DATA GENERAL
CORPORATION
Southboro,
Massachusetts 01772
(617) 485-9100

PROGRAM

STAND-ALONE DISK EDITOR

TAPE

Absolute Binary: 091-000083-00

ABSTRACT

The stand-alone disk editor permits every word in every block of disk space to be examined and/or modified. This editor requires a system with a minimum of 4K of core memory, a Teletype or video display console, and a disk. Optionally, program output can be directed to a line printer.

093-000092-00

Original Release - September, 1973

INTRODUCTION

The stand-alone disk editor permits users to examine and/or modify words and blocks of words on both fixed and moving head disks. This editor references disk words by logical addresses (a composite of block and word numbers) or by physical addresses (consisting of sector, head, and cylinder numbers). Thus this editor is contrasted with the disk file editor, OEDIT, which is file-oriented and does not permit the accessing of all disk space.

The stand-alone disk editor permits every word of disk space to be accessed, and thus is useful in resolving links in sequential files which have been altered by a system malfunction. This editor might also be used to patch or restore file information found in a system directory which might have been altered inadvertently. Use of the disk file editor on disk space managed by a disk operating system presupposes that the user is intimately familiar with the structure of disk files, directories, and initial disk block assignments of his system.

The following publications will assist users of the disk editor:

93-000056	Real Time Operating System User's Manual
93-000075	Real Time Disk Operating System User's Manual
93-000083	Introduction to the Real Time Disk Operating System
93-000084	Octal Editor Manual

TABLE OF CONTENTS

CHAPTER 1

DISK EDITOR INITIALIZATION

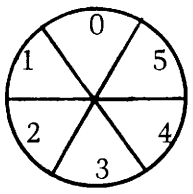
The stand-alone disk editor (091-000083) permits users to examine and/or modify words found in any portion of disk space on any disk device used with Nova family computers. The editor operates by accepting keyboard commands from the user and by displaying or modifying disk words on a disk block basis. That is , the editor reads a block of 256 words into a core-resident buffer and writes this buffer back to disk when a new block is specified to be operated upon.

Users of the disk editor on moving head units must know the number of heads found on the disk unit whose storage is being examined, and they must know into how many sectors per track the disk space is divided. The following list summarizes this information on four current disk types.

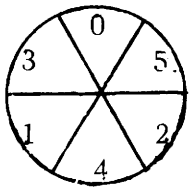
DGC Part No.	Disk Name	Number of Heads	Number of Sectors
4047A	Diablo 31	2	12
4047B	Diablo 33	2 (for each unit)	12
4048A	Century 111 (similar to IBM 2311)	10	6
4057A	Century 114 (similar to IBM 2314)	20	12

All disk sector/tracks contain 400_8 words each, and all moving head disks contain 203 cylinders per disk surface. All 203 cylinders are used by DGC disk operating systems.

One final concept of disk space addressing must be clarified before using the disk editor: mapped sector addressing. Mapped addressing refers to the practice of numbering sectors alternately in moving head systems. That is, in mapped addressing a logical sector address is used which does not correspond to the physical sector address. Instead of being numbered consecutively (and counter clockwise), logical sector addresses are assigned so that an intervening block (or blocks) is found between consecutive sector addresses.



Six Sector Unmapped Addressing



Six Sector Mapped Addressing

The following lists addressing schemes employed by various revisions of DGC disk operating systems:

Operating System Name and Revision	Sector Addressing
DOS, all revisions	Unmapped
RDOS, revision 0	Mapped
RDOS, revision 1	Mapped
RDOS, revisions 2 and higher	Unmapped
RTOS, all revisions	Unmapped

Loading the Disk Editor

The stand-alone disk editor is supplied on a length of paper tape in absolute binary form. Conventional binary load procedures are followed to load the editor.

When the program is loaded it will self-start. Location 2 contains the starting address, and location 3 is used to restart the program. After being loaded, the program will output the message

DISK UNIT?

The user responds with one of the following disk mnemonics:

Disk Type	Mnemonic
Fixed Head	DKn (n is the controller number)
Moving Head	DPn (n is the unit number)

The disk unit name may be changed at any point in the editing process by typing the command

U ddd

where ddd represents a valid disk device mnemonic.

If the unit selected is a moving head disk, the program outputs the message

MAPPED (Y or N)?

The user types Y if mapped sector addressing is used, and N if sector addressing is unmapped. Consult the list given earlier of addressing conventions used by DGC operating systems. Having stated to the editor whether or not mapped addressing is employed, the user need not concern himself further with the concept. Mapped

addressing does not affect logical block addresses, and the editor calculates the effective physical block addresses from given block addresses automatically. Knowing whether or not mapped addressing is used merely enables the editor to perform the effective, physical address calculation when a logical address is given to it.

Default Settings

Upon responding to the sector addressing question, the user may select a disk block and then proceed to issue any of the disk editor commands given in the following chapter.

The editor presumes that the disk unit is either a DGC 4047A or 4047B, i.e., that it is a Diablo type and has 12 sectors/track, 2 heads/unit, and 203 cylinders. If the device being examined has a different configuration, the user must issue the command

C s,h,c

where s is the octal number of sectors/track, h is the octal number of heads, and c is the octal number of cylinders.

To display the current disk configuration, the command

C/

is issued. This command causes the current configuration to be displayed with parameters given in the same order as was specified earlier for changing the configuration.

