TEXT EDITOR

User's Manual

093–000018–08

TAPES

Binary:	091-000001
RDOS Dumps:	088-000050
	088-000106
SOS Relocatable Binaries:	089-000104
	089-000080
	099-000010

The DGC Text Editor allows the user to edit input to produce updated output. It is most commonly used to modify program source tapes in preparation for a new assembly.

Ordering No. 093-000018

Printed in the United States of America
Rev. 08, February 1975

NOTICE

Original Release - February 1969
First Revision - May 1969
Second Revision - August 1971
Third Revision - June 1972
Fourth Revision - August 1972
Fifth Revision - November 1972
Sixth Revision - June 1973
Seventh Revision - June 1974
Eighth Revision - February 1975

This revision of the DGC Text Editor, 093-000018-08, is a major revision and supersedes any previous revisions.

INTRODUCTION

The purpose of the DGC Text Editor is to create, update and modify ASCII text. This is done by the use of simple commands.

The commands used in the Editor are divided into groups, those that input and output the contents of the edit buffer and those that modify the contents of the buffer. The edit buffer is an area of memory in which input is stored while being modified. Input commands bring the text into the buffer, modification commands are used to change the contents of the buffer, and output commands transfer the updated files to the output device.

The command structure is versatile enough to allow the modification of a single character, several characters, or a whole line.

Once loaded, the Stand-alone Text Editor is self-starting and in a 4K machine, provides over 3000 characters of text storage.

There are four editing systems:

1. Stand-alone Edit
2. SOS Edit
3. RDOS Edit
4. MEDIT

The first three are single-user Text Editors. MEDIT is the multi-user Text Editor. MEDIT was created to allow several users to be active simultaneously. The multi-user Editor is described in Chapter 4.

TABLE OF CONTENTS

CHAPTER 1

CONCEPTS

EDITING PROCESS

The editing process consists of the following steps:

1. Call or load the Editor.
2. Specify I/O devices or files.
3. Bring a page into the buffer from the input device or file.
4. Modify the page.
5. Transfer the page from the buffer to the output device or file.

Figure 1-1. Editor I/O

When the Text Editor is loaded (step 1) it prints a prompt ("*") on the input console. The user must then specify the I/O devices or files (step 2). Another prompt is issued and an input command is issued by the user (step 3). Whenever a prompt appears on the console, it signifies the Editor is ready to take another command. To modify the page the user issues commands from the console (step 4). Step 5 is accomplished by issuing output commands.

If the capacity of the buffer is exceeded, while reading input into the edit buffer, an error message is printed on the console. Error messages are described later in the manual.

INPUT

The DGC Text Editor takes a continuous string of characters as input. These characters form lines, where a line is a string of characters up to and including a carriage return ( ). A page of input is a string of characters up to but not including a form feed.

While using the DGC Text Editor in the Real Time Disk Operating System (RDOS) or the Stand-alone Operating System (SOS), input is in the form of an input file or any input device supported by the operating system. When editing in Stand-alone Operation, input is in the form of paper tape.

OUTPUT

Output is the information that is transferred from the buffer to the output device or file. RDOS output disk files are updated after every page. This saves a part of the edited file in the event of a system failure.

CHARACTER POINTER

The DGC Text Editor maintains an implicit pointer called the Character Pointer (CP). This pointer resides between two characters and is used to facilitate locating the exact position for modification or examination. For example:

```
ABCDF
  ↑
  CP
```

Any modifications performed are placed at the current location of the CP (i.e., insertion of an E produces, ABCDEF).

There is also an implicit line numbering system that is maintained by the Text Editor. This numbering system starts with the number 1 and is continually updated as lines are inserted and deleted.

TABBING

The Editor is equipped to simulate tabs with spaces. This feature is usually used if the console being used does not have a tab key. If tabs are simulated, the pre-defined tab positions occur at 1, 9, 17, 25, etc. The Editor is initialized to simulate tabs. It may be complemented, by typing a CTRL P (↑P), in a command string. Each occurrence of ↑P causes the tabbing feature to be complemented, thus ↑P↑P has no effect.

MODE

The Editor can operate in one of the following two modes:

Page Mode	File input is page oriented.
Window Mode	File input deals with a set number of lines of input (RDOS only).

At any instant the Editor is in one of the following two states:

Command Mode	The Editor is ready to accept a command.

MODE (Continued)

Execution Mode	Commands issued to the Editor are being carried out.

RDOS NUMBER REGISTER

The Editor maintains a register represented by "#", which may be used to modify text when operating in the Real Time Disk Operating System. The commands for the manipulation of this register are found under Special Editing Commands Used in RDOS in Chapter 2.

PARITY CHECK

Some of the input commands can check parity. A parity bit is a binary bit that indicates the total number of binary "1" digits in a character. When the total number of "1" bits, including the parity bit, is even, the system is an even parity system.

When parity is checked, the parity bit is looked at to see if it indicates odd or even parity. If the parity bit is 0 then it has odd parity and an error message is printed on the teletype.

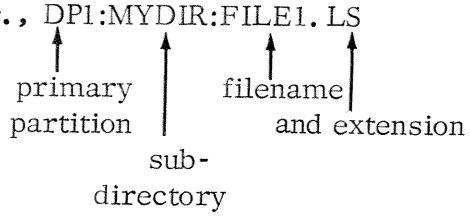
NOTATION CONVENTIONS

Console Representation	Results from
↑letter	Pressing CTRL and some letter e.g., CTRL E.
$	Pressing ESC.

Notation Conventions Used in Formats in the Manual

NOTATION	MEANING
)	Represents pressing RETURN on console.
[ ]	Enclose optional arguments to a command.
variable	An argument in lower case and underlined is a variable for which the user substitutes the appropriate literal.

Notation Conventions Used in Formats in the Manual (Continued)

NOTATION	MEANING
filename	The variable argument filename is usually replaced with the name of a file such as FILE 1. However, under RDOS, filename may represent the format: { primary partition:} { sub-partition:} { sub-directory:} filename { extension } e.g., DP1:MYDIR:FILE1.LS  (RDOS only)

CHAPTER 2

COMMANDS

As explained on page 1-1, once the user loads the Editor and the Editor issues a prompt the user is in command mode. It is now possible to issue input, modification, and output commands. These commands are described in this Chapter.

The DGC Editor is used in three operating environments:

1. Stand-alone Operation
2. Real Time Disk Operating System - RDOS
3. Stand-alone Operating System - SOS

The operating procedures for these environments are given in Chapter 3. Certain commands are used in all three environments, while some are relevant only to specific environments.

COMMAND STRUCTURE

ESC Key

The ESC (escape) character is represented by a $ on the output console. It is used for two purposes:

1. To signal the end of a command or command string. This is done by typing two consecutive escapes.
2. To delimit a string from the next command character or from another string. This is done by typing a single escape.

Single Command

A single command has the format:

{n} code {string $}

where: n is an optional integer in the range $-2047 < n < +2047$ in SOS and in the range $-2^{15} \leq n \leq +2^{15} - 1$
code is a letter, two letters, or special graphic
string is an optional string of characters terminated by an ESC.

The only exceptions to this are the change and UR commands which have two strings following the code:

Cstring$string$

URstring$string$

COMMAND STRUCTURE (Continued)

Command String

A command string is two or more commands stacked for sequential execution by the Editor. Each single command may be delimited from the next by a single escape. The format is:

where: [n] code [string$] [n] code [string$] [n] code [string$]...

n is an optional integer in the range -2047 $\leq$ n $\leq$ +2047 in SOS and in the range $-2^{15} \leq n \leq +2^{15}-1$ in RDOS
code is a letter, two letters, or special graphic
string is an optional string of characters terminated by an ESC.

