SUPEREDIT

User's Manual

093–000111–00

Ordering No. 093-000111

Printed in the United States of America
Rev. 00, August 1974

NOTICE

Original Release - August 1974

TABLE OF CONTENTS

CHAPTER 3 - SUPEREDIT COMMANDS

CHAPTER 1

INTRODUCTION

1.1 GENERAL

SUPEREDIT is an extremely versatile and powerful ASCII text editor specifically designed for use on Data General computers running under an RDOS system.

Some of the major features of SUPEREDIT are:

- Multi-buffer editing
- Multiple I/O files
- Macro programming
- Numeric variables

It is assumed that the user is on line with an operating RDOS system. The system is described in the following document:

093-000075	RDOS Real Time Disk Operating System User's Manual

1.2 DOCUMENTATION CONVENTIONS

1.2.1 Underlining

Where clarification is required in the examples used in this manual, underlined copy denotes entries output by SUPEREDIT. Copy not underlined indicates user input.

1.2.2 Control Characters

Control characters such as CTRL-A are typed by holding down the CTRL key and pressing the letter A. Control characters, such as CTRL-A are represented in this manual and echoed by SUPEREDIT as ↑A.

1.2.3 Symbols

The symbols listed in Table 1-1 are used throughout this manual for clarity.

Table 1-1. Special Symbols

Symbol	Character Represented	Explanation
)	Carriage Return	Pressing the RETURN key generates an automatic line feed in addition to the carriage return.
$	ESCape or ALTmode	The dollar sign ($) is echoed on the terminal as a result of pressing the ESC or ALT key on your terminal.
Δ	Space	Sometimes used in this manual to emphasize a space character.
↑	Character Pointer (CP)	Used to represent the position of the Character Pointer. Also used in control character (e.g., ↑A) representation.
→|	Tab	Represents use of the TAB or ↑I character on the terminal.
[ ]	Broken Brackets	Enclose optional arguments to a command.
{ }	Braces	Indicate a choice of items enclosed.

1.2.4 Terminals

The use of the word "terminal" throughout this manual implies a teletypewriter, CRT, or an equivalent interactive device.

1.3 RUNNING SUPEREDIT

SUPEREDIT is initialized for ECLIPSE™* line computers by typing

```
SPEED)

 or

SPEED filename)

!                              (the exclamation mark is
                                the SUPEREDIT prompt.)
```

If "filename" exists, a UYfilename$ command is executed (see paragraph 3.1.3). If "filename" does not exist, a GWfilename$ command is executed (see paragraph 3.1.1) and the message CREATING NEW FILE is typed out.

For NOVA®* line computers type

```
NSPEED)

 or

NSPEED filename)

!
```

*ECLIPSE is a trademark, and NOVA is a registered trademark of Data General Corporation, Southboro, Massachusetts.

CHAPTER 2

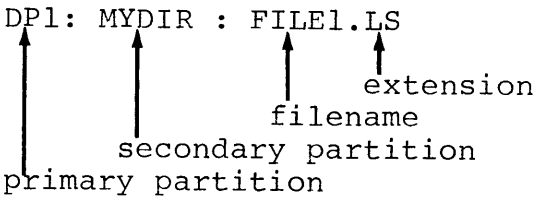
CONCEPTS

2.1 DATA FILES

The ASCII user input files referred to in this manual are those which have been created on an RDOS system. As such the generalization "filename" used to indicate a user file is more specifically defined as follows:

```
{primary partition:} {secondary partition:} {sub-directory:}...)
 filename   {.extension}
```

For example:



2.2 DATA FORMAT

An ASCII input file contains a string of ASCII characters. Characters form lines which are terminated by a carriage return (). A string of characters up to, but not including, a form feed character is defined as a page. Files may be divided into pages or windows containing a fixed number of lines.

2.3 EDIT BUFFERS

SUPEREDIT provides 36 Edit Buffers which may be used for editing user files. Commands are available for transferring data from input files to any buffer, transferring data from one buffer to another and for outputting from buffers to output files. Buffers may contain textual data, commands to be executed, or both.

2.3 EDIT BUFFERS (Continued)

Buffers have single character names (A to Z and 0 to 9), and only one buffer is the current buffer at any given time. Initially, buffer 0 is the current edit buffer. Any edit buffer referenced during an editing session is considered to be active.

A detailed explanation of buffer commands is provided in section 3.11.

2.4 THE EDITING PROCESS

The editing process consists of the following steps:

1. Read text from an input file to an edit buffer or type in text for a new file.

2. Modify the text in the edit buffer.

3. Write the contents of the edit buffer to an output file.

Step 3, above, is extremely important. The output commands described in section 3.9 must be used to insure preservation of edited material.

2.5 CHARACTER SET

The character set used by SUPEREDIT is the full ASCII character set in Appendix A. Both the octal and decimal values of the characters are given and should be referenced when using the nI variation of the Insert (I) command discussed in section 3.7.

2.6 CHARACTER POINTER (CP)

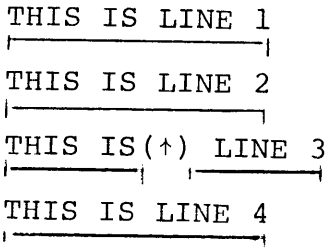
The Character Pointer (CP) is the mechanism used by SUPEREDIT to keep track of the current position within an edit buffer. The CP may be located at the beginning of an edit buffer, between two characters in the buffer, or at the end of the buffer. The symbol "↑" is used in some examples to show the

2.6 CHARACTER POINTER (CP) (Continued)

position of the CP. Character Pointer Commands described in Chapter 3 are used to reposition the CP to any location within an edit buffer. Most modification commands will alter the edit buffer at the CP position. An insert (I) command for example, will insert a text string at the position of the CP. The effect of a command on the CP is described with the respective command.

2.7 INTERPRETATION OF LINE-ORIENTED COMMAND ARGUMENTS

SUPEREDIT considers a line as a string of characters ending with a carriage return. For those commands which are line-oriented (e.g., K and L) SUPEREDIT treats the numeric argument of these commands in the following manner.



In the above illustration the CP is located in the middle of line 3. The command -2K kills lines 1, 2 and the characters up to the CP in line 3. The command 2K kills the characters in line 3 starting from the CP plus the entire line 4. A 0K command kills the characters from the beginning of line 3 up to the CP. A 1K command kills the characters starting from the CP up to the end of line 3. A -1K command kills line 2 and the characters up to the CP in line 3.

2.8 COMMAND STRUCTURE

The commands used in SUPEREDIT are of the general form:

$$\left[\begin{Bmatrix} n \\ m,n \end{Bmatrix}\right] \quad \{\text{modifiers}\} \quad \text{code} \quad \{\text{string}\}\$$

or

$$\{\text{modifiers}\} \left[\begin{Bmatrix} n \\ m,n \end{Bmatrix}\right] \quad \text{code} \quad \{\text{string}\}\$$

where: n is an optional numeric argument which may precede certain commands.
m,n is an optional double argument allowable on certain commands.
modifiers may be used to alter the normal operation of the command.
code is the character or characters which represent a SUPEREDIT command. Code may be either upper case or lower case characters.
string is a character string argument required by certain commands.
$ is an ESCape character which terminates a single command.

A command string consists of one or more concatenated commands. The last command in the string is always terminated by a double ESC ($$) which signals SUPEREDIT to begin processing the command.

$$!\left[\begin{Bmatrix} n \\ m,n \end{Bmatrix}\right] \text{ code } [\text{string}]\$\left[\begin{Bmatrix} n \\ m,n \end{Bmatrix}\right] \text{ code } [\text{string}]\$...\$\$$$

Example:

Assume the current line is

SOMEONE↑ΔNEEDED

the command string

!-3D$IΔHELP$T$$

would delete ONE, insert ΔHELP and type the line including the current position of the CP shown as "(↑)".

SOME HELP(↑) NEEDED

Commands which never require string arguments following the command code (e.g., L,T,D) do not require an ESC ($) to delimit the command within a command string.

Example:

!3DL2TITEXT$T$$ is equivalent to

!3DL2T$ITEXT$T$$

2.9 COMMAND ARGUMENTS AND MODIFIERS

There are two categories of command arguments: Alphanumeric arguments which follow commands and numeric arguments which always precede commands.

2.9.1 Alphanumeric Arguments

Alphanumeric arguments are the character string arguments which follow such commands as Insert (I) or Search (S).

For example:

!IJUNE$S1974$$	Both "JUNE" and "1974" are alphanumeric arguments to the commands I and S respectively.

An ESC ($) must always be used to terminate an alphanumeric argument and thereby delimit the argument from the next command in the command string.

2.9.2 Numeric Arguments

Numeric arguments which precede commands are always evaluated to a decimal integer value before the command is executed. A numeric argument may be an expression, may contain any digit, the numeric operators described in Table 2-1, numeric variables as described in section 3.14 and special characters that represent numeric values as described in paragraph 2.9.4.

Table 2-1. Numeric Operators

Operator	Function
+	Addition
-	Subtraction
*	Multiplication
/	Division

2.9.2 Numeric Arguments (Continued)

Evaluation of numeric arguments which contain numeric operators is performed from left to right.

Single numeric arguments are represented in this manual by n and double arguments by m,n. The double argument must always be separated by a comma and m must always be less than or equal to n. Generally, if the numeric argument (n) is omitted from a command, n is assumed to be 0.

Example:

!3+1,60T$$	Type out the 5th through 60th characters in the buffer.

Numeric arguments may be positive or negative and are interpreted as a value in the range of -32768 to +32767 with the following exceptions:

a) Double argument commands (m,n)
b) \
c) J
d) =

which are interpreted as being in the range 0 to +65535.

2.9.3 Numeric Variables

There are 10 numeric integer variables (named 0 to 9). Each of the 10 variables is initially set to 0. The variables can be individually set to a value, incremented or decremented. They can be used as numeric arguments to commands, alone, or in argument expressions. Numeric variable commands are described in section 3.14.