Also by default, values output during the editing process are in octal. To change them to ASCII, the user types A. To change back to octal, the user types O. When words are printed in ASCII mode, they are printed as pairs of ASCII characters:

```
RE AL SV
FO RT
```

The octal values of non-printable characters are displayed within angle brackets, e.g.,

<000><100> <000><003> LF E<000>

Default Settings (Continued)

By default, the teletype printer or video display is the output device. To change to the line printer, the user types

L

To change back to the teletype printer, the user types

T

When the line printer is the output device, all program output--even the results of keyboard calculations--is directed to the line printer. The editor will reject the "L" command in a system lacking a line printer by issuing a "?" response.

*****

CHAPTER 2

DISK EDITOR OPERATION

This chapter summarizes and describes the commands and command operators available in the stand-alone disk editor program. The disk editor does not permit command input to be in free format; users are cautioned to be precise in specifying input commands, especially with regard to the use of spaces and carriage returns. Depressing the RUBOUT key deletes the most recent alphanumeric input character but cannot be used to delete single command operators or command characters. The editor will output a question mark upon receipt of an invalid command.

The triangle symbol (Δ) will be used to denote a single ASCII space (040), and the left arrow ( ) will be used to denote a single ASCII carriage return (015).

A summary of operators and commands is found at the end of this chapter.

KEYBOARD CALCULATIONS

The disk editor has several commands which allow it to be used as an octal calculator to perform addition, subtraction, and exclusive ORing of octal values. These operations are useful throughout the editing process, especially in determining link words of sequentially organized files. All output goes to the current listing device, either the teletype printer or the line printer.

To add two octal integer values a and b, the user types

a+b=

and the editor will output the result to the current listing device immediately after the equals sign, followed by a carriage return. Note that no spaces separate the + and = signs.

To subtract two octal values a and b, the user types

a-b=

and the editor outputs the result immediately after the equals sign.

Similarly, to obtain the exclusive OR of two values, the user types

a!b=

and the editor outputs the result immediately after the equals sign.

Values preceding the equals sign may occur in combination with several operators, like a-b+c+d!e+f= . If the exclusive OR operator is used in such a combination it may be used only once per calculation; the algebraic sum of all operands following the ! sign become the second operand in the exclusive OR operation.

Single operands can be deleted by depressing the RUBOUT key for each character. Attempting to delete an operator causes the entire command line to be deleted.

DISK EDITING

Word and Block Pointers

The disk editor maintains two pointers, a current block pointer and a current word pointer. The block pointer is the number of the current disk block which is being examined and/or modified by the user. Only one block may be operated on at a time. The word pointer contains the address of the current word being accessed within the current block. If the current word pointer is incremented beyond 377_8, the current block is written back to disk and the next block is read in, with the block pointer incremented. The word pointer is incremented modulo 377_8.

The symbol for the current word pointer is a period (.).

The current word pointer (.) may be used as an argument with either the addition, subtraction, or exclusive OR calculator commands. Keyboard calculation operations do not affect the word or block pointers.

To display the current block number the user types B:. To display the current block physical address, the user types B , .

The current word pointer is changed only by explicitly entering a new word value or by modifying a current value by means of the +, -, or ! operators or /, ↑, or line feed command characters. Changing word pointers is described more fully in a following section.

Deleting Input Characters and Command Lines

To delete characters input as part of an editor command, the operator types one or more RUBOUTs. Each time the RUBOUT key is typed, a left slash (\ ) will be printed on the teletype console and the most recent remaining character is deleted. Deletion of an operator or command character causes the entire command to be deleted.

Using Command Operators

Each of the operators described previously for use in performing keyboard calcu' tions can be used in combination with any disk editor command characters. Thus the command

```
.+3/
```

would cause the contents of the third word from the present location to be displayed. (/ is the editing command used to display disk words).

Opening and Examining Disk Words in Logical Blocks

Any location on any disk surface may be opened and examined by typing its address and following this by a right slash. Addresses may be expressed as either a word offset within a logical block or as an offset within a physical block.

When the disk editor is first entered, its block pointer is set to -1 and its word pointer is set to 0. Thus to examine a word, both the word and the block pointers must be set to the desired values.

To set the word and block pointers for the referencing of a word within a logical block, the logical block number and word offset within the logical block must be separated by a colon. Thus, 10:377 sets the block pointer to 10_8 and the word pointer to 377_8. The command

```
10:377/
```

causes the contents of the last word in logical block 10 to be printed on the current listing device. To display the contents of any other word in this block, all that must be typed is the address of the word within the block, followed by a right slash. Thus to display the contents of word 300 in this block, all that must be typed is

```
300/
```

since a missing logical block number causes the current block number to be used. If a different logical block number is specified, the current block will be written out to disk and the new block will be read into the editor buffer so that any word within the new block may be accessed.

The same word can be examined in either the ASCII or the octal modes, regardless which mode the editor is in, by typing one of two command characters. Typing an equal sign (=) causes the current word to be displayed as six octal digits. Typing a single quote (') causes a current word to be displayed as two ASCII characters. The octal equivalents of non-printable characters are displayed within angle brackets.

Opening and Examining Words in Physical Blocks

To examine a word which is specified as an offset within a physical block, the physical block address must first be specified by sector address, head number, and cylinder number in that order. These values must be separated from the word offset by a colon. To display the contents of word 377 in a block found in the 0th sector, 1st head, and cylinder number 2, the command

```
0,1,2:377/
```

must be issued.

