TEXT EDITOR REFERENCE MANUAL

OPERATING SYSTEM SOFTWARE

MAKES MICROS RUN LIKE MINIS

PHASE ONE
SYSTEMS, INC.

TEXT EDITOR REFERENCE MANUAL

Second Edition

Revised

Documentation by: C. P. Williams
Software by: Timothy S. Williams

OPERATING SYSTEM SOFTWARE

MAKES MICROS RUN LIKE MINIS

PHASE ONE
SYSTEMS, INC.

7700 EDGEWATER DRIVE SUITE 830
OAKLAND, CALIFORNIA 94621 USA

P R E F A C E

This manual describes the OASIS system text Editor. It provides sufficiently detailed information necessary to the use of this Editor in conjunction with the OASIS Operating System.

This manual, named EDIT , like all OASIS documentation manuals, has the manual name and revision number (if applicable) in the lower, inside corner of each page of the body of the manual. In most chapters of the manual the last primary subject being discussed on a page will be identified in the lower outside corner of the page.

Related Documentation

The following publication provides additional information required in the use of the OASIS Text Editor:

OASIS System Reference Manual

TABLE OF CONTENTS

Section **Page**

CHAPTER 1 INTRODUCTION 1

CHAPTER 2 INVOKING AND USING THE OASIS EDITOR 2

CHAPTER 3 GLOBAL COMMANDS 4
- 3.1 Case Command 4
- 3.2 Htab Command 5
- 3.3 Linemode Command 5
- 3.4 Number Command 6
- 3.5 Prompt Command 6
- 3.6 Renumber Command 6
- 3.7 Tabset Command 7
- 3.8 Truncate Command 7
- 3.9 Unnumber Command 8
- 3.10 Verify Command 8
- 3.11 Wrap Command 8
- 3.12 Zone Command 9

CHAPTER 4 TEXT POINTER POSITIONING COMMANDS 10
- 4.1 Bottom Command 10
- 4.2 The Carriage Return Command 10
- 4.3 Down Command 10
- 4.4 Down Arrow Command 10
- 4.5 Find Command 10
- 4.6 List Command 11
- 4.7 Locate Command 11
- 4.8 Next Command 11
- 4.9 Page Command 11
- 4.10 Top Command 12
- 4.11 Type Command 12
- 4.12 Up Command 12
- 4.13 Up Arrow Command 12
- 4.14 The nnnn Command 12

CHAPTER 5 TEXT MODIFICATION COMMANDS 14
- 5.1 Change Command 14
- 5.2 Combine Command 15
- 5.3 Delete Command 15
- 5.4 Duplicate Command 16
- 5.5 Get Command 16
- 5.6 Input Command 17
- 5.7 Put Command 17
- 5.8 Put and Delete Command 18
- 5.9 Replace Command 18
- 5.10 Split Command 18

CHAPTER 6 FILE MODIFICATION COMMANDS 19
- 6.1 File Command 19
- 6.2 Name Command 19
- 6.3 Quit Command 19
- 6.4 Save Command 20

CHAPTER 7 OTHER EDIT COMMANDS 21
- 7.1 Again Command 21
- 7.2 Column Command 21
- 7.3 Error Command 21
- 7.4 The Edit Macro Commands 21
- 7.5 Question Mark Command 23
- 7.6 CSI Command 23
- 7.7 Skip Command 24

CHAPTER 8 MODIFY COMMAND 25

APPENDIX A COMMAND SUMMARY 27

APPENDIX B GLOBAL COMMAND DEFAULT VALUES 28

APPENDIX C EDIT ERRORS AND MESSAGES 29

CHAPTER 1

INTRODUCTION

The OASIS text Editor allows you to create and maintain files for use by other system programs such as the EXEC language processor, the BASIC interpreter, the MACRO assembler, the SCRIPT processor, etc. Although the Editor is generally used to create or maintain files for these other processors there is not restriction on its use: you can maintain files to be used by your own programs.

A file that is created or maintained by the EDIT program for use by another system program generally contains a mixture of commands and data to that program. These commands or data should not be confused with the commands that the EDIT program uses. For example the operator may wish to create a file of commands and data to be used by the EXEC processor. All of these commands and data are treated as text to the EDIT program.

The OASIS Editor is a full featured text editor with commands that allow you to change, add or delete text from a file. All of these commands are oriented to the user, that is, these commands are English words whose meanings indicate the function that they perform in the Editor. For instance, the command that indicates that you wish to locate the next occurence of a sequence of characters is "LOCATE".

All of the commands may be abbreviated to the first character except those that might cause unrecoverable results and those whose first character would be ambiguous. To clarify this the syntax of each command is given with the minimum required characters in uppercase letters and the remaining unrequired characters in lowercase letters. The commands may actually be entered in upper or lowercase letters.

The OASIS Editor is a line oriented editor. This means that after a command is executed the text pointer is positioned at the beginning of the current line and the commands reference entire lines. For instance the "DELETE" command deletes a line of text; the "INPUT" command inputs lines of text; the "TYPE" command types lines of text; etc. There is a command available that allows you to use the Editor as a character oriented editor. This is the MODIFY command.

For purposes of documentaion and clarity the commands will be divided into six categories:

1) Global commands
2) Text pointer positioning commands
3) Text modification commands
4) File modification commands
5) Other commands
6) Modify command

Each of these categories is discussed in the following sections. The individual commands are discussed under the category headings.

Caution: Do not allow a CTRL/Z character to be placed in a file. This character is interpreted by most programs as meaning the end-of-file and will probably cause loss of data if placed in a file by an operator or user program.

CHAPTER 2

INVOKING AND USING THE OASIS EDITOR

To enter and use the OASIS Editor you use the CSI EDIT command in the following format:

EDIT <file-desc> [(<option>[)]]

Where:

option Indicates one of the two options available with the EDIT command: BACKUP and NOBACKUP. The BACKUP option, which is the default, will create a backup copy of the file being edited before allowing you to save it back on disk. The NOBACKUP option suppresses this feature.

When this command is executed the EDIT program is first loaded into memory and the file description is passed to the program. The Edit program then searches the directory (specified or default search sequence) and, when the file is found, reads the entire file into memory. If the entire file cannot fit into memory due to the amount of memory available the Edit program will display the message: "Available Memory Now Full:" followed by the last line of text that it was able to read. If this line of text is not the last line of text in the file then the operator should abort the edit session and reduce the size of the file by manipulating it with the COPYFILE command.

If memory is filled up when the file is read in and the last line of text is the line that is displayed after the error message the operator may continue the edit session if he first deletes some of the text lines before adding any new text.

If the file is not found on the specified directory or the default search directories then the Edit program will display the message "New File" before displaying the prompt character.

Due to the fact that the disk image of a file being edited is not updated until the operator either SAVEs or FILEs the memory image of the file the System Cancel command is redefined in the Edit session. If this key were not redefined then it is possible that hours of work may be lost by the inadvertant typing of this command. To abort an Edit session the operator uses the QUIT command.

EDIT Prompting Character

After the EDIT command has been executed an asterisk (*) will be displayed on the left side of the console terminal. This is the prompting character for the EDIT program and indicates that the EDIT program is waiting for a command.

EDIT Modes

The OASIS Editor has two modes of operation: the command mode, which is indicated by the Edit prompting character on the left side of the screen, and the text input mode, which is indicated by no prompting character on the left side of the screen.

Most Edit commands operate in one mode or the other, depending upon the command itself. Some commands can operate in both modes, depending upon the parameters given the command. The most notable of the latter include the INPUT and the REPLACE commands.

EDIT Commands, General

Most of the Edit commands have parameters following the command word. These parameters tell the Edit command interpreter what the operator wishes to do specifically. For instance:

```
     Example 1:     TYPE
     Example 2:     TYPE 5
```

In the first example there are no parameters included. The Editor will interpret this command as meaning that the operator wishes to type the current line. In the second example there is one parameter (5). The Editor will interpret this as meaning that the operator wishes to type the current line and the next four lines.

