM filename options
/FILE 1 UDS(uuuucccc) NREC(nnnnn) VOL(volume) NEW DELETE
/INCLUDE EDITOR
```

The information on the /PARM statement is the same as you would use on the EDIT command. The value of NREC should be large enough to make the work file (unit 1) be at least 2.1 times the size of the file to be edited. For example, to edit a file of 20000 80-byte records, NREC should be approximately

```
(2.1 * 20000 * 80) / 128 = 26250
```

See the MUSIC command called "BIGEDIT" for editing large files.

Editing User Data Set Files

To edit a user data set (UDS) file, execute the following job, using an appropriate /FILE statement pointing to your UDS file:

```
/FILE 4 UDS(dsname) VOL(volume) OLD
/INCLUDE EDITOR
```

Note that the /FILE statement uses unit 4 to point to your UDS file. The OLD on the /FILE allows you to write back the updated version. Use SHR if you only want to look at the file and want the system to protect you from writing on it.

The above control lines can themselves be saved in a Save Library file, in which case you would only have to type the Save Library file name at *Go time to run the edit program on your data set.

The Editor cannot edit files with a record size (RSIZ or LRECL) which is greater than 512 bytes.

The UDS Editor has enough working space to edit files that hold up to 4500 80-byte records or the equivalent. To edit larger files you must provide a temporary UDS file on unit 1 as shown in the following example:

```
/FILE 4 UDS(dsname) VOL(volume) OLD
/FILE 1 UDS(uuuucccc) NREC(nnnnnn) VOL(volume) NEW DELETE
/INCLUDE EDITOR
```

The value of NREC should be large enough to make the temporary UDS file at least 2.1 times the size (in bytes) of the UDS being edited. The size of a file in bytes can be calculated by multiplying the record length (RSIZ) by the total number of records (NREC). The data set name for the temporary file (uuuucccc in the above example), must agree with the usual data set naming conventions for UDS files.

You can provide a /FILE statement for a unit 3 if you wish to write on another file using the unit number option on the Editor FILE command, or to access the contents of another file using the unit number option on the MERGE or LIST command.

Additional Editor Command Input

Commands to be executed by the Editor come from three sources:

1. Commands in the input stream (unit 5), following the /INCLUDE EDITOR statement.

2. Commands (if any) on the same line as the EDIT command, separated from the filename and from each other by semicolons.

3. Normal Editor commands entered conversationally by the user.

Commands from these sources are processed in the order given above.

To supply input stream commands to the Editor, execute the following job instead of using the EDIT command:

```
/PARM filename options
/INCLUDE EDITOR
...Editor commands...
```

The lines following /INCLUDE EDITOR may be Editor commands, or /INCLUDE statements pointing to save files containing Editor commands. The Editor always starts off in command (Edit) mode (not Input mode) when processing these commands.

If you wish to have certain Editor commands always executed at the beginning of each edit, save (under your user code) a private file called EDITOR, consisting of /INCLUDE *COM:EDITOR followed by the desired commands. For example:

```
/INCLUDE *COM:EDITOR
USERS
FLAG SCRIPT
```

Then, whenever you use the EDIT command or /INCLUDE EDITOR (while on your user code), the commands USERS and FLAG SCRIPT will be performed at the beginning of the edit.

Specifying Editor Comments (* Command)

Any command line starting with an asterisk (*) is treated as a comment, and is ignored by the Editor.

Also, if an Editor command line has an asterisk in column 1, the line is not searched for command delimiter characters (normally semicolon, (;)). But if the * command is not the first one on the line, then delimiter characters are honoured. For example, TOP;*XXX;BOTTOM does a TOP then a BOTTOM, while *XXX;TOP is entirely a comment (no TOP is done).

Defining a TAB key

A function key can be defined to perform a tab operation for the Editor. For example, DEFINE F4 <TAB> allows you to advance the cursor to the next tab position, as defined by the TABIN command.

Note: A local tab key is more efficient, and should be used when available (3270 Entry Assist and PCWS).

Using Full-Screen Mode Without Function Keys

If you wish to use full-screen mode with a 3270-type terminal which does not have any program function keys, use the command FS NOPFK. This causes the Editor to put the cursor in the command area whenever the screen is displayed, thus facilitating the entry of commands. To move the cursor to the current line, leave the command area null and press the ENTER key, or simply use the arrow keys.

In addition, the commands UPWINDOW, DOWNWINDOW, UPPAGE and DOWNPAGE may be used in place of function key operations.

Simulating Additional Function Keys

It is possible to make use of all 24 function key operations even if your terminal only has keys for F1 to 12. The extra keys for F13 to 24 are simulated by placing the cursor into the left-hand margin of the screen, one column to the left of a screen line or the command area, and then pressing the corresponding function key (1 to 12). The Editor detects the special cursor position, and simulates the appropriate function key (it adds 12 to the number).

In this way, the LEFT ARROW (<--) key acts somewhat like a *shift* key for the function keys.

The special cursor position is screen column 4 (without line numbers) or column 7 (with line numbers) for a line of the file, and is column 9 for the command area line.

If the cursor is in the margin column when a real F13 to 24 is pressed, then the Editor will simulate the corresponding F1 to 12 key (it subtracts 12 from the number).

Blank-Filled Screen Display

Normally, when the screen is displayed, blanks at the end of each field (i.e. each line of the screen) are replaced by null characters. This allows easy use of the INS MODE key for inserting characters in the middle of a line. When adding characters to the end of a line, be careful to use the space bar to enter blanks, rather than the arrow keys, since null characters are not transmitted to the Editor.

However, for some applications such as entering tables or diagrams, it is more convenient to have trailing blanks retained. This is requested by using the FILL command. If the INS MODE key must be used when FILL is in effect, first use the ERASE EOF key to remove trailing blanks from the field. The NOFILL command reverts to the normal method of display.

An alternate name for the FILL command is NONULLS; an alternate name for the NOFILL command is NULLS. The command NULLS FLIP reverses the NULLS setting.

FILL (or NONULLS) mode is indicated by the ")" character at the end of the tab line.

Output Cursor Positioning

When the Editor displays the screen, it normally puts the cursor at the first position of the current line or the command area. However, commands such as CURSOR, LOCATE and SPLIT can cause the cursor to be displayed at a different position in the current line.

The CURSOR command controls output cursor placement:

```
CURSOR LOCATE
CU     NOLOCATE
       n
       n TEMP
       END
```

n is a number from 1 to 72. It is a column position number relative to the start of the screen field for the current line. 1 refers to the first character of the field, 2 to the second character, etc.

CURSOR LOCATE places the cursor at the start of the found string after any of the commands: LOCATE, UPLOCATE, FIND, UPFIND, SEARCH, HUNT. CURSOR NOLOCATE cancels this option. You can put a CURSOR LOCATE command into your private EDITOR file to make this option the default for all your edits. Abbreviations are LOC, NOLOC.

"CURSOR n" places the cursor at the specified column position whenever the cursor is not put out in the command area, provided a command such as LOCATE or SPLIT or CURSOR END or "CURSOR n TEMP" does not result in a different placement. This stays in effect for the remainder of the edit. To cancel the effect of this command, use CURSOR 1.

"CURSOR n TEMP" places the cursor at position n in the screen field for the current line, but only for the next screen display. Abbreviation T may be used for TEMP, as in CU 15 T. This command is intended mainly for function key definitions. For example, DEFINE F1 INSERT ABC---XYZ;CURSOR 4 TEMP.

CURSOR END is similar to "CURSOR n TEMP", except that the cursor is placed after the last non-blank character in the field.

The S Parameter on the SCAN and CHANGE Commands

When the *S* parameter is used on the SCAN or CHANGE command while in full-screen mode, each found or changed line is displayed on a new screen along with the preceding and following lines, and `More...` appears in the bottom right corner of the screen. Press the ENTER key to advance to the next screen. Screen changes are not allowed.

Example:

```
SCAN/ABC/S
CHANGE/ABC/XYZ/*S
```

If you wish to terminate the SCAN or CHANGE command before all the screens have been displayed, press the PA1 key.

Note: You may find it more convenient to use the LOCATE function key (F9) rather than the SCAN command. Enter a LOCATE or SEARCH command to find the first occurrence of the string, then press F9 to find each subsequent occurrence.

RIGHT and LEFT Parameters on the WINDOW Command

The parameter RIGHT or LEFT may be used on the WINDOW command to shift the window towards the right or left hand side of the records being edited. The width of the window is not changed. Optionally, the number of columns for the shift may be specified. Also, the FLIP parameter can be used to shift the window

to the extreme left or right, whichever is further from the existing window setting. These parameters may be used whether or not full-screen mode is in effect. The format of the commands is:

```
WINDOW RIGHT

WINDOW RIGHT n

WINDOW LEFT

WINDOW LEFT n

WINDOW FLIP
```

The parameter *n* is the number of columns by which the window is to be shifted. If *n* is omitted, the window is shifted by the maximum amount. RIGHT and LEFT may be abbreviated R and L. WINDOW FLIP shifts the window to the extreme opposite side.

When a WINDOW command is used in full-screen mode, the zone is automatically set to be from column 1 to the end of the window (refer to the ZONE command). So remember that if you have used a command such as ZONE 80, you must re-issue it after shifting the window left or right, otherwise the zone may be reset to 72. The ZONE setting affects which columns are changed by the CHANGE command.

Example: Assume the length of the records being edited is 80, and the current window is columns 1 to 72. Then "WINDOW RIGHT 2" results in a new window setting of 3, 74. WINDOW RIGHT or WINDOW FLIP results in a window setting of 9, 80.

Hexadecimal Input and Output

When the Editor begins, it operates in the normal alphanumeric input and output mode. Input and output may, however, be performed in hexadecimal form by using the commands described below. (Hexadecimal format represents each storage byte as two hex *digits*. Each hex digit can have 16 possible values. These 16 possibilities are represented by the sequence 0-9, A, B, C, D, E, F.)

HEX	Output in hexadecimal form only, 32 bytes per line.
ALPHA	Restoration of normal alphanumeric output format.
BOTH	Output in hexadecimal form with each alphanumeric character printed above the left-hand position of the corresponding hexadecimal representation.
XIN	Input in hexadecimal format (see below).
AIN	Restoration of normal alphanumeric input format.
NOTRAN	Normally all output from the Editor is translated to ensure that only printable characters are typed. NOTRAN suppresses this translation.
TRAN	Restoration of normal character translation.

Hexadecimal Input Format

During command mode with hexadecimal input format (XIN) in effect, each line typed by the user is considered to have two fields, separated by one or more blanks. The fields are interpreted as follows:

FIELD 1 Starts at the 1st character and extends to the first blank in the line (or to the logical end of the line if no blank is present). This field always specifies an edit command and is always interpreted as normal alphanumeric input.

FIELD 2 Starts after the first blank and extends to the logical end of the line. In this field any of the 256 EBCDIC characters can be specified by the corresponding hexadecimal value. Blanks are ignored and do not affect the interpretation of the line. Any non-blank character which is not a valid hexadecimal (that is, not one of 0-9, A-F) is taken as is, unless there is an error in the previous part of the line. Any characters, including blanks, may be specified literally by enclosing them in single quotes. To specify a literal single quote, two consecutive single quotes are typed.

In Input mode, the entire line is in FIELD 2 format.

Notes:

1. Although N6 and N 6 are equivalent in AIN mode, they are not equivalent in XIN mode.

   N6, N '6' and N F6 are all equivalent in XIN mode.

2. It is not necessary to close single quotes at the end of a line. End-of-line performs this function automatically.

3. The user using XIN mode must be careful not to code a delimiter in the middle of a string. For example, since a slash (/) is equivalent to hexadecimal 61, the command C /05A1/0461/ would be equivalent to C /05A1/04// which is not a valid command format.

4. The command delimiter character (normally semicolon) will not be recognized as a command delimiter if it is entered in its 2-character hexadecimal form.

Changing Screen Colors