Just as with logical blocks, when the physical block address has been specified, other words within that block can be examined by simply specifying the address of the word within the physical block. Thus to examine the 0th word in the physical block addressed earlier, the command

```
0/
```

is issued. After the right slash is typed, the contents of location 0 are displayed either as six octal digits or as two ASCII characters.

Examining Successive Words

Successive words, in forward or preceding locations, may be examined by typing one of two commands: line feed or up arrow (↑). Typing a line feed causes the next word to be displayed; typing an up arrow causes the preceding word to be displayed.

If by using either of these commands the current block pointer becomes changed, the current block becomes written to disk and the new block is read by the editor automatically. The current block pointer is changed by the editor if the word before 0 or the word after 377 is requested.

Examining a Group of Words

A group of disk words can be displayed by typing the beginning and ending addresses, separated by a left angle bracket (<). Thus the command

```
0< 377/
```

causes the contents of the current disk block to be printed on the current listing device. If block boundaries are crossed, block pointers are updated automatically.

If words at only a certain repeated offset within a given range are desired, the user may specify this increment by typing an at-sign (@) and the desired increment immediately before the right slash. Thus the command

Examining a Group of Words (Continued)

0< 377@ 11/

causes every 9th word in the current disk block, starting with the first word in the block, to be printed on the current listing device.

The operator can prematurely terminate output being printed by typing the escape key, ESC (a $ is echoed). This feature is especially useful when the teletype printer has been selected as the output device.

Modifying a Disk Word

After a disk location has been opened by means of the right slash command described earlier, the word contained in that location can be modified if desired. To change the contents of the current location, a user simply types the new word contents and a carriage return.

If the new contents are to be numeric, the user types six or less octal digits followed by a carriage return. Numeric values are right justified in the location.

To enter ASCII character pairs, these character pairs must be preceded by a double quote character (a trailing quote character is optional) and must be followed by a carriage return. To input a single left-justified ASCII character, the character must be both preceded and followed by a double quote character, followed by a carriage return. The right byte of the word receiving the ASCII character will be set to zeroes, i.e., it will be an ASCII null character.

The following table illustrates the contents of words after selected octal and ASCII input commands.

Command	New Contents
012345↓	012345
0123↓	000123
"AB↓	040502 (ASCII A and B)
"AB"↓	040502
"A"↓	040400 (ASCII A and null)

The carriage return terminator closes the location after the new contents are deposited there. In order to access the location again, it must be explicitly opened and re-examined by means of the right slash command. The line feed and up arrow characters may also be used to open or close locations. If these terminators are used they will cause the first location to be closed, the next (or previous) location to be opened, and its contents to be displayed.

Modifying a Disk Word (Continued)

Failure to follow the new contents by a terminator will cause new values -- intended as input to the opened location--to be interpreted as part of a new edit command. This feature prevents the inadvertent entering of spurious data into locations, and also allows several commands to be entered on the same line.

Thus the following three commands could be entered on the same line:

1/123456 7/000000 11/000432 ↵

Each of the three locations would be closed and their contents would remain unaltered.

Transferring a Disk Block

It is possible to copy the contents of a disk block into a new block of disk space. This is done by issuing the command

X nn ↵

where nn is the new logical block number. Thus the current block which has been modified and/or examined by the user will be written out to disk block nn. The current block pointer becomes set to nn.

Command Summary

Command	Meaning	Page Where Described
A	Change the output mode to ASCII.	1-3
B:	Display the current block number.	2-2
B,	Display the current block physical address.	2-2
C△argument	Change the disk configuration to that specified by the argument. The argument must be in the following format: number of sectors/track, number of heads, and number of cylinders (separated by commas).	1-3
C/	Display the current disk configuration. The configuration will be displayed in the following format: number of sectors/track, number of heads, and number of cylinders.	1-3
L	Make the line printer ($LPT) the output device.	1-4
O	Change the output mode to octal.	1-3
T	Make the console ($TTO) the output device.	1-4
U△argument	Change the unit number (but not the unit type) to the value specified by the argument.	1-2
X△argument	Write the current block specified by argument. argument may be a physical or a logical block address.	2-6
argument/	Display the contents of the location or locations specified by the argument.	2-3 ff
$argument_1$/△$argument_2$	Display the contents of the location specified by $argument_1$, and change these contents to those specified by $argument_2$.	2-5
↑	Close the current location and display the contents of the previous one.	2-4
(line feed)	Close the current location and display the contents of the next one.	2-4

Command	Meaning	Page Where Described
)	) is the conventional symbol for a carriage return. Close the current location.	2-1, 2-5
=	Display the contents of the current location in the numeric mode.	2-3
=	Determine the result of the previous operator combination.	2-1
'	Display the contents of the current location in ASCII mode.	2-3
RUBOUT	Delete an input editor command character.	2-2
ESC	Terminate printed output.	2-5

Operator Summary

Operator	Meaning	Page Where Described
a+b	add a and b	2-1
a-b	subtract b from a	2-1
a<b	locations a through b inclusive	2-4
"	ASCII characters will be input via the teletype keyboard.	2-5
.	current location	2-2
a:b	interpret this input as block a : word b	2-3
a,b,c	interpret this input as physical address sector a, head b, and cylinder c in that order.	2-4
a!b	exclusive OR a and b	2-1
@i	step output by increments of i (used with <). The default increment is 1.	2-5

ERROR MESSAGES

If a disk hardware error is detected during the operation of the editor, a brief diagnostic message will be output on the console, followed by the message:

– STATUS n

where n is the disk status word (see How to Use the Nova Computers, "Disks").