Parameters to the Edit commands may be of two types: numeric and string. Numeric parameters are always assumed to be decimal (base 10). String parameters are always enclosed within delimiters. A delimiter is a character that indicates the beginning or end of something. For more versatility the OASIS Editor allows many characters to be string delimiters, including all non-alphabetic, non-numeric characters. The delimiter may not be part of the string, and the terminating delimiter, when used, must be the same as the starting delimiter. For documentation purposes the slash character (/) will be used for the string delimiter.

If, while typing a line of text or command, the operator should wish to cancel what he has entered, he may type either the program cancel key or a CTRL/X. This will abort the line being typed with no change to the text file.

EDIT HELP Command

The OASIS Editor has a HELP command to assist you by listing the commands available while editing a program. The format of the HELP command is:

HELP

When this command is entered the Editor will display the commands available along with the general syntax of the commands on the console. Since the list of commands is longer than most console displays the Editor will wait at the bottom of each screen for you to enter a key.

Editing Program Files

The OASIS Editor allows the editing of program files (Assembly, BASIC, EXEC, etc.) with added intelligence. For each of these program types the global commands are initialized to special default falues (see section on global default values). In addition, the Editor "knows" that when a BASIC program file is renumbered all references to line numbers must be adjusted to reflect the new line number sequence.

For all of the program types the global CASEMODE command is set to uppercase, but, when editing an Assembly program or BASIC program the CASEMODE is temporarily set to mixed mode whenever the Editor detects that a REMARK (BASIC) or comment (Assembly) is being edited. Also, the Editor temporarily sets the CASEMODE to mixed when string literals are being edited.

CHAPTER 3

GLOBAL COMMANDS

The following OASIS Editor commands are global commands in the sense that they instruct the Edit program how to interpret all of the characters that are entered from the keyboard. These commands are global in effect.

All of the global commands have an initial value depending upon the file type of the file being edited. For instance, the LINEMODE command has a default of ON for the program file types such as BASIC and EXEC but has a default of OFF for file types indicating text files such as SCRIPT and all other file types.

3.1 Case Command

The CASE command instructs the Editor on whether to 'fold' the input from the keyboard to uppercase, lowercase or to accept the input as is.

The format of the CASE command is:

CASE [mode]

Where:

Mode	Meaning
U	'Fold' or change all alphabetic input from the keyboard to upper case. This is the default CASE mode for some program file types (EXEC, FORTRAN, etc.).
M	Accept all input from the keyboard with no translation of case mode. This is the default case mode for all non-program file types.
L	Accept all input from the keyboard with inverse translation. This is the inverse of the CASE mode M. All alphabetic characters typed from the keyboard as text input are translated to their inverse case before display. This feature is useful when the console keyboard does not have a shift lock key.

Program Case Modes

Program source files normally use only upper case characters for the lines of text (statements), but mixed case is usually desired for literals and comments or remarks. For this reason the OASIS Editor provides special case mode features for these situations. In program files a dual case mode is provided that allows you to specify the case mode for literals and comments or remarks separatly from the case mode for the other sections of the text. An example of this dual case mode is 'BL' which means BASIC over lower case mode. The 'B' indicates that the file is a BASIC program file and all non-literals and non-remarks are to be forced to upper case. The 'L' indicates that literals and remarks are to be translated to their inverse case mode.

It is necessary to specify which kind of language the program is written in primarily because comments or remarks are specified differently in various languages. BASIC always starts a remark statement with the characters REM; the ASSEMBLER language, however, always starts comments with a semicolon, etc.

This "intelligent" interpretation of case mode only occurs during multiline INPUT or REPLACEment of text. In all other subcommands and in the command mode of the Editor the alternate case mode is in effect. In the above examle this would be the 'L' mode.

The dual case mode may be specified at one time or separately.

A	Indicates that the text is to be treated as if it was for an Assembly source program. In this mode all characters are translated to uppercase unless they are within quotes (literals) or follow a semicolon in the line (comment). This is the default case mode for file types ASSEMBLE, MACRO, and COPY.
B	Indicates that the text is to be treated as if it was for a BASIC source program. In this mode all characters are translated to uppercase unless they are within quotes (literals) or follow the verb REM.
C	Indicates that the text is to be treated as if it were a COBOL source program. In this mode all characters are translated to uppercase unless they are within quotes (literals) or they follow an asterisk

or slant (/) in column seven (7) of the line.

blank — If no mode is specified then the OASIS Editor will display the current CASE mode.

3.2 Htab Command

The HTAB command instructs the Editor about which character is to be used as a tabulation character and whether or not to translate the character into the ANSI tab character (CTRL/I) or the proper number of spaces. The format of the HTAB command is:

HTAB [char] [ON|OFF]

Where:

char — Indicates the character to be used on input for the tabulation character.

ON — Specifies that the character, when input as text, is to be translated into the ANSI tab character (CTRL/I).

OFF — Specifies that the character, when input as text, is to be translated into the proper number of spaces, according to the current TABSET command.

<blank> — If no character or status is specified after the HTAB command then the Editor will display the current HTAB character and status.

The HTAB command only affects the character specified when input as part of the multiline INPUT or REPLACE command. (INPUT and REPLACE immediate and MODIFY are not affected by this command.)

3.3 Linemode Command

The LINEMODE command instructs the Editor on whether or not to insert line numbers before each line of text. This feature is especially useful when editing a program file and for that reason the LINEMODE command has a default of ON when the file type is BASIC or EXEC.

The format of the LINEMODE command is:

LINEMODE [ON|OFF]

Where:

ON — Indicates that the text is to contain a preceding line number. This is the default for program file types. When the LINEMODE is ON the operator does not type line numbers for each line of text added to the file. Instead the Editor prompts the operator with the next line number available followed by a space. The operator then enters the line of text for that line number. While LINEMODE is ON the operator cannot change the line number of a line of text except by deleting the line and then re-adding it with the new line.

When the LINEMODE is first set ON, ZONE is set to 6, WRAP is set OFF, and the Editor does a sequence check of the file to determine if all of the lines are in their proper ascending sequence. Missing line numbers are acceptable, but if the file contains lines that are out of sequence the Editor will display an error message along with the first line of text that it found out of sequence and set LINEMODE OFF.

If the Editor finds no sequence errors then LINEMODE is set ON.

For more information regarding LINEMODE and line numbers see the RENUMBER, ZONE, PROMPT, MODIFY, and nnnn commands.

OFF — Indicates that the Editor is not to prompt the operator with line numbers nor is it to perform any sequence checks. ZONE is set to 1.

blank — If no ON or OFF is specified for the LINEMODE command then the Editor will display the current status of LINEMODE.

3.4 Number Command

The NUMBER command allows you to add line numbers to a file that was created without line numbers. The format of the NUMBER command is:

NUMBER [start# [incr#]]

Where:

start#	Indicates that the current line numbers are to be numbered starting with this value. If the starting number is not specified the default value is the current value of the PROMPT increment.
incr#	Specifies the value to be added to the last line to produce the line number of the next line. When this value is not specified the current value of the PROMPT increment is used.

3.5 Prompt Command

The PROMPT command instructs the Editor on the increment value for line number prompts. The format of the PROMPT commmand is:

PROMPT [nnnn]

Where:

nnnn Indicates the increment value to be applied to the current line number when the Editor prompts the operator for the next line of input. Unless otherwise specified the increment value has a default of 10 for all file types.

The PROMPT nnnn command will always set the increment value but this value is never used unless LINEMODE is ON.

The value of nnnn must be an integer between 1 and 9999, inclusive.

When the Editor is accepting input of lines of text between two lines of current text and LINEMODE is ON the Editor first adds the prompt increment to the previous line number and checks the line number of the next line. If the prompt line number is greater than or equal to the line number of the next line number then the difference of the last line number and the next line number is divided by two and rounded down to generate the prompting line number.

blank If no increment value is specified after the PROMPT command then the Editor will display the current prompt value.

3.6 Renumber Command

The RENUMBER command instructs the Editor to renumber all line numbers of the text file. The RENUMBER command can not be be used unless LINEMODE is ON. The format of the RENUMBER command is:

RENUMBER [start# [incr#]]

Where:

start#	Indicates that the current line numbers are to be renumbered starting with this value. If the starting number is not specified the default value is the current value of the PROMPT increment.
incr#	Indicates that when the lines are renumbered this value will be added to the last renumbered line to produce the line number of the next renumbered line. When this value is not specified the current value of the PROMPT increment is used.

