Symbolic Editor
(SEDIT)
User's Manual

093-000160-00

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. 093-000160

Printed in the United States of America
Revision 00, March 1978

Symbolic Editor User's Manual

093-000160

Revision History:

Original Release - March 1978

Preface

This manual describes the Symbolic Editor - a disk file editing utility which allows you to examine, analyze, and modify the contents of disk files. The Symbolic Editor (SEDIT) works on random or contiguous files produced within Data General's Real-Time Disk Operating System (RDOS) or Diskette Operating System (DOS).

SEDIT resembles the RDOS symbolic debuggers very closely (although it cannot set breakpoints or run a program).

This manual is organized as follows:

Chapter 1	introduces SEDIT, describes special characters, and explains SEDIT error responses.
Chapter 2	tells you how to invoke SEDIT from the CLI; then it describes the commands which change output and enter data. Examples show you how to open, examine, and modify the contents of save, overlay, and text file locations.
Chapter 3	begins by describing the special SEDIT registers; it then explains how you can use these registers to find and modify locations.
Chapter 4	covers SEDIT symbol recognition: how to disable and enable global and local symbols.
Chapter 5	summarizes the SEDIT commands alphabetically. We have printed this chapter on yellow stock for easy reference.
Appendix A	lists the SEDIT error messages and explains their causes.

Reader, Please Note:

We use these conventions for command formats in this manual:

COMMAND required *[optional]* ...

Where	Means
COMMAND	You must enter the command (or its accepted abbreviation) as shown.
required	You must enter some argument (such as a filename). Sometimes, we use: {required$_1$ | required$_2$} which means you must enter *one* of the arguments. Don't enter the braces; they only set off the choice.
[optional]	You have the option of entering this argument. Don't enter the brackets; they only set off what's optional.
...	You may repeat the preceding entry or entries. The explanation will tell you exactly what you may repeat.

Additionally, we use certain symbols in special ways:

Symbol	Means
)	Press the RETURN key on your terminal's keyboard.
□	Be sure to put a space here. (We use this only when we must; normally, you can see where to put spaces.)

All numbers are octal unless we indicate otherwise; e.g., 35_{10}.

Finally, we usually show all examples of entries and system responses in THIS TYPEFACE. But, where we *must* clearly differentiate your entries from system responses in a dialog, we will use

THIS TYPEFACE TO SHOW YOUR ENTRY)
THIS TYPEFACE FOR THE SYSTEM RESPONSE

For example:

START/ *020440* ; *LDA 0 .OVLY*

Here, the person at the console typed START/ and SEDIT returned 020440. Then the person typed ; and SEDIT returned LDA 0 .OVLY.

End of Preface

Contents

Illustrations

Chapter 1
Introduction

Data General's Symbolic Editor (SEDIT) allows you to edit disk file locations symbolically in any random or contiguous file created on an RDOS or DOS system. SEDIT is compatible with the RDOS debuggers and shares many commands with them. (To emphasize this similarity, we use the terms "location" and "address" interchangably within this manual.)

With SEDIT, you can display and modify locations in octal, or in instruction, system call, byte, or ASCII format. You can display whole blocks of words and use special registers (described in Chapter 3) to find and modify locations. SEDIT works best on programs which have been loaded with a program symbol table, and on text files. You *can* patch operating system files with SEDIT, in octal -- but you may find the ENPAT and PATCH utilities easier to use for this.

SEDIT is unique among Data General disk editors in that it searches the disk file for symbolic values. For example, take the instruction JSR @ 50 in a program. Using symbols, SEDIT might display this as

JSR @ .SAV

which tells you about the symbolic address for the JSR. SEDIT can also access a program's overlay file from its save file.

Commands

Each SEDIT command is a single character, which you sometimes precede with ESC (echoed as $), an address, or a symbol. SEDIT has both global commands and local commands. Global commands direct searches, display ranges of locations, set registers, and enable or disable symbol recognition. Local commands open and close locations and display data words in different formats.

Note on Consoles

The following guidelines apply not only to SEDIT, but to all other RDOS disk editors and symbolic debuggers.

If you have an upper/lower case console, set it on ALPHA LOCK before trying to use these utilities, because they don't recognize lower-case letters.

One command common to all utilities has been called LINE FEED (↓) in our manuals. ANSI-standard terminals, like Data General's DASHER printer or display Model 6053, do not have a LINE FEED key; instead, they have a NEW LINE key, and you can press NEW LINE to enter this command. On a non-ANSI standard console, press the LINE FEED key. In this manual, we call this command NL/LF, and show it as (↓).

Another common command has been called SHIFT-N (↑). On a DG upper/lower case console, press SHIFT-6 instead of SHIFT-N to enter this command. In this manual, we call this command uparrow and show it as (↑).

Special Characters

As mentioned in the Preface, we use italic type in examples to indicate SEDIT output, and boldface type to signify your input. All numbers are octal, unless shown otherwise. Other special keyboard characters and their meanings are:

Character	Meaning
CTRL-A (Press CTRL and A keys)	Interrupt the current SEDIT command and return the SEDIT prompt.
CTRL-Q	Resume console display from the point at which CTRL-S suspended it.
CTRL-S	Pause while displaying output on the console. CTRL-S/CTRL-Q are useful for long displays on CRTs. They work on all DG consoles.
)	Carriage RETURN.
↓	NEW LINE or LINE FEED (NL/LF).
↑	Uparrow. Press SHIFT and 6 or SHIFT and N keys.
$	ESC. Press ESC key.

Typing Errors

If you make a mistake while typing input to SEDIT before you press), press the RUBOUT key. SEDIT will respond with U , close the current location, and display its prompt on a new line. You can then type another command.

Actually, you can use any undefined character to do what RUBOUT does, but RUBOUT is most convenient.

SEDIT Error Responses

SEDIT has two common error responses: U and ?. (Other SEDIT error messages are explained in Appendix A.)

U means that the characters you've typed are undefined. This can result from a simple mistake, or it may mean that your program doesn't include a symbol table, or that you have not enabled this symbol.

? means that your command syntax was wrong. Global commands must start on a new line, immediately after the prompt. This can also occur if you type more than six octal numbers, or any value which exceeds 2^{16} -1 (e.g., 277777 or 88000.), or numbers, or if you use a space in an expression (e.g., START □ +10).

After it detects an error, SEDIT closes the current location (if open), and outputs a carriage return and its prompt. These errors do no harm.

