DataGeneral

RDOS/DOS Text Editor

DataGeneral

RDOS/DOS Text Editor

Notice

Data General Corporation (DGC) has prepared this document for use by DGC personnel, customers, and prospective customers. The information contained herein shall not be reproduced in whole or in part without DGC's prior written approval.

DGC reserves the right to make changes in specifications and other information contained in this document without prior notice, and the reader should in all cases consult DGC to determine whether any such changes have been made.

THE TERMS AND CONDITIONS GOVERNING THE SALE OF DGC HARDWARE PRODUCTS AND THE LICENSING OF DGC SOFTWARE CONSIST SOLELY OF THOSE SET FORTH IN THE WRITTEN CONTRACTS BETWEEN DGC AND ITS CUSTOMERS. NO REPRESENTATION OR OTHER AFFIRMATION OF FACT CONTAINED IN THIS DOCUMENT INCLUDING BUT NOT LIMITED TO STATEMENTS REGARDING CAPACITY, RESPONSE-TIME PERFORMANCE, SUITABILITY FOR USE OR PERFORMANCE OF PRODUCTS DESCRIBED HEREIN SHALL BE DEEMED TO BE A WARRANTY BY DGC FOR ANY PURPOSE, OR GIVE RISE TO ANY LIABILITY OF DGC WHATSOEVER.

IN NO EVENT SHALL DGC BE LIABLE FOR ANY INCIDENTAL, INDIRECT, SPECIAL OR CONSEQUENTIAL DAMAGES WHATSOEVER (INCLUDING BUT NOT LIMITED TO LOST PROFITS) ARISING OUT OF OR RELATED TO THIS DOCUMENT OR THE INFORMATION CONTAINED IN IT, EVEN IF DGC HAS BEEN ADVISED, KNEW OR SHOULD HAVE KNOWN OF THE POSSIBILITY OF SUCH DAMAGES.

Ordering No. 069-400016

Rev. 00, February 1983

Preface

The RDOS and DOS text editors, Edit and Multiuser Edit (Medit), are programs you can use to create and change text files. Medit is an expansion of Edit and allows several users to edit at the same time. These editors run on the Real-Time Disk Operating System (RDOS) and the Disk Operating System (DOS).

You can use Edit and Medit to keep notes, reports, lists, tables, data, book chapters, or entire books. Your text files consist of characters (letters, numbers, punctuation marks, spaces, and control) that you type in at your keyboard. You can store your files on disk and reopen them any time for updates. In Edit letters display in uppercase only.

After editing your text you can print it out or, if your text is a source program, you can leave Edit and call other Data General utility programs to compile and load your program so that it is ready to execute.

Chapter 1 introduces you to the editing cycle and fundamental Edit concepts. Chapter 2 directs you through a sample editing session in which you use basic editing commands and learn how to construct macros. Chapter 3 directs you through an editing session using more complex commands. Chapter 4 describes loading single user Edit and loading and managing Medit. Chapter 4 assumes knowledge of mapped and unmapped foreground operations.

In Chapters 1 to 3, references to Edit refer also to Medit unless differences between them are specifically noted. Chapter 4 discusses Edit and Medit separately.

Related Manuals

The following manuals are a part of the series of books published on RDOS, DOS, and RTOS.

Introduction to RDOS (DGC No. 069-400011) describes the fundamentals of using RDOS and summarizes the features, utilities, and capabilities of the operating system.

Guide to RDOS Documentation (DGC No. 069-400012) describes all of the books that comprise the revised documentation set for RDOS, DOS, and RTOS and lists the previous books that each replaces.

How to Load and Generate RDOS (DGC. No. 069-400013) explains how to generate and maintain RDOS. Instructions cover preparing hardware, program loading, disk initialization, bootstrapping, tailoring, system backup, and system tuning.

How to Generate Your DOS System (DGC No. 093-000222-01) explains how to generate and maintain DOS. Instructions cover preparation of hardware, program loading, disk initialization, bootstrapping, tailoring, system backup, patching, and optimization.

RDOS/DOS Command Line Interpreter (DGC No. 069-400015) discusses the user interface with the operating system. It covers the Command Line Interpreter (CLI) features and command mechanisms, and instructs you on how to use CLI commands. Features and operation of the Batch monitor are also presented.

RDOS/DOS Superedit Text Editor (DGC No. 069-400017) introduces the commands and concepts of the Superedit Text Editor (Speed) which offers more powerful features for editing text.

RDOS System Reference (DGC No. 093-400027) describes RDOS system features, calls, and user device driver implementation for assembly language and high-level language programming.

DOS Reference Manual (DGC No. 093-000201-03) discusses all features of DOS for system programmers, including system control, memory management, and system calls.

RDOS/DOS User's Handbook (DGC No. 093-000105-04) summarizes the commands, calls, and error messages of the RDOS/DOS CLI, Batch monitor, and utility programs.

RDOS/DOS Assembly Language and Program Utilities (DGC No. 069-400019) details the Extended Assembler (ASM), Macroassembler (MAC), Extended Relocatable Loader (RLDR and OVLDR), and Library File Editor (LFE) utilities that aid you in programming.

RDOS/DOS Debugging Utilities (DGC No. 069-400020) describes five utilities that assist you in editing, debugging, and patching programs—the Symbolic Editor (SEDIT), Symbolic Debugger (DEBUG), Disk Editor (DISKEDIT), and Patch (ENPAT and PATCH).

RDOS/DOS Sort/Merge and Vertical Format Unit Utilities (DGC No. 069-400021) offers functional information on Sort/Merge (RDOSSORT), which helps you manipulate records in data files, and the Vertical Format Unit (VFU), which enables you to define data channel line printer page formatting.

RDOS/DOS Backup Utilities (DGC No. 069-400022) presents the features and operation of the utilities that involve disk and tape backup. These are BURST/TBURST, DBURST/MBURST/RBURST, DDUMP/DLOAD, FDUMP/FLOAD, and OWNER.

Typesetting Conventions

We use the following conventions for command formats in this manual:

COMMAND *required [optional] . . .*

Where	Means
COMMAND	Enter the command or its accepted abbreviation as shown. Uppercase letters indicate the command mnemonic.
required	Enter an argument (such as a filename). Lowercase italic letters indicate this argument. Sometimes we use: $required_1 \mid required_2$ You can choose between the arguments listed. Do not type the vertical bar; it merely separates the choices.
[optional]	Brackets mean that you have the option of entering the argument, indicated in lowercase italic letters. Command switches also appear in this format. Do not include the brackets in your code; they only set off the choices.
. . .	Repeat the preceding entry or entries. The explanation tells you exactly what to repeat.

- The process has continued without incident, and you may now take the next action described.

In examples of dialogue, we use:

THIS TYPEFACE TO SHOW YOUR ENTRY and
THIS TYPEFACE TO SHOW SYSTEM RESPONSES.

Additionally, we use certain symbols in special ways:

Symbol	Means
⟨CR⟩	Carriage Return. Press the CR key on your keyboard. NOTE: If you have a D100, D200, or G300 terminal, you should press the New Line key (NEW LINE) on your keyboard when you see ⟨CR⟩.
☐	Include a space at this point. We use this to clarify command lines in some cases. Normally, you can see where to put spaces.
CTRL-	Depress and hold the Control key (CTRL) while you press the character that follows CTRL-.
⟨NL⟩	New Line. Press the NEW LINE key on your keyboard.
R	RDOS/DOS Command Line Interpreter prompt.
*	The asterisk (*) is the Edit prompt and means that Edit is ready to accept a command.

All numbers are decimal unless we indicate otherwise; for example, to indicate octal 35, we use 35_8.

The keys defined as DEL and RUBOUT perform the same function. Depending on the console you are using, you will find one of these keys on your keyboard. In this manual, we use DEL to represent that function.

The up arrow symbol (↑) is also executed by different keys, depending on your console. You execute it by pressing either SHIFT-N or SHIFT-6. In this manual, we reference SHIFT-6 to execute the up arrow symbol.

We welcome your suggestions for the improvement of this and other Data General publications. To communicate with us, use the postpaid comment form at the end of this manual.

Table of Contents

Figures

Tables

Chapter 1

Text Editing Concepts

The general procedure for editing a text file follows:

1. Start the text editor Edit from the Command Line Interpreter (CLI). (Since Medit runs only the editor, Medit users never need to call it. See Chapter 4 for an explanation of how to start Medit).

2. Specify the filename(s) or input/output device(s) you wish to edit.

3. Insert text at your console or bring a page into the buffer from the input file or device.

4. Modify the text in the edit buffer.

5. Transfer the page from the edit buffer to the file or output device.

6. Repeat steps 3, 4, and 5 until you finish with the file.

7. Close the file(s).

After you start Edit (step 1), your console displays the asterisk prompt (*). Whenever this prompt appears, Edit is ready to accept your command. You must specify which file or input/output devices you want to use (step 2). When the prompt reappears, type your input command to insert or read in text (step 3). To edit the page, type modification commands from the console (step 4). Next, type output commands to transfer the contents of the buffer to the file or output device (step 5). If necessary, repeat steps 3, 4, and 5 (step 6). Close the file (step 7).

Concepts and Terms

The following paragraphs contain the basic information you need to start using Edit.

The Terminal

The terminal is your main editing instrument. It allows you to view the text and enter commands, and echoes everything you type in at the keyboard. The terminal can display the text of any file previously entered in the editing system.

When you edit a file, you view characters on the terminal from two different sources:

1. All characters you type in at the keyboard.

2. Whatever the editor displays from the present file or from any previously entered file.

Throughout the rest of this manual we refer to the terminal as the console.

ASCII Character Set

You communicate with Edit by means of the ASCII Character Set (American Standard Code for Information Interchange), which is composed of alphanumeric characters, punctuation marks, and control characters. Table 1.1 summarizes the number and type of characters.

Number	Character Type	Function
95	Graphic	Form text (letters, digits, punctuation marks, spaces). Advance cursor one position.
07	Format control	Direct Edit to perform a formatting action, such as start a new line or move to a tab stop. CTRL key plus one character.
26	Command control	Direct Edit to act on your text (search for or change a character string), to display part (or all) of what you have typed); one character in length.

Table 1.1 Characters used in Edit

Text

Text consists of a continuous string of ASCII characters (letters, numbers, carriage returns, control characters, tabulations, and spaces). Although the characters appear on your console as you type them, they are stored in memory as a continuous string of characters. The exception would be Carriage Returns and spaces which do not display as characters. You end a line of text with a Carriage Return ⟨CR⟩.

