Octal Editor User's Manual

093-000084-02

For the latest enhancements, cautions, documentation changes, and other information on this product, please see the Release Notice (085-series) supplied with the software.

Ordering No. 093-000084

Revision 02, July 1977

DataGeneral
SOFTWARE DOCUMENTATION

Octal Editor User's Manual
093-000084

Revision History:

Original Release - April 1973
First Revision - January 1975
Second Revision - July 1977

This document has been extensively revised from revision 01; therefore, change bars have not been used.

CONTENTS

OCTAL EDITOR

FEATURES

The OEDIT program provides you with powerful tools for editing a disk file. Using OEDIT, you can:

- Examine and modify disk words
- Search your file for a specific word
- Dump any portion of your file to terminal or printer
- Fill any portion of your file with any word
- Extend any random or sequential file

You may examine your file in any of four different formats, choosing any number base ranging from 2 through 16 for displaying and altering the contents of the file.

INVOKING OEDIT

Invoke OEDIT by typing:

```
OEDIT FILENAME )              ( ) = carriage return)
```

Filename specifies any file existing under RDOS. If you fail to enter "filename", OEDIT prints:

```
ERROR:  NO FILENAME SPECIFIED
```

and control returns to CLI. If OEDIT cannot find "filename" in your RDOS system directory, it prints:

```
ERROR:  A NON-EXISTENT FILE: FILE XXX
```

and returns control to CLI.

On finding "filename", OEDIT prints a carriage return/line feed and a period (.), indicating that you may now enter your commands.

FORMAT

Commands consist of "expressions" followed by a command code. You may use either of two types of commands: Local and Escape. Local commands concern specific operations on an expression (and only on that expression), while Escape commands perform more general functions, such as selecting a number base for OEDIT output. Local commands do not use the ESC character (shown in the manual as $); otherwise, both commands have the following format:

[expression] [$] command code

(where brackets, [ ], denote optional arguments).

Expression addresses some word location in your file, or it may be interpreted as a value to be stored to your file; or, expression may simply be a value or a sequence of numbers, operators or alphanumeric characters that you want calculated, with the result displayed in the format of your choice. The command code governs which interpretation OEDIT will use.

When you wish to address your file, "expression" has either of the following forms:

displacement

or

record number : displacement

Displacement addresses a word location in your file that you want accessed.

Record number : displacement also addresses some word location in your file, and is an optional addressing feature in OEDIT.

You may choose to consider your file as segmented into "records" of some specific length, and then access any or all of these records while editing. Rather than typing the word location for each entry, you may enter a record length into the record size register (see $L command), then address your file with record number : displacement, where displacement pertains to that record you are editing. This addressing scheme lends itself handily to certain file types, eliminates time-consuming external calculations, and enhances accuracy.

You can also specify a base address to be added into all address calculations. You set this with the base register (see $B command); it has the default value 0 until you set it.

OEDIT computes the "effective" file address using the following method:

base-address + (record-number * (record-length)) + displacement

The command code directs OEDIT to perform an operation. A discussion of the two types of command codes, Local and Escape, follows the next topic. The sequence ↑A (typed by depressing the CTRL and A key simultaneously) will abort any command.

KEYBOARD CALCULATIONS

OEDIT can calculate the value of any expression in a command. Expressions may contain operators +, - and ! (exclusive OR), numerics (including decimal digits), parentheses and alphabetic characters. You may request a calculation with both Escape and Local commands.

The period character (.) in OEDIT has two functions; used following numbers it indicates decimal notation, and used alone or in conjunction with operators it represents the current displacement value. For example:

```
  14+.=
```

prints a 21 if the current displacement (last file address accessed) is 5.

A period followed by an equal sign, .=, prints the current displacement. If you type 14.+.= OEDIT prints 23, again assuming the current displacement is 5. The first period tells OEDIT to interpret 14 as a decimal value; the second period is replaced by the value 5 in internal calculations. If you type:

```
  (10.+2)+(17-7):4+5$C
```

and the record register ($L) contains 100, OEDIT returns 24:11, and if you input:

```
  (2+5)-(4+2)=
```

OEDIT returns the value 1. Note that in the above examples, the expressions are simply values calculated and displayed. You are not accessing your file, nor are you storing some value to your file. The Local command =, and the Escape command $C specify the function: calculate and display the result.