End of Chapter

Chapter 2
Operating SEDIT

This chapter tells you how to execute the SEDIT program from the CLI. It also shows you how to include a symbol table (which you need to edit a save file symbolically), and describes the commands to change SEDIT output format and input new information. Then it explains editing an overlay file, and ends by showing you how to edit a text file.

Invoking SEDIT

You invoke the SEDIT utility with the command:

```
SEDIT filename )
```

where the filename is the name of any RDOS or DOS random or contiguous file.

SEDIT searches for the filename specified; if it can't find the filename and you omitted an extension (e.g., .SV), SEDIT searches for the filename with the .SV extension. Because SEDIT allows you to edit any overlay for any node from the save file, it is generally easier to edit an overlay file while editing the save file than to edit the overlay file directly.

When SEDIT finds the file, it opens it, then displays a revision number and a period prompt (.). You can then enter SEDIT commands. If the file is read- or write-protected, SEDIT displays a PROBLEM READING FILE error message and returns to the CLI when you try to read or modify the file.

When you have finished editing, type ESC Z (echoed as $Z) to terminate SEDIT and return to the CLI. For example:

```
SEDIT MYPROG )          Invoke SEDIT.
SEDIT REV x.xx
.                       The period is SEDIT's prompt.
.START/020451           Enter SEDIT commands.
$Z                      Return to the CLI.
DONE.
R
```

Global Switches:

/N Do not search for a symbol table. Use this switch along with global /Z to edit a text file, or if the program lacks a symbol table. SEDIT may display a NOT ENOUGH MEMORY FOR SYMBOL TABLE error message if the program lacks a symbol table and you use /Z without /N.

/Z This file starts at location zero. You *must* use this switch to edit a text file or stand-alone program file; do *not* use it for a conventional save file. Programs you can execute under RDOS begin at location 16; SEDIT assumes this if you omit /Z.

Including Symbols in Save and Overlay Files

If you want to edit a save file symbolically (instead of exclusively with octal locations), you must include user symbols. SEDIT will not recognize any symbols at all unless the program includes a program symbol table. You can instruct RLDR to include both a symbol table and debugger with the RLDR/D switch; or you can have it include a symbol table alone by writing a .EXTN .SYM. statement into one of the program modules.

If the program includes a symbol table, SEDIT automatically recognizes *global* symbols (those specified in a .ENT psuedo-op in the program). To use *local* symbols (those not identified by .ENT), you must insert the global /U switch in the assembler command line, and the *local* /U switch in the RLDR command line. Having included local symbols, you then tell SEDIT to *see* them with the SEDIT command name%.

For example, assume that you want to include all symbols from two source files (named MYPROG and MYPROG1) in save file MYPROG. You'd follow these steps:

```
{ MAC/U MYPROG; MAC/U MYPROG1 }
{ ASM/U MYPROG; ASM/U MYPROG1 } )

RLDR/D MYPROG/U MYPROG1/U $LPT/L)
.
.
R
```

This produces save file MYPROG.SV, and sends the load map to the line printer. (It's very helpful to have a copy of the map.) If either MYPROG or MYPROG1 included a .EXTN .SYM., you could omit the RLDR global /D switch.

To enable local symbols in overlays, you'd append global /U to each overlay name; e.g., RLDR/D MYPROG/U MYPROG1/U [OVLY0/U, OVLY1/U].

```
SEDIT MYPROG )
SEDIT REV x.xx
.MYPROG%
.
```

Chapter 4 describes enabling and disabling user symbols.

SEDIT Command Format

All SEDIT commands have the format:

[address] command *[reply] [new-contents] [ )]*

The *address* can be a number, symbol or period. Each SEDIT command is a single character. Commands which display an address are entered alone; these are called *local* commands. You enter other commands by pressing the ESC key, then the command letter; these are *global* commands. You can enter multiple local commands on one line, but you must start each global command on a new line. The *reply* is SEDIT's response -- often the contents of address. You enter the *new-contents* (in any format described below) only if you want to modify the current contents. A carriage return tells SEDIT to accept the *new-contents* (if any) preceding; it also closes the current address and displays a new prompt. SEDIT will output a carriage return if you make a syntactical mistake or enter an undefined symbol. Here are three local commands in one line:

.START/ *020440* ; *LDA 0 ER+10* LDA 0, ER+7)

START is an address, "/" is a local command which opens and displays the address symbolized by START and *020440* is SEDIT's response, which indicates the contents of the address. The semicolon (;) is another command, which tells SEDIT to display the contents in instruction format (instead of octal); *LDA 0 ER+10* is SEDIT's translation of *020440* into instruction format. The last entry made by the user, specifies the new contents for address START; the) installs the new contents and closes the address.

An example of a global command is:

.$H *000000* 1)

This opens the output register (described in Chapter 3), and stores 1 in it; this sends all output of search commands to the line printer. Global commands remain in effect until you change them.

Location and Display Commands:

Location commands open and close locations; display commands show the contents of these locations in different formats. When you open a location, it becomes the *current location;* this does not change until you access another location.

Location Commands

Command	Effect
addr/	Open address *addr* and display contents.
addr!	Open address *addr* and display nothing.
↓ (NEW LINE or LINE FEED) (NL/LF)	Close current address (if open), open and display next address. (Use NEW LINE if your terminal lacks a LINE FEED key.)
↑ (SHIFT-N or SHIFT-6)	Close current address (if open), open and display preceding address.
) (RETURN)	Close current location (if any); return prompt. The closed location remains the current location.
/ (slash)	Close current location (if open) and open location specified by *contents* of current location. This is useful for pointers (e.g., .LOAD/ *LOAD / LDA 0 TEMP*).
.=	Display current location in current radix (e.g., 5005).
.:	Display current location symbolically (e.g., LOOP+6).

When a location is open, SEDIT will generally try to insert whatever you type as the new contents for that location. If you don't want to modify the contents of an open location, be sure to close it with one of the commands above before typing new data. If you do type characters which you don't want SEDIT to try and insert in this location, press RUBOUT; this closes the current location without changing anything.

Display Commands

The display commands act on the *last entry* which appears on the console. This can be an entry which you have typed in, or which SEDIT has displayed; or it can be an address or the contents of an address.