Input: Lines and Pages

A line is made up of a string of characters terminated by CR. (CTRL-M performs the same function.) A page is made up of a string of lines terminated by a Form Feed (CTRL-L). Form Feed is an ASCII character (non-printing) which signals a printing device, such as a line printer, to move the paper to the top of the next sheet. CTRL-L clears a console screen.

Input File

An input file is the device or disk file you use for inputting text. When you edit in RDOS, you can use disk files for input and output, or you can use any input/output device that the operating system supports.

Edit Buffer

The edit buffer is a temporary storage area between input and output files in memory which holds text as you type or edit it.

Edit Commands

Edit commands fall into three categories: input, output, and buffer. Input commands input text to a buffer, output commands output text from a buffer to an output file, and buffer commands perform all tasks in the buffer (modify text, move Character Pointer (CP), or type out buffer contents).

Output Files

When you finish a session in Edit, you write your text to one or more output files. You can update the disk file that holds your text after you edit each page, so that if memory is destroyed (by a system failure), you lose no more than the current page.

When you want to work with your text, the output file becomes your input file, and you read its contents into memory in order to work on them. Figure 1.1 shows a diagram of the relationship among the edit buffer, files, and input/output devices.

Character Pointer

The Character Pointer (CP) is positioned between two characters (never at a character). Your console does not display the CP. To edit, you use the CP to specify the position at which you want to modify or examine text. Any text you insert or delete from the buffer is added or removed beginning at the current position of the CP. In this manual, we use the up arrow (↑) to show the position of the CP. For example,

ABCD↑F

To modify text, place the CP at the location where you want to begin to make changes. In the example above, you might insert an E to produce ABCDEF.

Line Numbering

Edit maintains an implicit line numbering system which starts with 1. Edit continually updates line numbers as you insert and delete lines. For example, your buffer might contain these seven lines.

THIS IS LINE A
THIS IS LINE B
THIS IS LINE C
THIS IS LINE D
THIS IS LINE E
THIS IS LINE F
THIS IS LINE G

If you delete lines 3 and 4,

THIS IS LINE C
THIS IS LINE D

then lines 5, 6, and 7 become lines 3, 4, and 5:

THIS IS LINE A
THIS IS LINE B
THIS IS LINE E
THIS IS LINE F
THIS IS LINE G

Escape Key

In Edit you press the Escape key (ESC) to:

- Signal the end of the command or command string (double ESC).
- Separate a string from the next command character or from another string (single ESC).

Your system console displays a dollar sign ($) when you press ESC.

Control Characters

You can type control characters by holding down the CTRL key and pressing the desired letter. Control characters, such as CTRL-A, appear on the console as ↑A.

Tabbing

If your keyboard does not have a TAB key, insert Tabs by holding down the CTRL key and pressing the letter I. Predefined tab positions occur at spaces 1, 9, 17, 25, and so on. Every time you press TAB or type CTRL-I, the CP moves to the next predefined tab position.

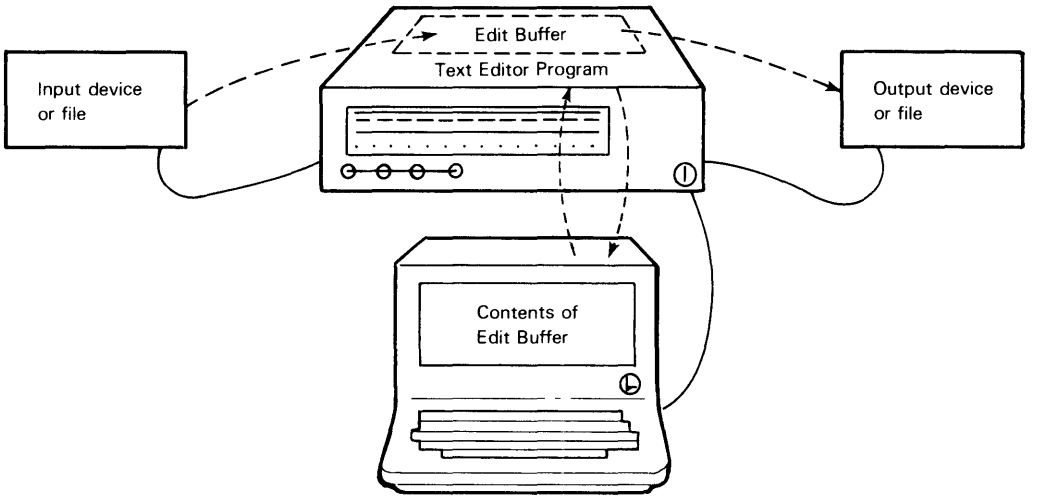

Figure 1.1 File input and output DG-09485

For example:

CTRL-ITHIS IS A TAB.$$

results in:

THIS IS A TAB.

NOTE: The TAB key also puts you into the Insert Text mode.

Mode

When you bring text into the buffer from the input file, Edit operates in either window or page mode. In page mode, your input forms pages (a page consists of all characters up to and including a Form Feed). In window mode (Edit only) you specify an exact number of lines as input, and Edit reads only the specified number from the input file.

You can input any amount from the keyboard to the current buffer. The mode in which you operate makes a difference in the amount you bring into the buffer from the input file.

Correcting Input Mistakes

If you want to delete a line before terminating the command with double Escape, use CTRL-X (↑X). CTRL-X deletes the current line of input. If you have typed more than one line of input without typing a double Escape, you can delete a line at a time with CTRL-X. For example,

```
*CGOOD TEXT$BAD TEXT↑X
*
```

↑X deletes the entire line, and you get the asterisk prompt.

Typing Errors

Use the Delete key (DEL) to delete characters when you make a typing error in a command line. (In this manual we refer to the Delete key, instead of the Rubout key.) Edit deletes the most recently typed character and then the next most recently typed character each time you press it. Edit displays the echo for each deleted character in the order in which you delete it. The display shows the initial text plus the deletions. The deleted characters show in reverse order. Read the letters as a check on your deletions, then continue typing text. See the following example:

THE ABS VALUE

To get rid of ABS, while keeping THE VALUE, position the CP after the S and press DEL three times. When you complete the deletions, your system console displays:

THE ABSSBA VALUE

You can check your deletions by typing:

1T$$

The 1 indicates that you want to see one line. T is the TYPE command, and $$ is the command delimiter. Your console displays:

THE VALUE

To view the entire contents of the buffer, type T$$.

When you use the Delete key to delete all the characters in a command line, Edit gives you the asterisk prompt.

Chapter 2

Sample Editing Session

The text editor, Edit, usually resides on the master device, where it was placed during Real-Time Disk Operating System (RDOS) or Disk Operating System (DOS) system generation. Start Edit at the Command Line Interperter (CLI) *R* prompt with this command:

```
Edit ⟨CR⟩
```

Your console displays the Edit asterisk prompt*.

Edit recognizes only editing commands. RDOS and DOS CLI commands do not work in Edit. To return to the CLI and get the *R* prompt, type:

```
H$$
```

H is the command to leave Edit and the double ESC ($$) is the command delimiter. In Medit, H returns only the master console to the CLI. You cannot leave Medit from a subordinate console. See Chapter 4 for details.

Insertion Command

You must insert the contents of the first buffer page into the system to have text to edit in our sample editing session. To do so, use the insert command described next.

Insert (I)

To insert a string of characters into your buffer, type I, followed by the string you wish to insert. Do not be confused when you find typographical errors in the sample text. They are there for you to correct. Figure 2.1 gives you practice text to put into the edit buffer. Remember to type Edit ⟨CR⟩. To correct your own typographical errors, use the Delete key. Remember that deleted characters show in reverse order on your console. To see the corrected line, type 1T$$.

You must include the insert command at the beginning of a text insertion. If you forget to type I, the double ESC (to end the command) does not enter the text into the buffer. Instead, Edit attempts to interpret the text as a command string, and you get an error message from the program. See Appendix A for error messages.

Character Pointer Commands

There are several character pointer (CP) commands. The sample session does not utilize all of them. For a complete listing, see Table 2.4 at the end of this chapter.

Beginning (B)

To move the CP to the beginning of the buffer, you must issue a B command. For example, to position the CP immediately before the tab in line 1 of the sample, type:

```
B$$
```

B is the command and $$ is the command delimiter.

Jump to the Start of a Line (J)

To position the CP before the first character in a given line, issue the command *n*J, where *n* represents the number of the line to which you want to move.

For example, to move the CP from the end of line 1 of the sample to the beginning of line 5, type:

```
5J$$
```

Move to End of Buffer (Z)

Use Z to move the CP to the end of the buffer. For example, to move the CP from line 1 of Figure 2.1 to follow the Carriage Return (CR) in line 10, type:

```
Z$$
```

Move Past a Number of Lines (L)

To move the CP *n* lines from a current position, issue the command *n*L, where *n* represents the number of lines you want the CP to move. A negative number means to move toward the beginning of the buffer, and a positive number means to move toward the end. For example, to move the CP from the end of the first buffer in the sample (the beginning of line 11) to the beginning of the fourth line, type:

```
-7L$$
```

```
*I  THE FOLLOWING PAGES GUIDE YOU THROUGH A SAMPLE EditING SESSION IN ⟨CR⟩
EDIT. SINCE OUT INTENTION IS TO MAKE THIS PROCESS AS ⟨CR⟩
SIMPLE AND AS PAINLESS AS POSSIBLE, THE TEXT THAT YOU ARE NOW READING WILL ⟨CR⟩
SERVE AS A SAMPLE PROGRAM. ALL COMMANDS IN THE SAMPLE EDITING SESSION ARE ⟨CR⟩
DIRECTED TOWARDS THIS TEXT. ERRORS THAT YOU SEE HERE ARE INTENTIONAL AND ⟨CR⟩
CAN BE CORRECTED BY SIMPLE COMMANDS FROM THE CONSOLE. ONCE THE ⟨CR⟩
TEXT EDITOR HAS BEEN LOADED AND IT DISPLAYS A PROMPT, YOU ⟨CR⟩
ARE READY TO ENTER THE FIRST COMMAND. KEEP IN MIND THAT THE COMMANDS ⟨CR⟩
DISCUSSED HERE ARE COMMON TO EDIT AND MEDIT. IN LATER CHAPTERS, WE'LL ⟨CR⟩
DESCRIBE COMMANDS FOR SPECIFIC OPERATING SYSTEMS. ⟨CR⟩

$$
```

Figure 2.1 Practice text DG-09484

To move the CP to the beginning of a line on which it already resides, type:

```
L$$
```

Search, Change, and Type Commands

Edit provides three search commands as a more sophisticated means for positioning the CP. Search commands position the pointer in the buffer by searching for the string you specify and placing the CP after that string if it is found. The first search command, which is explained below, searches only the current buffer contents. The second two search commands, N and Q, documented later in this chapter, cross page boundaries to search through all text not yet read into the edit buffer.