However, depending upon the command code, OEDIT can calculate the value of an expression (as in the above examples) and use the result to access your file. If you type:

```
  (10.+2)+(17-7):(4+.)/
```

OEDIT derives a file address of 24:11, and will display the contents of record number 24, displacement 11. The command code (in this case /) directs OEDIT to display the contents of the specified address for possible modification.

Calculation arguments have the following forms:

- Double precision octal or decimal number
- "charactercharacter, where character is a single alphanumeric. The characters are packed left to right in a word with null fill. For example:

```
    " =    0
    "A =   040400
    "AA=   040501
```

LOCAL COMMANDS

Local commands direct OEDIT to perform an operation on the expression you enter with the command. Escape commands (see the next section) generally have broader effects on OEDIT, such as specifying a certain number base for all output.

A summary of Local commands follows:

Local Command	Description
/	Open the current word, and display its contents in the current format. This command allows you to modify a word. For example: `.451/  040501` (Note the preceding period; this is a prompt for your command.) You requested a display of location 451 in your file. OEDIT printed its contents, in this case the value 040501. You may wish to modify the contents of location 451, or you may wish to see what ASCII characters are equivalent to 040501 - see the next command.

Local Command   Description

'

Apostrophe. Display the last word or expression in ASCII format. For example,

```
.451/  040501 ' AA
```

Apostrophe directs OEDIT to display the expression you enter, or the value it has just printed, in ASCII format. The apostrophe command does not open any location, nor does it store any value to your file. Apostrophe just translates expressions to their ASCII equivalent. (Thus, the ASCII equivalent of 40501 is AA.)

←

Left Arrow. Display the last word or expression in half-word format as two octal numbers. For example,

```
.451/  040501 ' AA ← 101 101
```

This example illustrates opening and displaying location 451 using local command /. OEDIT types 040501; then you request that same word in ASCII format using Local command ' (apostrophe). The last command, ←, requests half-word format in octal; OEDIT complies by printing 101 101.

=

Equal sign. Display the last expression in word format in the current base.

Again the local command =, like ' and ←, supplies you with a different "filter" (so to speak) to view an expression. If the current output format is ASCII and you type .451/, OEDIT will type AA. If you then wish to see what AA looks like in word format, then use the command = and OEDIT will type the word equivalent of AA (which happens to be 040501 in base 8).

The command line would look like:

```
.451/ AA = 040501
```

Local Command   Description

*

Asterisk. Display the last expression as an octal word.

This command overrides the current number base, and displays only in octal. (You control the number base with Escape command $N; see the next section for specifics.)

)

Carriage Return. Store the result of the expression entered into the currently-opened word. If you do not enter an expression, or if no word is open, then carriage return is a no-op. For example:

```
.451/ 040501 ' AA "BB)
```

Here you key in the value BB, which alters the contents of location 451 which previously held AA.

```
.451/ 040501 )
```

Here you close location 451; no modification is made.

↓

Line Feed. Store the result of the expression in the currently-opened word, and display the next word.

```
.451/ 040501 "BB)
.452/ 056411
```

This example illustrates opening location 451, modifying it, then opening location 452 with a new command. The following command accomplishes both:

```
.451/ 040501 "BB↓
      000452 056411
```

Line feed (↓) closes location 451 after modifying it to BB, then OEDIT prints the address of the next location, along with its contents.

↑

Up-Arrow. Store the result of the expression in the currently-opened word, and display the previous location.

```
.451/ 040501 ↑
000450 026377
```

Local Command	Description
↑ (cont'd)	In the above example, up-arrow closes location 451 without modifying it, since the programmer at the console did not enter any expression after OEDIT displays its contents. OEDIT then types the address of the previous location, along with its contents.
n]n	Byte format. You can enter bytes by opening a location, typing the value you want in the left byte, a right bracket, then the value you want in the right byte. For example: `10/000000 101]101)` `10/040501 ' AA`
$ (ESCape)	Escape. Enter Escape Command Mode; the character following ESC ($) is the command.

ESCAPE COMMANDS

Escape commands affect general functions in OEDIT. To call an Escape command, type the ESCape key, then the command code.

Escape commands terminate a command line. OEDIT expects one, and only one, character to follow escape. When you type that character, OEDIT performs some operation; no other information is requested of you. When OEDIT completes the task you called for with the command, it types a prompt, or opens a special register for modification.