Command	Effect
-	Display last entry in current radix (the default is octal). A .= command displays the current numeric location.
;	Display last entry in machine instruction format (e.g., MOVS 0 0).
' (apostrophe)	Display last entry in ASCII format (e.g., HI or <n><n> if nonprinting characters).
\ (backslash)	Display last entry in .SYSTM command format (e.g., .RDL 0).
← (backarrow or SHIFT-O)	Display last entry in half-word format (e.g., 14 17).
&	Display last entry in byte-pointer format (e.g., 3007 1).
:	Display last entry in symbolic format (e.g., LOOP+20).
*	Display last entry in symbolic format with bit 0 zeroed.
. =	Display current numeric location (e.g., 005005). (As with other utilities, you can enter a period (.) to indicate the current location.)
.:	Display current symbolic location (e.g., START). If no symbol is defined within 2000 locations, : displays location numerically.
n.=	Display decimal number n in current radix (default is octal). See Chapter 3, Number Register, for other conversions.
nxH=	Display hexadecimal number nx in current radix. n must be a digit from 0-9; x can be any hex number(s) or hex letter(s).

Figure 2-1 shows examples of the local commands shown above.

SEDIT MYPROG)	Invoke SEDIT.
SEDIT REV x.xx	
.START/ *020440*)	Open address START and display its contents (octal by default); close location with).
. .= *000452*)	Current address remains 452,
. .:*START*)	which is START symbolically.
.START!)	Open address START without displaying contents; close location. Address remains START.
. ↓	NL/LF closes current location (if open); then opens and displays next location.
START+1 126400 ↓	Close START+1; open START+2.
START+2 006017 ↑	Uparrow closes current location (if open); then opens and displays previous location.
START+1 126400)	) closes.
. .: *START+1* ↓	Current address remains START+1. NL/LF opens next: START+2 contains 6017.
START+2 006017	
./ *146032*)	Use contents of START+2 (6017) as address; open and display address 6017.
START+2/ *006017*	Open and display START+2 in octal,
; *JSR @ OVLY1+16*	display in machine instruction format
' *< 14 > < 17 >*	then in ASCII format (nonprinting values),
\ *.WRB 17*	then in .SYSTEM command format,
_ *14 17*	then in half-word format,
& *3007 1*	then in byte-pointer format,
: *6017*	then in symbolic format (not a symbol),
* *6017*	then in symbolic format, 0B0 (here there is no difference between : and *),
.: *START+2*	then, display current location symbolically.
.9999.= *023417*	Convert decimal 9999 (period indicates decimal) to octal.
0FH= *000017*	Convert hex numbers F and
2FFAH= *027772*	2FFA to octal.

Figure 2-1. Local Commands Example

You can convert most display commands into global commands by typing ESC, then the command. SEDIT then displays *contents of addresses* in the new format until you change it again. You can continue to use other display commands as usual. For example:

.START/ *020440*)	Octal mode.
.$;	Change display mode.
.START/ *LDA 0 ER+10* ↓	New mode
START+1 SUB 1 1 ↓	continues....
START+2 JSR @ OV1+16 ↓	
START+3 ISZ 0 \ *.OVOP 0*)	Display in other format.
.$=	Change mode.
.START+3/ *012000*	New mode continues.

You can enter new contents for an address in any mode, including the special data entry modes described next. For example:

.START+35/ *125120 ; MOVZL 1 1 MOVZR 1 1*
.START+35/ *125220 ; MOVZR 1 1*

Change contents of START+35 from MOVZL 1 1 to MOVZR 1 1, then verify the change. The display mode remains octal throughout.

SEDIT also allows you to examine radix 50 symbols. All user symbols are stored in radix 50 in the program symbol table; this allows any symbol to fit in two 16-bit words. You can use the left brackets to display the symbol name. The format is:

firstword[secondword[

For example, assume that you are stepping backward through a symbol table:

T.OVL+271 000000 ↑
T.OVL+270 131401 ↑
T.OVL+267 113014 131401 [113024[*ROOT*

(This works because RLDR builds the symbol table *downward* -- not upward as it does the rest of the save file.) You need not actually type the numbers before the brackets -- simply type the bracket after the appropriate number appears:

T.OVL+271 000000 ↑
T.OVL+270 131401 [↑
T.OVL+267 113014 [*ROOT*

To do this, remember to stay in synch -- every second bracket must follow the first word of the symbol. Symbol entries in the table are three words long.

SEDIT also offers two special data entry formats, which allow you to translate ASCII characters, or bytes into any format, or enter characters or bytes in the current open location. To enter ASCII characters, precede them with a quotation mark ("). You can (but need not) enter a closing quote. For example:

```
."AB= 040502;STA 0 ER+52 'AB )
.START/ 0220440"AB )
.START/ 040502 'AB
```

The first line displays an ASCII AB in different formats; the second line inserts AB in location START. You can use any two ASCII characters after the quote; e.g., "A= 040440. To enter a quote, you must type two sets of quotation marks. For example, to enter "A, you'd type """A" or """A.

To enter data in bytes, use the format n]n, where n is a byte of information. For example:

```
.10]1= 004001' <10><1> )
.START/ 020440 377]0 )
```

The first line displays the word containing 10 in its left byte and 1 in its right byte. The second line inserts a word containing bytes 377 and 0 in location START.

If n is too large, SEDIT includes only the digit(s) which fit into eight bits.

Calculations

SEDIT has two arithmetic operators, plus (+) and minus (-). You can use these operators for calculations, or to access locations. For example:

.4567+123= 004712	Add octal numbers.
.999.+5678.= 015025	Add decimal numbers, result in octal.
.START/ 020440	START contains 020440.
.START= 000452	START is location 452.
. .+12= 000464 / 126400	Add 12 to current location, result is 464, open location 464.
.ER= 00502	ER is location 502; current location remains 464.
. .-10= 000454 / 006017	Subtract 10, result is 454, open location 454.

Examples

Assume that you have determined with a debugger that you want to change an instruction in a program -- we'll call it WRITE.SV. You can't or don't want to change the source program and reassemble and reload this program; you simply want to run it.

```
SEDIT WRITE )
SEDIT REV x.xx
.WRITE%
```

Get into SEDIT and enable local symbols.

```
.START+10/ 020445;LDA 0 START+35 \.CCON 45
```

Display the contents of START+10 in octal, instruction, and system call format (the latter is meaningless).

```
START+11 0006017;JSR @ 17 ↓
START+12 021027; LDA 0 27 2
```

After examining succeeding locations, you decide to change the display mode to instruction format because you're looking for a machine instruction.

```
)
.$;
.START + 32/ JMP START + 45 ↓
START+33 LDA 2 START+71 ↓
START+34 LDA 1 START+74 ↓
START+35 AND 1 2  ANDS 1 2 )
```

Find the faulty instruction and change it to ANDS 1 2. Verify the change:

```
START+35/ ANDS 1 2 )
```

And leave SEDIT:

```
.$Z
DONE
R
```

You can now run the program.