The RENUMBER command is very powerful when used with a BASIC program. (For the Editor to recognize a file as a BASIC program it must have a file type of BASIC.) The Editor is programmed with the intelligence to update all references to line numbers that are changed; this includes any statement that transfers control to another line in the program, such as GOSUB, GOTO, IF-THEN, etc.

For example:

```
    >EDIT EXAMPLE BASIC S
    EDIT
```

```
*PAGE
  10 REM This is an example - do nothing program
  11 FOR I = 1 TO 10
  18 PRINT I
  22 NEXT I
  26 GOTO 11
  32 IF I>5 THEN 10 ELSE NULL
  40 ON I GOSUB 50,51,52,53,56,70,75
  41 STOP
  50 RETURN 41
  51 RETURN
  52 RETURN
  53 GOTO 50
  56 RETURN 41
  70 STOP\RETURN
  75 GOTO 50
 100 END
*REN 100 10
*TOP
*P
 100 REM This is an example - do nothing program
 110 FOR I = 1 TO 10
 120 PRINT I
 130 NEXT I
 140 GOTO 110
 150 IF I>5 THEN 100 ELSE NULL
 160 ON I GOSUB 180,190,200,210,220,230,240
 170 STOP
 180 RETURN 170
 190 RETURN
 200 RETURN
 210 GOTO 180
 220 RETURN 170
 230 STOP\RETURN
 240 END
```

3.7 Tabset Command

The TABSET command instructs the Editor what column numbers the tab character is to be translated to for display purposes. The TABSET command has different default values depending upon the file type of the file being edited. The format of the TABSET command is:

TABSET [list]

Where:

list — Is a list of column numbers separated by spaces or commas indicating the columns that a tab character (CTRL/I) will position the text buffer pointer to. This command is similar to setting the tab stops with a typewriter. When a tab character is encountered the buffer pointer and cursor are positioned to the next tab column. Only sixteen (16) columns may be specified as tab stops.

The default TABSET for ASSEMBLY Programs is: 10 16 28.

the default tabset for COBOL programs is: 8 12 16 20 24 28 32 36 40 44 48 52 56 60 64 68 72.

the default tabset for FORTRAN programs is: 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80.

The default TABSET for all other file types is: 6 11 16 21 26 31 36 41 46 51 56 61 66 71 76 81.

blank — When no list of column numbers is typed the Editor will display the current tab settings.

3.8 Truncate Command

The TRUNCATE command informs the Editor what the longest line of text will be. The format of the TRUNCATE command is:

TRUNC [nnn]

Where:

nnn	Indicates the column number that text is not to exceed. Whenever a line of text is INPUT, CHANGEd, or MODified this value is checked by the Editor and, if the length of the line exceeds this value the line will be truncated to this length and the message: "Truncated:" is displayed along with the line of text after truncation. The default value for the TRUNCATE command for FORTRAN and COBOL source files is 72. For all other file types the default value is 255.
blank	When no parameter is entered the Editor will display the current TRUNCATE value.

The TRUNCATE command is destructive in the sense that characters entered after the truncation column are lost before the line of text is saved in memory.

3.9 Unnumber Command

The UNNUMBER command removes line numbers from the file in memory. The format of the command is:

UNNUMBER

3.10 Verify Command

The VERIFY command instructs the Editor whether or not to display the current line after a command has been executed, or a line of text has been changed, and, if so, what column number to stop the verification on. The format of the VERIFY command is:

VERIFY [ON¦OFF] [nnn]

Where:

ON	Indicates that the Editor is to display the current line before the Edit prompt character is displayed. In addition, whenever a line is changed by the CHANGE command, the line will be displayed after the change has been made. This is the default status of the VERIFY command.
nnn	Indicates the column number of the last character that is to be displayed when a line is verified. The initial value of this parameter is 255. After the Editor is entered, this parameter is only changed by the operator by using the VERIFY command.
OFF	Indicates that the Editor is not to display text lines unless specified by a PAGE, LIST, or TYPE command. The operator may include a verify column number with the OFF option and the Editor will save the column number but not use it until VERIFY is set ON again.
blank	When no parameters are specified with the VERIFY command the Editor will display the current status of the VERIFY mode.

3.11 Wrap Command

The WRAP command informs the Editor that new lines of text may "wrap" from one line to the next, and that when this occurs the Editor is to clean up the word boundaries at line overflow time. This command is only helpful on text files, especially SCRIPT type files. The format of the WRAP command is:

WRAP [ON¦OFF]

Where:

ON	Indicates that the WRAP mode is to be set ON. This is the default setting for all non-program file types. When the WRAP mode is set ON the Editor will check the character input as the last character of the physical line. If this character causes the line length to exceed the physical line length (63 for the VDM) then the Editor checks to see if it needs to wrap the line to the next line. WRAP mode may not be set ON while LINEMODE is ON. If the character checked is a word delimiter (the space or tab character) the Editor will insert an end-of-line code, position the cursor to the next line and continue to accept input as if the

operator had finished the line with a carriage return.

If the character checked is not a word delimiter, the Editor will search backward to find the beginning of the word, and transfer this partial word to the next line of text. The partial word is erased from the current line in the buffer and on the screen; the line is closed off with an end-of-line code and the cursor is positioned to the next line after the partial word. The operator continues to enter text. All of this "text manipulation" occurs so fast that the operator will notice no degradation of input speed.

When WRAP is ON and the user enters a line of text that contains no word delimiters, the Editor will insert a word delimiter (space) at the end of the physical line. If this feature causes any problems the user should set WRAP OFF before entering a line of text that contains no word delimiters.

When wrap mode is on a multiline input or replace is being performed, the bell will sound five (5) columns before the truncate column.

OFF	Indicates that the Editor is not to allow lines of input text to "wrap" from one line to the next. This is the default mode for all program file types.
blank	When no parameter is specified the Editor will display the current WRAP mode.

3.12 Zone Command

The ZONE command instructs the Editor which column of text is to be treated as the first character. The ZONE column number is used by all of the line oriented, string related commands, except the FIND and MODIFY commands. These commands will ignore all characters before and including the ZONE column. The format of the ZONE command is:

ZONE [nnn]

Where:

nnn	Indicates the character position that the string related commands are to use as the first character of text. This number is automatically set to 6 by the LINEMODE ON command. ZONE may not be set to a value less than 6 while LINEMODE is ON.
blank	When no parameter is specified the Editor will display the current ZONE column number.

The ZONE command is non-destructive, that is, the characters before the ZONE column number are not lost, they are just ignored by the string related commands. For instance, if ZONE were 5 and the operator typed a LOCATE /string/ command the editor will only search for that string in each line of text after column 5.

CHAPTER 4

TEXT POINTER POSITIONING COMMANDS

The following OASIS Editor commands allow the operator to change the position of the text buffer pointer.

4.1 Bottom Command

The BOTTOM command allows the operator to position the pointer to the end of the text. The format of the BOTTOM command is:

BOTTOM

The BOTTOM command will position the pointer to the line before the end of file marker, and, if the VERIFY mode is ON, display the line. If there are no lines of text in the file then the BOTTOM command will display TOF: indicating that the line before the end of file marker is the top of file marker.

4.2 The Carriage Return Command

The carriage return command is provided as a quick and easy means of advancing the text pointer to the next line of text in the file. It is identical in effect to the DOWN 1 command or the NEXT 1 command. The format of the carriage return command is simply to enter a carriage return after the Edit prompt character.

4.3 Down Command

The DOWN command allows the operator to position the pointer down a number of lines or down to the next line that contains a specified string. The DOWN command, when used with a string parameter, a ?, or no parameter, is interpreted by the Edit program to be a LOCATE command. The format of the DOWN command is:

DOWN [nnn|/string |?]