A brief summary of each escape command follows; you'll find a more detailed description in the next section.

Escape Command	Description
$M	Open the mask register for display and modification. You will need this register to search your file; it allows you to ignore bit positions in words within your file when you're searching for a specified value.
$W	Open the word register for display and modification. You need this register for both Search and Fill commands.
$N	Open the number register for display and modification. This register defines the output format for OEDIT. Your entries to this register direct OEDIT to zero-suppress its display of file words; treat words in your file as signed values, printing them accordingly; and display any word in your file in any number base from 2 through 16.
$J	Open the increment register for display and modification. This register governs searches, dumps and fills to your file. If you desire, you may direct OEDIT to search, dump and fill every Jth position in your file. If this register contains 1, OEDIT accesses each word location in your file while performing these commands.
$L	Open the record size register for display and modification. An entry to this register allows you to refer to word locations in your file in terms of record number:displacement.
$B	Open the base address register for display and modification. This number will be added to each address.
$H	Open the output device register for display and modification. The output device register directs OEDIT output from search and dump commands to either the input device or line printer.
$S	Search the file for a specific 16-bit value.
$D	Dump the specified file locations.
$F	Fill the specified file locations.
$A	Print the record number for the segment of your file presently in core.
$C	Calculate the preceding expression and print the result in terms of record number:displacement. If no expression preceeds this command OEDIT prints the current location.
$Z	Close all files and return to CLI.

ESCAPE COMMAND DESCRIPTIONS

Word ($W) and Mask ($M) Registers

OEDIT allows you to search your file for any 16-bit value with the $S (search) command. As a prerequisite to searching your file, you must enter values into both the word and mask registers. The fill ($F) command also uses the word register for the value you want inserted into your file.

For searches, first set the word register to represent the value you're looking for in your file. Next, set the mask register to mask specific bits of the file words (if desired). While OEDIT searches, it ANDs the value in a file location with the mask register, then compares it to the contents of the word register. If they match, OEDIT types the file location and its contents.

Both word and mask registers initially contain 0. If you leave them that way, and you request a search, all file locations match and your entire file will be printed.

$W	Opens the word register
$M	Opens the mask register

Example:

```
.$W 000000 401)
```

Open the word register and store 401

```
.$M 000000 777)
```

Open mask register and mask bits 0 thru 6

```
.$S
```

Search the entire file, and print those locations that have a value 401 in bit positions 7 thru 15; ignore bits 0 thru 6.

Number Register ($N)

The number register governs the base of the number system (octal, decimal, etc.) OEDIT uses for printing file words. The number register can also specify zero suppress and sign control. You may choose any number base ranging from 2 through 16.

Code the number register as follows:

Bit 0 = 1	Treat the number as signed.
Bit 1 = 1	Suppress leading zeroes
Bits 12 - 15 = base	Any integer from 2 through 16.

Initially the number register contains 8., so OEDIT will print values in octal, will not suppress leading zeroes, and will treat numbers as unsigned. To open the number register for display and/or modification, type:

```
$N
```

Examples:

```
.33/  000035 )
.$N  000010 10.)
```

Display location 33 and close that location, then open the number register for display and modification. Current mode is octal 10 (base 8.); change base to decimal (10.) and close number register.

```
.33/  00029 )
.$N  00010 40000+10.)
```

Open location 33 for display; note that its contents are now displayed in decimal. Then open the number register, set bit 1 to 1 and set base to base 10. (decimal).

```
.33/ 29 )
.$N 16394 16.)
```

Open location 33 again; note that leading zeroes are suppressed. Then open the number register again. Viewing its contents in decimal seems misleading, but $16394_{10} = 40000_8 + 10$. Set number register to base 16., and close it.

```
.33/ 001D )
```

Now you see the contents of location 33 in base 16.

Increment Register ($J)

OEDIT uses the increment register in conjuction with three commands: DUMP, FILL, and SEARCH. After performing either a search, dump, or fill to the starting address in each of these commands, OEDIT adds the value in the increment register to derive the next location to search, dump or fill.

If you wish, for example, to examine every 10th word in your file, simply store 10. into the increment register rather than entering each word location separately, then request a dump. OEDIT will dump the first word in your file, then add the value in the increment register to the previous address, type out the contents of the new address and so on until it reaches the end-of-file.