A command string may be carried over to the next line by the use of a carriage return between two commands. Two consecutive ESC's terminate the command string.

Deleting a Command Character

The RUBOUT key is used to delete characters in a command. It deletes the last typed character from right to left, each time it is depressed. When a character is deleted, it is repeated on the output console. For example:

```
THE ABSSBAVALUE
    ↑↑↑
    rubouts
```

produces:

```
THE VALUE
```

When the RUBOUT key is used to delete all the characters in a command line a new prompt is issued by the Editor.

INPUT COMMANDS USED IN ALL ENVIRONMENTS

--- Y

The command to read input into the buffer until a form feed or window width is encountered is:

```
Y$$
```

In page mode, the form feed is read but not stored in the edit buffer. The CP is positioned at the beginning of the buffer and a prompt is issued on the output console.

If the capacity of the buffer is exceeded the resulting error message is:

TEXT BUFFER FULL WHILE READING

The buffer must then either be output or sections deleted from it before the rest of the page can be read in.

If parity is checked and a read fails parity, the message printed on the teletype is:

PARITY ERROR IN LINE n

For example, to bring the first 5 lines of the Sample Program (Figure 2-1) into the edit buffer, issue the command:

Y$$

The command to yank in a specified number of lines is:

nY$

where: n is the number of lines to be read into the edit buffer.
This command is not applicable in the Stand-alone Editor.

If the next page has fewer than n lines, the Editor will stop at the form feed.

In the Window mode both Y or nY brings in the window width. A form feed is read in as just another character.

--- A

The command to append a page or window width of input to the material already contained in the edit buffer, is:

A$

The Text Editor appends a page of input to the present contents of the buffer and returns a prompt on the console. The CP is located at the beginning of the appended page.

If the buffer is already full or its capacity is exceeded while appending, the resulting message is:

TEXT BUFFER FULL WHILE READING

INPUT COMMANDS USED IN ALL ENVIRONMENTS (Continued)

A (Continued)

In either case the buffer must be output or sections deleted before the rest of the page can be appended.

For example, if the contents of the second buffer of the Sample Program (Figure 2-1) is to be appended to the first buffer to form one buffer the command issued is:

A$$

INPUT COMMANDS USED IN RDOS

--- GR

The command to get for reading (input) a specified file is:

GRinputfilename$

inputfilename is the file to be input and must be an existing file. When this command is issued the input file, if any, is closed and the specified file is opened.

Note: This command does not read the first page of inputfilename into the buffer, the user must issue a command in order to bring the first page into the buffer.

If the file does not exist the resulting error message is:

NO SUCH FILE

If the file is read-protected the resulting error message is:

FILE CAN'T BE USED FOR INPUT

For example, when the command:

GRSAMPLE$$

is issued, the Editor opens the file SAMPLE for examination and/or modification.

--- UN

UNfilename$

is used to create a new file. This command creates a file with the specified name.

UN (Continued)

makes it the input file and makes it permanent for the duration of its editing. It also creates and opens filename.SC for output. A filename with suffix .SC or a file with the same name as a device name cannot be used. The attributes of the input file are restored only when a UE, US, UC or UH is issued.

For example, to create a permanent file named SAMPLE and open SAMPLE.SC for output, issue:

```
UNSAMPLE$$
```

--- UY

```
UYfilename$
```

is used to edit an existing file. This command makes the specified file permanent for the duration of its editing, opens it for input and yanks a page. It also creates and opens filename.SC for output. A filename with the suffix .SC or a file with the same name as a device name cannot be used. The attributes of the input file are restored only when a UE, US, UC or UH is issued.

For example, to change the attributes of SAMPLE to permanent, open it for input and yank the first page into the edit buffer, issue:

```
UYSAMPLE$$
```

In addition it creates and opens SAMPLE.SC for output.

Note: After opening a file with UY or UN a GRfilename can be used to read in from other files. At the end when a UE, UC, or US is issued, the original input filename will still be used for renaming purposes.

When the filename is given immediately after the Edit command in the CLI, it is equivalent to UYfilename. For example,

```
EDIT SAMPLE )
is equivalent to
EDIT )
* UYSAMPLE$$
```

INPUT COMMANDS USED IN SOS

--- GR

The command to get for reading (input) a specified file is:

GRinputfilename$$

inputfilename is the file to be input and must be an existing file. When this command is issued any current input file is closed and the specified file is opened.

Note: This command does not read the first page of inputfilename into the buffer, the user must issue a command in order to bring the first page into the buffer.

If the file does not exist the resulting error message is:

NO SUCH FILE

If the file is read-protected the resulting error message is:

FILE CAN'T BE USED FOR INPUT

For example, when the command:

GRSAMPLE$$

is issued, the Editor opens the file SAMPLE for examination and/or modification.

CHARACTER POINTER COMMANDS USED IN ALL ENVIRONMENTS

--- B

The command to move the CP to the beginning of the buffer is:

B$

There is no argument used with this command. If one is given, it will be ignored by the Text Editor.

For example, to position the CP right before the period in line 1 of the Sample Program (Figure 2-1), issue the command:

B$$

--- nJ

The command to position the CP before the first character in line n from the beginning of the buffer is:

nJ$$

where: n is the number of the line the CP is to jump to.

For example, the CP is positioned at the end of line 3 in the Sample Program (Figure 2-1), and the user wants it positioned at the beginning of line 5, the command issued is:

5J$$

--- nL

The command to move the CP n lines from its present position is:

nL$

where: n is the number of lines the CP is to be moved.

For example, the CP is positioned at the beginning of line 3 in the Sample Program (Figure 2-1), and the user wants it positioned at the beginning of line 5, the command issued is:

2L$$

The command to move the CP to the beginning of the line on which it is located is:

L$

--- nM

The command to move the CP n characters to the right or left is:

nM$

where: n > 0 The CP is moved n characters to the right
n < 0 The CP is moved n characters to the left

CHARACTER POINTER COMMANDS USED IN ALL ENVIRONMENTS (Continued)

nM (Continued)

For example, the CP is positioned before the T of TEMPORARY in line 5, of the Sample Program (Figure 2-1), the command to position it before the Y in line 5 is:

8M$$

--- Z

To move the CP to the end of the buffer, issue the command:

Z$

An example of this would be if the CP were positioned anywhere on line 2 of the Sample Program (Figure 2-1), the command to position it after the) in line 5 is:

Z$$

BUFFER EXAMINATION COMMANDS USED IN ALL ENVIRONMENTS

There are two commands to examine the buffer: T and nT. Neither of these commands have any effect on the CP.

The command to type the entire buffer on the console is:

T$

The command to examine a certain number of lines of the buffer is:

nT$

where: n is the number of lines to be examined from the present position of the CP.

MODIFICATION COMMANDS USED IN ALL ENVIRONMENTS

--- S

The command to search for a character in the current buffer is:

Sstring$

S (Continued)

The Editor searches for the string from the present CP position. The CP is then positioned immediately after the last character of the first occurrence of the string.

When the end of the buffer is encountered and the string has not been found the console prints the message:

STR NOT FOUND

The CP is then positioned at the beginning of the buffer and a new prompt is issued.

For example, the CP is located at the beginning of the Sample Program (Figure 2-1), and the user issues the command:

SSAVE$$

The CP is positioned directly after the E in SAVE in line 1.