Editing Overlays ($O)

Using SEDIT, you can edit any overlay as if it were part of the save file. SEDIT opens the overlay file and reads the overlay you specify into its node; after editing, it writes any changes you make back to the overlay file. You can read any overlay into its node at any time.

If you have identified an overlay with the .ENTO pseudo-op, you can read the overlay into its node by typing:

```
overlayname$O
```

If .ENTO was omitted from the overlay, you must specify the node number and overlay number:

```
node]overlay$O
```

where node is the node number (0 for the node defined by the first pair of brackets in the RLDR command line, 1 for the node defined by the second pair of brackets, and so on), and overlay is the overlay number (0 for the first binary within the square brackets, 1 for the second binary within the square brackets, and so

on). We used byte entry to enter the node and overlay numbers; you *can* use a word with the node number in the left byte and overlay number in the right byte (e.g., 0 for node zero, overlay zero, 1 for node zero, overlay 1, and so on).

For the rest of this section, we'll be using a save and overlay file created by the command line:

```
RLDR/D ROOT/U [OVLY0/U, OVLY1/U] )
```

This creates save file ROOT.SV, with one node, and overlay file ROOT.OL, which holds overlays OVLY0 and OVLY1. OVLY0 and OVLY1 contain a .ENTO OVLY0 and an .ENTO OVLY1 statement, respectively. RLDR's load map shows that the overlay node begins at location 517 and ends at 1117.

You can enable or disable symbols at any time, as described further in Chapter 4. When you enable symbols in one module, you disable symbols in the current module.

When you load another overlay after you have enabled symbols in an overlay, SEDIT displays symbols exactly as they were in the *original* overlay. For example, in the second overlay below, PRNTB might have really contained the instruction LDA 0, FILEA instead of LDA 0 B. This can be very confusing, so you can clarify overlay symbols by following this sequence when you want to edit successive overlays in the same node:

1. Enable local symbols if you need them for next step.
2. Get to node start and enable symbols in the desired overlay (ovlyname%).
3. Read in the desired overlay (ovlyname$O or node]ovly$O).
4. Edit the overlay.
5. Re-enable symbols in the root program (e.g., ROOT%); go to step 2.

```
SEDIT ROOT )
SEDIT REV x.xx
.ROOT%                          Enable local symbols.
.515/ 027117 '.O↓               Approach the node start (this is the end of the overlay file text string,
                                ROOT.OL.)
OFILE+4 046000 'L <0> ↓         Symbol OFILE points to the overlay file name.
PRNTB 000000 .= 000517 ↓        PRNTB is vacant, as is the rest of the node. There's a value for it in the first
PRNTB+1 000000 ↓                overlay.
PRNTB+2 000000 ↓
PRNTB+3 000000 )
.OVLY0$O                        Load OVLY0 into its node (without .ENTO, you'd use 0]0$O).
.515/ 027117' .O ↓
                                Approach node as before...
OFILE+4 046000' L <0> ↓         Same value at 516.
PRNTB 020407                    A value for PRNTB! It is an instruction. The overlay is in the node.
; LDA 0 PRNTB+7 ↓
PRNTB+1 006017                  PRNTB+1 holds an instruction, and PRNTB+2 holds a system call.
; JSR  @  NSW+16 ↓
PRNTB+2 010000 \ .PCHA 0 ↓
PRNTB+3 002430                  And PRNTB+3 holds another instruction.
; JMP  @  PRNTB+6 )
.OVLY0%                         Enable symbols in first overlay. SEDIT displays LDA 0 B instead of LDA 0
.PRNTB/ 020407 ; LDA 0 B ↓      PRNTB+7.
PRNTB+1 006017                  Enabling overlay symbols doesn't affect display of PRNTB+1 or PRNTB+2.
; JSR  @  NSW+16 ↓
PRNTB+2 01000 \ .PCHA 0 )
.OVLY1$O                        Load OVLY1 into node - it replaces overlay 0.
.ROOT%                          Enable ROOT's symbols; we won't enable symbols in this overlay for reasons
                                explained above.
.515/ 020407 '.O ↓              Approach node again...
OFILE+4 046000 'L <0> ↓         Same value at 516.
PRNTB 020407                    Same value -- (instruction) in PRNTB.
; LDA 0 PRNTB+7 ↓
PRNTB+1 006017                  Same value in PRNTB+1 -- (instruction). Is this the same old overlay?
; JSR  @  NSW+16 ↓
PRNTB+2 015401 \ .RDL 1         No. This used to be 010000, .PCHA 0. The second overlay is in the node.
```

Figure 2-2. Loading and Examining Overlays

Text Files

When you invoke SEDIT to edit a text file, use the global /Z and /N switches. Generally, you'll be displaying locations in ASCII mode ($'), and using the ASCII entry operator ("xx) and byte entry operators (n]n).

To help illustrate, assume that you have created a text file via the CLI by typing:

```
XFER/A $TTI FILEA/R )
MESSAGE HELLO;DISK )
CTRL-Z
```

You need the /R switch in RDOS to organize the file randomly. (The text editors automatically create random files.)

Now you invoke SEDIT by typing:

```
SEDIT/Z/N FILEA )
SEDIT REV x.xx
.
```

The /Z switch tells SEDIT that the file begins at location zero; if you forgot it, SEDIT would display the first 15 (octal) location contents as 0, and you could not edit the file accurately. The /N switch instructs SEDIT not to look for a symbol table; if you forgot it, SEDIT might display a NOT ENOUGH MEMORY FOR SYMBOL TABLE error message and return to the CLI.