Search (S)

To search for a character in the current buffer, issue the command S*string*. Edit searches for the string only in text following the present CP position, and repositions the CP immediately after the last character of the first occurrence of the string. For example, in this case, place the CP at the beginning of the buffer by typing:

```
B$$
```

Then type:

```
SPAGES$$
```

The CP is now located directly after the S of PAGES in line 1. If Edit does not find the string before it reaches the end of the buffer, your console displays:

```
STR NOT FOUND
```

Then it positions the CP at the beginning of the buffer and displays the prompt.

Change (C)

To change a string in the current buffer, you issue the command C*string1*$*string2*. This searches for *string1*, deletes it, and inserts *string2*. Edit positions the CP after the last character in *string2*. For example, change the practice text as follows:

```
CGUIDE YOU THROUGH$DEMONSTRATE$$
```

C is the CHANGE command. *GUIDE YOU THROUGH* is *string1* and *DEMONSTRATE* is the new string. With 1T$$ your console displays the corrected text.

```
THE FOLLOWING PAGES DEMONSTRATE A SAMPLE.....
```

The CP moves from its original position to the space following the E in DEMONSTRATE.

If Edit does not find *string* before reaching the end of the buffer, the console displays:

```
STR NOT FOUND
```

and repositions the CP at the beginning of the buffer.

Delete (D)

To delete a number of characters before or after the CP position, issue the command *n*D, where a positive number preceding D deletes that many characters to the right of the CP and a negative number deletes the stated number of characters to the left of the CP.

For example, place the CP after the R of PR OCESS in line 2 of the sample (do this with the search command SPR$$). Type:

```
1D$$
```

This eliminates the extra space and produces:

...TO MAKE THIS PR↑OCESS AS SIMPLE...

Conversely, with the CP at this point, if you type:

-2D$$

your console displays

...TO MAKE THIS ↑OCESS AS SIMPLE...

The CP resides directly after the last deleted character, in the above example, before the O.

Delete Lines

To delete whole lines from the buffer, issue the Kill-lines command *n*K. The *n* may be either a positive or a negative number. For example, 2K deletes the text on the line following the CP position, as well as the next line. If the CP is in the middle of a line, 1K deletes that line up to and including the CR. A negative number like -2 deletes the characters to the left of the CP in the current line plus the previous line.

For example, position the CP at the beginning of line 4 of the sample. To delete lines 4 and 5, type:

2K$$

The CP resides after the last deleted character.

Type on the Terminal (T)

Use the Type command (T) to examine the contents of your edit buffer. For example, to display the first buffer of the sample on your console, type:

T$$

Assuming that you have made no errors thus far, your console should display the contents of the first buffer minus lines 4 and 5. For the sake of continuity, retype (insert) the two lines with I, followed by the text.

To examine a certain number of lines in the buffer, type the command:

*n*T

The *n* represents the number of lines you wish to examine from the CP position. For example, if you wish to examine only lines 1 through 4 of the sample, type:

B4T$$

Neither T nor *n*T has any effect on the location of the CP.

Output Commands

The buffer only contains what has been inserted from the console. No changes are made in a file until the contents of the buffer are written to an output file. Edit provides output commands that you use when you have completed your changes and want to store your file. These are described in the following section.

Get for Writing (GW)

o store the contents of the edit buffer in an output file, use the GW (Get for Writing) command. Create a new filename and use it with the command. The format follows:

GW*outputfilename*$$

This gets the specified output file for writing (output). Call your new file SAMPLE. For example, to specify your current buffer for writing, type:

GWSAMPLE$$

Put Buffer in the Output File (P and PW)

Although you specify SAMPLE as your output file for writing, it is an empty file. To put the edit buffer into output file SAMPLE, use one of two commands. The first, P, enters a file of buffer content length; the second, PW, enters a file of specified length. Type the P command using one of two formats. The first follows:

P$$

Edit, starting from the CP, outputs however many lines you have into SAMPLE and ends the page with a Form Feed. The second format follows:

*n*P$$

In this case, Edit outputs a specific number of lines and ends them with a Form Feed.

To eliminate the Form Feed, use PW in place of P in either of these formats, depending upon whether you want to indicate the number of lines in the buffer:

PW$$ or *n*PW$$

Successive PW commands append material to the same page in the output file.

Get for Reading (GR) and Get and Close (GC)

When you want to output the contents of an input file to an output device (that is, to a line printer or magnetic tape), specify the device name after the GW command. In our

example, we use additional commands to close SAMPLE and print out its contents.

GRSAMPLEGWLPT$$

and

P$GC$$

GR is the Get for Reading command which opens the file SAMPLE but does not read text into it. The section on Input Commands in this chapter explains how GR works. $LPT is the name of the line printer; P is the command to put the edit buffer SAMPLE to the output file. GC is the Get and Close command which closes both the input and output files. The section Get and Close in this chapter presents more on the GC command.

If you make a mistake, Edit sends out an explanatory error message. For a complete listing of error messages and their remedies, see Appendix A.

Clearing the Buffer

Always clear the buffer after outputting its contents; otherwise, the buffer contents remain and get scrambled with succeeding insertions. After implementing the P command, position the CP at the beginning of the buffer with B$$. Tell Edit how many lines to clear with *n*K, followed by the double Escape (ESC).

Type the command in this format:

B*n*K$$

where *n* equals (at least) the number of lines in the buffer. For example, to clear your current buffer, type:

B999K$$

Edit returns the CP to the beginning of the buffer and deletes as many lines as it can, up to 999. To check that the buffer is empty, type:

T$$

The console display is empty.

After clearing the contents of the first edit buffer, insert text into a second buffer, using the I command, followed by the text. Type the practice text, mistakes and all, exactly as shown in Figure 2.2.

To make additions and corrections, use the CP and the commands that enable you to examine, search for, and change text. Table 2.1 summarizes these for your convenience. Continue to practice using them until you feel completely at ease.

Command	Meaning
B	Move CP to start of buffer.
C*string1*$*string2*$	Find *string* 1 and change to *string* 2.
C*string*$$	Delete *string* .
[−]*n*D	Delete *n* characters from CP.
I*string*$	Insert *string* at CP.
[−]*n*J	Jump pointer to beginning of line *n*.
*n*K	Delete *n* lines from CP.
L	Move CP to beginning of line *n*.
*n*L	Move CP *n* lines from current position.
[−]*n*M	Move CP across *n* characters.
T	Type contents of edit buffer.
[−]*n*T	Type *n* lines of edit buffer.
	Move CP across *n* characters.
S*string*$	Search edit buffer for *string*.
*n*W	Set window mode of *n* lines. Reset editor to window mode if in page mode (default mode).
W	Set page mode. Reset editor from window to page mode.
↑X	Delete current command line.
Z	Move CP to end of buffer.

Table 2.1 Basic editing commands

When you are satisfied that you are familiar with the commands discussed so far, put the contents of the second buffer into SAMPLE by typing:

P$$

Clear the buffer by typing:

B50K$$

```
I YOU PROCEED THROUGH THE REXT..PAGE BY PAGE WHEN YOU MODIFY. ⟨CR⟩
IF YOU DISCOVER THAT YOU FORGOT TO CHANGE PART OF A PAGE WHICH ⟨CR⟩
YOU HAVE ALREADY OUTPUTTED. YOU CANNOT FLIP BACKWARDS THROUGH THE ⟨CR⟩
TEXT TO FIND IT. ⟨CR⟩
$$
```

Figure 2.2 The second buffer contents DG-09486

Get and Close (GC)

To close file SAMPLE, use the GC command. GC closes both the input and output files. To recall SAMPLE, you must use an input command called GR Get for Reading) described next.

Input Commands

When a file exists on the disk, as opposed to one you create in the buffer by using an I command and typing text, you must use an input command to get it into the buffer.

Get for Reading (GR)

To re-examine and modify an existing file, type the command in this format:

GR*inputfilename*$$

In this example, type:

```
GRSAMPLE$$
```

This command opens SAMPLE, but does not read text into the buffer.

> **NOTE:** To store modified text, you must open an output file using Get for Writing (GW) with the Get for Reading (GR) command. In the example we did not use GW to open an output file.

Yank in a Page (Y)

To read the first page of text into the buffer, use Y (Yank in a Page). For example, to yank in and examine the first page of SAMPLE, type:

```
Y$T$$
```

To read in the second page, output the first page with P$$ and type another Y command.

> **NOTE:** If you type a Y command without first outputting the buffer contents you have been editing, you will lose the entire contents of that buffer. The Y command clears the buffer and then brings in the next page.

This ends the sample editing session. We urge you to work on the contents of SAMPLE until you fully understand each command. Refer to Table 2.4 for a summary of these commands. Once you have practiced with the text, close the file with GC$$.

Page/Window Mode

Normally, Edit is in page mode, which means that it displays the entire contents of the buffer if you type a T command. The screen of your console may not be able to display the full contents of the buffer. The top lines of the file disappear as they reach the top of the screen.

To avoid this problem, you can set the window to the number of lines that fit on the screen. By setting the window, you set the maximum number of lines that the current edit buffer can hold. Window mode also allows you to edit text, window-length by window-length, without the interruptions of the Form Feeds that separate pages. Instead, you see the Form Feed indicator, CTRL-L, printed within the text at the appropriate places.

If you decide on window mode, set the window before you open a file. The window does not affect the file itself; you can revert to page mode at any time.

Window mode is most convenient for long files, or files with many Form Feeds. You can change the mode from page to window by using *n*W where *n* equals the number of input lines you want in the buffer. For example,

```
25WM$$
```

sets a window length of 25 lines. When you open the file, the first page appears in the length you desire, as do all succeeding pages. You can change the mode at any time while modifying the file. Table 2.2 shows the window mode commands.

Command	Meaning
*n*W	If currently in page mode, reset Edit to window mode.
W?	Display the page/window mode status.
W	Reset Edit from window to page mode.

Table 2.2 Window mode commands

If Edit is already in page mode, you get the error message:

ILLEGAL WINDOW WIDTH

Special Insertion Commands

The following pages describe several commands not explained in the sample editing session.

Insert *n*I and '*n*I

Edit does not accept certain ASCII characters as text characters in an inserted string. For instance, if you try to insert ASCII control characters as text in a text string, they perform a function instead of appearing as characters. You can use the insert code command *n*I$ or '*n*I$ to insert any ASCII character into the buffer at the CP location. For example, to include a control character in the midst of a text string, type the command in this format:

I*string*$*n*I$I*rest-of-string*$$

Edit interprets I*string* as "insert the string as usual," *n*I$ as "interpret the ASCII character identified by number (decimal) before the I as literal text." Edit interprets I*rest-of-string*$$ as "insert the identified special ASCII character between the I and the double Escape command delimiter as literal text."