OEDIT will interpret the value 0 in the increment register as 1, since 0 is meaningless in this context. You may enter negative values into the increment register to back up through your file while performing searches, dumps or fills. The contents of the increment register may not exceed the record length in the record size register; if record size = 0 then you can put any value in the increment register.

To open the increment register, type:

```
    $J
```

Example:

```
    .$J 000001 12)
```

This command opens the increment register for display nd modification, and stores the value 12_8 as the new increment. Any $S, $D or $F commands that follow will act only upon every tenth word location until you modify the increment register again.

Record Size Register ($L)

When you want to address your file with record number: displacement, type $L and enter a record length. The record size register is initially set to 0. Like all special registers, you can modify the contents of the record size register anytime; that is, you may choose to edit your file in terms of file displacement, then use record numbers, and later return to file displacement.

Example:

```
    .$L 000000 )
    .126/ 034761 )
    .$L 000000 100 )
    .1:26/ 034761 )
```

Location 1:26 references file location 126 when record size = 100.

Base Address Register ($B)

When a set of data begins at an offset into a file, you can use the base register to display addresses using displacements from the base. The base address register is initially set to zero. Like other special registers, you may modify it at any point in your editing.

Example:

```
    .$B 000000 )
    .126/ 034761 )
    .$B 000000 100 )
    .26/ 034761 )
```

Location 26 in the last line refers to base + 26, i.e., location 126.

You can also use the base register together with the record size register. In this case, record numbering starts with the base.

Example:

```
    .$B  000000 )
    .$L  000000 )
    .126/  034761 )
    .$B  000000 100 )
    .$L  000000 10 )
    .2:6/  034761
```

Location 2:6 equals the base + twice the record size + 6, where the base is 100 and the record size is 10.

Output Device Register ($H)

You may direct output from searches and dumps to the line printer by entering a nonzero value in the output device register. Initially this register contains zero.

Example:

```
    .$H 000000 1)
```

Open output device register, store a 1, and close the register. All output from SEARCH ($S) and DUMP($D) commands is directed to the line printer until you reset this register to zero.

SEARCH Command ($S)

This command searches all or part of your file for any 16-bit value, prints file locations that contain that value, and the contents of those locations. Execute the following steps to perform a search:

- Set the mask register to the mask desired for the search
- Set the word register to the value you're looking for
- Set the increment register to the desired increment value
- Specify the SEARCH command, $S.

The SEARCH command ($S) will search your entire file unless your command line specifies otherwise. (See below)

To find a match, the search ANDs a word from your file with the contents of the mask register. If the result equals the contents of the word register, it is a match; if the two differ, it is not a match.

Initially both mask and word registers are set to 0 and the increment register is set to 1.

After the first compare, OEDIT adds the value in the increment register to the starting address of the search for the next location. Comparing and incrementing continues until it has searched all the specified file area.

To invoke the SEARCH command, type:

```
$S
```

The $S will search your entire file; however, you can restrict the search limits by using one of the following command formats:

```
file address $S
```

searches your file from the beginning to the file address specified.

```
file address < $S
```

searches your file from the address specified to the end of the file.

```
file address < file address $S
```

searches your file between the two specified addresses.

Example: Find all words containing 016401 in your file.

```
.$W 000000  16401)
.$M 000000  -1)
.$J 0000005 1)
.$S
```

This sequence places the value you're seeking into the word register, sets a mask of no bits (-1 =177777), makes the search increment 1, and searches the entire file.

If it found any matches OEDIT would print them as:

```
000120 016401
000436 016401
007462 016401
```

In this example there were three matches on your file.

DUMP Command ($D)

This command dumps all or part of your file to an output device. Unless you specify otherwise in the command line, it dumps the entire file. The same formats apply to this command as with the SEARCH command:

```
$D
file address $D
file address< $D
file address < file address $D
```

DUMP uses the value in the increment register to dump only selected locations. DUMP output includes file addresses and the data within those locations.