--- N and Q

The command to continue searching for a string throughout the remainder of the input file is:

Nstring$

The Editor then searches through the current buffer for the string. If it is not found the buffer is output, the next page is yanked in, and the search is continued. This process is repeated until the string is found. When the Editor encounters the string the CP is positioned immediately after the last character of the string.

If the end of the program is encountered and the string is not found the message printed on the console is:

STR NOT FOUND

The CP is now positioned at the start of an empty buffer.

For example, the CP is positioned before the T in TEMPORARY on line 5 of the Sample Program (Figure 2-1), and the user issues the command:

N.MPYA$$

N and Q (Continued)

The Editor searches through the current buffer, punches it, yanks in the next page and continues the search until the string is encountered. The CP is then positioned after the A in .MPYA .

Another way to perform a continuous search for a string is the command:

Qstring$

This is the same as the N command except the pages before the page in which the string is found are not punched.

--- C

The change command has the format:

C$string_1$\$string_2$\$

This searches for $string_1$, deletes it and inserts $string_2$. The CP is positioned after the last character of $string_2$.

When the command:

Cstring$$

is issued the Editor searches for the string and deletes it.

If the string is not found, the message printed on the console is:

STR NOT FOUND

The CP is positioned at the beginning of the current buffer.

For example, the CP is positioned after the R in STORAGE on line 5 of the Sample Program (Figure 2-1).

CAGE$ED$$

produces:

```
(5)     .BC11     .BLK 3        ;TEMPORARY STORED
```

--- I

The command to insert the string at the present CP position is:

Istring$

The CP is positioned directly after the last character of the string.

For example, the CP is positioned directly after the N of RETURN in line 1 of the Sample Program (Figure 2-1).

IS$$

produces

(1) .CC03: 0 ;SAVE RETURNS

To insert an ASCII character into the text, using the decimal representation, issue the command:

nI$$

where: n is the decimal representation of an ASCII character.

To insert an ASCII character into the text, using the octal representation, issue the command:

''nI

where: n is the octal representation of an ASCII character.

For example, the CP is positioned directly after the N of RETURN in line 1 of the Sample Program (Figure 2-1).

83I$$

produces:

(1) .CC03: 0 ;SAVE RETURNS

I (Continued)

whereas

```
"123I$$
```

produces:

```
(1)   .CC03:    0                    ;SAVE RETURNS
```

Note: The nI command is primarily used to insert lower case ASCII characters into a text, using an upper-case keyboard.

--- nD

The command to delete a number of characters before or after the present position of the CP is:

```
nD$
```

where: n > 0 Deletes n characters to the right of the CP.
n < 0 Deletes n characters to the left of the CP.

The CP is positioned immediately after the deleted character.

For example, the CP is positioned after the E in RESULTS in line 2 of the Sample Program (Figure 2-1).

```
-2D$$
```

produces:

```
(2)   .BC10:    ,BLK 4   ;SAVE ABS(U) , SULTS OF MULTIPLY, AS RE-
```

whereas:

```
2D$$
```

produces:

```
(2)   .BC10     .BLK4    ;SAVE ABS(U) , RELTS OF MULTIPLY, AS RE-
```

--- nK

The command to delete whole lines from the buffer is:

nK$

where: n > 0 Deletes n lines forward from the present CP position
n < 0 Deletes n lines backwards plus any characters to the left of the CP in the current line from the present position of the CP.

The CP is positioned after the last deleted character.

For example, the CP is positioned at the beginning of line 3. To kill lines 3 and 4 issue the command:

2K$$

--- Macro Command

A macro command is a command which contains an arbitrary command string. The DGC Text Editor provides for the definition and execution of a macro command.

The command to define the contents of the macro command is:

XM$command_1$...$command_n$$

where command has the format:

{n} code {string$}

The macro definition is terminated by a double escape; therefore it must be the last command in a command string. Once the macro command is defined, it is retained by the Editor and can be called for future execution by means of a single code. The Editor allows for the handling of only one macro definition at a time.

Execution of the macro command is accomplished by issuing the command:

nX$

where: n is the number of times the macro command is to be executed.

MODIFICATION COMMANDS USED IN ALL ENVIRONMENTS (Continued)

Macro Command (Continued)

The command to delete a macro command is:

XD

If the macro command is undefined or the macro is recursive, (e.g., XMX$$ 1000X$$) the resulting error message printed on the console is:

MACRO ERROR

For example, the CP is positioned at the very beginning of the buffer containing the first 5 lines of the Sample Program (Figure 2-1), the command:

```
XMCA$E$II$$
1X$$
```

produces:

```
(1)    .CC03:    0    ;SEIVE RETURN
```

The command to type out the contents of the current macro is:

X?$

MODIFICATION COMMANDS USED IN RDOS

--- UD

The command to delete a specific file is:

UDfilename$

where: filename is the name of the file to be deleted.

An example of this is:

UDSAMPLE$$

This deletes the Sample Program (Figure 2-1).

MODIFICATION COMMANDS USED IN RDOS (Continued)

--- UR

The command to rename a file is:

URfilename$_1$$filename$_2$$

where: filename$_1$ is the current name of the file.
filename$_2$ is the name it is to be changed to.

For example,

URSAMPLE$PROGRAM$$

changes the name of the Sample Program (Figure 2-1) to PROGRAM.

--- UZ

The command to change the attributes of a file to non-permanent is:

UZfilename$$

where: filename is the name of the file to be changed.

For example,

UZSAMPLE$$

changes the attributes of SAMPLE to non-permanent

SPECIAL EDITING COMMANDS USED IN ALL ENVIRONMENTS

COMMAND	MEANING
:	Print the number of lines currently in the edit buffer.
.	Print the number of the line the CP is pointing to.
=	Print the number of characters contained in the edit buffer.
↑I string$	Insert string with tabulation.

SPECIAL EDITING COMMANDS USED IN RDOS

COMMAND	MEANING
↑A	When issued within a command, it scraps the entire command buffer. If the command is longer than 256 characters it issues the message: SAVE COMMAND BUFFER YES (1) OR NO (2) ? If the user responds with a 1, the message: ENTER FILENAME is printed and the contents of the command buffer will be written to the file the user specifies. The Editor then issues a prompt. The file name that is entered must be a non-existing file.
↑Z	In a search string matches all characters. (i.e., .BC↑Z searches for any occurences of .BC with any character that follows it.)
U?	Type out the I/O file names.
CANCEL (↑X)	Delete the current line of the command line. When CANCEL is issued in the first line, delete it and restart the command. This can be repeated to delete multiple lines.
n# ←	Reset the register to n, where n may be from 1 to 5 digits in the range 0 n 65,535.
#+	Increment the register by 1.
#-	Decrement the register by 1.
#?	Type out the contents of the register.
#0	Output the register as a five digit unsigned integer to the text buffer at the current location of the CP
n#!...$!...$	When the register does not match the number argument, skip the command characters up to the next escape followed by an exclamation.

SPECIAL EDITING COMMANDS USED IN RDOS (Continued)

COMMAND	MEANING
↑C	The ↑C is an RDOS break character. Once it is issued it terminates the Editor and returns command to the CLI.