Alternately, you may use octal numbers in place of *n* in this command format:

I*string*$'*n*I$I*rest-of-string*$$

Edit interprets I*string* as "interpret characters after the I as text";'*n*I$ as "interpret the ASCII character identified by number (octal) as text"; and I*rest-of-string* as "insert all characters, including the identified special ASCII character, after the I and before the double Escape command delimiter as text."

For example, to insert three zeroes, CR, and three slashes in sequence, type:

I000$13I$I///$$

The 13 represents a CR. With 2T$$, the console displays:

000
///

To insert an Escape character ($) into the middle of a sentence as a text character, you may use its octal code in the command with '*n*I as follows:

IA$'33I$ISEPARATES COMMANDS.$$

Edit interprets IA as "insert letter A."; '33I as "insert an Escape character symbol"; and ISEPARATES COMMANDS as "insert the character string that follows." (33 is the octal number for Escape.) Appendix B lists the decimal and octal numbers for the ASCII character set.

Moving the CP (M)

To move the CP *n* characters to the right or left, type the Move (M) command in this format:

*n*M$$

A positive number moves the CP *n* characters to the right, and a negative number moves *n* characters to the left.

For example, assume that the CP resides after the G of FOLLOWING:

FOLLOWING↑

The ↑ stands for the location of the CP. Your console does not display it. To move the CP to the position before the W of the same word, type:

-4M$$

Advanced Modification Commands

In the following section, Nonstop Search, Quick Search, and Macro Command are described.

Nonstop Search Command (N)

Use N only if you have created an output file. To search for a string throughout the entire input file from the current CP position, type the command in this format

N*string*$$

Edit first searches throughout the current edit buffer for *string*. If it does not find *string*, Edit outputs the buffer contents, yanks in the next page, and continues the search, page by page, until it either finds *string* or reaches the end of the file. When Edit finds *string*, it positions the CP immediately after the last character in *string*. If Edit does not find *string* by the time it reaches the end of the file, the console displays

STR NOT FOUND

and the CP resides at the beginning of an empty buffer. For example, place the CP at the beginning of the first page of SAMPLE with B$$ (if the CP is not there already). Type

NREXT$$

Edit searches through the current buffer, outputs it, yanks in the next page, and continues the search until it finds REXT. Then the CP resides after the T in REXT.

The N command is most useful when you wish to examine a large file. It enables you to bypass the tedious search through separate pages.

Quick Search Command (Q)

Use Q as another way to perform a continuous search for a string. It differs from the N command in that it scraps any pages it searches through, instead of outputting pages, one by one, to the output file. For example, assume that the CP resides at the beginning of the third line of the first page of SAMPLE. If you type

```
QPUTTEDYOU$$
```

Edit searches through the contents of the current buffer from the CP position. When it does not find the string (the space is omitted), it yanks in the next page without putting the first page to the output file. This destroys the first page. The search continues until Edit finds the string. If the CP resides at the beginning of the first page of a file and Edit does not find the string, Edit deletes the contents of the entire file.

Macro Command (XM)

Use XM to combine several command instructions within one command. In Edit you can define and execute a macro command. To define the contents of a macro, type the command in this format:

XM*command*$_1$\$*command*$_n$

COMMAND consists of:

[*n*]*code* [*string*\$]

To terminate the macro definition, press the Escape key twice. To execute the macro command, type *n*X, where *n* equals the number of times you wish to execute the macro.

For example, place the CP at the beginning of the first page of SAMPLE and type in the macro shown here:

```
XMSFLIP$-4M$CFLIP$MOVE$1L$$
2X$$
```

Edit interprets XM as "define the macro as"; SFLIP as "search for the string FLIP"; -4M as "move the CP four characters to the left"; CFLIP as "delete FLIP and replace it with MOVE"; 1L$$ as "move the CP to the beginning of the next line"; and 2X$$ as "execute the macro twice." Use these commands to check your work:

```
-3L$$
2T$$
```

Your console displays

IF YOU DISCOVER THAT YOU FORGOT TO CHANGE PART OF A PAGE WHICH YOU HAVE ALREADY OUTPUTTED, YOU CANNOT MOVE BACKWARDS THROUGH THE TEXT

showing that the corrections have been made.

Once you define a macro command, Edit retains it and you can call it for execution by typing *n*X. Edit can handle only one macro definition at a time.

To type out the contents of the current macro command, type:

```
X?$$
```

To delete a macro command, type:

```
XD$$
```

Advanced Output Commands

Read, Eject, Form Feed, and Go Home commands are described in the following section.

Read Out a Page and Read in a Page (R)

Using R combines the attributes of the P and Y commands. Edit outputs a page and clears the buffer when it yanks in the next page.

To output *n* pages and yank in *n* pages, type:

*n*R$$

If you omit the *n* argument, Edit outputs one page and reads in one page. For example, assume the CP resides at the end of the first page of SAMPLE. Type the following:

```
R$$
```

Edit puts out the contents of the first page to the output file and reads in the contents of the second page.

You can use this command even if there is no succeeding page to read. Your console displays this question:

??R??

This means that Edit cannot find another page to read. Edit performs the output and clearing function.

Eject the Remainder of the Input File (E)

To output the contents of a buffer, along with the remainder of the input file, use the E command. Edit follows the procedure outlined in Figure 2.3.

For example, assume that you wish to make a correction in the first line of the file SAMPLE and then copy, as is, the remainder of the input file. Make the correction and type:

E$$

This outputs the contents of the buffer.

Form Feed (F)

To output a Form Feed to the output file, type:

F$$

Use this command when you wish to output a Form Feed without using a P or R command. The contents of the buffer remain unaffected.

Go Home (H)

To return to the RDOS CLI after completing your editing session, type:

H$$

> **NOTE:** When you use Medit, the H command does not return control to the CLI. Instead, it closes out both the input and output files. Chapter 4 describes the procedure for leaving Medit.

Advanced Input Command

The following section discusses Append.

Append a Page to the Edit Buffer (A)

To append a page or window width to the contents of your present buffer, type:

A$$

Edit takes the next page from the input file and appends it to the present contents of the buffer; it then returns a prompt to your console. The CP resides at the beginning of the appended page.

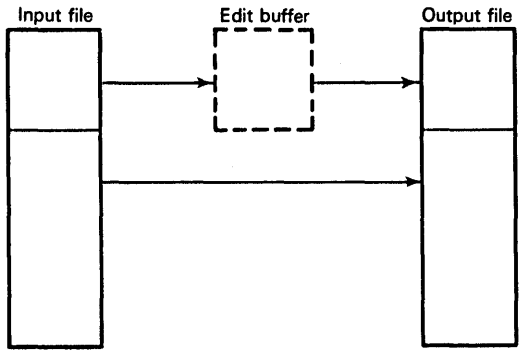

Figure 2.3 E command execution DG-08968

For example, if the edit buffer contains the first page of the sample and you wish to append the second page to the first page, type:

A$$

Local Buffer Commands

Edit maintains three registers with which you can obtain information on current buffer contents. Use the name of each register as a command when you want the console to display the numeric contents of the buffer. Table 2.3 describes the commands.

Command	Meaning
:	Print the number of lines in the current edit buffer.
.	Print the number of the line on which the CP resides.
=	Print the number of characters in the current edit buffer.

Table 2.3 Special local buffer commands

Terminate all special commands by pressing ESC twice ($$).

Command Review

By this time, you have probably noticed that there are two forms of command structure: the single command and the command string. The following text lists the formats for each command structure.

A single command is:

[n]code [string$]

[n] represents an optional integer between -32,768 and 32,767 in RDOS.

code represents a letter, two letters, or special graphic.

string represents an optional string of characters terminated by an Escape.

The only exceptions to this are the C (Change) and UR (Rename File) commands, which have two strings following the code and use the formats shown next:

Cstring1$string2$$

URfile1$file2$$

A command string consists of two or more commands stacked for sequential execution by Edit. You delimit each single command from the next by a single Escape.

A command string is:

[n] code [n] code [string$] [n] code [string$]

[n] represents an optional integer between −32,768 and 32,767.

code represents a letter, two letters, or a special graphic.

string represents an optional string or characters, terminated by an Escape.

You can carry a command string over to the next line by pressing CR between two commands. Two consecutive Escapes terminate the command string.

See Table 2.4 for a summary of text editing commands.

Mnemonic	Command Name	Meaning
A	Append	Append page or window width to current buffer.
B	Buffer	Move CP to the beginning of the buffer.
C*string1*$*string2*	Change	Find *string1* and change to
E	Eject	Put contents of buffer plus remainder of input file into output file.
F	Form Feed	Put Form Feed to output file (does not affect current buffer).
GC	Global Close	Closes input and output files.
GR*infile*	Get for Reading	Open input file *infile*.
GW*outfile*	Get for Writing	Open and write output file *outfile* (cannot already exist).
H	Home	Return to CLI (master console only in Medit).
I*string*	Insert	Insert *string* at CP.
*n*D	Delete	Delete *n* characters from CP.
*n*J	Jump	Move CP *n* lines from beginning of buffer.
*n*K	Kill	Delete *n* lines from the CP.
*n*L	Line Move	Move CP *n* lines forward or back from present position.
*n*M	Move Character Pointer	Move CP *n* characters.
R	Read	Output edit buffer and read next page.
*n*R	Read	Put *n* pages to output file and yank *n* pages into buffer. If *n* argument is not given, output one page and yank one page.
N*string*	Nonstop Search	Search for *string*. If not found, output current buffer and continue search through input file, outputting the file until string is found. (Position CP after last character of first occurrence of *string*.)
P	Put	Put contents of edit buffer to output file and end with a Form Feed.
PW	Put	Put contents of edit buffer without Form Feed.
Q*string*	Quick Search	Search for *string*; if not found, continue search through all input files but output nothing. (Overwrite and destroy pages leading up to *string*.)
S*string*	Search	Search edit buffer for *string*; position CP after last character of first occurrence of *string*.
T	Type	Type out contents of edit buffer.
XM*com1*$*com2*$$	Macro Command	Define current macro as commands *com1 com2*.
*n*X	Execute Macro	Execute current macro *n* times.
XD	Macro Delete	Delete macro.
X?	Current Macro	Type out current macro.
Y	Yank	Yank (read) an input page into buffer.
Z	End of buffer	Move CP to end of buffer.