Where:

nnn	Indicates that the Editor is to position down nnn lines. If the value of nnn is greater than the number of lines remaining in the file then the Editor will position down to the end of file marker.
string	Indicates that the Editor is to LOCATE the next occurrence of string. (The LOCATE command is actually executed.)
?	Indicates that the last valid LOCATE command is to be displayed on the console.
blank	When no parameter is specified after the DOWN command then DOWN 1 is assumed.

The DOWN command is identical in effect and format to the NEXT command.

4.4 Down Arrow Command

The down arrow command provides a quick and easy means of specifying that the text pointer is to advance one line of text. When the operator types the down arrow key the Editor will interpret it as the DOWN 1 command, display the message DOWN 1, and advance the text pointer one line of text. Some terminals label the down arrow key as "line feed" or "LF".

4.5 Find Command

The FIND command allows the user to locate or find a line of text that starts with a specific sequence of characters. The format of the FIND command is:

FIND [string|?]

The FIND command does not allow the string of characters to be surrounded with delimiters.

The FIND command searches the text file starting with the line following the current line, searching for a line of text that begins with string. Trailing blanks in string are ignored. Preceding blanks (other than the separator between FIND and string) are treated as part of the string.

The FIND command uses the value of the ZONE to determine the first character of each line. This is useful when program files are being edited.

When the FIND command is used with no string parameter, the last valid FIND command will be executed again.

The FIND command may be used in conjunction with the question mark, indicating that you wish the last valid FIND command displayed on the console. For example:

```
    *FIND The
    The quick brown fox jumped over the lazy black and pink dog.
    *FIND?
    The
    *F
    The beginning of this line starts with the word "The".
    *
```

4.6 List Command

The LIST command is a synonym of the TYPE command and is provided as an alternate means of displaying text. The syntax of the LIST command is identical to the TYPE command except that the command name is LIst.

4.7 Locate Command

The LOCATE command allows the user the ability to locate and position to the next occurrence of a sequence of characters. The format of the LOCATE command is:

LOCATE [/string[/]|?]

Where:

string	indicates the sequence of characters that the Editor is to search for. The search begins with the line following the current line. If the sequence of characters is found then the text pointer is positioned to the line that contains that string, and, if VERIFY is ON, the line is displayed. If the sequence of characters is not found then the Editor displays the message: "Not Found:" followed by the command that is was trying to execute. The text pointer is not changed when the string is not found.
blank	Indicates that the last valid LOCATE command is to be executed again.

The LOCATE command may be used in conjuntion with the question mark, indicating that you wish the last valid LOCATE command displayed on the console. For example:

```
    *LOCATE /The/
    The quick brown fox jumped over the lazy black and pink dog.
    *L?
    /The
    *
```

4.8 Next Command

The NEXT command is a synonym of the DOWN command and is provided as a more natural word for the function of advancing to the next line of text. The syntax of the NEXT command is identical to the DOWN command exécept that the command name is Next.

4.9 Page Command

The PAGE command allows the user to display a page of text and to position the text pointer to the end of the next page of text. The format of the PAGE command is:

PAGE

The PAGE command will display one screen of text.

The first line of text displayed by the PAGE command is the current line. The last line of text displayed on the screen by the PAGE command is determined by the class of terminal designated by the OASIS ATTACH command. For the CRT this would be 15 lines of text. At the end of the execution of the PAGE command the text pointer will be positioned to the beginning of the last line displayed. This means that if two PAGE commands are typed consecutively then the first line displayed by the second PAGE command is the last line displayed by the first PAGE command. In this manner it is easy to observe the flow of text in spite of the page break.

4.10 Top Command

The TOP command allows the operator to position the text pointer to the beginning of the text file. It has the inverse effect of the BOTTOM command. The format of the TOP command is:

TOP

Upon execution of the TOP command the text pointer will be positioned to the top of file marker which is before the first line of text. The Editor will display the message: "TOF:". This command is the only command that will allow the operator to enter lines of text before the first line of text. Refer to the INPUT command.

4.11 Type Command

The TYPE command allows the operator to view several lines of text and to position the pointer at the end of the text displayed. The format of the TYPE command is:

TYPE [nnn]

Where:

nnn	Indicates the number of lines of text to display. When no parameter is entered after the TYPE command the current line of text is displayed and the text pointer is not changed. The TYPE command always displays the current line of text as the first line displayed. After displaying the number of lines specified the text pointer will be positioned to the beginning of the last line displayed. If VERIFY mode is ON the current line will not be displayed again after the TYPEd display as this would be confusing.
blank	When no parameter is specified after the TYPE command then the default value of 1 is used and the current line of text is displayed.

4.12 Up Command

The UP command allows the operator to position the text pointer backward in the text file. The effect of the UP command is the reverse of the DOWN command. The format of the UP command is:

UP [nnn|/string |?]

Where:

nnn	Indicates that the Editor is to position up nnn lines from the current text pointer. If the value of nnn is greater than the number of lines that precede the current line then the text pointer will be postioned to the top of file marker and the message: "TOF:" will be displayed.
string	Indicates that the Editor is to search the file backward for the first occurrence of the sequence of characters indicated by string. The search starts from the current text pointer.
?	Indicates that the last valid UP command is to be displayed on the console.
blank	When no parameter is entered after the UP command then the last UP /string/ command is used. If there have been no prior UP /string/ commands executed then UP 1 is used.

4.13 Up Arrow Command

The up arrow command allows the user a quick and easy means of specifying that the text pointer is to back up one line of text. When the operator types the up arrow key the Editor will interpret it as an UP 1 command, display the message UP 1, and back the text pointer up one line of text. If the console terminal does not have an up arrow key then the user may use the CTRL/Z key instead.

4.14 The nnnn Command

The nnnn command allows the operator to position the text pointer to a specific line number in a text file that contains line numbers. The nnnn command can only be used when LINEMODE is ON. The format of the nnnn command is:

nnnn

The nnnn command does not position to absolute line number nnnn nor to relative line number nnnn. The UP and DOWN commands provide this ability. The nnnn command positions the text pointer to the line whose text begins with the line number nnnn and can only be executed when LINEMODE is ON. The line whose number is nnnn can be either before or after the current position of the text pointer. The Editor determines the direction of positioning by "knowing" that line numbers are in ascending sequence.

CHAPTER 5

TEXT MODIFICATION COMMANDS

The following OASIS Editor commands allow the operator to add, change, or delete lines of text from the file.

5.1 Change Command

The CHANGE command allows the operator to change a sequence of characters to a different sequence of characters. This change can be for one or more occurrences of the string in a line and for one or more lines. The format of the CHANGE command is:

CHANGE [/from-string/to-string[/ [n1 [n2 [n3]]]]]

Where:

from-string	Indicates the sequence of characters that the operator wishes to change.
to-string	Indicates the sequence of characters that the operator wishes the from-string to be changed to. The terminating delimiter is not necessary if the defaults are used for the following parameters.
n1	Indicates the number of lines that the CHANGE command is to examine and possibly change. When this number is not specified the default value of 1 is used and n2 and n3 cannot be specified but have values of 1 also. An asterisk (*) may be used instead of a number indicating that all lines from the current line through the end of the file are to be examined. After the CHANGE command has executed, whether any changes were made or not, the text pointer will be positioned down n1 minus one lines. In other words if n1 is 2 then the current line after the CHANGE command is the next line.
n2	Indicates the number of occurrences per line that are to be changed. When this number is not specified the default value of 1 is used and n3 cannot be specified. An asterisk (*) may be used instead of a number indicating that all occurrences from the starting occurrence are to be changed.
n3	Indicates the starting occurrence number of the from-string on each line that is to be changed. When this number is not specifed the default value of 1 is used.
?	Indicates that the last valid change command is to be displayed on the console.

Entering the CHANGE command with no parameters indicates that the last valid CHANGE command is to be executed again.

The following examples and explanations should clarify the use of the CHANGE command.

```
Example 1: CHANGE /abcdef/xyz

Example 2: CHANGE /abcdef/abc/ 1 1 2

Example 3: C /abc// 1 * 1

Example 4: C /abcd/dcba/ 2 1 3

Example 5: C /abcd/efgh/ 1 *

Example 6: CHANGE

Example 7: CHANGE "^Z"..."
```

Example 1 specifies to the Editor that the current line is to be examined and the first occurrence of the string "abcdef" is to be changed to the string "xyz". Only the first occurrence of this string on the current line is to be affected.