WINDOW MODE COMMANDS USED IN RDOS

nW	If currently in the page mode, this sets the mode to window mode with a width of n lines. If in the window mode it resets Editor to page mode.
W?	Displays the page/window mode status.
W	Resets the mode to page mode from window mode. If the Editor is already in page mode the resulting error message is: ILLEGAL WINDOW WIDTH

OUTPUT COMMANDS USED IN ALL ENVIRONMENTS

--- E

The command to output the contents of the buffer as well as the remainder of the input file to the output file is:

E$

For example, if the user makes a correction to the first line of the Sample Program (Figure 2-1), and wishes to copy, as is, the rest of the file, issue the command:

E$$

--- F

The command issued to output a form feed to the output file is:

F$

F (Continued)

When an argument is given:

nF$$

n inches of leader is output. If n is greater than 100, only 100 inches of leader is output.

Neither F nor nF has any effect on the CP.

--- P and PW

The command to output the entire buffer to the output device ending with a form feed is:

P$

If a numeric argument precedes the P command:

nP$

then starting from the CP, n lines will be output ending with a form feed. When operating in window mode there is no form feed. When n is greater than the number of lines contained in the buffer, punching will stop at the end of the buffer.

The command issued to eliminate the ending form feed in the above commands is either:

PW or
nPW

None of the Punch Commands have any effect on the CP.

--- nR

The command to output n pages and read in n pages is:

nR$

OUTPUT COMMANDS USED IN ALL ENVIRONMENTS (Continued)

nR (Continued)

This is equivalent to PY...PnYn . If the n argument is not given then one page is output and one page is read into the buffer.

For example:

R$$

outputs the contents of the first buffer of the Sample Program (Figure 2-1) and reads in the contents of the second buffer.

OUTPUT COMMANDS USED IN RDOS

--- GW

The command to specify an output file is:

GWoutputfilename$

This will get for writing (output) the specified file outputfilename. The outputfilename cannot already exist.

The Editor issues the following error messages:

OUTPUT FILE ALREADY ACTIVE
if the output file is active and has not been closed

FILE NAME IN USE
if the file name specified already exists

FILE CAN'T BE USED FOR OUTPUT
if the file is write-protected

ILLEGAL FILE NAME
if the specified file name does not conform to legal RDOS file name specifications.

GW (Continued)

For example,

GRSAMPLE$GWPROGRAM$$

Opens the input file SAMPLE for reading and the output file PROGRAM for writing.

--- GC

The command issued to close the input and output file upon completion of the editing of an input file to produce a new output file is:

GC$

Warning: If return to the operating system takes place without closing the output file, it has a byte count of zero.

When an output command is issued and no output file has been specified the resulting error message is:

NO OUTPUT FILE

To include the remaining input in the output file a P or E command must be issued before the GC command. The GC command does not force the writing of the last output page.

Note: Multiple files may be appended to produce one output file. This is accomplished by successive use of the GR command without declaring a new output file.

--- H

A return can be made to the CLI, upon completion of the editing by issuing the HOME command:

H$

Note: When using H$$ in MEDIT it is equivalent to the UH command in RDOS.

OUTPUT COMMANDS USED IN RDOS (Continued)

--- GO

The command to close the current output file, opened by GW, and open a new output file is:

GOfilename$$

where: filename is the file to be opened.

For example,

```
GRSAMPLE$GWEXAMPLE$$
        :
        :
GOPROGRAM$$
```

closes EXAMPLE and opens PROGRAM

--- UE

The command to copy the remaining input file, then close the input and output files, delete the input file and rename the output file with the input file name is:

UE$

For example,

```
UYSAMPLE$$
    :
    :
UE$$
```

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

--- US

The command to copy the input file until its end and close the input and output files is:

```
US$
```

This also renames the input file name to filename.BU, renames the output file to the input file name, restores the attributes of the input file and deletes the previous file.BU, if any.

For example,

```
UYSAMPLE$$
    :
    :
US$$
```

copies SAMPLE until its finish, closes SAMPLE and SAMPLE.SC, renames SAMPLE to SAMPLE.BU and SAMPLE.SC to SAMPLE and restores the attributes to SAMPLE .

--- UC

The command to close input and output files, restore the input file's attributes and rename the output file with a specified name is:

```
UCfilename$
```

where: filename is the specified name the output file is to be changed to.

For example,

```
GRSAMPLE$GWPROGRAM$$
                :
                :
UCEXAMPLE$$
```

closes SAMPLE and PROGRAM, restores the attributes to SAMPLE and renames PROGRAM to EXAMPLE.

OUTPUT COMMANDS USED IN RDOS (Continued)

--- UH

The command to close input and output files opened by UN or UY is:

UH$

Note: When using MEDIT H$$ is equivalent to UH$$.

OUTPUT COMMANDS USED IN SOS

--- GW

The command to specify an output file is:

GWoutputfilename$$

This will get for writing (output) the specified file outputfilename. The outputfilename cannot already exist.

The Editor issues the following error messages:

OUTPUT FILE ALREADY ACTIVE
if the output file is active and has not been closed

FILE CAN'T BE USED FOR OUTPUT
if the file is write-protected

ILLEGAL FILENAME
if the specified file name does not conform to legal SOS file name specifications.

For example, if the Sample Program is residing in CT0:1 and the user wants a paper tape punch of the output, the command issued is:

GRCT0:1GWPTP$$

This opens the input file residing on CT0:1 for reading and the output file $PTP for writing.

--- GC

The command issued to close the input and output file upon completion of the editing of an input file to produce a new output file is:

GC$

Warning: If return to the operating system takes place without closing the output file, it has a byte count of zero.

When an output command is issued and no output file has been specified the resulting error message is:

NO OUTPUT FILE

To include the remaining input in the output file a P or E command must be issued before the GC command. The GC command does not force the writing of the last output page.

Note: Multiple files may be appended to produce one output file. This is accomplished by successive use of the GR command without declaring a new output file.

--- H

If an H command is issued while using the Stand-along Operating System in cassette or magnetic tape environment an immediate return is made to the Core Image Loader. When issued in a paper tape environment the H command is ignored.

```
                 (1)    .CC03:   0          ;SAVE RETURN)
CONTENTS         (2)    .BC10:   .BLK 4     ;SAVE ABS(U), RESULTS OF MULTIPLY)
OF               (3)                        ;AS REMAINDER OF DIVIDE)
FIRST BUFFER     (4)    .BC11:              ;SAVE ABS(V))
                 (5)    .BC11    .BLK 3     ;TEMPORARY STORAGE )

                 (6)    .BC13:   0 )
CONTENTS         (7)    .BC14:   0)
OF               (8)    .BC20:   42)
SECOND           (9)    .BC30:   .MPYU)
BUFFER          (10)    .BC31:   .MPYA)
                (11)    .BC32:   .BC10)

                                 filename:  SAMPLE
```

*****

Figure 2-1. SAMPLE PROGRAM

CHAPTER 3

OPERATING PROCEDURES

There are three operating environments under which the Text Editor is used:

1. Stand-alone Operation
2. Stand-alone Operating System (SOS)
3. Real Time Disk Operating System (RDOS)
   a. Foreground
   b. Background

The initial loading and operation is different for each operating environment. Actual use of the Text Editor is the same except where noted within the manual.

STAND-ALONE OPERATION

The Text Editor object tape (091-000001) is loaded into core with the Binary Loader (091-000004). Once the Text Editor tape is loaded, control is transferred to the Editor which deletes its edit buffer and input buffers and requests the user to assign input and output devices.