Table 2.4 Text editing command summary

Chapter 3

More Editing in RDOS and DOS

This chapter presents commands that combine the functions of other commands; it also contains special control and number register commands. For your convenience, Table 3.1 shows the generalized steps you follow to start and end the Edit text editor. We assume you have started Edit and are ready to practice using these commands.

Command	Description
R	The *R* indicates that you are in the Command Line Interpreter, the system's interface with individual programs.
R Edit (CR)	Type Edit followed by a Carriage Return (CR) to call the editor. When you see an asterisk on the console, you have started Edit.
*xxx$$	Specify the name of the file you wish to access. The file might already exist or you may be creating a new file. Terminate the command with a double Escape.
*xxx (CR) xxxx$$	When Edit issues another prompt, issue your insertion and/or modification commands. Repeat this cycle until you complete your editing session.
*xx$$	When you finish the file, close it with an output command.
*H$$	If you want to return to the CLI, issue an H command.

*xxx means insertion, modification, or input/output commands.

Table 3.1 Starting and terminating Edit from the CLI

Input Commands

Compound input commands consist of a U with another letter, as explained in this section.

Create a New File (UN)

The UN command combines the GR and GW commands. To create a file in Edit, type the U command in this format:

UN*filename*$$

Edit creates an input file with the specified name and classifies it as permanent for the duration of the editing session. This means that no one can delete it while you are editing it. Edit also creates and opens a temporary file, FILENAME.SC, for output. Because of its temporary status, you do not have access to the .SC file. For an example of how UN works, create a permanent file named SAMPLE and open SAMPLE.SC for output. Type:

UNSAMPLE$$

> **NOTE:** Close files opened by a UN command with another one of the U commands, UE, US, UC, or UH, discussed in the section called Output Commands in this chapter.

Editing an Existing File (UY)

To edit an existing file, type the U command in this format:

UY*filename*$$

With UY$$, Edit executes three commands (GR, Y, GW) which do the following:

- Open an existing file for input (GR),
- Yank the first page from the input file to the edit buffer (Y),
- Create a new output file with the .SC extension (GW).

For example, if you type:

UYSAMPLE$$

Edit opens SAMPLE for input, opens SAMPLE.SC for output, and yanks the first page of SAMPLE into the edit buffer.

> **NOTE:** After you open a file with a UY or UN command, use GR*filename* to read text in from another file. When you have finished editing, use UE, UC, or US to close the file and do whatever else you want to do with it. Edit uses the original input filename for renaming purposes.

When you start Edit in the Command Line Interpreter (CLI), you can specify the desired filename with the Edit command. If you do this, you can omit the UY command.

```
R
Edit SAMPLE ⟨CR⟩
```

is equivalent to

```
R
Edit ⟨CR⟩
*UYSAMPLE$$
```

R is the CLI prompt; Edit ⟨CR⟩ starts the Edit program. The console displays the Edit prompt (*). The command UYSAMPLE instructs Edit to open the file SAMPLE.

Deleting and Renaming Files

The commands for deleting and renaming files are described in the following sections.

Delete a File (UD)

To delete a specific file, type the U command in this format:

UD*filename*$$

For example,

```
UDSAMPLE$$
```

deletes the file SAMPLE.

Rename a File (UR)

To rename a file, type the UR command in this format:

UR*filename1*$*filename2*$$

where *filename1* is the current name of the file and *filename2* is the new name you wish to give to the file. For example,

```
URSAMPLE$PROGRAM$$
```

changes the name of file SAMPLE to PROGRAM.

Remove Protective Attributes (UZ)

Certain protective attributes can be given to files through the CLI. To remove a CLI-assigned permanence attribute from an input file, type the U command in this format:

UZ*filename*$$

You can delete the file since you have eliminated its permanence.

Output Commands

Compound output commands use U or G with another letter or symbol.

Close and Update a File (UE)

The UE command:

- Copies the remaining input file (E),
- Closes the input and output files (GC),
- Deletes the input file,
- Renames the output file with the input filename.

For example, after typing the UY command, assume that you make a correction in the first line of SAMPLE:

```
UYSAMPLE$$
.
. (Correction)
.
UE$$
```

The UE command copies the rest of SAMPLE to SAMPLE.SC, closes SAMPLE and SAMPLE.SC, deletes SAMPLE, and renames SAMPLE.SC to SAMPLE.

Close a File, Update, and Save Backup (US)

Use the US command when you want to save both the input and output (original and modified) files. Backup is indicated by the extension .BU. To copy the full input file and to close the input and output files, type:

```
US$$
```

US does these tasks as well:

- Renames the input file to *filename*.BU,
- Renames the output file to the input file's name,
- Restores the attributes of the input file,
- Deletes the previous .BU (backup) file.

For example:

```
UYSAMPLE$$
.
. (Modifications)
.
US$$
```

copies the rest of SAMPLE to SAMPLE.SC, closes SAM-

PLE.SC, renames SAMPLE to SAMPLE.BU and SAMPLE.SC to SAMPLE, and restores the attributes to SAMPLE.

Close, Update, and Rename a File (UC)

The UC version of the U command offers you a means to save both input and output files and to rename the open (input) file. The UC command:

- Closes the input and output files,
- Restores the attributes of the input file,
- Renames the output file with a specified name.

Type the command in this format:

UC*filename*$$

filename is the new name for the output file. For example:

```
UYSAMPLE$$
  .
  . (Modifications and output commands)
  .
UCEXAMPLE$$
```

closes SAMPLE and SAMPLE.SC, restores the attributes of SAMPLE, and renames SAMPLE.SC to EXAMPLE.

> **NOTE:** Remember to issue an output command (P, E, or R) to copy the contents of your input file to the output file before issuing the UC command. Otherwise, Edit will not be able to recognize the output file.

Close a File (UH)

To close the input and output files opened by UY or UN, type:

UH$$

For example,

```
UYSAMPLE$$
  .
  .
  .
UH$$
```

closes SAMPLE and SAMPLE.SC.

> **NOTE:** In Medit, the H command performs the function that UH does in Edit.

Display Current File Names (U?)

To display the names of the current input and output files on your console, type:

U?$$

Close Output File and Open New One (GO)

To close a current output file opened by GW and to open a new output file, type:

GO*filename*

where *filename* is the new file you wish to open. For example,

```
GRSAMPLE$GWEXAMPLE$$
  .
  .
  .
GOPROGRAM$$
```

closes EXAMPLE and opens PROGRAM.

For detailed explanations of G commands, refer to the sample editing session in Chapter 2.

Special Control Commands

Special control commands, CTRL-C, CTRL-A, and CTRL-2, are described in the following section.

Control Change (CTRL-C)

CTRL-C is a Real-Time Disk Operating System (RDOS) break character that terminates Edit and returns control to the CLI. If you issue a CTRL-C command before you close out your file, Edit deletes any open file. Medit does not recognize CTRL-C except from the master console.

Abort this Command (CTRL-A)

CTRL-A is also an RDOS break character. In Edit, CTRL-A interrupts the currently-executing command. CTRL-A has no effect unless Edit is executing a command. While executing a command, Edit stores command components in its own command buffer. Typing CTRL-A during command execution scraps this buffer, aborts the command, and evokes the Edit prompt (*).

If the executing command string exceeds 256 characters, your console displays:

SAVE COMMAND BUFFER YES (1) OR NO (2)?

If you respond 1$$, which means yes, your console displays the message:

ENTER FILENAME

Edit writes the contents of the command buffer to the file you create (which cannot already exist) and gives you the asterisk prompt (*).

When you interrupt an ongoing process with CTRL-A, examine the text to decide which commands have already been executed.

> **NOTE:** Medit recognizes CTRL-A only from the master console.

Match One Unknown Character (CTRL-Z)

Occasionally, you want to find a string of a specific length, which contains a couple of characters you know and some you do not know. For each CTRL-Z you insert in a search or change command, Edit assumes any single character. For example, assuming you have at least one word to fit the description in your text, find the first occurrence of a 5-letter word, beginning with M and ending with Y. You would type:

SMCTRL-ZCTRL-ZCTRL-ZY$$

SM means search for a string that begins with M. Using CTRL-Z three times stands for any characters, Y is the last character in the string, and $$ means execute the command.

CTRL-Z can help you to find strings or formulas that contain variables (if you know how long the variables are). For example, assume that you want to find the occurrence of a formula, A + *n* = E, where *n* is two characters long. You would type:

SA + CTRL-ZCTRL-Z=E$$

The command SA + initiates a search for a string that begins with A + , followed by two-digit variable, and ending with =E.

RDOS/DOS Number Register

You can use the RDOS number register as a counter to:

- Number pages or lines within a macro command
- Count the occurrence of something throughout your text
- Display a set value contained by the register at a specified place in the text

The number register commands are shown in Table 3.2:

Command	Description
n#	Reset the register to *n*, where *n* may be from one to five digits in the range of 0 to 65,535.
#+	Increment the register by one.
#–	Decrement the register by one.
#?	Type out the contents of the register.
#O	Output the contents of the register as a 5-digit, unsigned integer to the edit buffer at the current location of the CP.
n#!...$!...$$	When the register does not match the *n* argument, skip the command characters up to the next Escape followed by an exclamation point.

Table 3.2 Number register commands

Register Example 1

The following commands permit you to edit a program containing 200 lines and to set up sequential line numbers throughout the program. The example,

0#$$

resets the number register to zero. Do this outside the macro, because you need to set it only once and because the register is a location in memory rather than a command.

BXM#+#O1L$$

Start from the beginning of the buffer, define a macro that increments the register by one (#+), prints the register contents at the CP location in the text (#O), and moves one line forward after the number has been printed (1L).

Type:

B200X$$

to execute the macro 200 times from the beginning of the buffer.

Register Example 2

This example shows how to insert a Form Feed at every 50th line of a sample program. To do this, you must use the final form of the register command:

```
0#$$
```

to reset number register to zero.

```
XM50#!ICTRL-L$0#$!1L#+$$
```

If the register does not match 50, skip the command characters up to the next Escape followed by an exclamation point. Move the CP to the beginning of the next line and increment the register by one. Once the contents of the register match the number 50, insert a Form Feed and reset the register to zero.

The following example:

```
B0#$$
```

moves the CP to the beginning of the buffer with 0 in the register. The following example:

```
400X$$
```

executes the macro 400 times.

Table 3.3 summarizes RDOS/DOS file, control, and number register commands.