The following error messages are output:

DATA LATE	The data channel has failed to respond in time to a request for access.
WRITE ERROR	The program has specified "Write" and the selected track-sector is write-protected.
NO SUCH DISK	The fixed head disk selected by the program is not connected to the bus.
DATA ERROR	The cyclic check word read from the fixed head disk differs from that computed by the control for the data in the block.
CHECKWORD ERROR	The cyclic check word read from the moving head disk differs from that computed by the control for the data in the block.
UNSAFE ADDRESS	There is a malfunction in the moving head disk drive.
SEEK ERROR	The selected drive failed to position its head as requested.
END OF CYLINDER	The moving head disk controller has reached the end of a cylinder but the sector counter is not zero, but the data operation has ended anyway.

If desired, disk editor commands can be input via the console to determine which areas of the disk storage are faulty.

*****

APPENDIX A

USING THE DISK EDITOR TO EXAMINE DIRECTORIES

DIRECTORY STRUCTURES

Since one of the most common uses of the disk editor will be to examine and possibly modify portions of RDOS directories, this appendix illustrates the use of the editor to trace through a collection of SYS.DR's. To prepare for this illustration, the following review of directory structures in RDOS, revision 02, is presented.

SYS.DR Structure

As described in the RDOS User's Manual, information required about files in a given partition or subdirectory is kept within a system file directory called SYS.DR . The information within every SYS.DR includes file names, the length in bytes of the files, and the files' attributes and characteristics.

System directories employ a hashing algorithm* to speed up access of directory entries. Moreover, an initial system directory area is allocated (at the time the system is fully initialized) for entries in the primary partition on a moving head disk. This area (called a frame) is a contiguous set of disk blocks; the set is contiguous to minimize moving head travel time. Subdirectories and secondary partitions allocate system directory storage as required.

The primary partition frame size is dependent upon the type of disk unit on which it is located. The following list gives frame sizes for different DGC moving head disk types:

Unit Type	Frame Size (in decimal)
4047	97
4048	193
4057	773

The structure of SYS.DR for both system file directories and subdirectories is identical. That is, SYS.DR is a randomly organized file, and the first word in each block of the file is the number of files that are listed in this block of SYS.DR . Following this word is a series of 22_8 word entries, called user file descriptions or UFDs, which describe each file. The following page gives an illustration of any block of SYS.DR.

*See Appendix B.

Word	Contents
0	Number of files in this block of the directory (16_8 maximum)
1 . . . 22	User file description
23 . . . 44	User file description
. . .	

SYS. DR BLOCK

Each 22_8 word UFD for regular files has the fixed structure shown below. The UFD contains information about the file describing its name, its two-character name extension, the file attributes and characteristics, the file size, address of the first block, and a logical device code describing the device associated with this file.

Word Displacement	Contents
0-4	File name
5	Extension
6	Attributes
7	Link access attributes
10	Block count -1
11	Byte count in last block
12	First address (i.e., logical address of first block in the file.)
13	Year/day last accessed
14	Year/day created
15	Hour/min created
16	UFD variable info
17	UFD variable info
20	User count
21	DCT link

User File Descriptions for link files are also 22_8 words in length, but their structure necessarily differs somewhat from that for regular files:

Word Displacement	Contents
0-4	File name
5	Extension
6	Attributes
7-13	Directory specifier (or 0)
14-20	Alias name (or 0)
21	Alias extension

All file names within each file directory must be unique. An attempt to add a file name to a directory when the same file name already exists in that directory causes an error indication to occur. Deleted files are indicated by file names whose first two characters have been replaced by nulls. A non-zero file use count indicates that one or more users have opened the file. If a hardware malfunction should occur, this count will often be erroneous and must be cleared (via CLI command CLEAR) to zero before its associated file can be renamed or deleted.

Map Directory Structure (MAP.DR)

Associated with each partition or subdirectory is a file which keeps track of the availability of disk blocks within each file space. This file is called a map directory, MAP.DR . Each bit of each word in MAP.DR indicates whether a specific block is in use or not. Block assignments are from left to right, in ascending block order starting with the first block of the contiguous disk space to be managed. MAP.DR is a contiguous file whose size, in the primary partition, is determined by the size of the disk where MAP.DR resides.

Word	Contents
0	Block allocation map, 1 bit/block, from left to right
.	in ascending block order starting with block number 6.
.	0 ⟷ block is available, 1 ⟷ block is in use.
.	
377	

Since MAP.DR has a UFD in its system directory, the size of any MAP.DR can be found by examining word 10 of its UFD.

Initial Disk Block Assignments

The first 20 octal blocks of disk storage on every disk device have fixed assignments, with the remaining blocks free for either system use or user file storage. Blocks 0 through 5 are reserved for the bootstrap. Block 6 is the first index block of the random file SYS.DR . Blocks 7 through 16 are reserved for random file indexes for program swaps occurring within the primary partition; block 17 is reserved for the first block of the contiguous file MAP.DR .