2.9.4 Special Character Numeric Arguments

These special characters stand for specific values associated with the Character Pointer (CP), the size of the text buffer or lines in the current buffer. The special characters are described in Table 2-2 and can be used as numeric arguments. Alphabetic characters can be either upper case or lower case.

Table 2-2. Special Character Numeric Arguments

Operator	Meaning
Z	Represents the CP position immediately following the last character in the buffer. This is equivalent to the total number of characters in the buffer.
.(period)	Represents current CP position. This is equivalent to the number of characters between the beginning of the buffer and the character to the left of the CP.
#	A special double argument which is equivalent to 0,Z (the entire buffer).
VL	Represents the number of the line which contains the CP.
VN	Represents the number of lines in the current buffer.

Example:

!6VS0$.,.+VI0T$$	Type out the 7 characters following the CP. This command uses numeric variable commands described in section 3.14.

2.9.5 Command Modifiers

Several command modifiers (@ and :) are used with various commands to extend the versatility of the commands by altering their normal interpretation. The command modifiers must precede the command code in the command string. The command modifiers are described in detail in paragraph 3.16.2.

2.10 MODES OF OPERATION

2.10.1 Input Mode

When SUPEREDIT outputs a prompt (!) to the terminal, SUPER-EDIT is ready to accept a command string which must be terminated by a double ESC ($$). Command strings may be typed on more than one line.

2.10.2 Execution Mode

When SUPEREDIT recognizes a double ESC ($$) it outputs a carriage return and begins execution of the commands in the command string. When SUPEREDIT finishes execution of the commands in a command string it outputs another prompt (!) indicating that the user may enter additional commands.

2.11 DATA INPUT MODES

SUPEREDIT has the facility to input file data by pages (form feed-to-form feed) or when in Window Mode by a set number of input lines. The commands which control Page and Window Mode operation are described in section 3.2.

2.12 CORRECTING COMMAND STRINGS

2.12.1 Erasing Characters and Lines

The RUBOUT key is used to delete erroneously typed characters from a SUPEREDIT command string. Each time the RUBOUT key is pressed, the right most character is deleted. When a character is deleted it is echoed on the terminal.

2.12.1 Erasing Characters and Lines (Continued)

For example:

!IHELLOOLLEHBYE$$	The user pressed RUBOUT once for each character in the word HELLO.

A single line may be deleted by typeing CTRL-X (↑X) and is equivalent to typing RUBOUT back to the beginning of the line. The line may not be recovered. A ↑X at the beginning of a line deletes the previous line.

Example:

!HELLO ↑X !IBYE$$	is equivalent to the previous example.

2.12.2 Aborting Command Strings

In Input Mode, an entire command string (on one or more lines) may be aborted by typing CTRL-A (↑A). SUPEREDIT will return the user to the prompt condition. If ↑A is inadvertently typed, the command string may be recovered by a ←n command (see paragraph 3.16.1) since SUPEREDIT always saves the last command string.

Example:

!ITEXT$ IMORETEXT$ ILOTSMORE TEXT ↑A !	↑A eliminates all three lines of the command string.

In execution mode, typing a ↑A will terminate execution of the command string and return SUPEREDIT to the Input Mode.

2.12.3 Verification Of Command Lines

A CTRL-R (↑R) can be used to immediately retype the last line of a command string for verification. This is useful for those lines which have been corrected by excessive use of RUBOUT and are difficult to read.

CHAPTER 3

SUPEREDIT COMMANDS

The commands described in this chapter provide the facility for editing, splitting, rearranging and merging files. Sections 3.1 through 3.10 describe the fundamental commands required for most editing sessions. Section 3.11 through 3.17 describe commands for the advanced user.

Occassionally the user may precipitate an error message due to incorrect command usage. Refer to section 4.2 for a complete explanation of error handling.

3.1 FILE SPECIFICATION COMMANDS

Files are considered to be either global or local. A global file is applicable to all 36 edit buffers while a local file is only active for one particular edit buffer. Local files take precedence over global files. Therefore, any command which accesses a file (e.g., File Input, N and Q Search, Output) will apply to an open local file if one exists. Otherwise it will apply to an open global file.

Note: "Filenames" shown in command formats may include extensions.

3.1.1 The GW Command

The GW (Get for Writing) command is used specifically for creating a new file. The general form of the command is:

!GWfilename$$

The "filename" specified in the command must not already exist. A sequential file named "filename" is opened for output. Only one global output file may be opened at one time. If another file is already opened for output it must be closed before a new GW command is specified and executed.

3.1.1 The GW Command (Continued)

Text which is inserted into edit buffers is subsequently copied to the "filename" specified in the GW command by appropriate output and file closing commands.

Another form of the GW command is:

```
!:GWfilename$$
```

This command is essentially the same as the GWfilename$$ command except that a random file is created and opened for output rather than a sequential file.

3.1.2 The GC Command

This command is of the form:

```
!GC$$
```

The command closes the current output file as is. It does not provide for writing from any edit buffer or input file before closing the output file.

3.1.3 The UY Command

This command opens an existing file for input, yanks a page from the input file into an edit buffer and creates a new file for output with the extension .SC.

The command is of the form:

```
!UYfilename$$
```

Example:

```
!UYTEST.EX$$          Open TEST.EX for input
                      Open TEST.SC for output
                      Y (Yank) a page
```

The "filename" specified for input in the UY command may not have a .SC extension. If the input "filename" is sequentially or contiguously organized then the new output file will be sequentially organized. If the input "filename" is randomly organized then the new output file will be randomly organized. Only one global input file may be opened at one time.

3.1.4 The GR Command

This command closes the currently open input file and opens for input the "filename" specified in the command.

The form of this command is:

!GRfilename$$

After the specified file is opened, a Yank (Y) or Append (A) command (see section 3.3) must be used to insert a page into the edit buffer. This does not occur automatically as in the UY command.

Another form of the GR command is:

!GR$$

This command closes the currently opened input file without opening another file.

3.1.5 The US Command

The US command is used to copy the content of the current buffer and the remainder of the input file to the output file (an E command) and then close both the input and output files. The current buffer is left cleared. The form of this command is:

!US$$

If the input and output files were opened by a UY command then the input file is renamed to filename.BU and the output file is renamed from filename.SC to the orginal filename of the input file. The result is a backup file of the original input file and a new edited file with the original filename.

3.1.5 The US Command (Continued)

Example:

```
!UYSAMPLE.JK$$          Open input file SAMPLE.JK
                        Open output file SAMPLE.SC
    .
    .                   Edit commands
    .
!US$$                   Copy current buffer and rest
                        of SAMPLE.JK to SAMPLE.SC
                        Close SAMPLE.JK
                        Close SAMPLE.SC
                        Rename SAMPLE.JK to SAMPLE.BU
                        Rename SAMPLE.SC to SAMPLE.JK
```

3.1.6 The UE Command

The UE command is used to copy the content of the current buffer and the remainder of the input file to the output file (an E command) and close the input and output files.

If the input and output files were opened by a UY command then the input file is deleted, and the output file is renamed to the name of the original input file. The current buffer is left cleared.

The form of this command is:

```
!UE$$
```

Example:

```
!UYSAMPLE.JK            Open input file SAMPLE.JK
                        Open output file SAMPLE.SC
    .
    .                   Edit Commands
    .
!UE$$                   Copy current buffer and rest
                        of SAMPLE.JK to SAMPLE.SC
                        Close SAMPLE.JK
                        Close SAMPLE.SC
                        Delete SAMPLE.JK
                        Rename SAMPLE.SC to SAMPLE.JK
```

3.1.7 THE U? Command

The U? command is the file status command. It provides a list of open global and local input and output files. The local files are those relevant to the current edit buffer. The form of the command is:

```
!U?$$
```

Example:

```
!UYSAMPLE.JK$U?$$

GLOBAL:
        INPUT FILE - SAMPLE.JK
        OUTPUT FILE - SAMPLE.SC
LOCAL:
        INPUT FILE - NONE
        OUTPUT FILE - NONE
```

3.2 DATA INPUT MODE COMMANDS

Data is read from an input file to the current edit buffer in either page lengths (form feed-to-form feed) or in window lengths (fixed number of lines). Initially SUPEREDIT is in Page Mode and any File Input command (e.g., Y) causes page lengths to be input. The commands described in Table 3-1 permit switching from one data input mode to the other.

Table 3-1. Page and Window Mode Commands

Command	Description
WM	Returns value of data input mode. If WM=0, SUPEREDIT is in Page Mode. If WM=n (where n > 0) SUPEREDIT is in Window Mode and n is number of lines per window.
nWM	Changes input from Page Mode to Window Mode with a window length of n lines. SUPEREDIT is initialized in Page Mode.
0WM	Changes input from Window Mode to Page Mode.

3.2 DATA INPUT MODE COMMANDS (Continued)

Data read from an input file in Window Mode will include any form feed characters embedded in the file exactly as they are encountered. In Page Mode SUPEREDIT remembers when the last character read by an input command was a form feed but does not place the form feed character into the edit buffer. The detection of the form feed character on input is pertinent to the R, nR and E output commands discussed in section 3.10.

3.3 FILE INPUT COMMANDS

The commands described in Table 3-2 read data from an input file into the current edit buffer. Data is read in either as pages or window lengths depending on the current data input mode (Page or Window).

Table 3-2. File Input Commands

Command	Description
Y	The Yank command. The current edit buffer is cleared, then a page or window length of the input file is read into the buffer. The CP is positioned at the beginning of the buffer.
A	Appends a page or window length from the input file to the current edit buffer. The CP position is unchanged.
:Y and :A	Essentially same as Y and A, respectively, except returns a command value of +1 if Yank or Append was successful and 0 if the command failed. No error message is output. Both :Y and :A may be used as numeric arguments to the next command. For example: !:Y*10T$$ If Yank is successful type 10 lines.