The Editor first asks the user to establish the type of I/O devices to be used. It does this by typing two questions:

```
TTO(1)  OR  PTP (2) ?
```

where: 1 is issued by the user for teletypewriter output
2 is issued by the user for high speed punch output

```
TTI (1)  OR PTR  (2) ?
```

where: 1 is typed by the user for teletypewriter input
2 is typed by the user for high speed paper tape reader

A problem arises when the teletype is used for edited output in Stand-alone Operation. If the punch is left in the "ON" position while typing editing commands, these characters are punched as part of the output. The Editor resolves this problem by halting with Carry light OFF, before transmitting a prompt. This signals the user to turn the teletype punch OFF. After doing so, the user presses CONTINUE causing the Editor to issue a prompt. When the command string is terminated for execution,

the Editor determines whether output is to occur. If it is, the Editor again halts, this time with the Carry light ON. This signals the user to turn the punch ON, and press CONTINUE, causing the output to be punched. This process is repeated between commands and output, thus eliminating the interference problem.

The next questions asked by the Editor determine if the I/O is to be punched with parity. The Editor issues the two questions:

PARITY OUT (1) OR NOT (2) ?

where: 1 is issued to punch the output tape with parity
2 is issued to punch the output tape without parity

PARITY IN (1) OR NOT (2) ?

1 is issued to read the input tapes with parity
2 is issued to read the input tapes without parity

Once this information is determined, the Editor prints a prompt. The user now issues input, modification, and output commands. The Editor returns a prompt after each command until the user returns command to the CLI.

If it is necessary to reassign the I/O devices and delete all buffers, the user performs the following steps:

1. Press RESET
2. Enter 000002 in the data switches
3. Press START

When the user wishes the Editor to stop the operation it is performing without losing the Editor's buffer contents or changing the I/O assignments, or macro definition, perform the following steps:

1. Press RESET
2. Enter 000003 into the data switches
3. Press START

The Editor now issues a new prompt.

REAL TIME DISK OPERATING SYSTEM

The single-user background Text Editor is named EDIT.SV and is supplied as part of dumped tape (088-000050). The relocatable binary EDIT.RB can be loaded

using RLDR.SV to obtain a foreground single user Editor. The multi-user Editor is supplied as a dump of relocatable binary MEDIT.RB which may be loaded as foreground or background. Under RDOS the Text Editor can be used in either a single user or multi-user environment. In both single and multi-user environments it can be used in either foreground or background.

The dumped EDIT.SV and EDIT.RB on paper tape can be loaded onto the disk by mounting tape 088-000050 on an appropriate input device and using the LOAD command:

LOAD { $PTR / $TTR })

When finished, the CLI responds with a ready message (R). The user now brings the Editor into core by issuing the command:

EDIT)

The Editor issues a prompt signaling the user to respond with the editing commands that specify I/O file names. These commands are found in Chapter 2.

Editing in the Foreground

To edit in a desired foreground partition under the Real Time Disk Operating System, the RLDR command is used to load the relocatable binary file EDIT.RB which is on RDOS Dump tape 088-000050. The format of the RLDR command line is:

RLDR NREL partition/F ZREL partition/Z EDIT [filename/S])

where: /F indicates that the preceding octal value is the foreground NREL partition address.
/Z indicates that the preceding octal value is in the foreground.
/S indicates that a save file will be produced with the name filename.

All RLDR loads are in the background partition unless otherwise specified. Foreground loads are indicated by the partition addresses with local /F and /Z switches.

The partition addresses define the starting ZREL and NREL addresses of the foreground load. The selected NREL partition must be equal to $16_8+(n*400_8)$, where n is a positive integer.

Editing in the Foreground (Continued)

An example of an RLDR command line producing a save file with the name EDIT.SV is:

```
RLDR  20000/F  374/Z  EDIT  FEDIT.SV/S  6/C
```

The /C local switch specifies that the preceding octal value (6) is the number of channels required. This value is then placed in USTCH of the User Status Table. Memory is essentially represented as shown in Figure 3-1 on the following page.

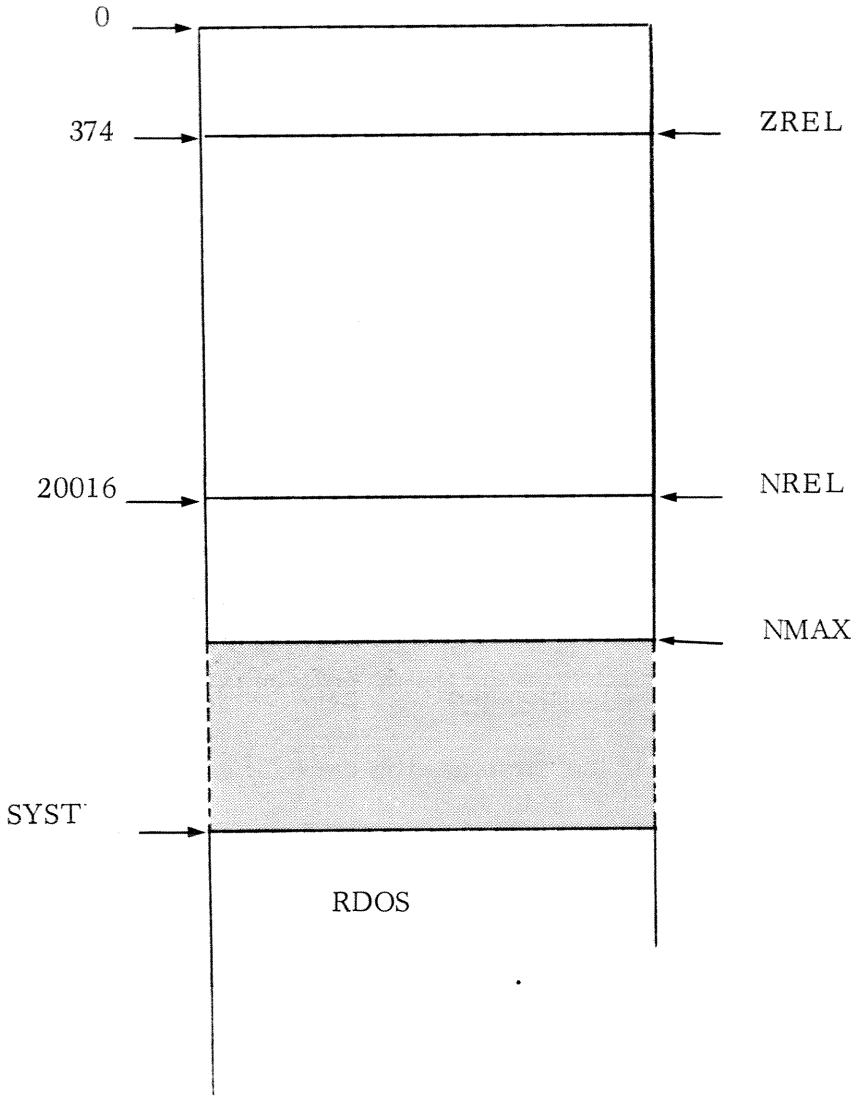


Figure 3-1. Memory Representation

REAL TIME DISK OPERATING SYSTEM (Continued)

In an unmapped system, the shaded area outlined with broken lines in Figure 3-1, will contain the User File Table, UFT . The unmapped foreground UST starts at the foreground partition. Location 12, USTP, points to the start of the UST belonging to the currently executing foreground program.

In order for the Editor to operate in the foreground partition in RDOS, the user should ensure that the Editor has enough room in core. This is done by finding the address of SYSTE in the symbol table of the load map of the RDOS system and then following the formula:

R_1 = address of SYST - number of channels * 41_8

R_2 = R_1 - number of Editor channels * 41_8

where: R_2 = top of Editor

R_3 = R_2 - size of Editor

where: R_3 is the approximate size required for the editor in core.