Now, check the locations and add a CLI command:

.0/ 046505 'ME)	The first two locations contain "ME".
.$'	Change to ASCII Display.
.↓	Display next location:
+1 SS ↓	And next;
+2 AG ↓	And next;
+3 E ↓	And so on;
+4 HE ↓	
+5 LL ↓	
+6 O; ↓	
+7 DI ↓	
+10 SK ↓	Here, the text message ends.
+11 <15><0> ↓	CR and null are the last nonzero words.
+12 <0><0>)	Now, to add a list command:
.11/<15><0> ";L)	Use " to insert ;L in location 11.
.11/ ;L ↓	Check 11 and open next location.
+12 <0><0> "IS↓	Place IS in 12; open next location.
+13 <0><0> "T/↓	Place T/ in 13, open next location.
+14 <0><0> "N<1 ?	You can't use " to enter more than two characters.
.14/<0><0> 116]15)	Try byte entry operator (n]n). 116=ASCII N.
.14/ N<15> ↓	Byte entry operator works!
+15 <0><0>)	Location 15 is unchanged.
.$Z	Leave SEDIT.
DONE.	
R	

Now, we've added the CLI command LIST/N to the file. Verify it:

```
TYPE FILEA )
MESSAGE HELLO;DISK;LIST/N
R
```

We can execute the commands by typing @FILEA@).

When you use SEDIT to edit a text file, it automatically extends the byte length of the file to an even multiple of 512_{10} bytes. If you open a location beyond the existing end-of-file (e.g., location 1001 in FILEA above), SEDIT automatically extends the file to the next multiple of 1000 (512_{10}) bytes.

End of Chapter

Chapter 3
Searches, Displays and Registers

The material in this chapter and the next relates primarily to editing *save* files, not text files.

Often you will want to search a file for a specific word, or display entire blocks of words on the console or line printer. This chapter explains the search and display commands and the special SEDIT registers which can help you use these commands effectively.

Search Command ($S)

With the Search command ($S), you can search forward in the file for the contents of any address (e.g., JSR @17). To search to a location, type **address$S**, and SEDIT will search forward from address zero and stop when it reaches **address**.

To search for the contents of an address, you must open the Word register with the **$W** command, type the word you want SEDIT to find, and close the Word register. Then you type **$M** to open the Mask register, enter a mask word, and close it. If the Mask register contains zero (default), the search will match all words in the file. To match a specific word, instead of a general group, enter -1 (177777) in the Mask register.

You can also set an increment for the search with the Search Increment Register; you can direct output to the line printer by placing 1 in the Output register.

Having set these registers (which are described later in this chapter), you enter your Search command, and SEDIT seeks matching contents in the file.

The formats of the Search command are:

addr$S	Search from location 0 to **addr**.
$S	Search entire file for value in Word register.
<addr$S	Search from location zero to **addr** for value in Word register.
addr<$S	Search from **addr** to last location for Word value.
addr<addr$S	Search from **addr** to **addr** for Word value.

If you omit **addr**, SEDIT assumes zero as the lowest address and 77777 as the highest address. When SEDIT finds a word matching the contents of the Word register, it displays **addr** and its contents in the current display mode. If you have enabled symbols, it displays locations symbolically, as in Figure 3-1.

```
SEDIT ROOT )
  SEDIT x.xx
  ROOT%                          Enable local symbols.
  .ER$S                          Display file to location ER.
  +0 000000                      SEDIT always searches for locations from location 0; thus it may take
  OVLY1 000000                   some time to reach location ER.....
  .
  .
  .
  RTURN+2 000401
  ER 006017                      Found!
  . .= 000503 .: ER+1 )          Current position is 503, symbolically ER+1.

  .$W 000000 JSR@17 )            Now to find locations containing JSR @ 17. First, place JSR @ 17 in Word
                                 Register.
  .$W 006017 ; JSR @ +17 )       Verify new value in Word register.
  .$M 000000 -1 )                To match a word specifically, place -1 in Mask register.
  .$S                            Search the save file for JSR @ 17. SEDIT displays the locations
  START+2 0006017                symbolically that contain JSR @ 17.
  START+6 006017
  .
  .
  DEBUG+01277 006017
  .$;                            Change display to instruction mode.
  .$S                            Search again.
  START+2 JSR OV1+16             SEDIT displays contents in instruction format.
  START+6 JSR OV1+16
  .
  DEBUG+1277 JSR @ OV1+16
  .$K                            Kill all symbols (see Chapter 4).
  .$S                            Search again.
  +454 JSR @ 17                  Display locations in octal, contents in instruction format.
  +460 JSR @ 17
  .
  2515 JSR @ 17
  .1$K                           Re-enable global symbols.
```

Figure 3-1. SEDIT $S Command Example

Another example is:

```
.$W 000000 .WRL 0 )    Insert .WRL 0 in Word register.
.$M 000000 -1 )        Insert -1 in Mask register.
.$\                    Change to .SYSTM call mode.
.400<$S                Search from location 400 upward.
5536 .WRL 0            Display in system call format.
7666 .WRL 0            There are two write lines to channel 0 in this program.
```

See the register sections below for more examples of the Search command.

Display Command ($D)

The Display command ($D) resembles the Search command, except that it cannot search for the contents of a location, and it displays the contents of eight locations on one line. You can display on the line printer by setting the Output register to 1, and you can set the Search Increment register to display locations at desired increments. The $D command always displays contents in octal; you cannot change the output of $D to other formats.

The formats for the Display command are:

addr$D	Display contents from location 0 to addr.
$D	Display contents of all locations.
addr<$D	Display from location addr to last location.
addr<addr$D	Display from addr to addr.

Figure 3-2 shows display examples. The first example displays locations one through 20, and locations 400 through 450 in a save file. Like most save files, this one uses only location 17 (in first range); its User Status Table, TCB(s), and overlay directory (if any) start at 400 (second range). The second example displays locations in a text file.

```
SEDIT ROOT )
SEDIT REV x.xx
0 20$D
+0 000000 000000 000000 000000 000000 000000 000000 000000
OVLY1+7  000000 000000 000000 000000 000000 000000 000000 001155
OVLY1+17  000000

 .400<500$D

"USTAD 000000 000060 006634 006114 006635 001120 001216 006635
USTAD+10 000000 177777 177777 000410 000424 000424 177777 00452
USTAD+20 000445 000000 177777 000000 001124 000000 000000 000000
USTAD+30 000000 000000 000000 177777 000000 000000 000000 000000
USTAD+40 000000 000000 000000 000000 000000 000001 177400 001001
USTAD+50 000000 000517 020440 126400 006017 012000 000424 020426
START+6 006017 010000 000420 020423 126400 006017 020000 000413
LOV0+5 002420 020416 126400 006017 020000 000405 002413 006017
RTURN+1 004400"

.
$Z
DONE.
R
SEDIT/N/Z TESTFILE)
SEDIT REV. x.xx
 .0 10$D
+0 046505 051523 040507 042440 052110 044523 020111 051440
+10 052105
.
```

Figure 3-2. Displaying Locations

SEDIT Registers

SEDIT registers are locations into which you can insert values. These values control I/O and numeric output, searches, and displays. There are five registers: Output, Number, Search Increment, Mask, and Word. You can use all the registers for searches; you can use the first three for displays.