Example: Dump words 10 through 20 in your file.

```
.$J 000001 )
.10<20$D
```

In this example, you first verify the contents of the increment register, then enter the DUMP command. If the current format specifies octal words, the output from this dump might look like:

```
000010 005126 000023 000000 004010
000014 001342 000000 003524 002041
000020 005322
```

Note that only one file address appears per line, but four file words print on each line in the octal word format. (If the format called for base 2, then only two words would print in each line.) In the above example locations 10, 11, 12, and 13 of your file print on the first line, words 14, 15, 16, and 17 on the second line and location 20 on the third line.

FILL Command ($F)

The FILL command writes the word in the word register to one or more words in your file. You first select the fill word and place it into the word register, and select the increment value and enter it into the increment register. When you key the $F command, it writes the word from the word register to your file. As a precautionary measure, OEDIT prints "REALLY DO IT?" Whenever you give the FILL command; you must respond with the character 'Y' to execute the command. Any other response aborts. See the SEARCH ($S) command for limiting formats and their descriptions.

Example: Fill your file from its beginning to location 377 with the null word (000000).

```
.$W 016401 0)
.$J 000012 1)
.377$F
REALLY DO IT? YES
```

In this example, you first enter the fill word into the word register, then enter the desired increment into the increment register, then type the correct command. When OEDIT completes the fill operation locations 0 through 377 of your file contain the null word (000000).

Print the Record Number Currently in Core ($A)

If you entered a value into the record size register, and you are thus editing your file in terms of record number:displacement, the $A command will print the record number for the portion of your file that you are presently editing. This command is useful when you change record lengths or when you are using a series of line feeds and operators in your expressions.

```
.5:31/ 034521 )
.$A
000006
.
```

Print Result of Expression Calculation ($C)

The $C command calculates an expression and prints the result as record number: displacement. If you key in no expression before the $C command, OEDIT prints the current file location.

```
.$C
10:53
.5+9.:.-2$C
16:51
```

In the first example, OEDIT prints the current location; in the second example, it calculates and displays the result.

Terminate and Return to CLI ($Z)

This command terminates OEDIT and returns control to the CLI.

Example:

```
 .$Z
R
```

CLI prints "R", its prompt.

Output Format Commands

These commands let you select the output format. The format you select remains in effect for all output until you explicitly change it. The general format of these commands is:

$x

Where x represents one of the following characters:

=	Word format
'	ASCII format
-	Half-word octal format
*	Octal word format (number register changed to 8.)

ESCAPE COMMAND DESCRIPTIONS

Word ($W) and Mask ($M) Registers

DISKEDIT uses both the Mask and Word registers to search the disk.

For searches, the user sets the Word register to represent the value sought on the disk. The mask register is set to mask specific bits of words (or to mask no bits). As the search proceeds, the value of each location is first ANDed with the Mask register, then compared with the contents of the Word register. If they match, the location and its value are printed out. When DISKEDIT starts, the Word and Mask registers each contain 0, which will print all disk locations in a search command.

The Word register is opened by the command:

```
$W
```

The Mask register is opened by the command:

```
$M
```

An example of each command is:

```
.$W 000000 1500)  Open Word register and
                  store 1500
.$M 000000 17700) Open Mask register and
                  mask bits 0-9.
```

Number Register ($N)

The Number register determines in which base numbers will be printed, when displayed in the word format. This register also defines the output format of numbers and is coded as follows:

100000	treat the number as signed
40000	suppress leading zeroes
base	base is an integer from 2 to 16, which defines the base of the number on output.

By default the Number register is set to 0, which prints out the register contents in octal. If the user sets the contents of the Number register to a base number, that number becomes the output base number. Only base 2 through base 16 are valid. Output formats are discussed at the end of this manual.

The Number register is opened for examination and modification by the command:

```
.$N
```

Some examples of the command are:

```
.$N 000010 10.)   - Open Number register and
                    set number base to 10 (note
                    period: 10.)
.$N 00010 16.)    - Open Number register and
                    set number base to 16.
.$N 0010 10 )     - Open Number register and
                    set number base to 8. (No
                    period follows base
                    specification).
.$N 000010        -Open Number register, sup-
     40000+10 )     press leading zeros, and
                    set base at 8.
.$N 40010)        - Open and examine the Num-
                    ber register.
```

Increment Register ($J)

DISKEDIT uses the Increment register to search the disk. The user specifies a value in this register which adjusts the increment between search locations. The value in this register can also be used in Fill and Dump commands. If the value is zero or negative, then 1 becomes the increment.