Example 2 specifies to the Editor that the current line is to be examined and the second occurrence of the string "abcdef" is to be changed to the string "abc". Only the second occurrence of this string on the current line is to be affected. After the current line is examined and changed if it qualifies then the text pointer is advanced one line and the new current line is examined in the same manner. No other lines are examined or affected.

Example 3 specifies that all occurrences of the sequence of characters "abc" on the current line are to be changed to the null string.

Example 4 specifies that the third occurrence of the sequence of characters "abcd" on the current line and the line following is to be changed to the string "dcba". A maximum of two occurrences could be changed with this command: one on the current line and one on the line following. After this command has been executed the text pointer is at the beginning of the line that follows the "old" current line.

Example 5 specifies that all occurrences on all lines that the current line, including the current line, are to be changed. In this example n3 has a default value of 1 meaning that the change will start with the first occurrence on each line.

Example 6 specifies that the last valid CHANGE command is to be executed again. In this case the last valid CHANGE command is example 5: CHANGE /abcd/efgh/1 *

Example 7 shows the special case of changing the end of line marker (specified with the CTRL/Z) to three periods. This does not actually change the end of line marker but adds the period character immediately in front of it. This provides a fast and convenient method of changing characters at the end of a line or group of lines.

The CHANGE command may be used in conjunction with the question mark, indicating that you wish the last valid CHANGE command to be displayed on the console. For example:

```
    *CHANGE /abc/def/1 *
    *CHANGE?
    /abc/def/1 *
    *
```

5.2 Combine Command

The COMBINE command joins the current two lines of text (the current line and the line immediately following) into one line of text. The format of the COMBINE command is:

COMBINE

For example, if the current line of text is:

Now is the time for all

and the next line of text is:

good men to come to the aid of their country.

then the COMBINE command would remove those two lines and replace them with:

Now is the time for all good men to come to the aid of their country.

5.3 Delete Command

The DELETE command allows the operator to delete whole lines of text from the file. The format of the DELETE command is:

DELETE [nnn|/string]

Where:

nnn	Indicates the number of lines to be deleted, including the current line.
string	Indicates that the current line is to be deleted and all lines that follow, up to, but not including, the line that contains string. If no occurrence of string is found following the current line then all lines are deleted from the current line to the end of the file.
blank	When no parameter is specified after the command DELETE then only the current line is deleted.

5.4 Duplicate Command

The DUPlicate command allows the user to duplicate the current line of text one or more times. The format of the DUPlicate command is:

DUP [nnn]

The DUPlicate command can only be used when LINEMODE is OFF as it would create duplicate line numbers otherwise.

The DUPlicate command will duplicate the current line nnn times and place the duplicate lines following the current line. When nnn is not specified the default value of 1 is used.

5.5 Get Command

The GET command allows the user to add lines of text to the file in memory, previously stored on a temporary work file or an external disk file. The format of the GET command is:

GET [<file-desc>] [/fr-string/|fr-count [/to-string/|to-count]]

Where:

file-desc	Indicates the external file description of a sequential disk file that is to be opened and added to the file in memory. When file-desc is not specified then the temporary work file used by Edit is used. Text is added to this temporary work file with the PUT command and only exists during the current edit session.
fr-string	Indicates a delimited string. The file is searched for this string and the first line of text that contains this string is the starting line of text transferred to the file in memory.
fr-count	Indicates that the first fr-count minus one lines of the work or external file are to be ignored. For instance, if fr-count is 5 then the fifth line of the work or external file is the first line transferred. When fr-string or fr-count is not specified to-string or to-count may not be specified. In this case the entire work or external file will be transferred.
to-string	Indicates that lines of text are to be transferred up to and including the line containing to-string.
to-count	Indicates the number of lines of text to be transferred.

The GET command may be used only when LINEMODE is OFF.

Example

If a file named SAMPLE.TEXT:S contains the following information -

```
The OASIS Editor allows the user to create and
maintain data files that are used by other
system programs such as the OASIS BASIC inter-
preter or the SCRIPT processor. The files
maintained by the EDIT program are not necessarily
restricted to use by these other system programs.
```

and the operator type this command:

```
*GET SAMPLE.TEXT:S /system/ 3
```

the following lines of text will be added to the file in memory -

```
system programs such as the OASIS BASIC inter-
preter or the SCRIPT processor. The files
maintained by the EDIT program are not necessarily
```

After the lines are added the current line will be the line starting with the word "maintained".

5.6 Input Command

The INPUT command allows the user to add new text to the file. It operates in two modes: single line and multiple line. Single line input mode is referred to as immediate mode. The format of the INPUT command is:

INPUT [string]

When the INPUT command is followed by a space and text the text will be input to the file as a new line after the current line and the new text will become the current line. Only one line of text may be input in this immediate mode.

When LINEMODE is ON then the line of text being input with the immediate mode must include a line number or an error message will be displayed. The line number must be followed by a space.

When a line of text is input using the immediate mode the space that separates the INPUT command and the text is not added to the file; it is only a delimiter.

When the INPUT command is followed by a space and no text a blank line will be added to the file after the current line and this blank line will become the current line.

When the INPUT command is not followed by any text but only a carriage return then the INPUT command will allow multiple lines of text to be added. These lines of text will always be added after the current line. As each line of text is added to the file it becomes the current line.

When the Editor is accepting multiple lines of text no prompting character is displayed on the left side of the console. This is because the prompting character indicates that the Editor is ready to accept a command and in this case it is not accepting commands but text. When LINEMODE is ON the operator is prompted with the next available line number at the beginning of each line of text. When LINEMODE is OFF no prompt of any kind is displayed.

It is only when the Editor is accepting multiple lines of text that the WRAP mode ON has any effect. In addition to the normal use of the WRAP mode ON as explained in the WRAP command, the user gains the ability to add "null" lines. This means that when you wish to have a blank line of text the operator does not have to type a space followed by a carriage return but may type a carriage return only.

There are two methods of returning to the command mode, depending upon the status of the WRAP mode. When WRAP mode is OFF the user merely types a carriage return with no text preceding it. When WRAP mode is ON the user must type the Program Cancel-key to end the input. If the Program Cancel-key is typed in the middle of a line of input that line will not be added to the file.

5.7 Put Command

The PUT command allows the operator to save lines of text in a temporary work file or an external disk file. The format of the PUT command is:

PUT [file-desc] [count|/to-string/|*]

Where:

file-desc	Indicates the file description of the external disk file. When this parameter is not specified the temporary work file is used.
count	Indicates the number of lines of text that are to be written to the file. The transfer of text always starts with the current line.
to-string	Indicates that lines of text are to be transferred to the file up through and including the line that contains this string. When this string is not found in the text then the remainder of the text is copied to the file.
*	Indicates that the remainder of the file in memory is to be copied to the file specified.
blank	Indicates that the current line only is to be copied to the file specified.

The PUT command is a destructive write. By this is meant that the previous contents of the external disk file or temporary work file are erased before the new text is written.

The text pointer will be positioned to the last line of text that was transferred to the file.

5.8 Put and Delete Command

The PUTD command allows the operator to save text in a temporary work file or an external disk file and to delete that text from the file in memory. The syntax of the PUTD command is identical to the PUT command with the exception that the command name is PUTD.

The PUTD command functions similar to the PUT command with the addition that the lines of text are deleted from the file in memory as they are transferred to the work or external file. The text pointer will be positioned to the line following those lines transferred.

5.9 Replace Command

The REPLACE command allows the user the ability to replace an existing line of text with a new line or lines of text. The format of the REPLACE command is:

REPLACE [text]

Similar to the INPUT command discussed above the REPLACE command has two modes: immediate and multiple line. In the immediate mode the REPLACE command is separated from the text by one space. When executed the text replaces the current line and the Editor returns to the command mode. The immediate mode of the REPLACE command is not allowed when LINEMODE is ON.