The value in each register is initially zero (except for the Search Increment Register). Any value you insert reverts to the default when you leave SEDIT and return to the CLI.

Output Register ($H)

By default, SEDIT sends Search and Display output to the console. You can direct this output to the line printer by entering the value 1 in the output register. Use the $H command to open and examine this register; you can then change it if you want. For example:

$H *000000* 1)

directs output of Search and Display commands to the line printer. When output of Search commands goes to the printer, SEDIT inserts the value (in current output mode) of the sought word before the contents of each address. Display command output is the same on both console and printer.

Number Register ($N)

By default, SEDIT displays numbers in octal, unsigned, with leading zeroes. You can tell it to output numbers with a leading sign, suppress leading zeroes, or select radix 8, 10, or 16 by placing the proper value in the number register. Use the $N command to open and examine this register; you can then insert a new value if you want.

New value:	Sets bit(s):	Effect:
100000	B0	Sign numbers.
040000	B1	Suppress leading zeroes.
000010	B12	Display in octal.
(or 000000)	none	Same.
000012	B12,B14	Display in decimal (10. has same effect).
000020	B11	Display in hexadecimal (16. has same effect).

You can enter combinations of these values in the number register. If you enter any value but those above, SEDIT uses the default radix (octal).

```
.START/ 020440 )          Default octal display.
.$N 000000 10. )          Specify decimal numbers.
.START/ 08480. )          Display in decimal.
.4440= 02336. )           Convert an octal number.
.$N 000010 140000 )       Insert sign, suppress leading
                          zeroes, display in octal.
.START/ +20440 )          SEDIT follows orders.
.$N -40000 16. )          Convert to hexadecimal.
.START/ 2120H             Terminal H means hex output.
15= 000DH
```

Search Increment Register ($J)

By default, SEDIT increments locations by one in Search and Display commands. You can specify a different increment by opening the Search Increment Register and inserting the increment you want. The $J command opens this register.

The default value in the Search Increment Register is 000001; a negative or 0 value is illegal, and will produce the default value.

```
.$N 000000 40000 )        Suppress leading zeroes.
.$M 0 -1 )                Set Mask register to -1.
.$W 0 STA 3, 50 )         Insert STA 3, 50 in Word
                          register.
.$S                       Search all locations for STA 3,
                          50.
DEBUG+527 54050
DEBUG+572 54050
DEBUG+1075 54050
.$J 1 4 )                 Specify increment of 4 in
                          Search Increment register.
.$S                       Search again.
DEBUG+572 54050           There's only one STA 3, 50 at
                          an increment of 4 from zero.
.$W 54050 0 )             Place 0 in Word register.
.$S                       Search again.
+0 0                      There are plenty of locations
OVLY1+3 0                 which contain 0.
OVLY1+7 0
OVLY1+13 0
.
.$J 4 10 )                Place increment of 10 in
                          register.
.<400$D                   Try a display command.
0. . . . . .
OVLY1+77. . . . . .
OVLY1+177. . . . .
OVLY1+277. . . . .
USTAD. . . . . .
.
.$J 10 1 )                Specify a search/display
                          increment of 1.
```

Mask and Word Registers ($M and $W)

SEDIT uses both the Mask and Word registers for searches. If you want to search for a specific instruction, you must set both of these registers. As you've seen earlier in this chapter, you insert the word you want matched in the Word register. After you issue the Search command, SEDIT searches each location and ANDs the value in the Mask register with the location's contents. If this value matches the Word register, SEDIT announces a match. The default value for both registers is 000000.

Because SEDIT ANDs the mask with the contents, and the default mask is 000000, there will be no match unless you set the Mask register. To match a specific word, insert -1 in this register. To match only a portion of a word, set the bits which contain that portion to 1 in the mask register, and set the other bits to 0.

The command

```
$M
```

opens the Mask register and displays the contents.

Place the value you want SEDIT to match in the Word register. You can enter this in any format (e.g., MOVS 0 0, 101300, or .WRL 0). The command

```
$W
```

opens the Word register and displays its contents. The following example shows application of the Word and Mask registers in Search commands.

```
SEDIT ROOT )
SEDIT REV x.xx
.$M 000000 -1 )           Set mask register to 177777 (16
                          ones).
.$W 000000 .RDL 0 )       Insert read line call in Mask
                          register.
.$S                       Search all locations.
5742 015400               One match.
.15400 \ .RDL 0           Check 15400; it's .RDL 0.
```

The Mask register can help you search for all occurrences of one *type* of instruction. For example, assume that you want to find all "MOVZL" instructions in a program. The octal instruction is 101120, but this applies only to MOVZL 0,0; with any other accumulators there will be no match. Bits 1 through 4 specify accumulators in MOV instructions, so you mask them out by inserting a word which contains 0 in bits 1,2,3, and 4, and 1 in the other bits. This word is 103777:

```
SEDIT SYS )
SEDIT REV x.xx
.$W 000000 MOVZL )        Insert MOVZL in Word
                          register.
.$M 000000 103777 )       Insert MOVZL mask in
                          Mask register.
.<4000$S                  Search from location 0 to
                          4000 for MOVZLs.
2535 155120               One match.
2536 175120               Two matches.
.155120; MOVZL 2 3 )      Check the match words.
.175120; MOVZL 3 3        The MOVZL mask works.
```

End of Chapter

Chapter 4
Enabling and Disabling Symbols

As described in Chapter 2, global symbols are those declared in .ENT statements in a program. To recognize global symbols, SEDIT requires only that a save file include a program symbol table. The RLDR/D command will include both a program symbol table and a debugger; a .EXTN .SYM. statement in any program module will include *only* the symbol table. SEDIT always recognizes global symbols (unless you disable them).

SEDIT will also recognize *local* symbols (those not declared with .ENT), *if* you assembled the program source file(s) using the global /U switch, and *if* you used local /U in the RLDR command line. To enable local overlay symbols, append local /U to the overlay name in the brackets; for example:

```
RLDR/D MYPROG/U MYPROG1/U [OVLY0/U, ↑)
OVLY1/U] )
```

When you proceed through a file, examining locations, SEDIT displays each location as it relates to the nearest previous symbol defined. However, if the nearest previous symbol is more than 2000_8 locations away, SEDIT displays the octal, not the symbolic, value of the location.