3.4 TEXT TYPE-OUT COMMANDS

Text Type-Out commands (described in Table 3-3) may be used to examine part or all of the current buffer.

When used with a single numeric argument, the T command is line-oriented. When used with a pair of numeric arguments (m,n), the T command is character oriented.

NOTE

The Text Type-Out commands do not move the CP.

Table 3-3. Text Type-Out Commands

Command	Description
T	Types the content of the entire current line (line containing the CP) and includes the character "(↑)" at the current location of the CP.
0T	Types current line from beginning to location of the CP.
nT	Types the content of the current buffer from the location of the CP through the next n carriage returns.
-nT	Types the content of the n lines preceding the current line plus the content of the current line up to the CP.
m,nT	Types the content of the current buffer from the m+1th character up to and including the nth character.
:0T :nT :-nT :m,nT	Forms of T commands modified by colon (:) print output on line printer rather than on terminal.

3.5 CHARACTER POINTER (CP) COMMANDS

The Character Pointer (CP) commands described in Table 3-4 may be used to move the CP between any two characters in the current edit buffer. The CP may be moved forward or backward, by characters or by lines. The CP is the means of specifying within an edit buffer the position at which insertions, deletions or corrections are to be made.

Table 3-4. Character Pointer (CP) Commands

Command	Description
nJ	Move the CP to the beginning of line n in the current buffer. n is always relative to the beginning of the current buffer. Line 1 is the first line of the buffer.
J, 0J or 1J	Move the CP to the beginning of the first line in the buffer.
nL	Move the CP forward across n carriage returns and position the CP at the beginning of the line following the nth carriage return.
-nL	Move the CP backward to the beginning of the nth line preceding the current line.
L or 0L	Move the CP to the beginning of the current line.
nM	This command moves the CP n characters to the right or left. If $n > 0$ the CP is moved n characters to the right. If $n < 0$ the CP is moved n characters to the left.

3.6 SEARCH COMMANDS AND CONTROL CHARACTERS IN SEARCHES

3.6.1 Search Commands

Search commands (see Table 3-5) are used to reposition the CP by means of a character string search. The Search commands cause SUPEREDIT to scan through the text until a specified string of characters is found, and then positions the CP after the last character in the string unless a ↑P is inserted into the search string. If the end of the edit buffer is reached during a S or C command and the search is not successful, an error message is printed on the terminal and the CP is repositioned at the beginning of the buffer. For N and Q searches an unsuccessful search will result in the entire file being read and the buffer being left empty. Text strings being searched for may not overlap page or window boundaries.

Table 3-5. Search Commands

Command	Description
Stext$	Searches the current buffer from position of CP for string "text". If string is found, CP is positioned after the last character in the string. If not found, CP is positioned before the first character in the buffer and an error message is printed.
nStext$	Similar to the S command except search is from CP through next n carriage returns. The CP is left at the beginning of the line after the last line searched if search is unsuccessful.
-nStext$	Similar to S command except search starts from beginning of nth line preceding the current line and ends at the current position of the CP. If the search fails, the CP is located at its original positions and an error message is printed.

Table 3-5. Search Commands (Continued)

Command	Description
0Stext$	Searches for string "text" from the beginning of the current line to the position of the CP. If the string is found, CP is positioned after the last character in the string. If not found, the CP is located at its original position and an error message is printed.
m,nStext$	Searches the m+1th through nth characters of the buffer inclusive for string "text". If string is found, CP is positioned after the last character in string. If not found, CP is positioned after the nth character and an error message is printed.
Ctext1$text2$	The C (Change) command is a combination search and modify command. A search is conducted from the CP position for "text1". If found, "text1" is deleted and "text2" is inserted in its place. The CP is positioned after the last character of "text2". If "text1" is not found in current buffer, an error message is printed and the CP is positioned at the beginning of the buffer.
nCtext1$text2$	Same as Ctext1$text2$ except search is from CP through next n carriage returns. The CP is positioned at the beginning of the line after the last line searched if search is unsuccessful.
-nCtext1$text2$	Similar to the Ctext1$text2$ command except the search starts from the beginning of the nth line preceding the current line and ends at the position of the CP. If the search fails, the CP is located at its original postion and an error message is printed.

Table 3-5. Search Commands (Continued)

Command	Description
0Ctext1$text2$	Searches for "text1" from beginning of current line to the position of the CP. If found, "text1" is deleted and "text2" is inserted it its place. The CP is positioned after the last character of "text2". If "text1" is not found, the CP is located at its original position and an error message is printed.
m,nCtext1$text2$	A search is conducted from the m+1th through nth characters of the buffer for "text1". If found, "text1" is deleted and "text2" is inserted in its place. The CP is positioned after the last character of "text2". If "text1" is not found an error message is printed and the CP is positioned after the nth character.
Ntext$	Performs same function as Stext$ command except search continues across page boundaries if necessary until the end of the input file is reached. The search begins at the CP position in the current buffer. If the end of the buffer is reached an R command is executed and the search continues until the string is located or an end-of-file is reached. If the search is not successful, an error message is printed.
Qtext$	Similar to an N command except a Y command is executed rather than an R command, if the string is not found in the current buffer. No output is done to the output file.

3.6.1 Search Commands (Continued)

Table 3-5. Search Commands (Continued)

Command	Description
@S%text%	Same as Stext$ except the text is delimited by the first character after S (arbitrarily shown as %). The @ modifier changes the command terminator from ESC($) to any other character (e.g., %) and may be used with any S, N, C or Q command.
:Stext$	Same as Stext$ except the command returns a value of +1 if it succeeds and 0 if it fails. No error message is output if search is unsuccessful. The command can be used as a numeric argument to the next command. The colon (:) modifier may be used with any S, N, C or Q command. The command modifiers may also be combined. For example, the following command strings are legal. @:S%text% :@C%text1%text2%

3.6.2 Control Characters In Searches

SUPEREDIT executes a search command by attempting to match the search command argument character-for-character with some portion of the current buffer. Several control characters are available for altering the normal search process and are described in Table 3-6.

Table 3-6. Control Characters In Searches

Command	Description
↑Z	When the ↑Z character is included in a search command string it is an indication that the position occupied by ↑Z is unimportant and any character is an acceptable match. For example: ABCDE matches AB↑ZDE
↑Nx	Any character in the position occupied by ↑Nx is acceptable except for the character x. For example: ABCDE matches ABC↑NEE ABCDE does not match ABC↑NDE
↑Vx	A ↑Vx in a search string indicates that any number of the character x (including 0) is an acceptable match. For example: ABCE and ABCDE and ABCDDDE are acceptable matches to ABC↑VDE
↑←	This control character causes the character following ↑← to be interpreted literally rather than as a special character. This control character is useful when conducting a search for other control characters (e.g., ↑Z, ↑N, etc.)

Table 3-6. Control Characters In Searches (Continued)

Command	Description
↑P	↑P in a search string is a character pointer (CP) positioning character. If the search is successful, the CP will be positioned where the ↑P was in the search string, instead of at the end of the matching string. For example: !Stext1↑PΔtext2$T$$ text1(↑)Δtext2!
↑T	This control character matches any number of spaces, or TAB (including 0). For example: ABCΔΔΔDE and ABCΔDE and ABC→|DE and ABCDE are all acceptable matches for ABC↑TDE

NOTE

Any other control character (except TAB (↑I), CR (↑M), FF (↑L), VT (↑K), ↑G, LF (↑J), ↑B) in a search string is flagged as an error unless preceded by a ↑P control character.

3.7 INSERTION COMMANDS

All insertion commands described in Table 3-7 cause a string of characters, specified in the command, to be inserted into the current buffer at the position of the CP. The CP is then repositioned to follow the last character of the insertion.

Table 3-7. Insertion Commands

Command	Description
Itext$	The string specified by "text" is inserted into the current buffer at the position of the CP and the CP is repositioned to follow the last character of the insertion.
↑Itext$	This command is similar to I command except that a TAB (→|) is inserted into the edit buffer before the text string. The command is equivalent to I→|text$.
nI	A special form of the I command, it allows insertion of a single character, where n is the decimal equivalent of any ASCII character given in Appendix A. The command is particularly useful for inserting characters which may not be available on the input terminal. It is the only way to insert the characters ↑B(2I) and ↑G(7I).
n\	Inserts the ASCII representation of the decimal number n into the buffer at the CP location. Leading zeroes are suppressed.
@I%text% @↑I%text%	Same as Itext$ and ↑Itext$ except that the text is delimited by the first character after I (arbitrarily shown as %). The @ modifier changes the command terminator from ESC ($) to any other character (e.g., %).

3.8 DELETION COMMANDS

The commands described in Table 3-8 are used to delete characters and lines from an edit buffer.

Table 3-8. Deletion Commands

Command	Description
nK	Deletes characters in the current buffer from the CP up to and including the next n carriage returns.
-nK	Deletes the n lines preceding the current line plus the characters up to the CP in the current line.
K or 0K	Deletes the characters from the CP back to the beginning of the line containing the CP.
m,nK	Deletes the m+1th through nth characters inclusive from the current buffer. The CP is positioned after the mth character.
nD	Deletes characters from the current buffer. If n is positive, deletes the n characters following the CP. If n is negative, deletes the n characters preceding the CP.

3.9 OUTPUT COMMANDS

The commands listed in Table 3-9 permit the user to output data from edit buffers and input files to output files.

NOTE

The P and PW commands do not clear the buffer after output unless modified by the colon (:) modifier.