When the REPLACE command is not followed by any text the operator may replace the current line with multiple lines of text. The current line is not actually deleted until it is replaced by at least one line. This feature allows the operator to recover from a bad replace if he has not actually finished typing the first line. When the first line is a return to the command mode with no text, the Editor restores the current line with no change to the text file.

The exit from the multiple line REPLACE command is identical to the INPUT command.

5.10 Split Command

The SPLIT command divides the current line into two lines of text. The format of the SPLIT command is:

SPLIT /string[/]

Where:

string	Indicates the string of characters after which the split is to be performed.

For example, if the current line of text is:

```
    Now is the time for all good men to come to the aid of their country.
```

and the command SPLIT /aid/ is given then the text will look like:

```
    Now is the time for all good men to come to the aid
    of their country.
```

with the second line becoming the current line.

CHAPTER 6

FILE MODIFICATION COMMANDS

The following OASIS Editor commands affect the disk image of the text file being edited. The various commands allow the user to update the file and return to the CSI; update the file and continue the EDIT session; return to the CSI without updating the file; interrogate the name of the disk file; change the name of the file. None of the update commands may be abbreviated due to the chance of an inadvertant typing error.

You may not SAVE or FILE a file that is delete protected (all non-private files have an implied delete protections).

6.1 File Command

The FILE command allows the user to terminate an EDIT session normally by updating the file and returning to the CSI. The format of the FILE command is:

FILE [<file-desc>]

Where:

<file-desc> Indicates an optional file description. When the <file-desc> is specified, only as much as is necessary need be entered, the omitted parameters will default to the current file description. The current file description may be interrogated with the NAME command.

When <file-desc> is specified the current file description is not changed but the <file-desc> specified is the description used by the FILE command.

When no <file-desc> is specified after the FILE command the current file description is used. In either case when the FILE command is executed the Editor writes the file in memory to the disk file. If the Editor finds an existing file on the specified disk with the same name and type as that being filed, the file type of that file is renamed to BACKUP, erasing any other file by that same description. When the Editor has successfully "filed" the text file the file description that it was saved as is displayed on the console terminal along with the message "filed". Control returns to the CSI.

6.2 Name Command

The NAME command allows the user to interrogate the current file description or to change the file description. The format of the NAME command is:

NAME [<file-desc>]

Where:

<file-desc> Indicates the file description to be changed to. Only as much of the description as is necessary need be entered. For instance if only the file name is to be changed then only the file name need be entered. If the file type is to be entered then both the file name and type must be entered because the parameters are position dependent. An asterisk may be used for the file name to indicate it is the same as the current file name.

blank When no file description is entered then the Editor will display the current file description.

6.3 Quit Command

The QUIT command allows the user to abort an Edit session without updating the file on the disk. This may be necessary for many reasons such as specifying the wrong file in the EDIT command or the operator has decided that the changes made to the file should not be saved. The format of the QUIT command is:

QUIT [command]|[return code]

When the QUIT command is executed with no command or return code following, the Editor will set the return code to zero and return control to the environment that invoked the EDIT command (CSI or EXEC) without updating the file. When the command parameter is specified the Editor will pass the command to the CSI without updating the file. The specified command must be a valid OASIS command with all parameter and options included. After the specified command has been executed control will return to the environment that invoked the EDIT command. When the return code is specified it must be a numeric value between 0 and 255. The value is assigned to the return code and control is returned to the environment that invoked the EDIT command.

6.4 Save Command

The SAVE command allows the user to save the current status of the text file on the disk and to continue the Edit session. The format of the SAVE command is:

SAVE [<file-desc>]

The <file-desc> option is the same as for the FILE command. When the SAVE command is executed the Editor updates the disk file using the file-desc specified or the current file description when not specified. Any file on the disk with the same description is renamed to file type BACKUP.

When the SAVE command has successfully updated the disk file the file description used is displayed on the console terminal along with the message "saved". Control returns to the Edit command mode.

The Program Cancel-key may be used during the execution of the SAVE command. When it is used the disk file being updated is erased to insure that the disk does not become misallocated and control is returned to the command mode.

CHAPTER 7

OTHER EDIT COMMANDS

This section discusses the OASIS EDITOR commands that do not fit into the previous categories due to the fact that these commands may invoke commands from various categories. These commands include the AGAIN command which re-executes the last command executed; macro commands which may be a series of commands from any category; the question mark (?) command which displays the last valid command executed; the CSI command which executes most of the OASIS commands.

7.1 Again Command

The AGAIN command re-executes the last valid, action type command executed. The format of the AGAIN command is:

AGAIN

All action type commands are repeatable by the AGAIN command. These commands include: BOTTOM, CASE, CHANGE, COMBINE, CSI, DELETE, DOWN, DUP, FILE, FIND, GET, HTAB, INPUT, LINEMODE, LIST, LOCATE, MODIFY, NAME, NEXT, PAGE, PROMPT, PUT, PUTD, REPLACE, SAVE, SPLIT, TABSET, TOP, TRUNCATE, TYPE, UP, VERIFY, and ZONE.

7.2 Column Command

The COLUMN command displays the column numbers across the console screen. The format of the command is:

COLUMN

For example:

```
*TYPE
Now is the time for all good men
*COL
         1         2         3         4         5         6         7
1234567890123456789012345678901234567890123456789012345678901234567890123456789
Now is the time for all good men
*
```

7.3 Error Command

The ERROR command, available only when used within a macro, allows you to specify an action to be taken when another command within the macro fails (specifically, CHANGE, FIND, and LOCATE commands). The format of the ERROR command is:

ERROR [count]

Where:

count Indicates the number of commands following the ERROR command to be skipped if the command following the ERROR command fails.

For example, the following macro will insert a line of text after the last line that starts with HI.

```
      ----------------------
      |                    |
      |                   \|/
  X ERROR 2&FIND HI&SKIP -3&I THIS IS A LINE OF TEXT
```

7.4 The Edit Macro Commands

A macro command is a stored Edit command or list of commands that can be recalled later for execution. The OASIS EDITOR has three macro commands available to the operator: X, Y, and Z.

The macro commands allow the user to repeat any command or sequence of commands.

The three macros are labeled X, Y, and Z. For documentation purposes only the X macro will be discussed. Each of the three macros can be used in the same manner as the other two. Each may be loaded with a command or list of commands and each may be later executed independent of the other two. One macro may execute another macro as a command in its list.

Loading A Macro

Before a macro command may be executed it must first be loaded with a command or list of commands. To do this the user types the macro name followed by the command or commands joined with the ampersand (&) character. For example:

```
     X CHANGE /abcd/dcba/
     X TOP&TYPE 5&DOWN 22
```

If one of the strings to be specified in a macro contains an ampersand character the operator must type two ampersand characters for the Editor to accept it correctly (&&).

The commands that are loaded into a macro may be abbreviated in the same form as if they were typed for normal execution. When the commands are loaded they are not executed but only stored in the macro.

Any valid Edit command may be loaded into a macro for later execution.

Each of the macros may contain up to 255 characters.

Executing A Macro Command

A macro command may be executed any time the Editor is in the command mode and the macro being executed has at least one command stored in it.

An Edit macro is executed by typing the macro name, optionally followed by the number of iterations that the macro is to be executed. For example:

```
     X
     X 3
```

When a macro is executed the commands are not displayed on the console terminal but the results of each command will be displayed if VERIFY mode is ON.

The execution of a macro does not change the contents of the macro.

A macro will be executed as many times as the user so specifies in the iteration number.

The execution of a macro may be aborted by the Editor if it contains a LOCATE, FIND, or CHANGE that is not successful.

The execution of a macro may be aborted by the user by typing the Program Cancel-key.

The current contents of a macro may be interrogated by typing the macro name followed by the question mark.

Edit Macro Examples

The following examples illustrate some of the uses of the macro commands.

```
          Example 1:  X L /OPEN/&U&I CLOSE
          Example 2:  X L /abcde/&C /XYZ/the/&U 2&P
          Example 3:  X
          Example 4:  X 4
          Example 5:  X LOCATE /abc/&DEL&X
          Example 6:  X ?
```

Example 1 loads the X macro with three commands. When this macro is executed it will first locate the string "OPEN", and then input a line before the line that contains the string "OPEN". If it cannot locate the string "OPEN" an error message will be displayed and the macro will be exited.