The Editor is now ready to be executed in the foreground. This is done by executing the following command line:

EXFG [/E] filename

where: filename is the name used in the RLDR command line.

To load a foreground save file from the disk into core and transfer control to the save file, EXFG is used. If the save file cannot be loaded without overwriting the CLI, the foreground save file will not be loaded. The optional /E switch specifies that the foreground and background partitions are to have equal priority.

In MRDOS no special loading is needed for foreground operation.

STAND-ALONE OPERATING SYSTEM

Under the Stand-alone Operating System, the Editor can be configured to produce an absolute binary tape and/or a save file on a master reel.

Producing an Absolute Binary Tape

The three binary tapes that are required for use of the Editor under SOS are:

1. 089-000104 - relocatable binary Text Editor
2. 099-000010 - SOS library
3. 089-000080 - SOS relocatable binary punch

The main SOS program and the driver routines necessary for all SOS I/O devices, except cassette and magnetic tape, are contained in the SOS Library tape (099-000010). For systems containing one or more cassette drives, a cassette Library tape (099-000041) is supplied. A magnetic tape Library tape (099-000042) is used for systems containing one or more magnetic tape drives.

In addition to the three tapes above a user-supplied binary tape is also required. This tape is produced by assembling the following program into the object program, TRIGGER.RB

TRIGGER.RB

```
.TITLE    TRIGGER     ;optional title statement
.EXTN     .CDRD       ;optional load of the card reader
.EXTN     .MTAD       ;optional load of magnetic tape unit 0
.EXTN     .MTU1       ;optional load of magnetic tape unit 1
.EXTN     .MTU2       ;optional load of magnetic tape unit 2
             .
             .
             .
.EXTN     .MTU7       ;optional load of magnetic tape unit 7
.EXTN     .CTAD       ;optional load of cassette unit 0
.EXTN     .CTU1       ;optional load of cassette unit 1
.EXTN     .CTU2       ;optional load of cassette unit 2
             .
             .
             .
.EXTN     .CTU7       ;optional load of cassette unit 7
.EXTN     .TTI1       ;optional load of 2nd $TTI and $TTO
          .PTPD       ;paper tape punch
          .PTRD       ;paper tape reader
          .LPTD       ;80 column line reader
.END
```

Producing an Absolute Binary Tape (Continued)

After the above program is assembled, the following procedure is taken to create the absolute binary SOS Text Editor.

1. Load the Extended Relocatable Loader (091-000038) via the Absolute Binary Loader (091-000004).

2. Load the SOS Relocatable Editor, SOSED.RB (089-000104) from the $PTR by issuing the "2" command.

3. Load the user produced relocatable tape, TRIGGER.RB, from the $PTR by issuing the "2" command.

4. Load the appropriate SOS Library tape (099-000042 or 099-000041 for magnetic tape or cassette support and 099-000010)from $PTR by issuing the "2" command.

5. Load the relocatable binary punch program, 089-000080, from the $PTR by issuing the "2" command.

6. Issue the "6" command causing the loader to print the load map on the $TTO.

7. Issue the "8" command causing the loader to terminate the load and initialize the load module.

8. Perform the following:

   A. Enter value of RBFP into data switches - this value is found in the load map.

   B. Press START.

9. An output device must be selected for the binary tape; type

   nH

   where: n = 0 if output device is $TTP

   n = 1 if output device is $PTP

Producing an Absolute Binary Tape (Continued)

10. Type the following:

    1,nnnnnP

    where: nnnnn is the value of NMAX which can be found from the the load map. This will punch leader on the tape followed by the specified memory range (1 through nnnnn) in absolute binary format. Upon completion, a carriage return/line feed is activated on the TTY.

11. Issue a "377E" command which produces a Start Block with an address of 377. Tape trailer is also produced. A reuseable SOS Text Editor tape now exists.

12. To activate the SOS Text Editor, perform the following

    A. Press RESET

    B. Enter 377 in the data switches

    C. Press START

    This will cause the prompt to be printed on the $TTO.

The user then responds to the prompt with Editor commands which specify input and output file names. Any file on any cassette or magnetic tape unit may be specified as the input file, and any file on any other cassette magnetic tape can be referenced as the output file.

Producing a Master Reel

To call the Editor via a master reel the user must first produce a master reel with the following utilities:

File 0 Core Image Loader/Writer
File 1 Relocatable Loader
File 2 Command Line Interpreter
File 3 Text Editor
File 4 Extended Assembler
File 5 Library File Editor
File 6 SYSGEN

Producing a Master Reel (Continued)

Once this is completed the Editor can be called using a CLI BOOT command or by using the Core Image Loader.

The following is the procedure to produce a master cassette or magnetic tape containing the configured SOS Editor. The necessary triggers (see page 3-7) should already be generated on paper tape.

1. Using the binary loader, load and start the absolute binary version of the Core Image Loader/Writer (091-000067 for cassette; 091-000068 for magnetic tape). When started this program will output the following message to the console device:

   LOAD UNIT 0; STRIKE ANY KEY

   Place the cassette or magnetic tape reel that is to become the master on unit 0 and depress any key on the console keyboard. This program will write a Core Image Loader/Writer (CILW) to file 0 of unit 0. When the loader has been successfully written the message,

   LOADER INSTALLED

   is printed at the console and the tape is rewound.

2. The binary loader remains in high core. Using this loader, load the absolute binary version of the SOS Relocatable Loader (091-00073 for cassette; 091-000076 for magnetic tape). This program outputs the prompt message (RLDR) to the console device; respond with the command line:

   ```
   CT0:1/S    $PTR/4    (for cassette)
   MT0:1/S    $PTR/4    (for magnetic tape)
   ```

   If the teletype reader is used, substitute $TTR for $PTR. As the relocatable loader requests them, mount the following paper tapes in order:

   a. The trigger to be used for the Relocatable Loader.
   b. The SOS Cassette Library, SOSCT.LB, (099-000041) for cassette or the SOS Magnetic Tape Library, SOSMT.LB, (099-000042) for magnetic tape.
   c. The SOS Library, SOS.LB, (099-000010).

d. The relocatable binary version of the SOS Relocatable Loader, RLDR.RB, (089-000120).

The loader will produce a core image of the tailored Relocatable Loader on file 1 of the master reel, type OK and HALT when finished. This version of the loader is now usuable for the remaining procedures. It will typically need to be reloaded via the Core Image Loader (CIL), but in this one case it is already present in core. Hence, to restart the loader, merely depress the CONTINUE switch on the master console.

3. The same procedure is followed to create the CLI core image file as given in 2. for the Relocatable Loader. The loader outputs the RLDR prompt and the user responds with

CT0:2/S	$PTR/4	or	CT0:2/S	$TTR/4
MT0:2/S	$PTR/4		MT0:2/S	$TTR/4

As the loader requests them, the user mounts tapes for the CLI just as he supplied them for the Relocatable loader, i. e.:

CLI trigger
SOSMT.LB or SOSCT.LB
SOS.LB
CLI.RB (089-000121)

The loader produces the CLI core image file and HALTs.

4. After creating the CLI core image file, the Relocatable Loader must be restored to core using the CIL. Restart the CIL by manually rewinding the tape unit containing the master reel. Set the appropriate device address into the data switches on the master console and depress RESET (cassette address = 34_8; magnetic tape address 22_8). Set data switch 0 to 1 and depress PROGRAM LOAD.