The Increment register is opened for examination and possible modification by the command:

```
$J
```

An example of the command is:

```
.$J 000000 12)   - Open Increment
                   register and store 12.
```

Search Command ($S)

This command searches all or part of disk for a specified value, and prints locations on disk which contain the value.

The following steps will perform a search:

- set the Mask register to the mask desired for the search.

  set the Word register to the value sought.

- set the Increment register to the desired increment value.

- specify the Search command. The Search Command ($S) will search the entire disk unless the search command line specifies otherwise (see next page).

The Search command combines the following information to find a match:

Mask-Word AND Disk Word = Word for match condition

Where: Mask is the contents of the Mask Register and Word is the contents of the Word Register. disk word is a word from the disk. AND represents logical AND.

When DISKEDIT starts, both Word register and Mask register contain 0, and the search increment is 1.

On matching conditions the location is printed. After a match, DISKEDIT adds the value in the Increment register to the match location, and continues searching. Comparing and incrementing continues until all the specified disk area is searched.

The Search Command is:

```
$S
```

The Search command can restrict a search to specific portions of the disk.

The command

```
disk address:displacement$S
```

searches the disk from the beginning to the specified address and displacement.

The command

```
disk address:displacement<$S
```

searches from the specified address and displacement to the end of the disk.

The command

```
disk address:displacement<disk address:
     displacement$S
```

searches between the two specified addresses and displacements.

An example of the Search command is:

To find all words containing 1 on the disk:

```
.$W 000000 1 )   - Set Word register for
                   search: value 1.
.$M 000000 -1 )  - Setting Mask register to
                   -1 causes default: register
                   will compare all bits of
                   words.
.$J 000000 )     - Set Search Increment
                   register-default increment
                   value is 1
.$S              - Search the disk
```

Results in:

```
000000:102\000001
001020:052\000001
```

This example indicates that only the two listed words contained a 1.

Dump Command ($D)

This command dumps all or part of disk to an output device. Unless the user specifies otherwise in the command line, the entire disk is dumped. The incremental value in the Increment register can dump selected incremental locations. All dumped outputs include the disk address and all the data within the location. Samples of dumping include:

The command:

```
$D
```

dumps the whole disk.

The command:

```
disk address:displacement$D
```

dumps from the beginning of the disk to the specified address

The command:

```
disk address:displacement<$D
```

dumps all locations from the specified address and displacement to the end of disk.

The command:

```
disk address:displacement<disk address:
     displacement$D
```

dumps all locations between the two specified addresses and displacements.

An example of this command is:

```
.1045:0<20$D     - Dump words 0-20 of block
                   1045.
```

The results are:

```
001045:000/005126 000023 000000 004010
     001342 000000 000000 003524
001045:010/000000 003013 002041 000000
     000000 006551 000000 003311
001045:020/005322
```

Fill Command ($F)

The Fill command writes a word from the word register into specific location(s). The user selects the Fill word and places it in the word register, and the $F command writes the word to specified locations. The incremental value in the Increment register can Fill incremental locations with this word. After specifying the Fill command the user is asked: "REALLY DO IT?"

Samples of the Fill Command are:

The Fill command alone:

```
$F
```

fills the entire disk with the value in the Word register.

The command:

```
disk address:displacement$F
```

fills the disk from the beginning to the specified address with the value in the Word register.

The command:

```
disk address:displacement<$F
```

fills all locations from the specified address to the end of disk with the value in the Word register.

The command:

```
disk address:displacement<disk
        address:displacement$F
```

fills all locations from the first address and displacement through the ending disk address and displacement with the value in the Word register.

Once the Fill command is specified, DISKEDIT asks:

"REALLY DO IT?"

Valid responses are Y (Yes) or N (No). Y writes the specified word into the specified location(s); N aborts the command:

An example of the Fill command is:

```
.$W 000000 -1 )     - set Word register.
.$J 000000)         - open Increment register for
                      display
.$F                 - no restriction; fill the entire
                      disk.
REALLY DO IT? Y
```

The Fill command then writes a -1 to all blocks of the disk.

Force Block to be Read Into Core Command ($R)

This command forces a user-specified disk block to be read into core. This command is useful when the original (unedited) version of the previously requested disk block is required. The edited copy of the block is eliminated when the original version is read into core.