Example 2 loads the X macro with four commands. When it is executed it will first locate the string "abcde" then change the first occurrence of "XYZ" on that line to "the". After the CHANGE command is executed the text pointer is positioned UP two lines and a PAGE of text is displayed. Again, if the first string cannot be located the macro will be exited.

Example 3 executes the X macro one time.

Example 4 executes the X macro four times. For the second through the fourth iterations the macro begins execution with the text pointer positioned as it was left by the previous iteration.

Example 5 loads the X macro with two commands and a call to itself. This macro, when it is executed, will LOCATE the line that contains the string "abc" and DELETE it. Then it will execute itself. This "recursive" sequence of events will continue until the string "abc" cannot be found. At this point the macro will be exited.

Example 6 will display the current contents of the X marco.

7.5 Question Mark Command

The question mark command (?) allows the user to display the last valid command executed. When the a question mark is entered at the command level, the Editor will respond with the last command executed. For example:

```
*CH /abcd/efgh/
*?
CH /abcd/efgh/
*up 5
*?
UP 5
```

In addition to the "stand alone" question mark command the question mark may be used in conjunction with certain other commands to display the current status of the command in 'question'. The commands that may be used with the question mark include: CHANGE, FIND, LOCATE, UP, DOWN, and the macros X, Y, and Z.

For example:

```
*LOCATE /abcde/
*LOCATE?
/abcde
*CHANGE /abc/def/1 *
*FIND The
*?
FIND The
*C?
/abc/def/1 *
*L?
/abcde
*FIND?
The
*?
FIND The
*X?
LOCATE /abc/&DEL&X
```

7.6 CSI Command

The CSI command allows the operator to execute most of the OASIS commands without leaving the Editor. The format of the CSI command is:

CSI **<command>**

The command is any valid OASIS command including all of the parameters and options desired. When the CSI command is executed by EDIT the status of the EDIT program and the text file itself is saved in high memory. Control is transferred to the Command String Interpreter along with the command.

When the first character of the command is a greater than character (>) the command will be displayed on the console. When the first character is not a greater than then the command is executed "silently". Output from the command is determined by the command itself.

When the command has completed its execution control is returned to EDIT. At this time EDIT reloads itself, restoring the status and text file from high memory.

You not use the CSI command to execute the larger OASIS commands such as EDIT, BASIC, etc.

CSI Example

```
*TOP
TOF:
*CSI >RENAME TEXT3A.BACKUP:S = OLD3A =

>RENAME TEXT3A.BACKUP:S = OLD3A =
TEXT3A.BACKUP:S has been renamed TEXT3A.OLD3A:S
*
```

7.7 Skip Command

The SKIP command, available only within a macro, allows you to specify branching within the set of macro instructions. The SKIP command is usually used in conjunction with the ERROR command. The format of the SKIP command is:

SKIP [[-]count]

Where:

count	Indicates the number of commands to be skipped counting from the command following the SKIP command.

For example, the following macro will skip back to the ERROR command after each successful execution of the FIND command:

```
    X ERROR 2&FIND HI&SKIP -3&I THIS IS A LINE OF TEXT
        /|\               |
         |                |
         ------------------
```

CHAPTER 8

MODIFY COMMAND

The Edit MODIFY command is discussed separately in this section for two reasons: it is the only command that allows the Editor to act like a character oriented text editor, and when the MODIFY command is being executed there is a new set of commands available to the user. The format of the MODIFY command is:

MODIFY [nnn|*]

Where:

nnn	Indicates the number of lines that are to be modified.
*	Indicates that all lines starting with the current line to the end of the text file are to be modified. To exit from this mode before the end of the file is reached you must use the Program Cancel-key.
blank	When no parameter is specified then the default value of one is used indicating that the current line is to be modified.

When the MODIFY command is executed the line to be modified is displayed on the console terminal and the cursor is positioned at the first character position. The Editor is now in a character oriented mode and the MODIFY command makes full use of the fact that the console terminal is normally a CRT with cursor positioning controls. The cursor indicates the current text pointer position in the line. Any changes made to the line are immediately indicated by the display of the line and the position of the cursor.

Any control characters imbedded in the line are expanded for display purposes to two characters: an up arrow character (^) followed by the ASCII representation of the control character. For example, a CTRL/I is displayed ^I. Even though the control character is displayed as two characters it actually is only one character. When a control character is deleted from the line both characters are erased from the display. When a control character is skipped over, both characters are skipped, etc.

The commands available to you when the Editor is modifying a line are one character commands, but still the one character is the first character of the word that it is an abbreviation for. When possible, it is the same character that would be used in the Edit command mode.

The following sub-sections discuss the commands available from the MODIFY command. When you type a command it is not displayed as that would disrupt the display of the line being modified. Any effect that a command has on the text in the line is immediatly displayed however.

Insert Characters Subcommand (I)

To insert new characters into the line the user types the insert command (I). Any characters typed after the I has been typed are added to the line before the current character. As each character is added to the line the remainder of the line is re-displayed.

To exit from the Insert character command you type a carriage return.

While in the insert character command you may backup one character position by typing the RUBOUT key. This backs the text pointer up one position, the cursor is backspaced, and that character is deleted. It is possible to backspace past the position that the insert command was given.

Delete Character Subcommand (D)

To delete a character from the line you use the delete character command (D). Every time a D is typed the current character is deleted from the line and the character is erased from the screen. This is an immediate command.

Replace Characters Subcommand (R)

To replace characters in the line you use the replace character command (R). After the R command has been typed each character that is typed replaces the current character and the text pointer is advanced one character position.

To exit from the replace character command you type a carriage return.

While in the replace character command, you may backup a character position by typing the RUBOUT key. This will back the text pointer up one position and the cursor will backspace. No characters are deleted and the Editor is still in the replace character command. It is possible to backspace past the position that the replace character command was typed and still remain in the replace character mode.

Advance Character Subcommand ( )

To advance the cursor and text pointer one position you type the advance character command ( ), a space. When the space is typed the text pointer and the cursor are advanced one character position. You may not advance past the end of the line, however you may insert new characters at the end of the line or replace characters at the end of the line.

The right arrow has the same effect as the space key and is more graphic in its meaning. Both may be used interchangeably.

Find Character Subcommand (F)

To advance the cursor and text pointer to the next occurrence of a specific character you use the find character command (F) followed by the character to be positioned to. When the F is typed followed by another character the cursor is advanced to the next occurrence of that character in the line being modified. The character specified must be entered in the same case as the character to be found. When the character can not be found the cursor will be positioned to the end of the line.

Backspace Character Subcommand (RUB)

To back the text pointer and cursor one character position you type the backspace character command. This command is any of the keys defined as the character delete keys, such as left-arrow, CTRL/H or RUB. The left arrow is more graphic in its meaning and is usually the key that is used for the RUBOUT. When this key is typed the text pointer and the cursor are backed up one character position.

Uppercase Character Subcommand (U)

To change the current character to uppercase you type the uppercase character command (U). When the U character is typed the current character is translated into uppercase, redisplayed and the text pointer and cursor are advanced one character position.

Lowercase Character Subcommand (L)

To change the current character to lowercase you type the lowercase character command (L). When the L is typed the current character is translated to lowercase, redisplayed and the text pointer and cursor are advanced one character position.

Beginning of Line Subcommand (B)

To position the cursor and text pointer to the beginning of the line being modified you use the beginning of line command (B). When the B is typed the cursor is positioned to the first character in the line.

End of Line Subcommand (E)

To advance the cursor and text pointer to the end of the line you type the end end of line command (E). When the E is typed the cursor and text pointer are advanced to the end of the line.

This command allows the user to easily add or change text at the end of the line.

Quit Subcommand (Q)

To quit the modification of the line and restore any changes made to the line you type the quit modify command (Q). When the Q is typed the line is re-displayed as it was before any changes were made and the modification of the line is terminated. If there are any iterations of the MODIFY command remaining then the text pointer advances one line and the next line is placed in the MODIFY mode.