Disk Block Number	
0, 1, 2, 3, 4, 5	Bootstrap
6	First Index Block of SYS.DR (random file)
7, 10, 11, 12	Background Program Swap File-index storage
13, 14, 15, 16	Foreground Program Swap File-index storage
17	First Block of MAP.DR

As mentioned earlier, the map directory, MAP.DR, is a file indicating which blocks of disk storage are currently in use and which are free for assignment.

Secondary Partition Structure

Secondary partitions are mutually exclusive subsets of total (primary partition) file space. Secondary partitions are created via the CLI command CPART, which requires that the partition be given a name and a specified size in disk blocks. The partition name is then inserted into the SYS.DR of the primary partition. A secondary partition's size becomes fixed when it is created, since secondary partitions are contiguous files.

In addition to having a SYS.DR, each secondary partition also contains a MAP.DR and program swap file index space. The minimum size of any secondary partition is 32_{10} disk blocks; this leaves 14 blocks for file storage within the partition. The system requires a minimum initial assignment of at least 8 blocks of SYS.DR storage for subdirectories and secondary partitions. This is required for the hashing of peripheral entries ($TTR, etc.) into the system directory at initialization time.

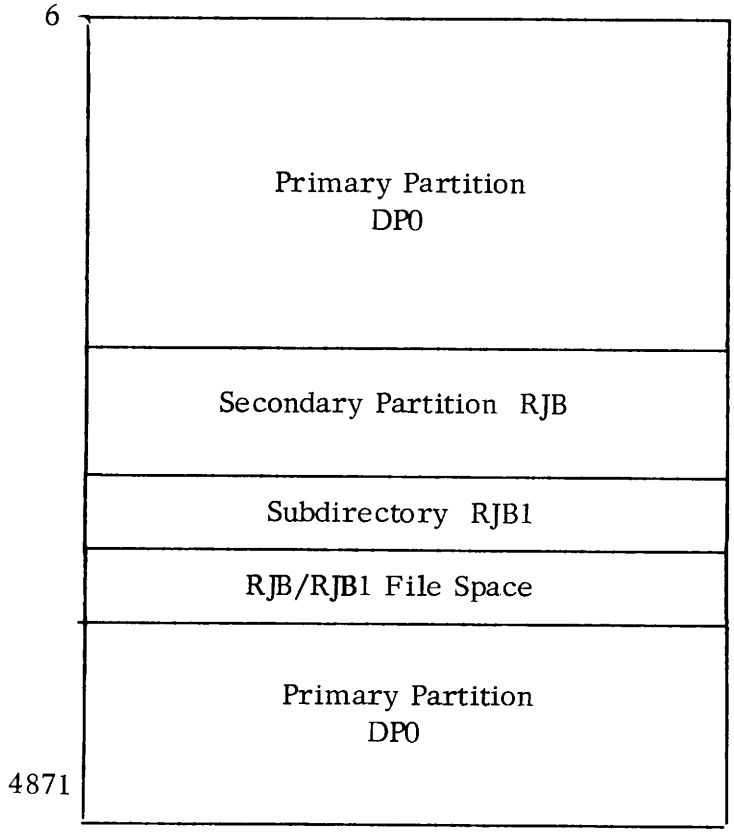
Subdirectory Structure

Subdirectories are mutually exclusive subsets of a parent partition's file space. Unlike secondary partitions, subdirectories have no defined amount of file space. Instead, subdirectories take file space from the parent partition as required and release the space when it is no longer needed.

Newly created subdirectories consist of three blocks: SYS.DR's initial index block and data blocks for the SYS.DR and MAP.DR entries. After a subdirectory is initialized, peripheral entries are inserted into it's SYS.DR. Since subdirectories have no file space of their own but borrow from the parent, the map directory entry in each subdirectory's SYS.DR points to the parent partition's MAP.DR entry.

TRACING THROUGH THE SYSTEM DIRECTORIES

The disk space to be examined in this appendix consists of a primary partition, DP0, a secondary partition, RJB, and a subdirectory, RJB1, within the secondary partition. For illustration purposes, RJB was created with the minimum size of 32 disk blocks. DP0, the primary partition, is a 4047a moving head disk; thus the initial file space extends from block 6 through block 4871. (HIPBOOT, the disk bootstrap, occupies blocks 0-5.) A block diagram of this disk space is shown below.



DISK SPACE ILLUSTRATION

The secondary partition and subdirectory were created by means of CLI commands CPART and CDIR. Issuing the CLI command DISK in each of these directories gives an interesting clue as to the structures of these directories. The following illustrates the commands used to create the directories, and the results of the CLI DISK commands:

```
CPART RJB 32
R
INIT RJB
R
DIR RJB
R
DISK
LEFT: 14, USED: 18
R


CDIR RJB1
R
(INIT,DIR) RJB1
R
DISK
LEFT: 4, USED: 28
R
```

Evidently 18 of the original 32 blocks in RJB are used for SYS.DR, MAP.DR, and push space file indexes. The creation of subdirectory RJB1 reduces to 4 blocks the amount of file space available to both directories.