Mnemonic	Command Name	Meaning
#+	Number Register	Increment the register by one.
#−	Number Register	Decrement the register by one.
#?	Number Register	Type out the contents of the register.
#O	Number Register	Output the register to the edit buffer at current loation of CP.
CTRL-A		Terminate command execution; scrap command buffer (ignored in Medit).
CTRL-C		Abort editor, return control to CLI; delete any open file (ignored in Medit).
CTRL-Z		Search for string matching a character.
CTRL-X(cancel)		Scrap current input on terminal.
GC	Get and Close	Close I/O files without data transfer.
GO*filename*	Close Output File and Open New One	Close current output file and open new output file.
GR*filename*	Get for Reading	Get input file for reading, closing the previous input file if any.
GW*filename*	Get for Writing	Create and open a new file for writing.
H	Home	Sign off and return control to the CLI.
n#	Number Register	Reset the number register to *n*.
n#!...$!...$$		If the register does not match *n*, skip the command characters up to the next Escape followed by an exclamation.
*n*W	Window	If currently in page mode, set mode to window mode with width of *n* lines.
U?	Display Current Filenames	Type I/O filenames.
UC*filename*	Close, Update, and Rename a File	Close I/O files, restore attributes of input file, and rename output file to *filename*.
UD*filename*	Delete a File	Delete *filename*. The file must be closed before deletion can occur.
UE	Close and Update File	Put remaining input file to output file, close I/O file, and rename output file with original input filename.
UH	Close a File	Close I/O files, restore attributes.
UN*filename*	Create a New File	Create a permanent file and open it for input; open *filename*.SC for output.
UR*filename1*$*filename2*	Rename a File	Rename *filename1 filename2*.
US	Close a File, Update, and Save Backup	Put remaining input file to output file, close I/O files, restore attributes of input file, change input filename to *filename*.BU, and rename output file with original input filename.
UY*filename*	Edit an Existing File	Make *filename* permanent, open it for input, and yank in the first page or window frame. Create and open *filename*.SC for output.
UZ*filename*	Remove Protective Attributes	Remove CLI-assigned permanent attribute of a file so it can be renamed or deleted.
W?		Display page/window mode status.
W		Reset editor from window to page mode.

Table 3.3 Edit file, control, and register commands

Chapter 4

Loading Procedures Under RDOS and DOS

This chapter has two primary sections: one on the Text Editor Edit and one on the Multiuser Editor (Medit). Each section describes loading for background operation, unmapped foreground operation, and mapped foreground operation. The Medit section also describes operating procedures. If you are running Disk Operating System (DOS), read only the next three paragraphs.

The Real-Time Disk Operating System (RDOS) Text Editor (Edit) and the Multiuser Editor (Medit) are supplied on magnetic tape or diskette with every RDOS system.

The DOS Text Editor (Edit) is supplied on diskette with each DOS system. Both editors are part of the RDOS support package, and Edit is part of the DOS package. Usually, the person who generates an RDOS/DOS system loads the editors as part of the system; this is an option. If they are loaded during RDOS SYSGEN, they are on the main portion of the disk that holds RDOS. This portion is called the primary partition of the master disk. If either (or both) are not loaded during RDOS SYSGEN, you can find them on magnetic tape or diskette and load them with the RDOS Command Line Interpreter (CLI), as described below.

If loaded during DOS SYSGEN, Edit is on the system diskette. If it is missing, find the proper DOS utilities diskette and load it as described below.

Most RDOS/DOS users can use Edit simply by typing its name from the CLI, as described in Chapters 2 and 3. Users in other directories (distinct from the primary partition) can easily link to the editor in its original directory. For more on this, see the LINK command in the *RDOS/DOS Command Line Interpeter* (DGC No. 069-400015.)

Edit and Medit were designed for execution in background memory. However, this chapter tells you how to execute either of them in the foreground. Also, Medit is a multiuser system, and requires special attention.

When you load either editor program, you can place it in whatever directory you want on disk. Generally, access to it is easier if you place it on the primary partition of a disk. The *RDOS/DOS Command Line Interpreter* describes directory hierarchy in RDOS. The actual location on the disk (primary partition, subdirectory) is a function of where it is put in a given installation.

Edit is supplied in two versions: the executable version and the relocatable binary version. You need the relocatable binary version *only* if you plan to operate the editor in an unmapped foreground. The filenames of Edit are Edit.SV (executable version) and Edit.RB (binary version).

Medit is supplied as a relocatable binary only: Medit.RB. You *must* load Medit.RB using the relocatable loader (RLDR) so that it can be executed.

Unmapped Foreground Operation

Before you can execute the editor in an unmapped foreground, you must configure it for operation in foreground memory. You do this by specifying starting normal relocatable location (NREL) and zero relocatable location (ZREL) address information in the RLDR command line, which you must execute on Edit.RB (you cannot use the .SV version of this). If Edit.RB is not on disk, load it from either your RDOS support software or the tape described above.

You must also have a foreground console to communicate with the foreground editor. For details on running an unmapped background, refer to *RDOS System Reference* (DGC No. 093-400027) and the RLDR and EXFG command in the *RDOS/DOS Command Line Interpreter* (DGC No. 069-400015).

In the RLDR command, you specify the starting NREL address with the local /F switch and the starting ZREL address with the local /Z switch. RLDR allocates NREL portions of memory in multiples of 400_8 from location 400_8; therefore, the address you specify in /F is always rounded (if necessary) to an integer multiple of 400_8. Edit requires four words of ZREL space. Foreground ZREL is allocated by counting backwards from 377_8. Make sure that your configuration leaves enough space in background memory for useful programs to operate, while the editor runs in the foreground. The editor requires about 6300_8 words, and the edit buffer should have about 2000_8 words.

The following sample RLDR command produces a foreground version of the editor.

RLDR 20016/F 374/Z Edit Fedit.SV/S 6/C 2/K

This command sequence produces Fedit.SV (the F distinguishes it from Edit.SV, which runs in the background). The editor has all memory above 20016 to the beginning of the RDOS buffers in which to operate; its page zero has addresses from 374_8 to 377_8 (start of the background's User Status Table).

The sample command specifies six I/O channels and two tasks for the editor; these are required. It also leaves the background about 7.5 Kbytes in which to operate. This suffices for many programs.

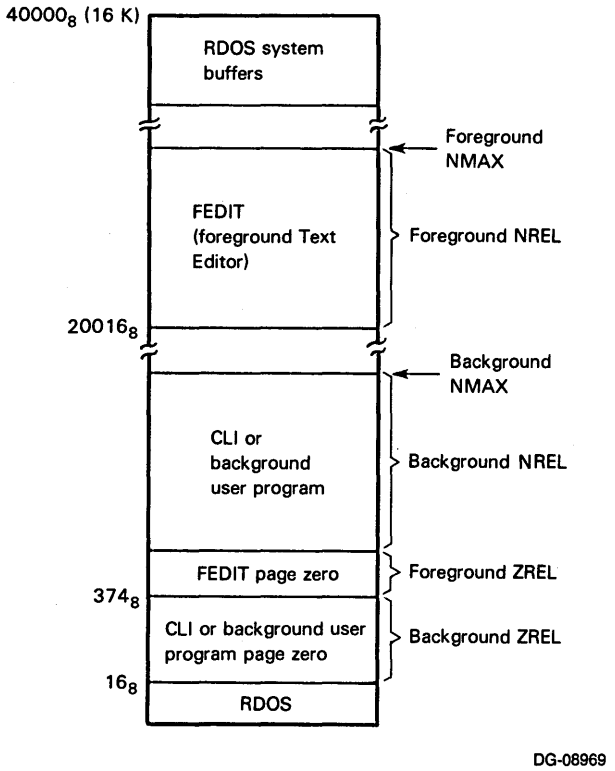

Figure 4.1 Unmapped foreground/background memory

The editor cannot operate in any portion of memory unless it has enough room. You can determine how much space it requires by examining file Edit.SV with the Octal Editor. (Refer to OEdit in the *RDOS/DOS Command Line Interpreter* (DGC No. 069-400015).

You can now execute the editor in the foreground. To load a foreground save file from the disk into core and execute it in the foreground, use the CLI command EXFG as follows:

EXFG *[/E] filename* ⟨CR⟩

The /E switch specifies that the foreground and background program receive equal priority. (If you omit /E, the foreground gets priority.)

For example:

EXFG/E Fedit ⟨CR⟩

executes the editor in the foreground. If RDOS cannot load the foreground editor without overwriting the CLI in the background, RDOS does not load the editor, and you receive an error message.

If the foreground text-editing console is operating online, the console displays the prompt after you type the EXFG command. The CLI prompt should return to the background console.

To terminate the foreground editor, type CTRL-F from the background console. You lose any open edit buffer in the foreground when you terminate the foreground. The H$$ command from the foreground terminal should also terminate the editor.

Mapped Foreground Operation

In a mapped system, you can execute the original version of the text editor, Edit.SV, directly in the foreground. As with an unmapped system, you must have a foreground console to communicate with the foreground editor. You also have two CLI commands to help apportion memory to the foreground and background programs.

These commands are GMEM, which means "get the current memory allotment for each ground," and SMEM, which means "set the allotment for each ground."

Check the memory requirement of Edit.SV with the OEdit utility, as described in the *RDOS/DOS Command Line Interpreter* (DGC No. 069-400015); then give the foreground at least as many 1K blocks as Edit requires. (Usually, four or five 1K blocks, including the block for the edit buffer, are necessary.)

You can then execute the original version of the editor, Edit.SV, in the foreground by typing:

EXFG [/E] Edit ⟨CR⟩

The /E switch specifies equal priority; if you omit it, the foreground has priority over the background.

If the foreground text-editing console is operating online, the prompt (*) should appear on it after you type the EXFG command. The CLI prompt should return to the background console.

To terminate the foreground editor, type CTRL-F from the background console. Any edit buffer open in the foreground editor is lost when you terminate the foreground.

Loading and Managing Medit

Medit runs only under RDOS. This section describes how to load and execute Medit in three environments: any RDOS background, the unmapped foreground, and the mapped foreground.

Because Medit is supplied only as a relocatable binary file, you must make it into an executable save file using the relocatable loader (RLDR) before you can execute it. When you execute the Medit program, you specify a number of consoles which will run it; a user can then edit on each console. Available memory space is divided equally among the consoles, for use as edit buffers. Medit itself requires about 6800_8 words of NREL memory and 13_8 words of ZREL memory.

To use Medit, your system must have an asynchronous line multiplexor (ALM, ULM, or QTY) and at least two terminals. If you have enough multiplexors and memory, Medit supports up to 20 text-editing terminals. You can load and call Medit from any CLI console. The console from which you evoke Medit becomes the master console, and you can terminate Medit and return to the CLI only through that console. From a subordinate console, Medit has the same commands as Edit, with the following exceptions:

- CTRL-A is ignored.
- CTRL-C is ignored.
- On *all* Medit consoles including the master, H$$ does not return control to the CLI. H, in Medit, is the equivalent of the command UH$$ (Close I/O File and Restore Attributes). CTRL-F returns control to the CLI.