Enabling Symbols

If the program has a symbol table, SEDIT will understand all global symbols when you invoke it. To enable local symbols in any program module or overlay, type:

```
name%
```

where name is the module name, as assigned by the .TITL pseudo-op. The name is not necessarily the filename. If .TITL was omitted from the module, the default name is .MAIN. The name% command enables all local symbols in this module. It also disables local symbols in only one module at a time.

To edit overlays symbolically, see "Editing Overlays" in Chapter 2.

To enable global symbols after you have killed all symbols ($K, below), type:

```
n$K
```

where n is a single octal digit (1 is convenient).

In any module, to enable all symbols that you have not individually disabled, type:

```
name%
```

as above.

Disabling Symbols

If there are a lot of local symbols, they can cloud your overview while you are editing; also, an individual symbol can interfere with editing if its value happens to fall at the wrong point. Occasionally, even global symbols hamper editing. You can disable all symbols, or individual symbols, with the command:

```
$K
```

For example:

```
SEDIT MYPROG )          Invoke SEDIT.
SEDIT REV x.xx
.START/ 0240440         START was .ENTered.
.$K                     Disable all symbols.
.START/ U               START is now Undefined.
```

To disable any single symbol, type:

```
name$K
```

where name in the symbol name. Note that SEDIT will still recognize the symbol when you type it in, but won't display it when you proceed forward or backward

through locations in the file. Instead, SEDIT displays the disabled symbol's locations as it relates to the preceding active symbol (if the previous symbol is less than 2000_8 locations away - see above).

When you disable output of a single symbol, you cannot re-enable it during this SEDIT session. To re-enable the disabled symbol, leave SEDIT and return to SEDIT.

For example, assume that program ROOT starts at symbolic location START, and the next symbol is LOV0. (Here, it doesn't matter whether START and LOV0 are global or local symbols.)

```
SEDIT ROOT )
SEDIT REV x.xx
.ROOT%                      Enable all symbols.
.START/ 020440 ↓            Open and display location
                            START, then next location.
START+1 126400 ↓            Continue...
START+2 006017 ↓
      .
      .
      .

START+10 000420 ↓
LOV0 020423 )               Proceed to location LOV0.
.LOV0$K                     Disable LOV0 for this session.
.START+10/ 000420 ↓         Check START+10
                            (unchanged).
START+11 020423             What SEDIT displayed as
                            LOV0, it now displays as
                            START+11.
.LOV0/ 020423               SEDIT still recognizes LOV0
                            on input, but does not display
                            it.
.$Z                         Leave SEDIT.
DONE.
R
SEDIT ROOT )                Re-enter SEDIT.
SEDIT REV x.xx
.ROOT%                      Enable local symbols again.
.START+10/ 000420 ↓         Check LOV0 display...
LOV0 020423                 LOV0 display is enabled.
```

Managing Symbols

To enable global symbols while editing, you need do nothing if you have used the global /D switch in the RLDR command, or an .EXTN .SYM. statement in a source module.

To enable local symbols, type name% (you must have used the assembler global /U and RLDR local /U switches).

To disable all symbols, type $K.

To disable the display of any symbol, type symbolname$K.

To re-enable global symbols type 1$K (or any digit instead of 1). The sequence $K, 1$K disables all symbols and re-enables globals.

To re-enable all but explicitly disabled symbols, type name%.

By default, when you include the global /D switch in the RLDR command, RLDR places the debugger directly above your program, and places the symbol table (which either SEDIT or the debugger needs), directly above the debugger. If you use a .EXTN .SYM. in a program module, and omit RLDR global /D, the symbol table will be directly above the program. The load map (if you specified one with the RLDR local /L switch) will tell you the location of each global symbol, the start address of the debugger (if present) and the start and end address of the symbol table. If the program lacks a symbol table, you *can* use the addresses in the load map to patch, but this is much more awkward than using symbols.

Symbol Example

Figure 4-1 shows various attempts to disable a distracting symbol.

```
SEDIT MYPROG )
  SEDIT REV x.xx
  .$N 0000Q0 40000 )          Suppress leading zeroes (to help fit this example into one column).
  START/ 20440;LDA 0 ER + 10  Display location START.
  .$;                         Display in instruction mode.
  .MYPROG%                    Enable all symbols.
  .START/ LDA 0 OFILE ↓       Local symbol OFILE replaces global ER + 10.
  START+1 SUB 1 1 ↓           Examine succeeding locations.
  .
  .
  .LOOP+4/INC 0 0 ↓           Examine LOOP+4, then next location.
  LOOP+5 COM 0 0 SZR ↓        And next...
  RTCNS JMP RTCNS+10 ↓        Encounter symbol RTCNS, but we find RTCNS a little confusing.
  RTCNS+1 STA 0 RTCNS+15 )
  .= 5005 )                   Check current location in octal, and symbolically.
  . .: RTCNS+1 )
  .$K                         Disable all symbols and enable global symbols; this disables RTCNS if it is a
  .1$K                        local symbol.
  . .:RTCNS+1 )               Check current location symbolically; find that RTCNS is still enabled -- thus
                              it is global.
  .RTCNS$K                    Disable RTCNS specifically.
  . .:START+27 )              RTCNS is not displayed.
  .MYPROG%                    Re-enable local symbols.
  .LOOP+6/JMP LOOP+16 ↓       Continue editing.
  .
  .
  $Z
  DONE.
  R
```

Figure 4-1. Disabling and Enabling Symbols

End of Chapter

Chapter 5
SEDIT Command Summary

This chapter summarizes SEDIT commands in Table 5-1. First, it presents character commands (e.g.,. &), then letter commands in alphabetical order.