Table 3-9. Output Commands

Commands	Description
P	Outputs the entire contents of the current buffer to the output file with an appended form feed character. The CP does not move.
nP	Outputs n lines from the position of the CP in current buffer to the output file with an appended form feed character. The CP does not move.
-nP	Outputs the preceding n lines plus the characters up to the location of the CP in the current line to the output file with an appended form feed character. The CP does not move.
0P	Outputs the characters on the current line, from the beginning of the line to the location of the CP, to the output file with an appended form feed character. The CP does not move.
m,nP	Outputs the m+1th to nth characters inclusive from the current buffer to the output file with an appended form feed character. The CP does not move.
PW	Outputs the entire contents of the current buffer to the output file without appending a form feed. The CP does not move.
nPW	Outputs n lines from the position of CP in the current buffer to the output file without appending a form feed. The CP does not move.
-nPW	Outputs the preceding n lines plus the characters up to the location of the CP in the current line to the output file without appending a form feed. The CP does not move.

Table 3-9. Output Commands (Continued)

Command	Description
0PW	Outputs the characters on the current line, from the beginning of the line to the location of the CP, to the output file without appending a form feed character. The CP does not move.
m,nPW	Outputs the m+1th to nth characters inclusive from the current buffer to the output file without appending a form feed. The CP does not move.
:P	Modified form of P command clears buffer after output. Equivalent to P#K. The colon (:) modifier may be used with all P command variations.
R	Outputs the content of the current buffer, appends a form feed only if detected as last character read on the previous input (A or Y) to the buffer, clears the buffer and yanks another page or window length into the buffer.
nR	Performs an R command n times.
:R and :nR	Modified forms of R and nR commands return a value of +1 if input is successful and 0 if it fails. No error message is output. The commands may be used as numeric arguments to the next command.
E	Outputs the contents of the current buffer and the remainder of the input file to the output file. Any form feeds detected on input are copied to the output file. The current buffer is left cleared.

3.10 EXIT COMMANDS

The commands described in Table 3-10 are used to exit from SUPEREDIT.

Table 3-10. Exit Commands

Command	Description
H	Exit from SUPEREDIT and return to the CLI. SUPEREDIT closes all files before returning to CLI. This command is used to make an orderly exit from SUPEREDIT at the end of an editing session.
↑C	A control character which interrupts SUPER-EDIT, creates a BREAK.SV file and returns the user to the CLI level. Should only be used as an emergency exit from SUPEREDIT.

3.11 BUFFER COMMANDS

Buffer commands provide the flexibility for manipulating blocks of data which can be stored in any of the 36 edit buffers (0 to 9 and A to Z) available in SUPEREDIT. One edit buffer is current at any time. Commands are available to open input and output files which apply only to a given edit buffer. These files are called local files. Files which do not specifically apply to a given buffer are called global files. Any commands which act on files (e.g., Y, P, etc.) will apply first to open local files. Therefore a local file must be closed before a global file can be used for a particular command.

Buffer commands are described in Table 3-11.

Table 3-11. Buffer Commands

Commands	Description
BSx	This command changes the current edit buffer to edit buffer x. x is a single character name for any one of the 36 edit buffers (A to Z and 0 to 9). Only one edit buffer may be current at a time. The status of the CP position and of any open local files is saved when the current edit buffer is changed, and is restored for the new edit buffer. Commands such as Yank (Y) and Append (A) input data from an input file to the current edit buffer.
B?$	The buffer status command provides a list of all active buffers and their length (characters) and indicates which edit buffer is the current buffer with a right angle bracket (>). For example: !B?$$ BUFFER I - 36 > BUFFER A - 18749 BUFFER G - 1902 Buffer A is the current buffer.
B?x	Types out status of buffer x.
BCx	Copies the entire contents of the current buffer to buffer x after clearing buffer x. The content and CP position of the current buffer remain unchanged. Buffer x CP is positioned at the beginning of buffer x.

Table 3-11. Buffer Commands (Continued)

Command	Description
nBCx	Copies the next n lines starting from the present position of the CP in the current buffer to buffer x after clearing buffer x. For example: !BS3$5BCA$$ — Copies 5 lines from buffer 3 to buffer A after clearing buffer A. The content and CP position of buffer 3 remain unchanged. Buffer A CP is positioned at the beginning of buffer A.
-nBCx	Copy the n lines preceding the current line plus the characters on the current line up to the CP to buffer x after clearing buffer x. The content and CP position of the current buffer remain unchanged. Buffer x CP is positioned at the beginning of buffer x.
m,nBCx	Copies the m+1th through nth characters inclusive from the current buffer to buffer x after clearing buffer x. The content and CP position of the current buffer remain unchanged. Buffer x CP is positioned at the beginning of buffer x.
BTx nBTx -nBTx m,nBTx	Same as BCx, nBCx, -nBCx and m,nBCx respectively except characters moved from current buffer are deleted (buffer transfer).
BKx	Inactivate buffer x. Buffer x may not be the current buffer.

Table 3-11. Buffer Commands (Continued)

Command	Description
BGx {filename} $	Where Gx is GR, GW or GC. The filename specified is opened or closed local to the current edit buffer. Only one local file may be open for input and one local file may be open for output for each active buffer. The U? command provides a status report of global and local input/output files for the current buffer.
BUx {filename}	Where Ux is UY, UE or US. The local input and output files are opened or closed in accordance with the respective UY, UE or US command.
BAx	Activate buffer x. All buffers except for the current buffer are stored on disk unless explicitly deleted by a BKx command. In the event of an error, it is possible to recover all the active buffers in a subsequent invocation of SUPEREDIT by use of the BAx command for each buffer to be activated. The buffer status (i.e., content and CP position) is retained. However local and global file status will be lost.

3.12 COMMAND STRING INSERTION COMMANDS

The commands described in Table 3-12 are used to insert the contents of a file or edit buffer into the command string.

Table 3-12. Command String Insertion Commands

Command	Description
↑Gfilename$	The inclusion of this command in a command string will cause the contents of the file named "filename" to be inserted in the command string in place of the ↑Gfilename$ command. The file may contain text to be inserted, commands to be executed, or both. The file must be less than 16K bytes long.
↑Bx	This command in a command string will cause the contents of buffer x to be inserted in the command string in place of the ↑Bx command. Buffer x may contain text to be inserted, commands to be executed, or both. Buffer x cannot be the current buffer.

The ↑Bx and ↑Gfilename$ commands may be nested up to 10 levels deep. When the ↑Bx or ↑Gfilename$ commands are used as arguments to Search (section 3.6) or Insertion commands (section 3.7) the characters up to the first ESC ($) in the buffer or file are treated literally as text characters rather than as commands.

3.13 ITERATION COMMANDS

3.13.1 Construction

A command string may be executed any number of times by placing the command string between angle brackets and specifying the number of iterations with a numeric argument preceding the brackets. A command string within angle brackets is called a command loop. The general form of a command loop is:

3.13.1 Construction (Continued)

n <command string>

If $n \leq 0$, the commands within the brackets are skipped. If $n > 0$, the commands are repeated n times. If n is omitted, the commands are repeated indefinitely.

For example:

!:SDATEΔ$ <IJUNE$> $$	Searches for DATEΔ and inserts JUNE if found; otherwise CP is repositioned to beginning of buffer. No error message if string is not found. If DATEΔ is found, value of :SDATEΔ is +1 and loop is iterated one time. If search fails :SDATEΔ is 0 and loop is skipped.

Nesting of command loops is permitted up to 10 levels deep.

Search commands and File Input commands within command loops are treated as though they were modified by the colon (:) modifier (paragraph 3.16.2). That is, rather than type out an error message if the command fails, the command returns a value of +1 if successful, 0 if it fails. Even if the search fails, command execution within the command loop continues. Therefore Search commands within command loops should be written so as to provide a numeric argument to the next command or be used in conjunction with the semicolon (;) command described in the following paragraph.

3.13.2 Semicolon Command

A semicolon (;) command can be used to prematurely terminate a command loop. The command transfers control out of the current command loop to the command immediately following the command loop if the last preceding Search command (S,C,N or Q) failed. Otherwise command execution within the command loop is continued. The semicolon (;) applies to the last previous search even though other commands may appear in the command string between the search command and the semicolon.

For example:

!<CJUNE$JULY$;>T$$	This command string changes all occurrences of JUNE to JULY in the current buffer and types out the first line in the buffer when through.
!Y<<S↑P;$;1KI) $>R;>$$	This command string removes the comment field from all pages of a source code file.

Semicolon (;) commands may only be used in command loops and are otherwise flagged as an error.

The semicolon command can also be of the form:

n;

where: n is any numeric argument
If n≤0 then n; terminates the command loop.
If n>0 the n; is disregarded.

If a Search command in a command loop is not used as a numeric argument to another command or followed by a semicolon command, the n; command may be used as an alternate means of testing for an exit from the command loop. Good SUPEREDIT programming practice would dictate that one of these techniques be used for determining the results of a Search command in a command loop. Otherwise, command interpretation may be incorrect.

A colon (:) command modifier can be used with the semicolon (;) and n; commands. The form is:

:; or n:;

and reverses the action of the semicolon (;). That is, control is transferred out of the command loop if the last search was successful or n>0.

3.14 NUMERIC VARIABLE COMMANDS

The commands which alter the ten numeric integer variables (0 to 9) are described in Table 3-13. Each command returns a command value. Therefore, the commands may be used alone, in numeric expressions, or as numeric arguments to other commands.

Table 3-13. Numeric Variable Commands

Command	Description
Vv	Represents the current value of variable v.
VIv	Increments variable v and represents the incremented value.
VDv	Decrements variable v and represents the decremented value.
nVSv	Sets variable v to value n and returns that value.

Example:

!5VS1$V1+1T$$	Set variable 1 to 5 and type out 6 lines.

3.15 FLOW CONTROL COMMANDS

SUPEREDIT commands may be combined in a manner to provide unconditional branching to predefined labels and to provide branching based on the evaluation (true or false) of a number of branching conditions. These features permit the user to write SUPEREDIT "programs" to solve complex editing problems.

3.15.1 Command String Labels

The facility used for naming locations within a command string is the command string label which has the form:

```
!label!
```

Label is a string delimited by a pair of exclamation points and may be placed anywhere in a command string except in text string arguments.