Mapped and Unmapped Operation

You must make Medit into an executable save file using RLDR before you can execute it.

Since Medit runs multiple tasks (multiple terminals), you must give RLDR task and I/O channel information. Type the RLDR command in this format:

RLDR Medit *channels*/C *tasks*/K [*newname*/S] ⟨CR⟩

The variable *channels* is the octal number of I/O channels allowed for Medit. The minimum value is three times the number of text-editing terminals plus two.

The variable *tasks* is the number of tasks allowed for Medit. The minimum value is the number of text-editing terminals plus one.

The variable *newname* will rename the Medit save file to whatever filename you choose. This is optional; if you omit it, Medit.SV is the save filename. If you choose a new name, use it to evoke Medit.

The CLI responds with the prompt R when loading is complete.

To start Medit from the CLI (from the proper directory), type:

Medit *terminals* [*clock-units*] ⟨CR⟩

(If you chose *newname* in the RLDR command, type *newname* instead of Medit.)

The variable *terminals* is the number of text-editing terminals for Medit to support. The first terminal is number 0; if you specify 8, for example, terminals 0 through 7 are supported.

The variable *clock-units* is optional. It is the number of RDOS real-time clock units after which task rescheduling will be forced (Medit will be suspended).

If all consoles are online, each displays The CLI prompt (*).

To terminate Medit and return control to the CLI, type CTRL-A or CTRL-C at the master console. An open edit buffer on any console is lost when you terminate Medit.

Unmapped Foreground Operation

Running an interactive program like Medit in an unmapped foreground demands special considerations. See the *RDOS System Reference* (DG No. 93-400027), and the dictionary of *RDOS/DOS Command Line Interpreter* (DG No. 069-400015).

In the RLDR command, you will need to specify starting foreground NREL and ZREL addresses for the foreground Medit, as described previously in the Edit section of this manual in Unmapped Foreground Operation. You will also need to specify a number of channels and tasks, and you should specify a new name, to distinguish the foreground Medit from any background Medit you might want to use. The RLDR command format is:

RLDR Medit *lowest-NREL-address*/F ↑
lowest-ZREL-address/Z *channels*/C *tasks*/K ↑
[*newname*/S]

The variable *lowest-NREL-address* in octal should leave enough room for Medit to run, yet allow useful background programs room to execute beneath it. Medit requires slightly less than 10000_8 words, and each editing terminal should have at least 2000_8 words for an edit buffer (this allows a page size of 2,048 characters per terminal).

The variable *lowest-ZREL-address* (octal) should be no higher than 360_8 or so; Medit requires 13_8 words of ZREL memory.

The variable *channels* is the octal number of I/O channels allowed for Medit. The minimum value is three times the number of text-editing terminals plus two.

The variable *tasks* is the octal number of tasks allowed for Medit. The minimum value is the number of text-editing terminals plus one.

The variable *newname* is optional; it names the Medit save file to whatever filename you specify. We recommend giving the unmapped foreground Medit a different name (for example, FMEDIT).

↑ is the RDOS line continuator. When a command exceeds the line width for your console, you use ↑ as a command continuator. The continuator prevents the first part of a 2-line command from being executed with a Carriage Return (CR). The CLI responds to the command string with an *R* prompt when loading is complete.

If the foreground text-editing consoles are online, the prompt (*) appears on each of them; the CLI prompt (*R*) appears on the background console.

Terminate Medit from the background console by typing CTRL-F.

On any console an open edit buffer is lost when you terminate Medit.

Mapped Foreground Operation

To load, use the same RLDR command that you would for the background. Before starting Medit in a mapped foreground, you must be sure that the foreground size is greater than or equal to four plus the number of terminals times the buffer size of each terminal (4 + *n*T × B *size*). The buffer size per terminal is measured in 1K blocks (1,024 words); Medit occupies slightly less than four blocks. A one-block buffer for each editing terminal allows a page size of about 2K characters per terminal (two characters per word).

To determine how much memory is in the system, type the CLI command:

GMEM ⟨CR⟩

which displays the number (in decimal) of 1024-word blocks in background and foreground:

BG: background blocks FG: foreground blocks

If your version of Medit requires more or fewer 1K blocks to operate, you can change the memory allotments with the SMEM command. The format of the SMEM command is:

SMEM *background-blocks* ⟨CR⟩

All blocks not allocated for the background go to the foreground.

Once you have loaded the foreground Medit using RLDR, set memory allotment, and made the checks discussed above, start the program with a command similiar to the following:

EXFG[/E]Medit*terminals [clock-units]* ⟨CR⟩

The /E switch specifies equal priority; if you omit it, the foreground will have priority over the background.

If you chose *newname* in the RLDR command, type *newname* instead of Medit.

The variable *terminals* is the decimal number of text-editing terminals for Medit to support. The first terminal is number 0; if you specify 8, for example, terminals 0 through 7 will be supported.

The variable *clock-units* is optional. It is the number of RDOS real-time clock units after which task rescheduling is forced (Medit is suspended).

If the foreground text-editing consoles are online, each displays the prompt (*). The background console displays the CLI prompt.

You can terminate Medit from the background console by typing CTRL-F.

> **NOTE:** An open edit buffer on any console is lost when you terminate Medit.

Appendix A

Error Messages

While you are editing, various errors may occur. Possible Edit and Medit error messages and their meaning are shown below. Some error messages occur only in specific operating environments; we list those appropriate environments in parentheses. Other error messages can occur in any operating environment.

COMMAND BUFFER FULL EXECUTING COMMAND

Command string exceeds capacity of edit buffer.

TEXT BUFFER FULL DURING INSERT

Insert string exceeds capacity of edit buffer.

TEXT BUFFER FULL WHILE READING

You tried to append a page when the buffer is full. During a Read (R) or Append (A), the buffer capacity is exceeded. The editor has read in or appended as much of a partial page as it could.

FILE CAN'T BE USED FOR INPUT

You attempted to read a read-protected file.

FILE CAN'T BE USED FOR OUTPUT

You attempted to write to a write-protected file.

FILENAME IN USE

You attempted to create an output file when that filename already exists in the file directory.

ILLEGAL FILENAME

Filename does not conform to a legal operating system filename.

ILLEGAL WINDOW WIDTH

You tried oto change the editor back to page mode, using the W command, when it is already in page mode.

MACRO ERROR

The specified macro is undefined or recursive.

NO OUTPUT FILE

You issued an output command before specifying an output file.

NO SUCH FILE

You specified an input file which does not exist.

OUTPUT ALREADY ACTIVE

You tried to Get for Writing (GW) an output file that has not been closed and is still active.

PARITY ERROR IN LINE n

During a read, a parity error occurred in line *n*. On the console, the character in error will be shown as two quotation marks (" ").

STR NOT FOUND

The editor could not find your string. Check that you inserted spaces properly. The CP is positioned at the beginning of the buffer.

??COMMAND STRING

The editor cannot understand or cannot execute your command. It outputs the remainder of the string to which the message refers. If this message occurs and the command buffer contained at least 256 characters, the following message is printed, and you may follow the procedure outlined below to recover your command buffer.

SAVE COMMAND BUFFER YES (1) OR NO (2)?

The command buffer contained at least 256 characters when either an illegal command or an RDOS interrupt was detected. If you respond with anything other than 1 ⟨CR⟩ the editor prints the message:

ENTER FILENAME

This instructs you to specify a new (original) output filename. The editor then writes the contents of the command buffer to the specified file. If the specified filename is not valid, the editor reprints the message:

ENTER FILENAME

If you respond by pressing the CR twice, then the editor deletes the buffer and reissues the prompt, waiting for a new command.

Appendix B

ASCII Character Set

DECIMAL	OCTAL	HEX	KEY SYMBOL	MNEMONIC
0	000	00	↑@	NUL
1	001	01	↑A	SOH
2	002	02	↑B	STX
3	003	03	↑C	ETX
4	004	04	↑D	EOT
5	005	05	↑E	ENQ
6	006	06	↑F	ACK
7	007	07	↑G	BEL
8	010	08	↑H	BS (BACKSPACE)
9	011	09	↑I	TAB
10	012	0A	↑J	NEW LINE
11	013	0B	↑K	VT (VERT.TAB)
12	014	0C	↑L	FORM FEED
13	015	0D	↑M	CARRIAGE RETURN
14	016	0E	↑N	SO
15	017	0F	↑O	SI
16	020	10	↑P	DLE
17	021	11	↑Q	DC1
18	022	12	↑R	DC2
19	023	13	↑S	DC3
20	024	14	↑T	DC4
21	025	15	↑U	NAK
22	026	16	↑V	SYN
23	027	17	↑W	ETB
24	030	18	↑X	CAN
25	031	19	↑Y	EM
26	032	1A	↑Z	SUB
27	033	1B	ESC	ESCAPE
28	034	1C	↑\	FS
29	035	1D	↑]	GS
30	036	1E	↑↑	RS
31	037	1F	↑—	US

DECIMAL	OCTAL	HEX	KEY SYMBOL
32	040	20	SPACE
33	041	21	!
34	042	22	" (QUOTE)
35	043	23	#
36	044	24	$
37	045	25	%
38	046	26	&
39	047	27	' (APOS)
40	050	28	(
41	051	29	)
42	052	2A	*
43	053	2B	+
44	054	2C	, (COMMA)
45	055	2D	-
46	056	2E	. (PERIOD)
47	057	2F	/
48	060	30	0
49	061	31	1
50	062	32	2
51	063	33	3
52	064	34	4
53	065	35	5
54	066	36	6
55	067	37	7
56	070	38	8
57	071	39	9
58	072	3A	:
59	073	3B	;
60	074	3C	<
61	075	3D	=
62	076	3E	>
63	077	3F	?
64	100	40	@

DECIMAL	OCTAL	HEX	KEY SYMBOL
65	101	41	A
66	102	42	B
67	103	43	C
68	104	44	D
69	105	45	E
70	106	46	F
71	107	47	G
72	110	48	H
73	111	49	I
74	112	4A	J
75	113	4B	K
76	114	4C	L
77	115	4D	M
78	116	4E	N
79	117	4F	O
80	120	50	P
81	121	51	Q
82	122	52	R
83	123	53	S
84	124	54	T
85	125	55	U
86	126	56	V
87	127	57	W
88	130	58	X
89	131	59	Y
90	132	5A	Z
91	133	5B	[
92	134	5C	\
93	135	5D	]
94	136	5E	↑ OR ^
95	137	5F	← OR —
96	140	60	` (GRAVE)