```
COLOR
COLOR fieldname=color fieldname=color...
COLOR Base
COLOR Defaults
```

(Alias: COLOUR). Abbreviations: B for BASE and D for DEFAULT.

The COLOR command redefines the color of various parts of the editor screen. This assumes that the terminal supports 3270 extended data streams (i.e. the Start Field Extended order). Not all terminals support all the possible colors.

The BASE parameter turns off use of 3270 extended attributes (SFE). It reverts to the basic 4 colors (using

SF). Previous extended color settings are still remembered, and are reinstated when a COLOR command without the BASE parameter is used (including a COLOR command with no parameters).

The DEFAULTS parameter turns off use of extended attributes (SFE) and also resets all the field colors to their defaults. This undoes any previous use of the COLOR command.

"fieldname" is the name of a field or group of fields on the editor screen. Only the first 3 characters are significant.

MISCEL IDLINE ID I TAB STATUS ST	Title line, tab line, "Command", "Input Mode", status word ("Reading" etc.) (default: blue)
MSG	Message area (white)
CURLINE CU CURREC	Current line, unprotected (red)
BCURLINE BCURREC	Current line, protected (as in BROWSE) (white)
FILEAREA FI F RECORDS	Non-current lines, unprotected (green)
BFILEAREA BF BRECORDS	Non-current lines, protected (as in BROWSE) (blue)
TOFEOF TO	Top-of-file and end-of-file lines (white)
MARGIN MARK NUM PTR	Arrow pointer, protected line numbers, mark character (blue)
PREFIX PR	Prefix area (green)
CMD CM	Command area (green)
SHOW SH	SHOW text, not highlighted (blue)
HISHOW HSHOW	SHOW text, highlighted (white)

"color" is a 3270 color name or number. The following colors can be used. Only the first 8 characters are significant in the color names.

Color #	Color Name	Abbreviations or aliases	
0	DEFAULT	DEF DE D	<-- usually green
240	NEUTRAL		<-- usually green or black
241	BLUE	BL B	
242	RED	RE R	
243	PINK	P	
244	GREEN	GR G	
245	TURQUOISE	TURQ T	
246	YELLOW	Y	
247	WHITE	WH W	
248	BLACK		
249	DEEPBLUE		
250	ORANGE		
251	PURPLE		
252	PALEGREEN		
253	PALETURQUOISE		
254	GRAY	GREY	
255	BRIGHTWHITE	HIWHITE HIGHWHITE HIWH HIW HWHITE HWH HW	

Example:

```
pref on
color pre=y
```

If the SFE (Start Field Extended) 3270 order is not supported by the terminal, or if an unsupported color is specified, an FSIO error message may appear, and the editor may revert to the base colors (COLOR BASE). Most newer 3270s support SFE and color numbers 0 and 241 to 247. Versions of Net3270 dated Sept./90 or later support SFE and all color numbers. PCWS does not support SFE.

With Net3270, you can define the PC-PS/2 color to be displayed for a given 3270 color name. For example, to define 3270 yellow as PC-PS/2 brown, press Alt-O to invoke Net3270's options screen, and enter the command "color 3270-yellow brown". You can also use Net3270's NETCOLOR program to define the red, green and blue components of a PC-PS/2 VGA color such as brown. This lets you customize the exact appearance of each color.

Editor Macro Facility in REXX

Editor macros can be written in the REXX language. The macros can issue Editor commands, issue MUSIC commands, extract information from the Editor using the EXTRACT command, call other macros, and use the MUSIO and PANEL commands available in REXX. The EDITOR/REXX interface must first be enabled via the REXX ON command (this is the default setting). Once enabled, any command that is not a valid Editor command is passed to REXX. REXX searches for the source of the macro in a PDS structure called MACLIB. If the REXX macro has the same name as an Editor command, the user can use a % symbol preceding the command to distinguish it from the Editor command. If it cannot find the macro, the command is executed as if it were a MUSIC command. This macro facility requires that the Editor be run in at least a 256K region, which is the default. Large macros may require more storage.

Writing Editor Macros

The macros are written in the standard REXX language. Unlike normal REXX programs, the /LOAD REXX or /INC REXX statements that normally appear at the beginning should NOT be present. Also, unlike normal REXX programs, Editor macros cannot issue regular MUSIC commands. Valid commands are MUSIO, PANEL, EXTRACT, and any Editor commands, including any other macros. The parameter entered with the macro command is available through the regular REXX PARSE ARG statement. Editor command strings issued from the macros must contain only one Editor command, since the interface does not support commands separated by the Editor command delimiter. Multiple Editor commands can however be placed on the same line in the macro by using the REXX statement delimiter. Note that quotes and upper case should be used. For Example:

```
'TOP;L XXX;DEL 3'          is invalid.
```

however.....

```
'TOP'; 'L XXX'; 'DEL 3'    is valid.
```

Return Codes

Most Editor commands, macros, and MUSIC commands set a *return code* to indicate the success or failure of the operation. Usually 0 indicates success and a non-zero value indicates an error or an unusual condition. After executing a command or macro, a REXX procedure can test the return code, which is in the special REXX variable RC. To see the various return code values set by Editor commands type "help return" in the Editor's command area.

The EXTRACT Command

The EXTRACT command is used by macros to get information about the current edit session. It causes the Editor to set the requested REXX variables to the appropriate values. The format of the EXTRACT command is:

```
'EXTRACT /varnam1/varnam2/...../varnamx/'
```

This command instructs the Editor to set the variables *varnam1* through *varnamx* to the appropriate values.

The variables that can be set and the values that they are set to are listed below. The entire command is enclosed in quotes to prevent REXX from interpreting the parameter string and performing substitutions. Any delimiter character may be substituted in place of the slash (/), but the slash is recommended to make programs consistent. The command name EXTRACT and the variable names may be in upper or lower case (but upper case is the convention).

The EXTRACT command sets a non-zero return code (rc) if there is a syntax error in the command or if any of the variable names are invalid. An exception is that variable names of the form $xxx (where xxx is a SETV variable - see below) do not cause a nonzero return code, even if xxx is undefined.

EXTRACT variables

Variables set by the extract command are governed by the following convention. For example, if the variable name used on the EXTRACT command is VAR, then the REXX variable VAR.0 will be set to the number of values associated with VAR. The variables VAR.1, VAR.2, ..., VAR.n (where n is the number returned in VAR.0) will be set to those values. The value returned in each VAR.i is a whole number or a character string.

For example, the command "EXTRACT /EOF/" in an editor macro sets REXX variable eof.0 to the number 1 and REXX variable eof.1 to the character string "ON" or "OFF", depending on whether or not the editor is positioned at End-of-File.

Below is a description of variables that can be used with the EXTRACT command. The minimum abbreviation for each variable name is shown in capital letters.

ACTion — Indicates whether changes have been made to the file and its attributes. Each variable is set to the character string ON or OFF.

- ACTION.0 — 4
- ACTION.1 — ON|OFF
  Indicates whether or not the text of the file has been changed since the beginning of the edit (or the last SAVE command). Value ON indicates there are unsaved changes.
- ACTION.2 — ON|OFF
  Indicates whether the file name has been changed by the NAME command since the beginning of the edit.
- ACTION.3 — ON|OFF
  Indicates whether the file tag has been changed by the TAG command since the beginning of the edit.
- ACTION.4 — ON|OFF
  Value ON is returned if the editor work file was too small to hold the edited file at the start of the edit. ON means that records will be lost if a SAVE or FILE command is done.

AUTOSKip — Returns the current settings for the AUTOSKIP command.

- AUTOSKIP.0 — 2
- AUTOSKIP.1 — ON|OFF
  Indicates whether AUTOSKIP INP is in effect.
- AUTOSKIP.2 — ON|OFF
  Indicates whether AUTOSKIP CMD is in effect.

CASE — returns the current case setting.

- CASE.0 — 2
- CASE.1 — UPPER|MIXED
- CASE.2 — RESPECT|IGNORE

CD — returns the current directory name. Returns "\" for the root directory, or else "\dirname" or "\dirname1\dirname2" etc.

CD.0	1
CD.1	current directory (without trailing blanks).

COLOR — returns the current color settings. The alternate spelling COLOUR may be used, in which case the returned variables are also COLOUR.i.

COLOR.0	1
COLOR.1	BASE|EXTENDED

CSRPOS — returns the cursor position and screen size.

CSRPOS.0	6
CSRPOS.1	The cursor column number when the 3270 action key was pressed, relative to the start of the record (not the display window). Value 0 is returned if the cursor was not in a line of the file (e.g. in the command area). Example: if the WINDOW is 11 to 50 and the cursor is at the 3rd character of the displayed text, then CSRPOS.1 is 13.
CSRPOS.2	Cursor row number (absolute position on the screen).
CSRPOS.3	Cursor column number (absolute position on the screen).
CSRPOS.4	Screen size in rows.
CSRPOS.5	Screen size in columns.
CSRPOS.6	Number of screen rows used by the editor, not counting the SHOW text lines. For example, the command area is on row (CSRPOS.6)-1 of the screen.

CURLine — returns the contents of the current line and information about its location in the file.

CURLINE.0	3
CURLINE.1	line number in file. This is always 1 or more. If at End-of-File, (total number of lines) + 1 is returned.
CURLINE.2	line number on screen
CURLINE.3	contents of current line (null string if at End-of-File)

DEFENTer — Gives the command string defined for the Enter key by the command DEFINE ENTER xxx, including trailing blanks. If no string is defined for Enter, a null (0 length) string is returned.

DEFENTER.0	1
DEFENTER.1	The command string or null.

DEFINPut — Gives the command string defined for Input Mode by the command DEFINE INPUT xxx, including trailing blanks. If no string is defined for Input Mode, a null (0 length) string is returned.

DEFINPUT.0	1
DEFINPUT.1	The command string or null.

DELCHAR — Returns the line delete character (normally ":"), or a blank if there is none. The delete character is defined by the DELCHAR command and is used by the MINSERT and MDELETE commands.

DELCHAR.0	1
DELCHAR.1	"x" (where x is the delete character or a blank)

DELIM — returns the command delimiter character (normally ";"), or a blank if there is no command delimiter.

DELIM.0	1
DELIM.1	"x" (where x is the delimiter character or blank)

EOF — returns ON or OFF depending on whether the current line pointer is at the end of file position.

	EOF.0	1
	EOF.1	ON|OFF
FLAG	returns ON or OFF depending on whether modification flagging is active.	
	FLAG.0	1
	FLAG.1	ON|OFF
FLScreen	returns the first and last line numbers on the screen.	
	FLSCREEN.0	2
	FLSCREEN.1	the line number (in the file) of first line displayed on the screen.
	FLSCREEN.2	the line number (in the file) of last line displayed on the screen.
FName	returns the current file name, or a null string if no name is defined.	
	FNAME.0	1
	FNAME.1	current file name (without trailing blanks).
FSMODE	returns ON or OFF depending on whether 3270 full-screen (FS) mode is in effect.	
	FSMODE.0	1
	FSMODE.1	ON|OFF
HEX	Return the settings for hexadecimal input (XIN/AIN commands) and output (HEX/ALPHA/BOTH commands).	
	HEX.0	2
	HEX.1	ON|OFF Indicates whether hex input is enabled.
	HEX.2	HEX|ALPHA|BOTH Indicates the setting for hex output.
KEYHIT	Indicates which 3270 action key was pressed. This applies to full-screen (FS) mode only).	
	KEYHIT.0	1
	KEYHIT.1	The PF key number, or 0 if the Enter key was pressed.
LENgth	Returns the length of the ZONE part of the current line excluding trailing blanks, or 0 if at Top-of-File or End-of-File. The "zone" is columns 1 thru n, where n is set by the command "ZONE n". In full-screen mode on an 80 character wide screen, the zone is columns 1 thru 72, by default.	