Labels are ignored unless specifically referenced by an unconditional branch command (paragraph 3.15.2). Therefore, they may also be used as comments throughout a SUPEREDIT program.

3.15.2 The O Command

The unconditional branch command is the O command and is of the form:

```
!Olabel$
```

The command causes an unconditional branch to be performed to !label! in the command string and continues processing the command string from the first command immediately following !label!. The label must contain fewer than 48 characters and may not be within a command loop.

However, an unconditional branch command may be used to exit from a command loop.

3.15.3 Conditional Branching

The general form of a conditional branch command is:

```
n"xstring'
```

where: n is a numeric argument
x is one of the conditions (defined in Table 3-14) which is checked against the argument n, and
string is a command string.

If the condition is true, the commands until the next single quote (') are executed. If the condition is false, the commands before the next single quote (') are skipped and execution begins with the command which follows the single quote (').

Table 3-14. Conditional Branching Commands

Conditional Branching Command (x)	Meaning
n"G	True if n > 0
n"L	True if n < 0
n"E	True if n = 0
n"N	True if n ≠ 0

Conditional Branching commands can be nested and can also contain or be contained within a command loop. If included within a command loop, the single quote (') associated with the double quote (") of the Conditional Branch command must be within the same command loop level. That is, when command loops are nested, a Conditional Branch command may not start at one command loop level and end at another.

Acceptable	Not Acceptable
!<SLDA$;0L1T>	<SLDA$"N0L1T>'

3.16 SPECIAL COMMANDS AND COMMAND MODIFIERS

3.16.1 Special Commands

There are several special commands available. These commands are described in Table 3-15.

Table 3-15. Special Commands

Command	Description
?	The Trace command. Used primarily for debugging complex SUPEREDIT macros. The first appearance of a "?" command in a command string turns tracing on and causes SUPEREDIT to print each command and its numeric arguments as it is executed. Subsequent "?" commands in the command string complement the trace mode flag thereby allowing the user to turn tracing on or off within a command string.
Xstring$	Execute "string" as a CLI command. For example: !XDISK$$ R LEFT = 692, USED = 332 !
n=	Numeric typeout command. Causes the value of n to be printed on the terminal. n may be any numeric argument or expression.
←n	When used as the first command after a prompt, causes the previous command line to be placed in buffer n. This command is particularly useful for recovering from an inadvertant ↑A command (see section 2.12). If a ←n was intended as the first command but something else was typed, recovery may be accomplished by a ↑X or sufficient RUBOUTs to rubout the remainder of the line. If more than four characters had been typed, however, part of the old command line will be lost.

3.16.2 Command Modifiers

There are two command modifiers which alter the normal interpretation of commands. The modifiers may be used individually or in combinations. The modifiers are described in Table 3-16.

Table 3-16. Command Modifiers

Command Modifer	Description
@	Used with Insert (I) and Search (S,C,N,Q) commands to change the command terminator from an ESC ($) to any other character. This modifier is particularly useful for inserting or searching for a single ESC ($). NOTE A double ESC ($$) cannot be inserted using the @ modifier since it will terminate the command string. To search for a double ESC ($$) one must use the special control character "↑←". When the @ modifier is used, the new command terminator is defined as the first character following the I, S, C, N or Q command. Examples: @I/text/ / is terminator @S%text% % is terminator @C3text13text23 3 is terminator
:(colon)	a) Can be used with any Search (S,C,N,Q) File Input (A,Y) and the R and nR Output commands to return a command value which then may be used as an argument to the next command. Command value is +1 if command was executed successfully; value is 0 if execution was unsuccessful. No error message is typed out if command is unsuccessful.

Table 3-16. Command Modifiers (Continued)

Command Modifier	Description
:(colon)	a) (Continued) Example: !:5Stext$T$$ — If search was not successful nothing is printed because the CP is at the beginning of the 6th line following the current CP position. If search was successful, types balance of line following "text" (1T). Therefore the success or failure of the search command provides a 0 or +1 numeric argument for the T command. b) The colon (:) can be used with the Text Type-Out commands (see paragraph 3.4) to print the data on the line printer rather than the terminal. :0T, :nT, -:nT, and :m,nT are modified forms of the 0T, nT, -nT, and m,nT commands respectively.

Table 3-16. Command Modifiers (Continued

Command Modifier	Description
:(colon)	b) (Continued) Example: !@:5S%text%:T$$ — If search was not successful nothing is printed because the CP is at the beginning of the 6th line following the current CP position. If search was successful print balance of line following "text" on line printer (1:T). c) Can also be used with semicolon (;) and n; iteration commands (described in section 3.13) to reverse the action of the semicolon (;). That is, control is transferred out of the command loop if last search was successful of if n>0. d) Modifies the GWfilename$ command to create a randomly organized output file rather than a sequentially organized file. Example: !:GWNETSAK.JR$$ — Creates new randomly organized output file named NETSAK.JR.

Table 3-16. Command Modifiers (Continued)

Command Modifier	Description
:(colon)	(Continued e) Modifies the P, nP, -nP, 0P, m,nP, PW, nPW, -nPW, 0PW and m,nPW commands to clear the entire current buffer after output. Example: !:5,500P$$ — Output the 6th through 500th characters in the current buffer to output file with appended form feed and delete the entire buffer. Equivalent to !5,500P#K$$.

3.17 CASE CONTROL COMMANDS

The case control command, and its variations, are used to create and edit upper and lower case files from an upper case terminal. The command is of the form:

{n}WC{x}{y}

where: n may be 0 to deactivate case control, a positive value for up-shifting, or a negative value for down-shifting.
x is the shift character.
y is the shift-lock character.

Case control command variations are listed in Table 3-17. Examples and a discussion of the commands follow the table.

Table 3-17. Case Control Commands

Command	Description
WC	Return value of case control mode. 0 = deactivated 1 = up-shifting -1 = down-shifting
0WC$$	Deactivate case control. Characters are read from terminal exactly as typed without any translation.
nWCx$$	When n is positive, all characters are translated to lower case except when preceded by shift character x which leaves the character upper case.
-nWCx$$	When n is negative, all characters are left as upper case except when preceded by shift character x which translates the character to lower case.
nWCxy$$ -nWCxy$$	Same as nWCx$$ and -nWCx$$ except y is a shift-lock character which translates all characters following the shift-lock character (y) as all upper case (if shifting-up) or all lower case (if shifting-down) until next appearance of y or a double ESCape.

Examples:

!1WC'$$	Set case control mode to shift-up. Use single quote as shift character.

Examples: (Continued)

!I'HELLO$0T$$ 'HELLO!	Insert "Hello" and type out line. On typeout, the same shift character precedes any upper case character so that output appears identical to input.

If the user actually wanted to insert the shift character itself, he would precede it with another shift character which, in this situation, is a single quote. The two adjacent single quote characters would be treated as one. The same is true if the user wants to insert a shift-lock character, i.e., if preceded immediately by a shift character, the shift-lock character itself is inserted into the command string.

!-1WC#$$	Set case control mode to shift-down. User number sign(#) as shift character.
!IS#UPE#RED#I#T$$	Insert "SuPErEDit".
!0WC$$	Deactivate case control.

Case translation operates at the instant of type-in and type-out. Thus, if a "shifted" character is "rubbed out," the echo includes the shift character. If a user types a shift character preceding a character other than a letter or another shift character, shifting has no effect. It is as if the shift character were never typed. The only exception to this rule is the rubout character. Obviously, the rubout cannot be shifted so when a shift character precedes a rubout, the shift character itself is echoed and removed from the input stream. Shift-lock characters may not be rubbed out but can be nullified by typing a second shift-lock character.

As a general rule, it is best to terminate all case control commands with a double ESC ($$) to avoid the possibility of confusion about when the command takes affect.

Examples:

!WC#$$	Set case control mode to shift-up.
!IABC##D#E#E	User typed a rubout after number sign(#) and after E.

Once case control is active, it affects everything which is typed in, including WC commands.

Examples:

!-1WC#$$!1WC##$$	Changing modes with the same shift character requires a double shift character in order for the command to be properly interpreted.
!-1WC#$$!0WC$$!1WC#$$	Deactivating before changing modes is an alternative to the preceding example.
!-1WC#$$!1WC%$$	Changing modes and changing the shift character avoids confusion.
!1WC'&$$	Set shift-up mode using single quote as shift character and ampersand as shift-lock character.

```
!I'THE &NOVA& LINE COMPUTERS HAVE &ALGOL&.$$
```

Insert "The NOVA line computers have ALGOL."

```
!0T$$

! 'THE 'N'O'V'A LINE COMPUTERS HAVE 'A'L'G'O'L.!
```

CHAPTER 4

IMPLEMENTATION NOTES AND EXAMPLES

4.1 CORE UTILIZATION

Initially, SUPEREDIT allocates 10K (decimal) of core. This allows room for approximately 14,000 characters for ECLIPSE line computers and 13,000 in the NOVA line computers. This space is divided, as needed, between the current edit buffer and the command input line buffer. If more room is needed, 1K more core is obtained, and the message "**CORE**" is typed on the console. Each additional 1K of core is enough storage space for 2K characters.

When all available memory is exhausted, during command execution, SUPEREDIT types out -- "MEMORY SPACE EXHAUSTED" and the current command is aborted. Enough room is left at this point to execute some commands, but an output command should be executed shortly.

If, when a command line is being typed in, available core space is exhausted, the error message "MEMORY SPACE EXHAUSTED" is typed out and a new prompt is issued. The old command line can be saved at this point by use of the "←n" command (see paragraph 3.16.1).

4.2 ERROR HANDLING

There are two classes of errors in SUPEREDIT - editor errors and RDOS errors. Editor errors are defined in Appendix C. There is a text file which contains the text of all these messages. If the file is present in the directory that SUPEREDIT is running in, errors are typed in the form "ERROR:text", where "text" is the message shown in the Appendix. If the text file is not present, the message "ERROR MESSAGE TEXT FILE NOT FOUND" is typed before the first prompt and editor errors are of the form "?nn" where nn is the error code shown in the Appendix.