Table 5-1. SEDIT Command Summary

Command	Meaning	Example
/	Use contents as addr, open and display	400/ *000000*
addr/	Open addr, display contents.	START/ *LDA 0 OV1*
addr!	Open addr, display nothing.	START!
NL/LF	Open next addr, display contents.	↓ *START + SUB 1 1*
↑ (SHIFT-6) or (SHIFT-N)	Open previous addr, display contents.	↑ *START LDA 0 OV1*
) (RETURN)	Close current location (if open), insert new value (if any), give prompt.	*AND 1 2* ANDS 1 2)
.=	Display current location in current radix.	. = *000402*
.:	Display current location symbolically.	.:*START*
=	Display last entry in current radix.	SUB 1 1 = *126400*
$=	Change display mode to numeric, current radix (default is 8).	$=
;	Display last entry as an instruction.	126400 ; *SUB 11*
$;	Change display mode to instruction.	$;
'	Display last entry in ASCII.	046505' *ME*
$'	Change display mode to ASCII.	$'
\ (backslash)	Display last entry in .SYSTM format.	017000\ . *WRL 0*
$\	Change display mode to .SYSTM format.	$\
:	Display last entry in symbolic format.	020423: *LOV0*
$:	Change display mode to symbolic format.	$:

Table 5-1. SEDIT Command Summary (continued)

Command	Meaning	Example
.	Display last entry as symbol, 0B0.	177777. *77777*
n.=	Convert decimal n to current radix.	9999.= *023417*
nxH	Convert hex nx to current radix (n=0-9).	0EF2H= *007362*
word 1 [word2]	Convert 2 radix 50 words to symbol.	131401 [113034[*ROOT*
"xx=	Display ASCII xx in current radix.	"HI= *044111*
"xx	Insert ASCII xx in open addr.	*000000* "HI)
byte]byte=	Display byte, byte in current radix.	10]1 = *004001*
byte]byte	Insert byte, byte word in open addr.	*000000* 10]1)
+	Add one number to another or to an address.	4567 + 123 = *004712*
-	Subtract one number from another or an address.	ER-10 = *00452*
name%	Enable output of all local and global symbols in module name.	ROOT%
$D	Display all locations in file.	$D
addr$D	Display locations from 0 through addr.	<40$D
addr $D	Display locations from addr to end.	77000<$D
addr addr$D	Display locations addr through addr.	400<500$D
$H	Open output register, display contents. 0 sends $D and $S output to console, 1 to line printer ($LPT).	$H *000000* 1)
$J	Open Search Increment register (used in $D and $S), display contents.	$J *000001* 4)
$K	Disable output of all symbols.	$K
n$K	Enable all global (.ENT) symbols.	1$K
name$K	Disable output of symbol name.	TEMP2$K
$M	Open Mask register (used with $W, in S commands), display contents. Insert -1 to match a specific word.	$M *000000* -1)
$N	Open Number register, display contents. Insert 100000 to sign output, 40000 to drop leading zeroes, 12 or 10. for decimal output, 0,8. or 10 for octal output and 20 or 16. for hex output.	$N *000000* 140010)
ovname$O	Load overlay named by .ENTO into node.	OVLY0$O
$O	Load overlay 0 into save file node 0.	$O
node]ov$O	Load overlay ov into node node. Node number goes in left byte, overlay number in right byte.	0]1$O or 1$O
$S	Search all locations for value in $W (also use value in $M, $J, $H for all Searches).	$S

Table 5-1. SEDIT Command Summary (continued)

Command	Meaning	Example
addr$S	Search locations from 0 to addr.	<40$S
addr<$S	Search locations from addr to end.	77000<$S
addr<addr$S	Search locations addr through addr.	400<500$S
$W	Open Word register, display contents. S commands search for the value in the Word register.	$W *000000* STA 3,50)
$Z	Stop SEDIT and return to CLI.	$Z
CTRL, A keys	Stop current SEDIT command, return prompt.	CTRL-A
CTRL, Q keys	Resume console display, suspended by CTRL-S.	CTRL-Q
CTRL, S keys	Suspend console display, wait for CTRL-Q.	CTRL-S

End of Chapter

Appendix A
SEDIT Error Messages

? or _?

Your SEDIT command syntax was wrong, or you pressed RUBOUT. SEDIT outputs a line feed and awaits another command.

ERROR IN ACCESSING PRINTER

When you have directed output to the line printer (by setting $H to nonzero), SEDIT could not open the printer or could not write to it. Perhaps another program is using it.

FILE IS SEQUENTIAL: filename

SEDIT cannot edit a sequential file.

FILE NOT FOUND: filename

SEDIT searched for the file specified, then tried searching for the file with the .SV extension, but could not find the file.

HAD PROBLEM OPENING FILE: filename

SEDIT could not file, or could not open, the file specified.

ILLEGAL OVERLAY ADDRESS

This is an internal SEDIT error. The node address is somehow incorrect.

ILLEGAL OVERLAY NODE NUMBER

Either you have specified a node or overlay number higher than the file contains, or there is no overlay directory.

INSUFFICIENT MEMORY FOR OVLY TABLE

On an $O command, SEDIT tried to read the overlay directory into memory, but there was not enough memory available.

NO INPUT FILE NAME GIVEN IN COMMAND LINE

You must specify a filename for SEDIT to edit.

NO OVERLAY LOADED

No overlay has been loaded ($O) into the node you specified. This message appears only after *some* overlay has been loaded. For example, your program has two overlay nodes, you loaded an overlay into one node, then tried to access the other node.

NOT ENOUGH MEMORY FOR SYMBOL TABLE

The program's symbol table won't fit into memory. Alternatively, the file is not a save file and you forgot the global /N switch, or inappropriately used the /Z switch.

PROBLEMS OPENING OVERLAY FILE

SEDIT could not read the overlay directory, or could not open the overlay file. SEDIT expects the overlay file to have the same name as the save file, with the .OL extension.

SOME PROBLEMS READING SOURCE FILE: filename

An error occurred as SEDIT tried to open the file or read/write to it.

U

You have entered an undefined command or symbol. SEDIT outputs a carriage return and awaits another command. To enable *local* user symbols, use the *global* /U switch in the assembler command line and use the *local* /U switch in the RLDR command line; then type modulename% from SEDIT.

End of Appendix

Index

Within this index, the letter "f" following a page number means "and the following page"; "ff" means "and the following pages".

Chapter 5 is a summary of all SEDIT commands, but we have not entered Chapter 5's entries in this index.

DataGeneral

Software Documentation Remarks Form

How Do You Like This Manual?

Title _____ No. _____

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

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

Who Are You?

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

What programming language(s) do you use? _____

How Do You Use This Manual?

(List in order: 1 = Primary use)

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

Do You Like The Manual?

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

Comments?

(Please note page number and paragraph where applicable.)

From:

Name _____ Title _____ Company _____

Address _____ Date _____

SD-00742

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