DECIMAL	OCTAL	HEX	KEY SYMBOL
97	141	61	a
98	142	62	b
99	143	63	c
100	144	64	d
101	145	65	e
102	146	66	f
103	147	67	g
104	150	68	h
105	151	69	i
106	152	6A	j
107	153	6B	k
108	154	6C	l
109	155	6D	m
110	156	6E	n
111	157	6F	o
112	160	70	p
113	161	71	q
114	162	72	r
115	163	73	s
116	164	74	t
117	165	75	u
118	166	76	v
119	167	77	w
120	170	78	x
121	171	79	y
122	172	7A	z
123	173	7B	{
124	174	7C	|
125	175	7D	}
126	176	7E	~ (TILDE)
127	177	7F	DEL (RUBOUT)

Figure B.1 ASCII character set

DG-05495

Index

D

E

F

G

H

I

J

L

M

N

O

P

Q

R

S

T

U

W

X

Y

Z

DG OFFICES

NORTH AMERICAN OFFICES

Alabama: Birmingham
Arizona: Phoenix, Tucson
Arkansas: Little Rock
California: Anaheim, El Segundo, Fresno, Los Angeles, Oakland, Palo Alto, Riverside, Sacramento, San Diego, San Francisco, Santa Barbara, Sunnyvale, Van Nuys
Colorado: Colorado Springs, Denver
Connecticut: North Branford, Norwalk
Florida: Ft. Lauderdale, Orlando, Tampa
Georgia: Norcross
Idaho: Boise
Iowa: Bettendorf, Des Moines
Illinois: Arlington Heights, Champaign, Chicago, Peoria, Rockford
Indiana: Indianapolis
Kentucky: Louisville
Louisiana: Baton Rouge, Metairie
Maine: Portland, Westbrook
Maryland: Baltimore
Massachusetts: Cambridge, Framingham, Southboro, Waltham, Wellesley, Westboro, West Springfield, Worcester
Michigan: Grand Rapids, Southfield
Minnesota: Richfield
Missouri: Creve Coeur, Kansas City
Mississippi: Jackson
Montana: Billings
Nebraska: Omaha
Nevada: Reno
New Hampshire: Bedford, Portsmouth
New Jersey: Cherry Hill, Somerset, Wayne
New Mexico: Albuquerque
New York: Buffalo, Lake Success, Latham, Liverpool, Melville, New York City, Rochester, White Plains
North Carolina: Charlotte, Greensboro, Greenville, Raleigh, Research Triangle Park
Ohio: Brooklyn Heights, Cincinnati, Columbus, Dayton
Oklahoma: Oklahoma City, Tulsa
Oregon: Lake Oswego
Pennsylvania: Blue Bell, Lancaster, Philadelphia, Pittsburgh
Rhode Island: Providence
South Carolina: Columbia
Tennessee: Knoxville, Memphis, Nashville
Texas: Austin, Dallas, El Paso, Ft. Worth, Houston, San Antonio
Utah: Salt Lake City
Virginia: McLean, Norfolk, Richmond, Salem
Washington: Bellevue, Richland, Spokane
West Virginia: Charleston
Wisconsin: Brookfield, Grand Chute, Madison

DG-04976

INTERNATIONAL OFFICES

Argentina: Buenos Aires
Australia: Adelaide, Brisbane, Hobart, Melbourne, Newcastle, Perth, Sydney
Austria: Vienna
Belgium: Brussels
Bolivia: La Paz
Brazil: Sao Paulo
Canada: Calgary, Edmonton, Montreal, Ottawa, Quebec, Toronto, Vancouver, Winnipeg
Chile: Santiago
Columbia: Bogata
Costa Rica: San Jose
Denmark: Copenhagen
Ecuador: Quito
Egypt: Cairo
Finland: Helsinki
France: Le Plessis-Robinson, Lille, Lyon, Nantes, Paris, Saint Denis, Strasbourg
Guatemala: Guatemala City
Hong Kong
India: Bombay
Indonesia: Jakarta, Pusat
Ireland: Dublin
Israel: Tel Aviv
Italy: Bologna, Florence, Milan, Padua, Rome, Tourin
Japan: Fukuoka, Hiroshima, Nagoya, Osaka, Tokyo, Tsukuba
Jordan: Amman
Korea: Seoul
Kuwait: Kuwait
Lebanon: Beirut
Malaysia: Kuala Lumpur
Mexico: Mexico City, Monterrey
Morocco: Casablanca
The Netherlands: Amsterdam, Rijswijk
New Zealand: Auckland, Wellington
Nicaragua: Managua
Nigeria: Ibadan, Lagos
Norway: Oslo
Paraguay: Asuncion
Peru: Lima
Philippine Islands: Manila
Portugal: Lisbon
Puerto Rico: Hato Rey
Saudi Arabia: Jeddah, Riyadh
Singapore
South Africa: Cape Town, Durban, Johannesburg, Pretoria
Spain: Barcelona, Bibao, Madrid
Sweden: Gothenburg, Malmö, Stockholm
Switzerland: Lausanne, Zurich
Taiwan: Taipei
Thailand: Bangkok
Turkey: Ankara
United Kingdom: Birmingham, Bristol, Glasgow, Hounslow, London, Manchester
Uruguay: Montevideo
USSR: Espoo
Venezuela: Maracaibo
West Germany: Dusseldorf, Frankfurt, Hamburg, Hannover, Munich, Nuremburg, Stuttgart

DataGeneral

Ordering Technical Publications

TIPS is the Technical Information and Publications Service—a new support system for DGC customers that makes ordering technical manuals simple and fast. Simple, because TIPS is a central supplier of literature about DGC products. And fast, because TIPS specializes in handling publications.

TIPS was designed by DG's Educational Services people to follow through on your order as soon as it's received. To offer discounts on bulk orders. To let you choose the method of shipment you prefer. And to deliver within a schedule you can live with.

How to Get in Touch with TIPS

Contact your local DGC education center for brochures, prices, and order forms. Or get in touch with a TIPS administrator directly by calling (617) 366-8911, extension 4086, or writing to

Data General Corporation
Attn: Educational Services, TIPS Administrator
MS F019
4400 Computer Drive
Westborough, MA 01580

TIPS. For the technical manuals you need, when you need them.

DGC Education Centers

Boston Education Center
Route 9
Southboro, Massachusetts 01772
(617) 485-7270

Washington, D.C. Education Center
7927 Jones Branch Drive, Suite 200
McLean, Virginia 22102
(703) 827-9666

Atlanta Education Center
6855 Jimmy Carter Boulevard, Suite 1790
Norcross, Georgia 30071
(404) 448-9224

Los Angeles Education Center
5250 West Century Boulevard
Los Angeles, California 90045
(213) 670-4011

Chicago Education Center
703 West Algonquin Road
Arlington Heights, Illinois 60005
(312) 364-3045

User Documentation Remarks Form

Your Name _____ Your Title _____

Company _____

Street _____

City _____ State _____ Zip _____

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

Manual Title *RDOS/DOS Text Editor* Manual No. *069-400016-00*

Who are you? ☐EDP Manager ☐Analyst/Programmer ☐Other _____
☐Senior Systems Analyst ☐Operator _____

What programming language(s) do you use? _____

How do you use this manual? *(List in order: 1 = Primary Use)* _____

___ Introduction to the product ___ Tutorial Text ___ Other
___ Reference ___ Operating Guide _____

About the manual:	Yes	Somewhat	No
Is it easy to read?	☐	☐	☐
Is it easy to understand?	☐	☐	☐
Are the topics logically organized?	☐	☐	☐
Is the technical information accurate?	☐	☐	☐
Can you easily find what you want?	☐	☐	☐
Does it tell you everything you need to know	☐	☐	☐
Do the illustrations help you?	☐	☐	☐

If you have any comments on the software itself, please contact Data General Systems Engineering.
If you wish to order manuals, use the enclosed TIPS Order Form (USA only).

Remarks:

Date

CUT ALONG DOTTED LINE

FOLD FOLD

TAPE TAPE

FOLD FOLD

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 26 SOUTHBORO, MA. 01772

Postage will be paid by addressee:

Data General
User Documentation, M.S. E-111
4400 Computer Drive
Westborough, Massachusetts 01581

DataGeneral

users group

Installation Membership Form

Name _____ Position _____ Date _____

Company, Organization or School _____

Address _____ City _____ State _____ Zip _____

Telephone: Area Code _____ No. _____ Ext. _____

CUT ALONG DOTTED LINE

1. Account Category

- ☐ OEM
- ☐ End User
- ☐ System House
- ☐ Government

2. Hardware

	Qty. Installed	Qty. On Order
M/600		
MV/Series ECLIPSE®		
Commercial ECLIPSE		
Scientific ECLIPSE		
Array Processors		
CS Series		
NOVA® 4 Family		
Other NOVAs		
microNOVA® Family		
MPT Family		

Other _____
(Specify) _____

3. Software

- ☐ AOS
- ☐ AOS/VS
- ☐ AOS/RT32
- ☐ MP/OS
- ☐ MP/AOS
- ☐ RDOS
- ☐ DOS
- ☐ RTOS
- ☐ Other

Specify _____

4. Languages

- ☐ ALGOL
- ☐ DG/L
- ☐ COBOL
- ☐ Interactive COBOL
- ☐ PASCAL
- ☐ Business BASIC
- ☐ BASIC
- ☐ Assembler
- ☐ FORTRAN 77
- ☐ FORTRAN 5
- ☐ RPG II
- ☐ PL/1
- ☐ APL
- ☐ Other

Specify _____

5. Mode of Operation

- ☐ Batch (Central)
- ☐ Batch (Via RJE)
- ☐ On-Line Interactive

6. Communication

- ☐ HASP
- ☐ HASP II
- ☐ RJE80
- ☐ RCX 70
- ☐ RSTCP
- ☐ 4025
- ☐ X.25
- ☐ SAM
- ☐ CAM
- ☐ XODIAC™
- ☐ DG/SNA
- ☐ 3270
- ☐ Other

Specify _____

7. Application Description

○ _____

8. Purchase

From whom was your machine(s) purchased?

- ☐ **Data General Corp.**
- ☐ Other

Specify _____

9. Users Group

Are you interested in joining a special interest or regional Data General Users Group?

○ _____

DataGeneral

Data General Corporation, Westboro, Massachusetts 01580, (617) 366-8911

FOLD *FOLD*

TAPE *TAPE*

FOLD *FOLD*

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 26 SOUTHBORO, MA. 01772

Postage will be paid by addressee:

Data General

ATTN: Users Group Coordinator (C-228)
4400 Computer Drive
Westboro, MA 01581

Data General

Data General Corporation, Westboro, Massachusetts 01580

069-400016-00