4.2 ERROR HANDLING (Continued)

RDOS errors are described in the RDOS User's Manual. If one of these occurs, "RDOS ERROR n" is typed, where n is the number of the error code.

After any error, up to nine characters of the command line are typed, with the first character typed being the command in error. The only exception to this occurs when the error is in a buffer or insert file being executed. In this case, the part of the command line typed will be the invocation of the file in error, e.g., if buffer A is being executed and an error occurs, what will be typed is "↑BA".

There is one other error that can occur, namely a fatal internal error. If this happens, a return will be made to the CLI, with the error code shown as the address of the command in error, thus causing the CLI to type "UNKNOWN ERROR CODE = XXXXX", where XXXXX is the address of the error. This can only be caused by a malfunction in RDOS or SUPEREDIT. If this ever occurs, please save all details on how to recreate the error and notify your local DGC Applications Engineer.

4.3 RDOS CHANNEL USAGE

The editor is initially built with eight RDOS I/O channels. This is sufficient for most normal editing functions. However, complex editing may require more channels. (With 36 buffers, SUPEREDIT has the potential to use more than the RDOS limit of 64 channels, since it needs one channel for each open input, output, command or insert file). When SUPEREDIT runs out of channels, error code 21 (NO MORE CHANNELS) is output. The number of channels can be changed by modifying location 413_8 in the save file using CEDIT (see OEDIT User's Manual, 093-000084). The right byte in this word contains the number of channels which are initialized for the editor.

4.4 CHANGING THE ESCAPE CHARACTER

SUPEREDIT uses the ASCII ESC character (octal 33) as its escape character. Older model teletypewriters may generate another code from the ESC (or ALT MODE) key. In order to use SUPEREDIT from such a terminal, the user must change location 45_8 in the save file using OEDIT (see OEDIT User's Manual, 093-000084) so that it contains the octal code for the character to be used as the escape character.

4.5 EXAMPLES

The following examples are provided to illustrate the use of as many SUPEREDIT commands as possible. As such, the examples do not necessarily represent the most efficient method of solving the stated problems. In addition, some problems are done by two methods to further illustrate the use of SUPEREDIT commands.

4.5 EXAMPLES (Continued)

Example 1: Change file containing random size pages to one which contains 50 line pages (Method A).

```
!50W$UYfilename$<<S↑L$;-1D>ZJI↑L$R;>US$$
```

Explanation of Example 1:

!50W$	Set window mode - 50 lines long.
UYfilename$$	Open "filename" for input and output. Yank 50 lines.
<<S↑L$;-1D>	Search buffer for all occurences of ↑L (form feed) and delete. Exit inner command loop when search fails.
ZJI↑L$	Position the CP at end of buffer and insert a form feed.
R;>	Output buffer to output file. Yank another 50 lines and repeat outer loop. If R fails exit loop.
US$$	Close I/O files with back-up.

Although "Z" is a character oriented argument and "J" is a line oriented command, use of the "ZJ" combination is acceptable and provides a useful technique for positioning the CP at the end of the buffer. The combinations "ZM" and "VN+1J" will also position the CP at the end of the buffer.

Example 2: Change file containing random size pages to one which contains 50 line pages (Method B).

```
!50W$UYfilename$BS1$12I$BS0$
<<S↑B1$;-1D>ZJI↑B1$R;>US$$
```

Explanation of Example 2:

!50W$	Set window mode - 50 lines long
UYfilename$	Open "filename" for input and output. Yank 50 lines.
BS1$12I$BS0$	Put the ASCII decimal representation of form feed in edit buffer 1 and return to edit buffer 0.
<<S↑B1$;-1D>	Search edit buffer 0 for all occurences of the contents of edit buffer 1 (form feed character) and delete. Exit inner command loop when search fails.
ZJI↑B1$	Position the CP at end of buffer and insert form feed character from buffer 1.
R;>	Output buffer to output file. Yank another 50 lines and repeat outer loop. If R fails, exit loop.
US$$	Close I/O files with back-up.

4.5 EXAMPLES (Continued)

Example 3: Put consecutive page numbers at the top of each page in file (Method A).

```
!UYfilename$
0VS00VS10VS2
!PAGE!7<↑I$>IPAGEΔ$
VI0-10"E0VS0VI1-10"E0VS1VI2''
48+V2I48+V1I48+V0II)
$:R"NOPAGE$'US$$
```

Explanation of Example 3:

UYfilename	Open "filename" for input and output. Yank first page.
0VS00VS10VS2	Set units, tens and hundreds counter to zero.
!PAGE!7<↑I$>IPAGEΔ$	Insert seven tabs and insert "PAGEΔ".
VI0-10"E	Increment units counter and subtract 10. If not equal to 0 skip commands to matching apostraphe (').
0VS0VI1-10"E	If units counter reached 10, zero the units counter, increment the tens counter and subtract 10. If not equal to zero skip commands to matching apostrophe (').
0VS1VI2''	If tens counter reached ten, zero the tens counter and increment the hundreds counter.
48+V2I48+V1I48+V0II)	Insert the ASCII decimal equivalent of the integers stored in the hundreds, tens, and units counters followed by a carriage return.
$:R"NOPAGE$'US$$	Output page and yank another page. If yank was successful jump to location PAGE in macro. Otherwise close I/O files with back-up.

4.5 EXAMPLES (Continued)

Example 4: Put consecutive page numbers at the top of each page in file (Method B).

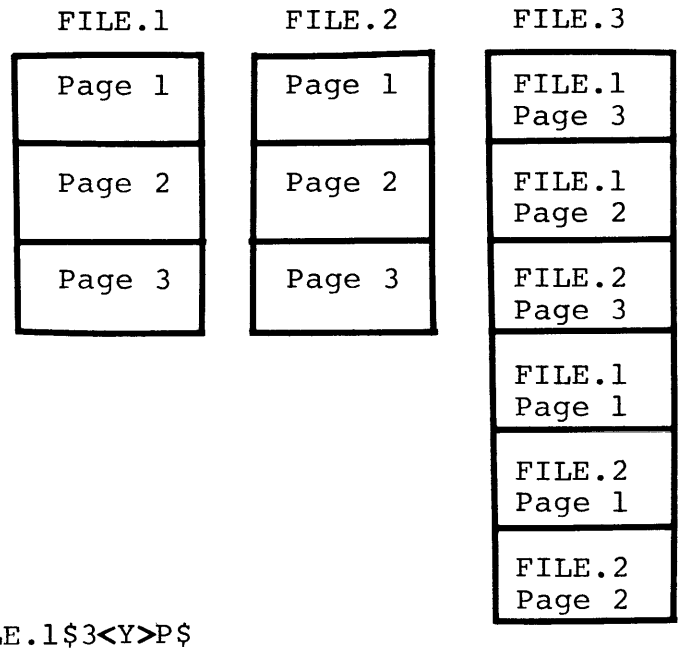
```
!UYfilename$0VS0$<7<↑I$>IPAGEΔ$VI0\$R;>US$$
```

Explanation of Example 4:

!UYfilename$	Open "filename" for input and output. Yank a page.
0VS0$	Set variable 0 to 0.
<7<↑I$>IPAGEΔ$	Space over seven tabs and insert "PAGEΔ".
VI0\$	Increment variable 0, convert integer to decimal ASCII representation and insert.
R;>	Output page to output file and yank another page. If yank is successful, repeat loop. Otherwise, exit loop.
US$$	Close I/O files with back-up.

4.5 EXAMPLES (Continued)

Example 5: Create file (FILE.3) comprised of rearranged pages from two other files (FILE.1 and FILE.2).




```
!GWFILE.3$GRFILE.1$3<Y>P$
GRFILE.1$2<Y>P$GRFILE.2$3<Y>P$
GRFILE.1$YPGRFILE.2$YRP$GR$GC$$
```

Explanation of Example 5:

!GWFILE.3$	Create new output file FILE.3.
GRFILE.1$3<Y>P$	Open FILE.1 for input, get the third page and output to FILE.3.
GRFILE.1$2<Y>P$	Close and reopen FILE.1, get second page and output to FILE.3.
GRFILE.2$3<Y>P$	Close FILE.1 and open FILE.2 for input. Get third page and output to FILE.3.
GRFILE.1$YP	Close FILE.2 and open FILE.1 for input. Yank and output first page to FILE.3
GRFILE.2$	Close FILE.1 and open FILE.2 for input.
YRP$	Yank first page and output. Yank second page and output
GR$GC$$	Close FILE.2 and FILE.3.

4.5 EXAMPLES (Continued)

Example 6: Create macro of Example 4 in a buffer which can be called for reuse by a ↑Bx command.

```
!BS1$@I%0VS0$<7<↑I$>IPAGE∆$VI0\$R;>US%BS0$$
!UYfilename$↑B1$$
```

Explanation of Example 6:

!BS1$@I%...%	Make buffer 1 current and insert command string delimited by "%" character.
BS0$$	Return to buffer 0.
!UYfilename$↑B1$$	Open "filename" for input and output and execute macro in buffer 1. I/O files are closed by macro. This command line may be repeated for any "filename" without the need for entirely retyping the macro contained in buffer 1.

Example 7: Create macro of Example 4 in a file which can be called for reuse by a ↑Gfilename$ command. The file can be saved for use in subsequent editing sessions.

```
!GWMACRO.1$@I%0VS0$<7<↑I$>IPAGE∆$VI0↖$R;>US%PWUE$$
!UYfilename$↑GMACRO.1$$
```

Explanation of Example 7:

!GWMACRO.1$	Open file MACRO.1 for output.
@I%...%	Insert command string delimited by "%".
PWUE$$	Write content of buffer to output file and close file. Use PW to avoid form feed which would be interpreted as an erroneous command in the file.
!UYfilename$↑GMACRO.1$$	Open "filename" for input and output and execute commands in file MACRO.1.