To further aid us in our examination of these directories, we must use the CLI in yet another way before beginning our use of the disk editor proper. In order to determine the relative positions of entries within the SYS.DR file indexes of DP0, RJB, and RJB1, we must* request unsorted lists of entries within each of these three system directories. This we do by means of the CLI command LIST/A/E/L. By examining the relative position of an entry in the SYS.DR LIST, we can approximate the position of that entry's UFD within its SYS.DR random file index. A portion of the DP0 list, and the entire RJB and RJB1 lists are given below and on the next page.

```
MAC.SV          14792   SD      7/23/73 9:35    7/23/73    [000524]   0.
RTSYSLIST.        901           6/20/73 9:5     6/20/73    [001462]   0.
TIMEC.RR          214   D       6/20/73 9:5     6/20/73    [001464]   0.


RJB.DR          16384   CTY     7/23/73 9:43    7/23/73    [002030]   0.
THOOT.SV         1530   S       6/19/73 12:59   6/19/73    [001466]   0.
MAC.PS          16832   D       7/23/73 9:35    7/23/73    [000562]   0.
```

Portion of DP0 List

*Unless we apply the hashing algorithm. See Appendix B.

```
RJB1.DR          512   DY      7/23/73 9:43   7/23/73   [0   052]   0.
$TTI.              0   APW     7/23/73 9:43   7/23/73   [0   000]   0.
SYS.DR         39936   APWDYI  7/23/73 9:43   7/23/73   [002053]   2.
MAP.DR             4   APWCI   7/23/73 9:43   7/23/73   [002041]   0.
$PTP.              0   RAP     7/23/73 9:43   7/23/73   [000000]   0.
$PTR.              0   APW     7/23/73 9:43   7/23/73   [000000]   0.
$TTO.              0   RAP     7/23/73 9:43   7/23/73   [000000]   0.
$TTP.              0   RAP     7/23/73 9:43   7/23/73   [000000]   0.
$TTR.              0   APW     7/23/73 9:43   7/23/73   [000000]   0.
$LPT.              0   RAP     7/23/73 9:43   7/23/73   [000000]   0.
```

RJB LIST

```
$TTI.              0   APW     7/23/73 9:43   7/23/73   [000000]   0.
SYS.DR         39936   APWDYI  7/23/73 9:43   7/23/73   [002052]   2.
MAP.DR             4   APWCI   7/23/73 9:43   7/23/73   [002041]   0.
$PTP.              0   RAP     7/23/73 9:43   7/23/73   [000000]   0.
$PTR.              0   APW     7/23/73 9:43   7/23/73   [000000]   0.
$TTO.              0   RAP     7/23/73 9:43   7/23/73   [000000]   0.
$TTP.              0   RAP     7/23/73 9:43   7/23/73   [000000]   0.
$TTR.              0   APW     7/23/73 9:43   7/23/73   [000000]   0.
$LPT.              0   RAP     7/23/73 9:43   7/23/73   [000000]   0.
```

RJB1 LIST

Having gotten the CLI LIST information, we are now ready to proceed with our use of the disk editor proper. Upon being loaded, the standard editor prologue is output:

```
DISK UNIT?  DP0
MAPPED(Y OR N)? N
```

Unmapped addressing (N) was selected, since addressing in RDOS 02 is unmapped.

The first step in this illustration is to examine the SYS.DR of DP0 to determine where the initial contiguous area of SYS.DR for the primary partition resides. To do this, we specify output to the line printer (and output is in octal by default):

```
L
```

A carriage return is echoed to acknowledge receipt and execution of the command.

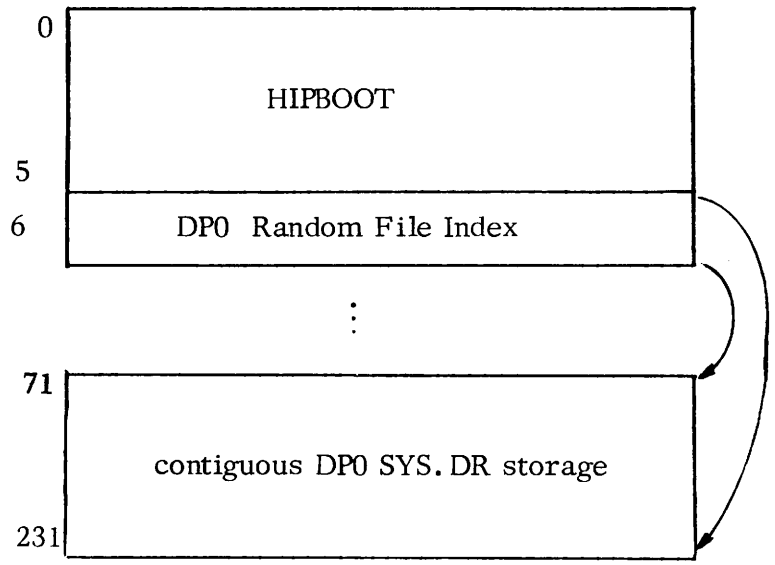
We now request that the contents of DP0's SYS.DR random file index be printed:

6: 0<377/

This command causes the following output to be printed on the line printer (only locations 0-157 are reproduced here, since the last non-zero entry is found in word 140):

```
000/  000072  000073  000074  000075  000076  000077  000100  000101
010/  000102  000103  000104  000105  000106  000107  000110  000111
020/  000112  000113  000114  000115  000116  000117  000120  000121
030/  000122  000123  000124  000125  000126  000127  000130  000131
040/  000071  000132  000133  000134  000135  000136  000137  000140
050/  000141  000142  000143  000144  000145  000146  000147  000150
060/  000151  000152  000153  000154  000155  000156  000157  000160
070/  000161  000162  000163  000164  000165  000166  000167  000170
100/  000171  000172  000173  000174  000175  000176  000177  000200
110/  000201  000202  000203  000204  000205  000206  000207  000210
120/  000211  000212  000213  000214  000215  000216  000217  000220
130/  000221  000222  000223  000224  000225  000226  000227  000230
140/  000231  000000  000000  000000  000000  000000  000000  000000
150/  000000  000000  000000  000000  000000  000000  000000  000000
```

The printout indicates that the DP0 SYS.DR is found in blocks 71 through 231. If this SYS.DR ever became full of UFD entries, new disk blocks would be linked as needed to SYS.DR as blocks are linked in a sequential file. By printing the contents of block 6, we can now construct a partial disk block map of our illustration, shown on the following page.