The CIL is invoked and types its prompt message,

#

to the console device. (Thereafter the CIL may be invoked by setting the address of the last location in core into the data switches, and depressing RESET and then START). Respond to this prompt with the command line:

0:1)

Producing a Master Reel (Continued)

4. (Continued)

The user-configured version of the SOS Relocatable Loader will be returned to core.

5. Now, the core image of the Text Editor can be created by following the same procedure used to create the CLI core image file. The loader outputs the RLDR prompt and the user responds with

CT0:3/S	$PTR		CT0:3/S	$TTR/4
MT0:3/S	$PTR	or	MT0:3/S	$TTR/4

As the loader requests them, the user mounts tapes for the Editor just as he supplied them for the Relocatable Loader and the CLI, i.e.:

EDITOR trigger
SOSMT.LB or SOSCT.LB
SOS.LB
EDIT.RB (089-000104)

The loader produces the Editor core image file and HALTs.

6. The same general procedures are used to produce core images of the Extended Assembler, Library File Editor, and SYSGEN on the master reel. These are described in the SOS User's Manual, 093-000062.

The user can now call the Editor by either of the following commands:

BOOT EDIT)

or

#Ò:3)

CHAPTER 4

MULTI-EDITOR

The Multi-User Text Editor (MEDIT) allows several users to be active at the same time. Each user is served as though he/she were the only user, except for a possible slight degradation in response time.

MEDIT has the same commands as the Single-User Text Editor (EDIT) except that the RDOS QTY-driver ignores the keyboard interrupts ↑A and ↑C, and the H command does not return to CLI.

The following sections describe the loading and calling procedures for the Multi-User Text Editor in four operating environments. They are:

1. Unmapped RDOS Background
2. Unmapped RDOS Foreground
3. Mapped RDOS Background
4. Mapped RDOS Foreground

The dump tape 088-000106 supplies the relocatable binary form of MEDIT (MEDIT.RB).

The system using MEDIT must have a QTY (asynchronous-terminal multiplexor). MEDIT supports up to twenty text-editing (QTY) terminals.

When execution begins, MEDIT divides all available memory above location (octal) 6743 + NREL Load Point (MEDIT's NMAX) equally among the text-editing terminals as buffer.

MEDIT needs 13 (octal) page zero words of memory.

LOADING AND CALLING MEDIT

MEDIT is loaded and called from any CLI console. Loading is done by the relocatable binary loader using the RLDR command described in the Real Time Disk Operating System User's Manual, document number 093-000075. The following describes the RLDR command used in loading MEDIT and the commands for executing MEDIT.

Unmapped RDOS Background

The following command creates an executable save file from the relocatable binary file MEDIT.RB:

LOADING AND CALLING MEDIT (Continued)

Unmapped RDOS Background (Continued)

RLDR MEDIT bmedit/S channels/C tasks/K)

where: bmedit is a user-assigned name for the MEDIT save file that is to run in the unmapped background.
channels is the number of channels to be used, in octal. This value must be greater than or equal to (3 times the number of text-editing terminals) +2.
tasks is the number of tasks that MEDIT requires, in octal. This value must be greater than or equal to the number of text-editing terminals +1.

Then call the program by:

bmedit terminals [ticks])

where: bmedit is a user-assigned name for the MEDIT save file that is to run in the unmapped background. This name must be the name given in the RLDR command.
terminals is the number of text-editing terminals to be supported, in decimal. The programmer can only access a terminal within the specified range, e.g., if the programmer specifies 10 terminals, only 0-9 can be accessed.
ticks is an optional argument specifying the number of system clock ticks at which task rescheduling will be forced.

Termination: ↑A or ↑C on $TTI background console.

Unmapped RDOS Foreground

Load by:

RLDR MEDIT fmedit/S channels/C tasks/K NREL-partition/F ZREL-partition/Z)

where: fmedit is a user-assigned name for the MEDIT save file that is to run in the unmapped foreground.
channels is the number of channels to be used, in octal. This value must be greater than or equal to (3 times the number of text-editing terminals) +2.
tasks is the number of tasks the MEDIT requires, in octal. This value must be greater than or equal to the number of text-editing terminals +1.
NREL-partition is the foreground NREL partition address, in octal.
ZREL-partition is the foreground ZREL partition address in octal.

LOADING AND CALLING MEDIT (Continued)

Unmapped RDOS Foreground (Continued)

Call by:

EXFG fmedit terminals [ticks]

where: fmedit is a user-assigned name for the MEDIT save file that is to run in the unmapped foreground. This name must be the name given in the RLDR command.
terminals is the number of text-editing terminals to be supported, in decimal. The programmer can only access a terminal within the specified range, e.g., if the programmer specifies 10 terminals, only 0-9 can be accessed.
ticks is an optional argument specifying the number of system clock ticks at which task rescheduling will be forced.

Termination: ↑A or ↑C on $TTI1 foreground console, or ↑F on $TTI background console.

Mapped RDOS Background

Load by:

RLDR MEDIT mmedit/S channels/C tasks/K

where: mmedit is a user-assigned name for the MEDIT save file that is to run in the mapped background or foreground.
channels is the number of channels to be used, in octal. This value must be greater than or equal to (3 times the number of text-editing terminals) +2.
tasks is the number of tasks the MEDIT requires, in octal. This value must be greater than or equal to the number of text-editing terminals +1.

call by:

medit terminals [ticks]

where: mmedit is a user-assigned name for the MEDIT save file that is to run in the mapped background or foreground. This name must be the name given in the RLDR command.
terminals is the number of text-editing terminals to be supported, in decimal. The programmer can only access a terminal within the specified range, e.g., if the programmer specifies 10 terminals, only 0-9 can be accessed.
ticks is an optional argument specifying the number of system clock ticks at which task rescheduling will be forced.

Termination: ↑A or ↑C on the console $TTI.

Mapped RDOS Foreground

The load command is the same as for the mapped RDOS background.

Before calling MEDIT, the user must ensure that the foreground size in 1024-word blocks is greater than or equal to 4 + (terminals times buffer-size-in-blocks-per-terminal). (MEDIT occupies slightly less than four blocks.) A one-block buffer for each of four text editing terminals allows a page size of about 2K characters.

The background size determines what can be run in the background. The CLI needs about 5K words.

To determine how much memory is in the system, the user must issue the command

GMEM)

which displays

BG: background blocks FG: foreground-blocks

showing the number of 1024-word (in decimal) blocks in background and foreground.

If necessary, the memory partition can then be moved by:

SMEM background-blocks foreground blocks

where the sum of the two arguments equal the sum of the two arguments displayed by GMEM.

MEDIT can be called from the foreground CLI by:

mmedit terminals { ticks }

MEDIT can be started in an inactive foreground from the background CLI by:

EXFG mmedit terminals {ticks}

where:

mmedit is a user-assigned name for the MEDIT save file that is to run in the mapped background or foreground. This name must be the name given in the RLDR command.

terminals is the number of text editing terminals to be supported in decimal. The programmer can only access a terminal within the specified range, e.g., if the programmer specifies 10 terminals, only 0-9 can be accessed.

ticks is an optional argument specifying the number of system clock ticks at which task rescheduling will be forced.

Termination:

↑A or ↑C on foreground $TTI1 or ↑F on background console $TTI.