APPENDIX A

ASCII CHARACTER SET

CHARACTER	OCTAL	DECIMAL
NULL	000	0
↑A	001	1
↑B	002	2
↑C	003	3
↑D	004	4
↑E	005	5
↑F	006	6
↑G	007	7
↑H	010	8
TAB (↑I)	011	9
LINE FEED (↑J)	012	10
VERT. TAB (↑K)	013	11
FORM FEED (↑L)	014	12
CARRIAGE RETURN (↑M)	015	13
↑N	016	14
↑O	017	15
↑P	020	16

CHARACTER	OCTAL	DECIMAL
↑Q	021	17
↑R	022	18
↑S	023	19
↑T	024	20
↑U	025	21
↑V	026	22
↑W	027	23
↑X	030	24
↑Y	031	25
↑Z	032	26
ESC or ALT MODE or CTRL-SHIFT-K	033	27
CTRL-SHIFT-L	034	28
CTRL-SHIFT-M	035	29
CTRL-SHIFT-N	036	30
CTRL-SHIFT-O	037	31
SPACE	040	32
!	041	33
"	042	34
#	043	35
$	044	36
%	045	37
&	046	38

CHARACTER	OCTAL	DECIMAL
'(apostrophe)	047	39
(	050	40
)	051	41
*	052	42
+	053	43
, (comma)	054	44
- (minus)	055	45
.	056	46
/	057	47
0	060	48
1	061	49
2	062	50
3	063	51
4	064	52
5	065	53
6	066	54
7	067	55
8	070	56
9	071	57
:	072	58
;	073	59
<	074	60

CHARACTER	OCTAL	DECIMAL
=	075	61
>	076	62
?	077	63
@	100	64
A	101	65
B	102	66
C	103	67
D	104	68
E	105	69
F	106	70
G	107	71
H	110	72
I	111	73
J	112	74
K	113	75
L	114	76
M	115	77
N	116	78
O	117	79
P	120	80
Q	121	81
R	122	82

CHARACTER	OCTAL	DECIMAL
S	123	83
T	124	84
U	125	85
V	126	86
W	127	87
X	130	88
Y	131	89
Z	132	90
[	133	91
\ (SHIFT-L)	134	92
]	135	93
↑	136	94
← or _	137	95
`	140	96
a	141	97
b	142	98
c	143	99
d	144	100
e	145	101
f	146	102
g	147	103
h	150	104

CHARACTER	OCTAL	DECIMAL
i	151	105
j	152	106
k	153	107
l	154	108
m	155	109
n	156	110
o	157	111
p	160	112
q	161	113
r	162	114
s	163	115
t	164	116
u	165	117
v	166	118
w	167	119
x	170	120
y	171	121
z	172	122
{	173	123
|	174	124
}	175	125
~(tilde)	176	126
RUBOUT or DELETE	177	127

APPENDIX B

SUPEREDIT COMMAND SUMMARY

B.1 FILE SPECIFICATION COMMANDS

Command	Description	Page Reference
GWfilename$	Create and open a new output file in sequentially organized format.	3-1
:GWfilename$	Create and open a new output file in randomly organized format.	3-2
GC$	Close current output file.	3-2
UYfilename$	Open filename for input, filename.SC for output and Yank (Y) a page.	3-2
GRfilename$	Close current input file and open filename for input.	3-3
GR$	Close current input file.	3-3
US$	Transfer remainder of input file to output file. Close input/output files. If opened by UY command, create back-up.	3-3
UE$	Transfer remainder of input file to output file. Close input/output files. If opened by UY command, delete input file and rename output to input filename.	3-4
U?$	Type global and local file status.	3-5

B.2 PAGE AND WINDOW MODE COMMANDS

Command	Description	Page Reference
WM	Return value of data input mode. (0 or n).	3-5
nWM	Change from Page to Window Mode with n line window length.	3-5
0WM	Change from Window to Page Mode.	3-5

B.3 INPUT COMMANDS

Command	Description	Page Reference
Y	Clear buffer and read one page from input file.	3-6
A	Read one page from input file and append to current edit buffer.	3-6
:Y and :A	Same as Y and A respectively except command returns a value of +1 if it succeeds and 0 if it fails instead of an error message.	3-6

B.4 TEXT TYPE-OUT COMMANDS

Command	Description	Page Reference
T	Type current line showing position of CP.	3-7
0T	Type from beginning of current line to position of CP.	3-7
nT	Type from CP through next n carriage returns.	3-7
-nT	Type n lines preceding current line plus characters on current line up to CP.	3-7
m,nT	Type the m+1th through nth characters inclusive.	3-7
:0T, :nT, :-nT, :m,nT	Same as 0T, nT, -nT, and m,nT except output to line printer instead of terminal.	3-7

B.5 CHARACTER POINTER COMMANDS

Command	Description	Page Reference
nJ	Move CP to beginning of line n.	3-8
J or 0J	Move CP to beginning of buffer.	3-8
nL	Move CP to beginning of line following the nth carriage return.	3-8
-nL	Move CP to beginning of nth line preceding current line.	3-8
L or 0L	Move CP to beginning of current line.	3-8
nM	Move CP across n characters.	3-8

B.6 SEARCH COMMANDS AND CONTROL CHARACTERS IN SEARCHES

Command	Description	Page Reference
Stext$	Starting at CP search for text delimited by S and ESC($) in current buffer.	3-9
nStext$	Search from CP through next n carriage returns for text.	3-9
-nStext$	Search preceding n lines plus characters up to CP on current line for text.	3-9
0Stext$	Search current line from beginning of line up to CP for text.	3-10
m,nStext$	Search m+1th through nth characters for text.	3-10
Ntext$	Search rest of input file for text. Output buffer to output file for each page until text is found.	3-11
Qtext$	Same as Ntext$ except buffer is not output.	3-11
Ctext1$text2$	Search for text1 and replace with text2.	3-10
nCtext1$text2$	Search from CP through next n carriage returns for text1 and replace with text2.	3-10
-nCtext1$text2$	Search preceding n lines plus characters up to CP on current line for text1 and replace with text2.	3-10
0Ctext1$text2$	Search current line from beginning of line up to CP for text1 and replace with text2.	3-11

B.6 SEARCH COMMANDS AND CONTROL CHARACTERS IN SEARCHES (Continued)

Command	Description	Page Reference
m,nCtext1$text2$	Search m+1th through nth characters for text1 and replace with text2.	3-11
@S%text%	Same as Stext$ except text delimited by first character after S (arbitrarily shown as "%"). Other variations are: @nS%text% @-nS%text% @0S%text% @m,nS%text% @N%text% @Q%text% @C%text1%text2% @nC%text1%text2% @-nC%text1%text2% @0C%text1%text2% @m,nC%text1%text2%	3-12
:Stext$	Same as Stext$ except command returns a value of +1 if it succeeds and 0 if it fails instead of an error message. Other variations are: :nStext$:-nStext$:0Stext$:m,nStext$:Ntext$:Qtext$:Ctext1$text2$:nCtext1$text2$:-nCtext1$text2$:0Ctext1$text2$:m,nCtext1$text2$	3-12

B.6 SEARCH COMMAND AND CONTROL CHARACTERS IN SEARCHES (Continued)

Command	Description	Page Reference
@:S%text%	Combines the @ and : modifiers described above. Other variations are: @:nS%text% @:-nS%text% @0S%text% @:m,nS%text% @:N%text% @:Q%text% @:C%textl%text2% @:nC%textl%text2% @:-nC%textl%text2% @:0C%textl%text2% @:m,nC%textl%text2%	3-12
↑Z	Accept any character in this position.	3-13
↑Nx	Accept any character except x in this position.	3-13
↑Vx	Accept any number of the character x in this position.	3-13
↑←	Intrepret next character literally, rather than as a special character.	3-13
↑P	Position CP at location of ↑P in search string if search is successful.	3-14
↑T	Accept any number of spaces or tabs in this position.	3-14

B.7 INSERTION COMMANDS

Command	Description	Page Reference
Itext$	Insert text delimited by I and ESC($) at the current position of the CP.	3-15
↑Itext$	Insert tab plus text delimited by ↑I and ESC($).	3-15
nI	Insert character whose ASCII decimal equivalent (from Appendix A) is n.	3-15
n\	Insert ASCII representation of the decimal number n into the buffer at the CP location.	3-15
@I%text% @↑I%text%	Same as Itext$ and ↑Itext% respectively except inserted text is delimited by first character after I (arbitrarily shown as "%").	3-15

B.8 DELETION COMMANDS

Command	Description	Page Reference
nD	Delete n characters starting at CP position.	3-16
nK	Delete characters starting at CP position through n carriage returns.	3-16
-nK	Delete preceding n lines and characters up to the CP on the current line.	3-16
m,nK	Delete the m+1th through nth characters inclusive.	3-16
K or 0K	Delete characters starting at CP position back to beginning of line.	3-16

B.9 OUTPUT COMMANDS

Command	Description	Page Reference
P	Output edit buffer with appended form feed to output file.	3-17
nP	Output n lines from CP to output file with appended form feed.	3-17
-nP	Output preceding n lines plus characters up to the CP on current line to output file with appended form feed.	3-17
0P	Output current line from beginning of line up to the CP to output file with appended form feed.	3-17
m,nP	Output m+1th through nth characters inclusive to output file with appended form feed.	3-17
PW	Output edit buffer to output file.	3-17
nPW	Output n lines from CP to output file.	3-17
-nPW	Output preceding n lines plus characters up to the CP on current line to output file.	3-17
0PW	Output current line from beginning of line up to the CP to output file.	3-18
m,nPW	Output m+1th through nth characters inclusive to output file.	3-18

B.9 OUTPUT COMMANDS (Continued)