	LENGTH.0	1
	LENGTH.1	Length of ZONE part of current line excluding trailing blanks.
LIne	Returns the line number of the current line in the file. This is always 1 or more. If at End-of-File, (total number of lines in the file) + 1 is returned.	
	LINE.0	1
	LINE.1	current line number
LOCSTR	returns the last locate string that was issued with a LOCATE, FIND, SEARCH, HUNT, ULOCATE, or UFIND command.	
	LOCSTR.0	1
	LOCSTR.1	last locate string, or null if no previous locate operation was done.
LOG	Returns information about editor logging. Logging is done to enable an edit restart if failure occurs.	
	LOG.0	2
	LOG.1	ON|OFF Indicates whether logging is enabled.
	LOG.2	The log file id number. This is a number from 1 to 9, if logging is on. It identifies which log file is being used.

LRecl — Returns the logical record length in effect for the edit session.

LRECL.0	1
LRECL.1	Record length.

MArk — Returns the starting and ending line numbers of the marked area. If no lines are marked, zero is returned.

MARK.0	2
MARK.1	starting line number, or 0 if no marked lines.
MARK.2	ending line number, or 0 if no marked lines.

MSG — Returns the text of the last message line put out by the editor. Note that if the complete message was put out as 2 or more lines, only the last line is returned. The text is returned even if the message display was suppressed by the MSGS OFF command.

MSG.0	1
MSG.1	text of last message.

MSGS — Returns OFF if editor messages are currently suppressed by the MSGS OFF command, otherwise returns ON.

MSGS.0	1
MSGS.1	ON|OFF

NEWFILE — Returns ON if the NEW option was used on the EDIT command (or the file name was NULFIL). This means that the editor started without any data.

NEWFILE.0	1
NEWFILE.1	ON|OFF

NULLs — returns ON if 3270 screen fields have trailing nulls (rather than blanks).

NULLS.0	1
NULLS.1	ON|OFF

NUMber — returns ON if line numbering is active, that is, line numbers are displayed.

NUMBER.0	1
NUMBER.1	ON|OFF

PFn — Returns the current program function key definition for function key *n*. Where *n* is a number between 1 and 24. If "*" is specified for *n* instead of a number, (e.g. "EXTRACT /PF*/"), the definitions of all 24 function keys are returned in the variables PF1.i, PF2.i, ..., PF24.i.

PFn.0	2
PFn.1	DEFINED|UNDEF Indicates whether the function key is defined. The returned string is 8 characters long, with trailing blanks.
PFn.2	Function key definition, including trailing blanks. If the key is not defined, a null string is returned.

PREFIX — Returns whether the PREFIX command is active, and other information relating to prefix operations.

PREFIX.0	3
PREFIX.1	ON|OFF indicating whether PREFIX command is active.
PREFIX.2	The color number for the prefix area, as set by the COLOR command.
PREFIX.3	Is "ON" if a prefix operation was executed during this same screen interaction, or if a macro xxx (for "DEFINE INPUT xxx") was invoked to process Input mode lines during this interaction. "ON" indicates that the new current line is as set by the prefix operation or Input mode macro, rather than by the cursor position when the action key was pressed.

RDONLY — Returns ON if the editor is in read-only mode (i.e. the BROWSE command, or command subset 2).

RDONLY.0 1
RDONLY.1 ON|OFF

RECFM Returns the record format for the edit. This is the record format of the file being edited, or the record format specified on the EDIT command. It is the record format that will be used when a file is written by the FILE or SAVE command. If a UDS (user data set) is being edited, or the UIO option was used on the EDIT command, the returned record format is always FC.

RECFM.0 1
RECFM.1 F|FC|V|VC

SHOW returns information about SHOW text. This is the text displayed by the SHOW command at the bottom of the screen.

SHOW.0 6
SHOW.1 actual number of SHOW lines, or 0 if NOSHOW.
SHOW.2 file name used on SHOW command, or null string if none.
SHOW.3 starting line number used on SHOW command, or 1 if none.
SHOW.4 ending line number used on SHOW command, or 99999999 if none. 99999999) if none
SHOW.5 color number for non-highlighted show text, as set by the COLOR command.
SHOW.6 color number for highlighted show text, as set by the COLOR command.

SIZe returns the total size of the file in terms of the number of records. This is the size of the editor's current working copy of the file, not the original file.

SIZE.0 1
SIZE.1 number of records.

SUBSET Returns the command subset number, as set by the SUBSET command. A nonzero subset number may indicate that some commands or command operands are not allowed.

SUBSET.0 1
SUBSET.1 Subset number.

TAG Returns the current file tag, from the original file or from the last TAG command. A null string is returned if there is no tag or the tag is unknown.

TAG.0 1
TAG.1 File tag (64 characters) or a null string.

TERMinal returns DISPLAY or TYPEWRITER depending on the terminal used.

TERMINAL.0 2
TERMINAL.1 DISPLAY|TYPEWRITER
DISPLAY indicates a 3270-type terminal (able to accept 3270 data streams and able to use editor full-screen mode). TYPEWRITER indicates some other type of connection.
TERMINAL.2 BATCH|TERM
BATCH indicates the editor is running as a batch job.

TOF returns ON or OFF depending on whether the current line pointer is at the top of file position.

TOF.0 1
TOF.1 ON|OFF

UDSEDIT returns ON if a UDS (user data set) file is being edited.

UDSEDIT.0 1
UDSEDIT.1 ON|OFF

UIO	Returns ON if the UIO option was used on the EDIT command. UIO.0 1 UIO.1 ON|OFF
WASINPut	Returns ON if the editor was in full-screen Input Mode when the 3270 action key (Enter or a function key) was pressed. WASINPUT.0 1 WASINPUT.1 ON|OFF
Window	Returns the first and last column numbers of the text display area. These column numbers are relative to the file record, not to the screen. WINDOW.0 2 WINDOW.1 Starting column number in record. WINDOW.2 Ending column number in record.
Zone	returns the number of columns of the current zone setting, as set by the ZONE command. ZONE.0 1 ZONE.1 current zone value.
$xxx	An EXTRACT name starting with $ returns the string assigned to the corresponding SETV name. xxx is the 1 to 8-character name used on the editor command "SETV xxx yyy". The EXTRACT command sets the Rexx variable xxx to yyy. Note that no ".n" numeric suffix is used. This allows editor macros to remember things between calls (but only within the same edit session). For example, if "setv abcd this value", then "EXTRACT /$ABCD/" sets Rexx variable ABCD to "this value". If xxx is not currently defined by a SETV command, EXTRACT sets xxx to a null (zero length) string.

Example

The following sets the variables FNAME, CASE, and HEX.