End Modify Subcommand (RET)

To end the modification of the line and save any changes made to the line you type the end modify command (carriage return). When the carriage return is typed the line is re-displayed with all changes saved and the modification of the line is terminated. If there are any iterations of the MODIFY command remaining then the text pointer advances one line and the next line is placed in the MODIFY mode.

APPENDIX A

COMMAND SUMMARY

Command		Page
AGAIN	Repeat last command executed.	21
BOTTOM	Locate end of file.	10
CASE	Set upper, lower, mixed, BASIC, or ASSEMBLE case input.	4
CHANGE	Change text.	14
COLUMN	Display column numbers.	21
COMBINE	Join two lines together.	15
CSI	Execute system command and resume edit session.	23
DELETE	Remove lines of text.	15
DOWN	Advance line pointer.	10
DUP	Duplicate lines of text.	16
ERROR	Action to be taken when macro sub-command fails.	21
FILE	Save text file on disk and return to system.	19
FIND	Locate line starting with specified string.	10
GET	Add text from disk file or temporary work file.	16
HELP	Display commands and syntax.	3
HTAB	Specify tabulation character and action.	5
INPUT	Accept text from keyboard.	17
LINEMODE	Set status of line numbering switch.	5
LIST	Display portion of file on console.	11
LOCATE	Locate specified string in text.	11
MODIFY	Allow character-by-character change of current line.	25

```
         B     Beginning of the line.     Q     Quit modification.
         D     Delete one character.      R     Replace characters.
         E     Advance to end of line.    U     Convert to upper case.
         F     Position to character.     <sp>  Advance one character.
         I     Insert characters.         <DEL> Backup one character.
         L     Convert to lower case.
```

Command		Page
NAME	Display name of text file.	19
NEXT	Advance line pointer.	11
NUMBER	Add line numbers to file.	12
PAGE	Display one screen of text on console.	11
PROMPT	Set line number increment value.	6
PUT	Add text from text file to external file.	17
PUTD	Move text from text file to external file.	18
QUIT	Abort this edit session.	19
RENUMBER	Resequence line numbers of text file.	6
REPLACE	Replace current line with text from keyboard.	18
SAVE	Save text file on disk and resume edit session.	20
SKIP	Branch within macro command.	24
SPLIT	Divide one line into two.	18
TABSET	Set tab stops for input and display.	7
TOP	Locate top of file.	12
TRUNCATE	Set maximum line length of input text.	7
TYPE	Display portion of file on console.	12
UNNUMBER	Remove line numbers from file.	8
UP	Position line pointer backwards in file.	12
VERIFY	Set verify mode.	8
WRAPMODE	Set wrap mode.	8
X,Y,Z	Execute or load macro.	21
ZONE	Set column number that CHANGES and LOCATES can examine.	9
?	Display last command executed.	23
<CR>	Advance line pointer one line.	10
<up-arrow>	Position line pointer up one line.	12
<LF>	Position line pointer down one line.	10

APPENDIX B

GLOBAL COMMAND DEFAULT VALUES

Command	Type ASSEMBLE	Type COPY/MACRO	Type BASIC	Type EXEC	Type FORTRAN	Type COBOL	Other
Case	AL	AL	BL	U	U	CL	M
Linemode	OFF	OFF	ON	OFF	OFF	ON	OFF
Prompt	10	10	10	10	10	10	10
Truncate	255	255	255	255	72	72	255
Verify	ON 255	ON 255	ON 255	ON 255	ON 255	ON 255	ON 255
Wrap	OFF	OFF	OFF	OFF	OFF	OFF	ON
Zone	1	1	6	1	6	1	1
Tabset	10 16 28	10 16 28	every 5	every 5	every 5	every 4	every 5

APPENDIX C

EDIT ERRORS AND MESSAGES

Message Explanation
==

Available Memory Now Full: Information message - occurs during input of text or when EDIT is loading the file into memory. Indicates that the text was accepted and saved in memory but there is not sufficient space available for any more text input.

The operator should delete some text to make space available or divide the file into multiple files (if the purpose of the file will allow multiple files).

BACKUP File is Protected Error message - Indicates that the Editor cannot rename the input file to BACKUP because a BACKUP file already exists and it is delete protected.

Can't save a BACKUP File Error message - An attempt was made to file or save a file with file type BACKUP.

Disk Full Error message - During an attempt to SAVE or FILE the file being Edited the disk became full. When this occurs the input file has already been renamed to file type BACKUP. Use the CSI command to erase some of the unused files on the disk or use the NAME command to designate that a different disk drive is to be used.

DOWN Information message - displayed in response to the operator typing the down arrow in the command mode.

EDIT: Information message - occurs when the Editor re-enters the Edit command mode after a multiple line INPUT or REPLACE has been exited.

End of Memory Reached: Information message - occurs when the Editor detects that there is probably insufficient memory available to add more lines of text.

EOF: Information message - displayed when the Editor encounters the end of file marker.

File Name Missing Error message - occurs when operator has specified a file description with a missing name field.

File Type Missing Error message - occurs when the operator has specified a file description with a missing type field.

fn.ft:fd filed Information message - displayed after the Editor has successfully FILED the text file.

INPUT: Information message - displayed when the Editor enters a multiple line input or replace mode.

Invalid command syntax Error message - occurs during edit command mode. This error message indicates that the operator has typed a valid command name but has used the wrong parameters or has made a typing error.

Invalid filename Error message - occurs when operator has specified a file description. File names must start with an alphabetic character, contain only alphbetic, numeric or the $ character and can be no more than eight characters in length.

Invalid numeric Error message - occurs when the command syntax requires a numeric parameter but the operator has entered a alpha character.

Line number not in range 1-9999 Error message - operator has specified a negative, zero or a line number greater than 9999. Retry operation.

Must be ON or OFF Error message - occurs when user types a global command that requires a parameter of ON or OFF and something else was specified.

Must be U, L, M, A, B, C, Ax, Bx, or Cx Error message - occurs during CASE command and operator has not specified a valid CASE mode.

NEW FILE: Information message - displayed when the EDIT program is first entered and the specified file to be edited is not found.

No prior macro in effect: Error message - occurs when the operator specifies the execution of a macro that has not been loaded with any commands.

No Room: Error message - occurs when the Editor detects that there is insufficient memory available to save the line of text just entered by the operator.

Not Found: Information message - displayed when the Editor can not LOCATE or FIND the sequence of characters specified.

Not Sequential Error message - occurs when an attempt is made to EDIT an indexed or direct file.

Not while LINEMODE is OFF: Error message - the operator has attempted to execute an EDIT command that requires LINEMODE to be ON. These commands include: RENUMBER and nnnn.

Not while LINEMODE is ON: Error message - The operator has attempted to execute an EDIT command that requires LINEMODE to be OFF. These commands include: INPUT immediate, REPLACE immediate, DUPlicate, WRAP ON, and ZONE less than 6.

Out of Sequence: Information message - occurs when an attempt is made to set LINEMODE ON when the file contains no line numbers or it is out of sequence.

Renumber Lines: Information message - occurs when LINEMODE is ON and lines are being added. Indicates that the next line has a line number one greater than the current line and there are no numbers available to assign to new lines.

Required parameter missing: Error message - occurs when the operator has not entered all of the parameters required by the command syntax.

fn.ft:fd saved Information message - displayed after the Editor has successfully SAVED the text file.

Space required following command Error message -

TOF: Information message - displayed when the Editor encounters the top of file marker.

Too many parameters: Error message - occurs during edit command mode.

Truncated: Error message - the operator has INPUTed, CHANGEd, or MODified a line whose new length is greater than the length allowed by the TRUNCATE length. The line has been truncated to the specified length and is displayed following the error message.

Unrecognized command: Error message - occurs during edit command mode. This error is usually caused by the operator typing text when the Editor is expecting a command. Also occurs when a command abbreviation is used and not separated by a space from text that follows it.

UP Information message - displayed in response to the operator typing the up arrow in the command mode.