Command	Description	Page Reference
R	Output edit buffer, clear buffer, and yank next page from input file.	3-18
nR	Repeat R command n times.	3-18
:R and :nR	Same as R and nR except command returns a value of +1 if it succeeds and 0 if it fails instead of an error message.	3-18
:P	Same as P except buffer cleared after output. Other variations are: :nP :-nP :0P :m,nP :PW :nPW :-nPW :0PW :m,nPW	
E	Output buffer and rest of input file.	3-18

B.10 EXIT COMMANDS

Command	Description	Page Reference
H	Normal exit from SUPEREDIT. Returns user to the CLI.	3-19
↑C	Emergency exit from SUPEREDIT. System break command returns user to the CLI.	3-19

B.11 BUFFER COMMANDS

Command	Description	Page Reference
BSx	Change current buffer to buffer x.	3-20
B?$	Type out buffer status for all active buffers.	3-20
B?x$	Type out status of buffer x.	3-20
BCx	Copy entire content from current buffer to buffer x.	3-20
nBCx	Copy the next n lines from CP position in current buffer to buffer x.	3-20
-nBCx	Copy the preceding n lines plus the characters up to the CP in the current line to buffer x.	3-20
m,nBCx	Copy the m+1th through nth characters inclusive from current buffer to buffer x.	3-20
BTx, nBTx, -nBTx, m,nBTx	Same as BCx, nBCx, -nBCx and m,nBCx respectively except characters moved from current buffer are deleted.	3-20
BKx	Delete buffer x.	3-20
BGx{filename}$	Execute a Gx command (where Gx= GR, GW or GC) local to the current buffer.	3-21
BUx{filename}$	Execute a Ux command (where Ux= UY, UE or US) local to the current buffer.	3-21
BAx	Activate buffer x.	3-21

B.12 COMMAND STRING INSERTION COMMANDS

Command	Description	Page Reference
↑Gfilename	Inserts contents of file filename into command string in place of ↑Gfilename command.	3-23
↑Bx	Inserts content of buffer x into command string in place of ↑Bx command.	3-23

B.13 ITERATION COMMANDS

Command	Description	Page Reference
n<command string>	Perform enclosed commands n times. If n ≤ 0 skip command loop.	3-23
;	Jump out of command loop if last Search command failed.	3-24
n;	Jump out of command loop if n ≤ 0.	3-25
:;	Jump out of command loop if last Search command was successful.	3-25
n:;	Jump out of command loop if n > 0.	3-25

B.14 NUMERIC VARIABLES AND SPECIAL CHARACTERS

Command	Description	Page Reference
Vv	Represents the current value of variable v.	3-26
VIv	Increments the value of variable v and represents the incremented value.	3-26
VDv	Decrements the value of variable v and represents the decremented value.	3-26
nVSv	Set variable v to value n and return that value.	3-26
Z	Represents the total number of characters in the current buffer.	2-7
.(Period)	Represents the current CP position.	2-7
#	Equivalent to double argument 0,Z (entire edit buffer).	2-7
VN	Represents the number of lines in current buffer.	2-7
VL	Represents the number of the line which contains the CP.	2-7

B.15 FLOW CONTROL COMMANDS

Command	Description	Page Reference
!string!	Define a label named "string" in the command string.	3-27
Ostring$	Transfer control to label "string".	3-27
n"Gcommand string'	Execute command string if n > 0.	3-27

B.15 FLOW CONTROL COMMANDS (Continued)

Command	Description	Page Reference
n"Lcommand string'	Execute command string if n > 0.	3-27
n"Ecommand string'	Execute command string if n = 0.	3-27
n"Ncommand string'	Execute command string if n ≠ 0.	3-27

B.16 SPECIAL COMMANDS AND COMMAND MODIFIERS

Command	Description	Page Reference
?	Complement trace mode flag.	3-29
Xstring$	Execute "string" as a CLI command.	3-29
n=	Type out value of numeric argument n.	3-29
←n	Place previous command string in buffer n. (Must be first command after prompt).	3-29
@	Modify Insert and Search commands to change text delimiter.	3-30
:	Modify Search and File Input commands (A,Y,R,S,N,Q,C) to return value of 0 if Search or Input fails; +1 is successful.	
	Modify Type commands (nT,-nT, m,nT) to output to line printer.	
	Modify Output commands (P,nP,m,nP, PW,nPW,m,nPW) to clear buffer after output.	
	Modify GWfilename$ command to create randomly organized output file.	3-30

B.17 CASE CONTROL COMMANDS

Command	Description	Page Reference
WC	Return value representing case control mode (0, 1, or -1).	3-33
0WC$$	Deactivate case control.	3-33
nWCx$$	When n is positive shift-up using x as shift character.	3-33
-nWCx$$	When n is negative, shift-down using x as shift character.	3-33
nWCxy$$	When n is positive, shift-up using x as shift character and y as shift-lock character.	3-33
-nWCxy$$	When n is negative shift-down using x as shift character and y as shift-lock character.	3-33

APPENDIX C

SUPEREDIT ERROR MESSAGES

Error Number	Error Message	Explanation
1	ILLEGAL FILE NAME	
2	SYNTAX ERROR	Format of command incorrect (e.g., 2#T).
3	ILLEGAL VARIABLE NAME	Legal variable names are 0 through 9 only.
4	ILLEGAL NUMBER OF ARGUMENTS TO COMMAND	Commands such as m,nR are not legal.
5	ILLEGAL BUFFER NAME	A through Z and 0 through 9 are the only legal buffer names.
6	BUFFER IS INACTIVE	An attempt to BKx or ↑Bx an inactive or empty buffer.
7	MAXIMUM ITERATION LEVEL EXCEEDED	Command loop nesting level is greater than 10.
10	NO OPEN FILE	Attempt to use A, Y, P, PW, E, R, UE, or US command without an open file.
11	FILE ALREADY EXISTS	
12	FILE DOES NOT EXISTS	
13	FILE ALREADY OPEN	Attempt to open an output file with a GW or UY command without first closing a previously opened output file.

Error Number	Error Message	Explanation
14	NO MORE CHARACTERS IN INPUT FILE	End of file reached by an A, Y, or R command.
15	UNSUCCESSFUL SEARCH	
16	MAXIMUM INSERT DEPTH EXCEEDED	Nesting level of ↑Bx and ↑Gfilename$ command exceeded.
17	SEARCH STRING OR <> BROKEN OVER TWO LEVELS	Search command or command loop starts at one command insert level and ends at another level.
20	INSERT FILE TOO LONG	File larger than 16K bytes.
21	NO MORE CHANNELS AVAILABLE	All I/O channels in use. See section 4.3.
22	INPUT LINE TOO LONG	Maximum line length is 132 characters. A line longer than 132 characters read by an A, Y, R, or E command. Warning message only; execution of the command string continues.
23	ATTEMPT TO DELETE CURRENT BUFFER	A BKx command may not be used for the current buffer.
24	PARITY ERROR	The character in error is replaced by a backslash (\). Warning message only; execution of the command string continues.
25	STACK OVERFLOW	
26	MEMORY SPACE EXHAUSTED	No more core available. See section 4.1.
27	ATTEMPT TO EXECUTE CURRENT BUFFER	A ↑Bx command may not be used for current buffer.

Error Number	Error Message	Explanation
30	UNTERMINATED STRING	@S or @I command without second delimiter.
31	< WITH NO CORRESPONDING >	
32	" WITH NO CORRESPONDING '	
33	LABEL NOT FOUND	Label referenced by unconditional jump (O) command not found.
34	UNABLE TO OPEN $LPT	
35	STRING ARGUMENT TOO LONG	In Ostring$ command, string is > 48 characters.
36	FIRST ARGUMENT GREATER THAN SECOND ARGUMENT	In commands which have two arguments, the first must be less than or equal to the second (e.g., m,nT).
40	RENAMING ERROR	
41	ILLEGAL COMMAND	A character or characters were used which are not defined as a legal SUPEREDIT command.
42	ILLEGAL ARGUMENT TO COMMAND	Command which only accepts a positive argument was given a negative argument.
43	ILLEGAL CONTROL CHARACTER IN SEARCH STRING	A character other than those described in paragraph 3.6.2 appeared in a search command argument.
44	FILE READ PROTECTED	
45	FILE WRITE PROTECTED	
53	DIRECTORY SPECIFIER UNKNOWN	

DataGeneral

Software Documentation Remarks Form

How Do You Like This Manual?

Title _____ No. _____

We wrote the book for you, and naturally we had to make certain assumptions about who you are and how you would use it. Your comments will help us correct our assumptions and improve our manuals. Please take a few minutes to respond.

If you have any comments on the software itself, please contact your Data General representative. If you wish to order manuals, consult the Publications Catalog (012-330).

Who Are You?

- ☐ EDP Manager
- ☐ Senior System Analyst
- ☐ Analyst/Programmer
- ☐ Operator
- ☐ Other _____

What programming language(s) do you use? _____

_____

How Do You Use This Manual?

(List in order: 1 = Primary use)

- ____ Introduction to the product
- ____ Reference
- ____ Tutorial Text
- ____ Operating Guide
- ____ _____

Do You Like The Manual?

Yes	Somewhat	No	
☐	☐	☐	Is the manual easy to read?
☐	☐	☐	Is it easy to understand?
☐	☐	☐	Is the topic order easy to follow?
☐	☐	☐	Is the technical information accurate?
☐	☐	☐	Can you easily find what you want?
☐	☐	☐	Do the illustrations help you?
☐	☐	☐	Does the manual tell you everything you need to know?

Comments?

(Please note page number and paragraph where applicable.)

From:

Name _____ Title _____ Company _____

Address _____ Date _____

FOLD DOWN FIRST FOLD DOWN

FIRST
CLASS
PERMIT
No. 26
Southboro
Mass. 01772

BUSINESS REPLY MAIL

No Postage Necessary if Mailed in the United States

Postage will be paid by:

Data General Corporation

Southboro, Massachusetts 01772

ATTENTION: Software Documentation

FOLD UP SECOND FOLD UP

STAPLE