Changes to a block are not written onto the disk until a different block is accessed, or until an $O or $Z command is entered.

After the original version of the block is in core, editing can continue.

The format of the $R command is:

```
disk address:$R
```

An example of the $R command is:

To force current block to be read in again type:

```
.$R
.
```

Force Block to be Output to Disk ($O)

This command forces the block currently in core to be output to disk. $O is often used to clear core for a new block or to change a block's location on the disk. After specifying the $O command, the user is asked "REALLY DO IT?"

The command to force the current block in core back to where it came from is:

```
$O
```

The command to force the current block in core to a new location is:

```
disk address:$O
```

DISKEDIT will then output this message:

```
    REALLY DO IT?
```

Valid responses are Y (Yes) or N (No). A Y response outputs the block; an N response aborts the command.

Print the Block Number Currently in Core ($A)

This command prints the block number currently in core, in the selected block number format.

The format of this command is:

```
    $A
```

For example:

```
    .5:01/000000)        - Type block into core
    .$A                  - Print number of block in
                           core
    000005
    .102:53/126412)      - access another block
    .$A                  - print its number
    000102
```

Set the Current Core Block to a Different Address($B)

Ordinarily, the block in core is sent back to its original disk address. The $B command specifies a different address for the core block; the $O command will then write the block to the address specified in $B.

The format of the $B command is:

```
    disk address:$B
```

For example:

```
    .5:01/002300)        - Type block into core
    .$A                  - Print block
    000005
    .10:$B               - Set block to different
                           destination
    .$A                  - Print new destination
    000010
    .01/002300LINE FEED
    000010:0021/123456
```

Print Out Results of Expression Evaluation ($C)

This command calculates an expression and prints the results.

$C command formats include:

```
    $C
```

which prints the current address and displacement.

The command

```
    disk address:displacement$C
```

Calculates the specified expression and prints the results.

Three examples are:

```
    .10:53/100245        - displays contents of disk
                           location
    .$C                  - evaluate expression using all
                           defaults
    000010:053
    .5+5:50+30$C         - evaluate expression and
                           display result
    000012:100
    .$C                  - display current address and
                           displacement.
    000010:053
```

Restart the DISKEDIT Program ($U)

This command restarts the program at the "DISKEDIT DISK DRIVE MODEL NUMBER?" query, and resets all registers to their initial values.

The command format is $U.

Write Modified Core Block and Halt ($Z)

This command writes any current modified core block back to disk and halts the DISKEDIT program. The CONTINUE console switch restarts the program at the "DISK TYPE" question.

The format of this command is:

```
    $Z
```

OUTPUT FORMAT COMMANDS

These commands enable the user to specify and change the form of his output. A specified format will apply to all future printouts unless it is explicitly changed.

The format of these commands is:

```
$x
```

Where: x represents one of the following characters:

```
= - word format
' - ASCII format
← - half-word octal format
* - word format (set number register to 10
    for octal retrieval)
: - output disk block numbers in double
    precision output format
\ - output as two octal words
, - output in head, sector, cylinder format
    (Fixed-head disks only)
```

Examples of the commands include:

```
.$*                     - set to output in octal word
                          format
.15+2+3=000022)         - display results in octal
.$N 000010 10.)         - set to output numbers in
                          decimal
.15+2+3=00018)          - display expression in
                          decimal
.$N 00010 40000+        - output numbers in decimal,
    10.)                  suppress leading zeroes
.15+2+3=18)             - try calculation again
.$*                     - set to output in octal word
                          format
.0:0/040501_'AA)        - display a disk word, examine
                          it in ASCII
.$'                     - set to display words in
                          ASCII
.0/AA=040501←           - display disk word in ASCII,
    101 101)              then in half-word format
.$←                     - set to display word in half-
                          word format
.0/ 101 101)            - do it
.$=                     - set to display in word
                          format
.0/040501)              - try it
.$:                     - set to output in double
                          precision
.15:45+2$C              - calculate disk octal address
                          and display result
000015:047
.$,                     - set to output disk address in
                          head, sector cylinder
                          format
.5:45$C                 - display disk address
0,5,0:045
.$\                     - set to output in double word
                          format
.5432:53$C              - disk address, calculate
000000\005432:          - try it
    053
```

END OF MANUAL