```
'EXTRACT /FNAME/CASE/HEX/'
```

Sample Macros

The following sample macros are available for inspection in the files name.MAC, where "name" is the macro name. The files may be slightly different from the listings shown here. Other interesting sample macros are CALC, CURFIRST, FILEINFO, XL.

1. GET macro.

   The GET macro saves the current file being edited and brings in a new file to be edited.

```
parse arg fname
"EXTRACT /ACTION/"
if action.1 = "ON" then do
   say ' File has unsaved changes. Do you wish to save them?'
   pull resp
   if substr(resp,1,1) = "Y" then "SAVE"
end
"NAME" fname
"TOP"
"DEL 9999999"
"MERGE" fname
rcsave=rc
"TOP"
exit rcsave        /* exit with return code from MERGE */
```

2. BROW, BU, and BD Macros.

   The following three macros (BROW, BU, and BD) are used to allow browsing a file using the show command, while editing another file. Note the use of PF key definitions to preserve information between calls to the macros. (The SETV command could also be used.) The NOSHOW command can be used to cancel this.

   BROW issues the initial show command and sets up PF1 and PF2 to scroll up (BU) and down (BD).

```
/* BROW - init edit browse */
PARSE ARG FNAME
DEF PF1 BU FNAME 1
DEF PF2 BD FNAME 1
SHOW FNAME 1 10
```

   BU scrolls up and redefines the PF keys for next time.

```
/* BU - do edit browse up */
parse arg fname n1
n1=n1-10
if n1<1 then n1 = 1
n2 = n1+9
def pf1 bu fname n1
def pf2 bd fname n1
show fname n1 n2
```

   BD scrolls down and redefines the PF keys for next time.

```
/* BD - do edit browse down */
parse arg fname n1
```

```
n1=n1+10
n2 = n1+9
def pf1 bu fname n1
def pf2 bd fname n1
show fname n1 n2
```

3. The KEYS macro provides a full screen approach to determining and modifying the current function key settings in the EDITOR. The panel used by keys is in the file PFSCR.

```
/* This sample macro allows the user to inspect and modify the
   program function key definitions. The new definitions are
   saved in the user's EDITOR file if required */
'EXTRACT /PF*/'
j=0
do i=1 to 24 by 2
  j=j+1
  pfx='pfscr.' || i || '=pf' || j ||'.2'
  pfy='pfscr.' || i+1 || '=pf' || j+12 ||'.2'
  interpret pfx
  interpret pfy
end
do forever
  'PANEL PFSCR'
  if aid='A1' then exit
  if aid=3 then do
    call getpfk
    call setpfk
    exit
  end
  if aid=10 then do
    call getpfk
    call setpfk
    call savpfk
    exit
  end
end
getpfk:
  j=0
  do i=1 to 24 by 2
    j=j+1
    k=j+12
    l=i+1
    pfk.j=pfscr.i
    pfk.k=pfscr.l
  end
return
setpfk:
  do i=1 to 24
    'DEF PF'||i pfk.i
  end
return
savpfk:
  'MUSIO READ *USR:EDITOR ALL'
  'MUSIO CLOSE *USR:EDITOR'
  if queued()=0 then do
    queue '/SYS REG=256'
```

```
    queue '/FILE MACLIB PDS(*.MAC)'
    queue '/INCLUDE *COM:EDITOR'
    queue 'REX ON'
  end
  j=queued()
  k=0
  do i=1 to j
    pull x
    if substr(x,1,3)¬='DEF' then do
      k=k+1
      rec.k=x
    end
  end
  do i=1 to k
    queue rec.i
  end
  do i=1 to 24
    queue 'DEF PF'||i pfk.i
  end
  'MUSIO WRITE *USR:EDITOR ALL'
  'MUSIO CLOSE *USR:EDITOR'
return
```

Defining Prefix Macros

The prefix area of the Editor is turned on with the PREFIX ON command. There are several commands already defined for this area. Details about using the prefix area can be found earlier in this chapter under the heading "Prefix Area".

The standard prefix commands (I, D, C, MM, etc.) are predefined. You would use the DEFINE PREFIX command only to add your own prefix operations, or to undefine or rename existing ones. When parameters tt and mmmmmmmm are omitted, the prefix command pppp is made undefined.

The types tt for DEFINE PREFIX are:

1	Non-paired command, with no target required (e.g. In, Dn)
1T	Non-paired, requiring a target (e.g. Cn, Mn)
1L	Same as 1, except the operation is done last, after other prefix operations (e.g. /).
2	Paired, no target (e.g. DD, >>)
2T	Paired, requiring a target (e.g. CC, MM)
2L	Same as 2, except the operation is done last, after other prefix operations.
F	Target, of type "following" (e.g. F)
P	Target, of type "preceding" (e.g. P)

You can use REXX to define your own prefix commands. For example, suppose you wanted to use XX to print a group of lines. XX is entered in the prefix area of the first and last lines of the group. You would define it by:

```
define prefix xx 2 myxxmac
```

"myxxmac" is the name of the macro to be invoked (any 1 to 8-character macro name), and "2" is the type of prefix operation (in this case, a "paired" command). You would have to write the editor macro in Rexx, and store it as file MYXXMAC.MAC. The macro would do whatever is needed to print the lines. It would be

invoked by the editor as:

```
MYXXMAC n1 n2 0 0 XX op
```

n1 and n2 are the line numbers of the first and last lines of the group; XX is the prefix command name; op is the parameter (if any).

The prefix command name can contain special characters. For example, the . and .xxx prefix commands are defined by:

```
define prefix . 1 $$point
```

When the prefix command starts with a special character (a non-letter), the command name is ended by the first letter, digit, or blank encountered in the prefix area. For example, /10 and /e both have "/" as the command name, and the rest ("10" or "e") as the parameter.

If the prefix command name on the DEFINE command is already defined, the new definition replaces the old one.

To make a prefix command name undefined, omit the type and macro name parameters. Example: define prefix ff

The pre-defined prefix operations use macro names $$xx, where xx is the command name, e.g. $$I.MAC, $$MM.MAC, ...

The Rexx macro for a prefix operation is invoked with a standard set of parameters, which completely define the parameters for the operation. When the operation (such as CC or MM) involves more than one entry, the macro is invoked only once. The macro is invoked by the following command line:

```
macname num1 num2 targetnum targetcmd cmdname parameter
```

Where:

macname	is the name of the macro.
num1	is the line number (8 digits) of the first (or only) line of the file on which the operation acts.
num2	is the line number (8 digits) of the last line of the file on which the operation acts, or is 0 when only one line is involved.
targetnum	is the target line number, or 0 if no target is involved. The target line number has already been converted to F ("following") form, by subtracting 1 if a P-type ("preceding") prefix command was used. However, when P is entered on the Top-of-file line, 0 is used (not -1).
targetcmd	is the prefix command name (normally P or F) by which the target was specified, or 0 if no target is involved. The name is in upper case.
cmdname	is the prefix command name entered, in upper case.
parameter	is the parameter of the prefix command, as entered (if any). For example, for In or >>n, the parameter is n.

Line number 0 refers to the Top-of-file line. A line number one more than the number of lines in the file refers to the End-of-file line.

For examples of prefix macros, see files $$I.MAC, $$C.MAC, and $$CC.MAC.

Editor Restart Facility

Introduction

A traditional problem with interactive systems has been the effect of system and terminal failures on users. Often much work and input data is lost when the system goes down or the user's terminal is disconnected.

The restart facility of the Editor addresses this problem. The idea is to record all edit activity (commands, input lines, etc.) in a log file, so that if the edit session is terminated abnormally, the changes can be reapplied to the original file when the user signs back on the system.

The restart feature is active by default, and is usually transparent to the user (until a restart is needed).

General Description

At the start of each edit session, the Editor opens a log file (a new file is created if one does not exist). This file is stored in the user's library, and is named @ELOG.sss or @ELOG.sss.n, where *sss* is the user's subcode and *n* is a number from 2 to 9. Some special characters such as the / in the subcode are changed to 0. During the edit session, all Editor commands, input lines, 3270 full-screen changes, etc. are written sequentially to the log file. At normal termination of the edit session, the data in the log file is cleared (although the log file itself is not deleted). A log file name of the form @ELOG.sss.n is used if the normal file (@ELOG.sss) is in use by another edit session on the same userid and subcode.

If the edit terminates abnormally, say because of a system failure or a terminal disconnect, the log file is not cleared. The next time the user starts an edit, using the same userid and subcode as before, the Editor notes that log file data already exists (i.e. the log is not empty) and tells the user that the previous edit may be restarted. The user has the choice of stopping the new edit and restarting the previous one (YES or Y response), or of clearing the log file and continuing the new edit with a fresh log (NO or N response). Note that after a NO response, the log data is gone and thus the restart cannot be done later.

If the user chooses to restart the previous edit, the Editor executes a restart program. This utility program examines the log file and generates an Editor job (in save file @EXEC) which will redo the old edit session up to the point of the failure. The starting point for this Editor job is the last successful FILE command in the edit session, or else the beginning of the session if no successful FILE was done.

The generated Editor job is then executed. This reapplies the Editor commands and file changes, using a special mode of the Editor called *restart mode*. Once the restart is complete, the user is in regular command mode, at the approximate point of the original failure, and may continue editing. Subsequent commands and input lines are added to the old log data. Also, the @EXEC file containing the generated Editor job is deleted.

Note that if the Editor is cancelled during MUSIC attention mode (ATTN), using the /CANCEL command, this is equivalent to an abnormal end, and the next edit will ask whether or not a restart should be done. If the /CANCEL was intentional, simply reply NO to the question.

Logging is not done if the BROWSE command is used, or if any of the option RO, UIO, or NOLOG is used on the EDIT or TEDIT command.

Extent of Recovery

Recovery of the edit session when an edit restart is done should be complete except for the last few commands and input lines entered before the failure. These last lines are not written to the log file because they are still in a main storage buffer at the time of the failure, even though they have been processed by the Editor. The maximum number of log lines lost in this way may be set by the LOG command, described in the next section. The default value is 25 lines for 3270-type terminals and 20 lines for other terminals.

Also, in Editor Input mode (in other than 3270 full-screen mode), the Editor gets control only after each *n* lines of input (where *n* is the number set by the LOG command, or 20 by default). This is because the Editor does spooled conversational reads when in input mode (refer to the section "Spooled Conversational Reads" in *Chapter 4. File System and I/O Interface*. For this reason, up to n additional lines may be lost if failure occurs during Input mode.

On a 3270-type terminal in full-screen mode, screen changes are transmitted to the system only when an action key (such as ENTER or a program function key) is pressed. Therefore all changes entered on the screen since the last action key are always lost.

The LOG Command

The format of the LOG command is:

```
      LOG
or    LOG n
or    LOG END
or    LOG PURGE
```

When used without a parameter, the LOG command forces any log lines being held in main storage buffers to be written to the log file. This ensures that all edit activity prior to the LOG command will be available for restart if necessary.

When used with a number *n* as a parameter, the command defines the maximum number of log lines which will be held in main storage. Initially, the Editor assumes a value of 25 lines, i.e. "LOG 25", for 3270-type terminals and 20 lines for other terminals.

The number n specified on the LOG command is also the maximum number of lines which may be entered in Input mode (in non-full-screen mode) before the Editor gets control. This may be noticeable as a slight pause after each group of n lines entered in Input mode.

Using LOG without a parameter does not change the value n.

As explained in the previous section, additional lines may be lost if the Editor is in Input mode or Full-Screen mode.

If the parameter END is specified, the Editor will clear and close the log file. The log file can then be purged by issuing the command "PURGE @ELOG.sss" to save file space. (*sss* is the user's subcode. Special characters such as the / in the subcode should be replaced by 0).

If the parameter PURGE is specified, it is equivalent to specifying the parameter END and the PURGE command as mentioned above. That is, the log file will be cleared, closed, and purged all at the same time.

Effect on Performance

Apart from the restart process itself, the overhead due to the Editor restart feature is the extra input/output involved in opening and writing to the log file during an edit session. The amount of input/output to the log file is largely controlled by the value *n* used in the "LOG n" command. The smaller the value of *n*, the more input/output and the more overhead. Thus there is a trade-off between efficiency and the amount of data lost when a failure occurs.

With a value for *n* such as 20, the extra overhead should not be a problem, since log file input/output will normally occur only once every few Editor interactions, and each interaction is a time slice involving considerable input/output anyway.

Suppressing the Restart Feature

If you wish the Editor not to create a log file or look for an existing one, use the NOLOG option on the EDIT or TEDIT command. This may be useful if you need to edit something without disturbing the existing log file. For example:

```
EDIT FILEXYZ NOLOG

TEDIT TEXTFILE NEW NOLOG
```

To suppress logging and restart for an edit of a user data set (UDS) file, use the following control statement at the beginning of the job:

```
/PARM *UDS NOLOG
```

When the Editor is run from batch, logging and restart are never done.

Restrictions

- In order to restart an edit, the user must sign on with the same userid and subcode as before.
- If the user has reached the file space limit for the userid, the Editor will not be able to create a log file, and logging will not be done. Logging is also terminated if the log file becomes full and space cannot be added to it. Similarly, a restart may fail if the restart utility program cannot create the file @EXEC because the save file space limit has been reached.
- If a file is specified on a MERGE command, and the file is later purged (deleted) or renamed during the same edit session, the restart may not be successful. One way to avoid this problem is to do a SAVE command after the PURGE or RENAME command.
- If an edit job is run using a /FILE statement for unit 3, and a command such as "MERGE (3)" is used to merge lines from unit 3 into the file being edit, then the edit session cannot be correctly restarted. This is because the generated Editor job will not have a /FILE statement for unit 3, and the MERGE command will fail during restart mode.
- If column number limits are used on an Editor FILE command, the command may still be used as a restart point. This may cause the unsaved columns to be lost when a restart is done.

Sample Terminal Session

In the following example, the user begins by editing the file MYFILE. During the edit, the system goes down before any changes are saved. When the system comes back up, the user signs on again and tries to use the Editor. The Editor sees that log data already exists, and allows the user to restart the original edit. The user then continues editing MYFILE.

Lines entered by the user are shown in lower cases, while messages from the system are in upper case.

```
/id ccccsss    (the user signs on)
...
edit myfile
...
...  (edit commands, etc.)
...
---------- (the system goes down at this point) ----------
...
/id ccccsss    (the user signs on again, same code and subcode)
...
edit myfile    (or edit of any other file)

*** LOG FILE EXISTS FROM PREVIOUS EDIT, AND MAY BE
    USED FOR RESTART (RECOVERY OF UNSAVED DATA).

DO YOU WISH TO CONTINUE THE PREVIOUS EDIT?  (ANSWER YES OR NO)
?
yes

-- EDIT RESTART --     CODE,SUBCODE = CCCC SSS

EDIT-LOG FILE REPRESENTS EDIT AT 14:13 (TODAY)
OF FILE MYFILE

ENTER A BLANK LINE TO PROCEED (OR ENTER /CANCEL TO STOP)
?
               (user enters a blank line)

EDIT SESSION WILL BE RESTARTED FROM THE BEGINNING.

AN EDIT RESTART JOB HAS BEEN GENERATED AND
WILL NOW BE EXECUTED.  ONCE THE RESTART HAS BEEN
DONE, YOU WILL BE IN COMMAND MODE.  PLEASE CHECK THE
FILE NAME AND CONTENTS BEFORE SAVING IT.


...
...  (messages, etc. from re-executed Editor commands)
...

*** RESTART COMPLETED.  PLEASE CHECK YOUR FILE.
...
...  (the user continues the interrupted edit of MYFILE)
...
```

Editor Restart for Multiple Sessions

If multiple sessions are started on the same terminal, or if the same user code and subcode is signed on to more than one terminal, the Editor uses an alternate log file name if the normal log file is in use. The normal (primary) log file name is @ELOG.sss, where *sss* is the subcode. Alternate log file names are @ELOG.sss.n, where *n* is 2 to 9. @ELOG.sss.2 is used if @ELOG.sss is busy. @ELOG.sss.3 is used if @ELOG.sss.2 is busy, etc.

If more than one log file contains restart data when an edit restart is done, the user is asked which edit session to restart. Note: the Editor looks for restart data only in the primary log file. If the primary log file is empty or does not exist, the user must type RESTART.EDIT when in *Go mode in order to do secondary restarts.