Having located the contiguous area of SYS.DR entries, we must now find the entry for secondary partition RJB. The algorithm used to place entries in SYS.DR places these entries over the entire SYS.DR storage area, and each block of SYS.DR will contain approximately the same number of entries as each other block. Since our initial list of DP0 was unsorted, the order in which entries were listed corresponds to their relative positions within SYS.DR. Thus, since RJB.DR appeared approximately two thirds through the list, we might expect that RJB.DR would be found near block 171. Issuing the print command 171:0<377/, we receive the following output (only the pertinent portion is reproduced):

```
000/  <000><002>  RT  SY  SL
004/  IS  T<000>  <000><000>  <000><000>
010/  <000><000>  <000><001>  <001><207>  <003>2
014/  <007><316>  <007><316>  <011><005>  <000><000>
020/  <000><000>  <000><000>  <000><033>  TI
024/  ME  C<000>  <000><000>  <000><000>
030/  RB  <000><004>  <000><000>  <000><000>
034/  <000><326>  <003>4  <007><316>  <007><316>
040/  <011><005>  <000><000>  <000><000>  <000><000>
044/  <000><033>  <000><000>  <000><000>
```

Upon examining this printout, we can see that block 171 contains two entries: RTSYSLIST and TIMEC. Since these entries immediately precede RJB.DR on the DP0 list, RJB.DR must be found on block 172. Although it does not occur here, any UFD entry in any SYS.DR which has 2 leading nulls in the filename (word 0) indicates that the file has been deleted from the directory. Printing the first few words of block 172, we find the RJB.DR entry:

```
000/  <000><001>  RJ  B<000>  <000><000>
004/  <000><000>  <000><000>  DR  <014><010>
010/  <000><000>  <000><037>  <002><000>  <004><030>
014/  <007><357>  <007><357>  <011>+  <000><000>
020/  <000><000>  <000><000>  <000><033>  <000><000>
024/  <000><000>  <000><000>  <000><000>  <000><000>
```

Having found the UFD for secondary partition RJB, let us examine this UFD more closely. Word 0 of this block contains the value 1, indicating that there is only one UFD in this block of the DP0 SYS.DR. The UFD for RJB therefore extends from word 1 through word 22. Words 1 through 5 contain the name of the directory, RJB, with trailing null bytes. The extension, DR, is found in word 6 and indicates that RJB is a directory. Word 7's contents, expressed octally, equal 006010. This word contains the file (directory's) attributes. The attributes of RJB are that it is a directory, a partition, and is contiguously organized. A complete list of file attributes is given in the RDOS User's Manual in the discussion of the system call .CHATR, and these attributes are also listed in the RDOS User Parameters, PARU.SR .

Word 10 contains the link access attribute's word; there are no link access attributes. The next word contains the number of blocks in the directory, less one; the directory is a randomly organized file consisting of 40_8 blocks. The next word indicates the number of bytes of free storage assigned to the last block of RJB; there are 1000_8 such bytes (256_{10} words). Word 13 contains the address of the first block in directory RJB; this is 2030, as we indicated earlier.

The next two words, 14 and 15, contain information describing the date the directory was last accessed and the day that it was created. Since the last access date is maintained by the system only for SYS.DR entries within directories, the information in word 14 is not meaningful in this case. The creation date in word 15 is meaningful, however. This value represents the number of days since January 1, 1968, that the file was created. In this case, 4438_8 is equivalent to July 23, 1973. Word 16 contains

information describing the hour and minute that the directory was created. RJB.DR was created at the 9th hour (left byte) and the 43rd minute ("+" equals 053 octal).

Word 16 is used by the operating system to store attribute words temporarily when links are accessed, and when files are opened either exclusively or for reading only so that the original attributes can be restored when the file is closed. Word 17 is never used by RDOS revision 02, and thus it too contains zeroes.

The last word in the UFD, word 22, contains the device code of the primary partition where the file (directory) is found. The device code of DP0 is 33_8.