CHAPTER 5

ERROR MESSAGES

During the editing process various errors may occur; the outputted error messages and their meanings are shown below. Where error messages are only output in a given operating environment, the appropriate environment(s) are listed in parentheses. Other error messages are applicable in all operating environments.

ERROR MESSAGE	MEANING
COMMAND BUFFER FULL; EXECUTING COMMAND. TEXT BUFFER FULL DURING INSERT	Command string exceeds capacity of edit buffer.
TEXT BUFFER FULL WHILE READING	Attempting to append a page when the buffer is full. During a read or append the buffer capacity is exceeded. A partial page has been read in or appended.
FILE CAN'T BE USED FOR INPUT	Attempt to read a read-protected file. (RDOS, SOS)
FILE CAN'T BE USED FOR OUTPUT	Attempt to write to a write-protected file. (RDOS, SOS)
FILE NAME IN USE	Attempt to create an output file when that file name already exists in the file directory. (RDOS)
ILLEGAL FILE NAME	File name does not conform to a legal operating system file name. (RDOS, SOS)
ILLEGAL WINDOW WIDTH	Editor is already in page mode when a W command was issued. (RDOS)
MACRO ERROR	Undefined or recursive macro.
NO OUTPUT FILE	Attempt to issue output command, without first specifying an output file. (RDOS, SOS)

ERROR MESSAGES (Continued)

ERROR MESSAGE	MEANING
NO SUCH FILE	Attempt to specify an input file which doesn't exist. (RDOS, SOS)
OUTPUT ALREADY ACTIVE	Attempt to get for writing an output file which has not been closed, and is still active. (RDOS, SOS)
PARITY ERROR IN LINE n	During a read, a parity error occurred in line n. When examined, the character in error will be replaced by " ".
STR NOT FOUND	Unsuccessful string search.
??command string	Editor cannot understand or cannot execute command. It outputs the remainder of the string to which the message refers. If this message occurs while using one of the operating systems and the command buffer contained at least 256 characters when this message occured, the following message will be printed, and the user may follow the procedure outlined below to recover his command buffer:
SAVE COMMAND BUFFER YES (1) OR NO (2) ?	The command buffer contained at least 256 characters when either an illegal command or a RDOS interrupt was detected. If the user responds with anything other than a 1) then the command buffer is deleted and the prompt character "*" is reissued to await a new command. If the user responds with 1) the message: ENTER FILE NAME is printed instructing the user to specify

ERROR MESSAGE	MEANING
SAVE COMMAND BUFFER YES (1) OR NO (2) ? (Continued)	the file name in the following manner: xxx) where xxx is any valid output file. The contents of the command buffer will then be written to the specified file. When this operation is complete the Editor issues the prompt character "*" to await a new command. If the specified filename is not a valid output file, the message: ENTER FILENAME will be printed again.

APPENDIX A

COMMAND SUMMARY

COMMANDS USED IN ALL ENVIRONMENTS

COMMAND	MEANING	PAGE
↑P	Change tab simulation (initially simulated).	1-2
RUBOUT	Cancel and echo previous character entered.	2-2
.	Display the line number the CP is pointing to.	2-15
:	Display the number of lines in the buffer.	2-15
=	Display number of characters in the buffer.	2-15
I string $	Insert with tabulation in front of CP.	2-15
A	Append a page, or windows to buffer.	2-3
B	Move CP to beginning of the buffer.	2-6
Cstring$string$	Change string, set CP.	2-10
nD	Delete n characters ahead of CP.	2-12
E	Copy contents of input file to output file.	2-17
F	Issue a form feed to output file.	2-17
nF	Output n inches of leader.	2-17
Istring $	Insert string in front of CP.	2-11
nI	Insert character using the decimal representation.	2-11
''nI	Insert character using octal representation.	2-11
nJ	Jump CP to line n.	2-7
nK	Delete n lines ahead of CP.	2-13
L	Move CP to beginning of the line.	2-7
nL	Move CPn lines from CP.	2-7
nM	Move CP n characters.	2-7
Nstring $	Perform a continuous search for string, starting at CP, punch pages.	2-9
P	Punch the buffer ending with a form feed. In window mode, just output the buffer.	2-18

COMMAND	MEANING	PAGE
nP	Starting at CP, punch n lines ending with a form feed	
	in window mode, just output n lines.	2-18
PW	Punch the buffer.	2-18
nPW	Punch n lines, starting at CP.	2-18
Qstring $	Perform a continuous search for string, from CP.	2-9
nR	Punch n buffers and read n pages or window frames.	2-18
Sstring $	Search for string, beginning at CP.	2-8
T	Type buffer.	2-8
nT	Type n lines of buffer, starting at CP	2-8
XM	Create a macro.	2-13
nX	Execute a macro n times.	2-13
XD	Delete a macro command.	2-14
X?	Types out contents of current macro.	2-14
Y	Yank a page or window frame.	2-2
nY	Yank n lines.	2-3
Z	Move CP to end of buffer.	2-8

COMMANDS USED IN RDOS

COMMAND	MEANING	PAGE
n#←	Resets the number register to n.	2-16
#+	Increments the register by 1.	2-16
#-	Decrements the register by 1.	2-16
#?	Types the contents of the register.	2-16
#0	Outputs the register to the text in front of CP.	2-16
n#!...$!...$	If the register doesn't match n, skip the command characters up to the next exclamation escape.	2-16
↑A	Scraps command buffer.	2-16
↑C	Terminates the Editor.	2-17
↑Z	Search for string matching all characters	2-16
CANCEL (↑X)	Scraps current line of the command.	2-16
GC	Closes the input file and the output file without data transfer.	2-20
GRfilename$	Gets for Reading, closing the previous input file if any.	2-4
GOfilename$	Closes current output file and opens new output file.	2-21
GWfilename$	Creates and opens a new file for writing.	2-19
H	Return to CLI.	2-20
U?	Types I/O file names.	2-16
UCfilename$	Closes I/O files, restores input file's attributes and renames the output file.	2-22
UDfilename$	Deletes a file.	2-14
UE	Punches remaining input file, closes I/O files, deletes input file, and renames output file according to original input.	2-21
UH	Closes I/O files, resets attributes.	2-23
UNfilename$	Creates a permanent file and opens it for input; opens filename.SC for output.	2-4

COMMANDS USED IN RDOS (Continued)

COMMAND	MEANING	PAGE
URfilename$ filename$	Renames a file.	2-15
US	Punches remaining input file, closes I/O files, restores the attributes of the input file, changes its name extension to .BU, and rename the output according to original input.	2-22
UYfilename$	Makes filename permanent, opens it for input and yanks in the first page or window frame. Creates and opens filename.SC for output.	2-5
UZfilename$	Removes the permanent attributes of a file, so it can be renamed or deleted.	2-15
W	Resets mode.	2-17
W?	Displays mode status.	2-17
nW	Set window mode with a width of n lines.	2-17

COMMANDS USED IN RDOS

COMMAND	MEANING	PAGE
GC	Closes input file, and output file.	2-24
GR	Get for Reading, closing previous input file if any.	2-6
GW	Get for Writing.	2-23

INDEX

DataGeneral

SOFTWARE DOCUMENTATION
REMARKS FORM

Document Title	Document No.	Tape No.

SPECIFIC COMMENTS: List specific comments. Reference page numbers when applicable. Label each comment as an addition, deletion, change or error if applicable.

GENERAL COMMENTS: Also, suggestions for improvement of the Publication.

FROM:

Name	Title	Date

Company Name

Address (No. & Street)	City	State	Zip Code

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