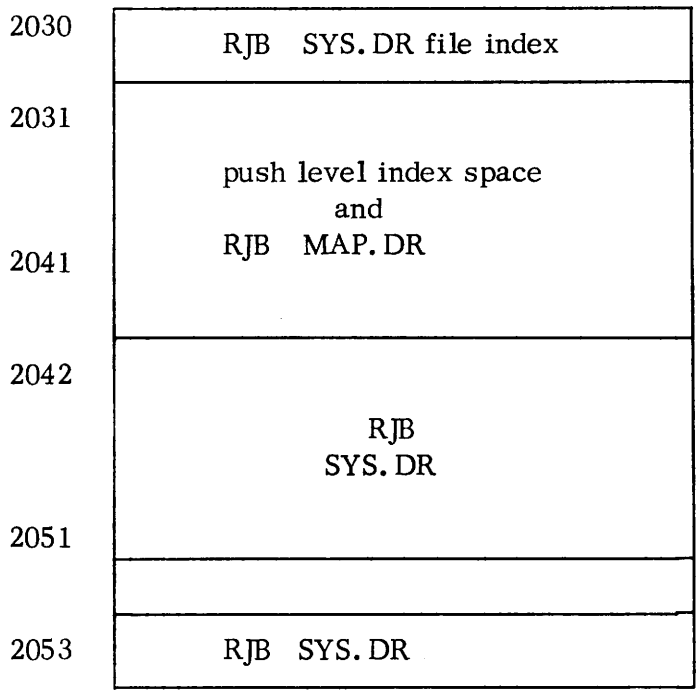
Having examined RJB's UFD, we can now determine where that directory is and can proceed to analyze its structure. Word 13 of the UFD indicated that the starting block address of that contiguous file was block number 2030. Since the file is contiguous, it extends from 2030 through 2067.

As with any user directory, the first block contains the random file index for the system directory, SYS.DR. Issuing the print command 2030:0 < 377/, we receive the following on the line printer (only the useful area is reproduced here):

```
000/  000000  000000  000000  000000  000000  000000  000000  000000
010/  000000  000000  000000  000000  000000  000000  000000  000000
020/  000000  002053  000000  000000  000000  000000  000000  000000
030/  000000  000000  000000  000000  000000  000000  000000  002043
040/  002042  000000  002050  000000  002047  002044  002046  000000
050/  002045  000000  000000  000000  000000  000000  000000  000000
060/  000000  000000  000000  000000  000000  000000  000000  000000
070/  000000  000000  000000  000000  000000  000000  000000  000000
100/  000000  000000  000000  000000  000000  000000  000000  000000
110/  000000  000000  000000  000000  000000  002051  000000  000000
```

An examination of this index indicates that the RJB SYS.DR is found on blocks 2042 through 2051 and block 2053. Note that SYS.DR blocks in secondary partitions and subdirectories are assigned as needed; there is no necessary initial contiguous

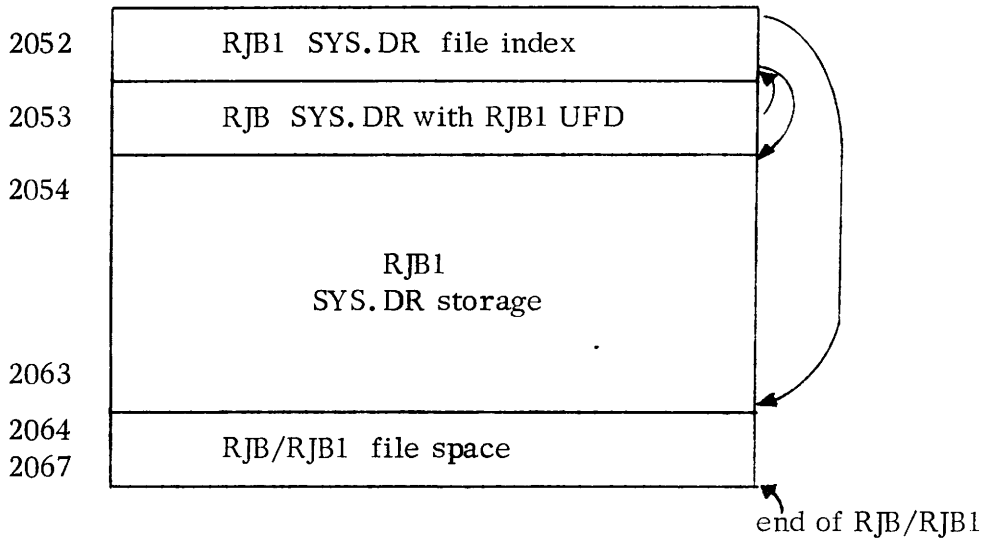
SYS. DR allocation. The intervening disk storage blocks 2031 through 2041 are therefore allocated for program swap file index storage and for the map directory. Thus a preliminary map of RJB looks as follows; we will fill in more details as we proceed to examine RJB and then subdirectory RJB1:



Going through a process similar to that we performed earlier with DP0 SYS. DR, we find the UFD for subdirectory RJB1 in block 2053. Looking at the starting logical address of the subdirectory, we find that its random file index is located in block 2052. We now print that random file index, and find that RJB1's SYS. DR is found in blocks 2054 through 2063:

```
000/  000000  000000  000000  000000  000000  000000  000000  000000
010/  000000  000000  000000  000000  000000  000000  000000  000000
020/  000000  000000  000000  000000  000000  000000  000000  000000
030/  000000  000000  000000  000000  000000  000000  000000  002055
040/  002054  000000  002062  000000  002061  002056  002060  000000
050/  002057  000000  000000  000000  000000  000000  000000  000000
060/  000000  000000  000000  000000  000000  000000  000000  000000
070/  000000  000000  000000  000000  000000  000000  000000  000000
100/  000000  000000  000000  000000  000000  000000  000000  000000
110/  000000  000000  000000  000000  000000  002063  000000  000000
```

We can now sketch the disk block map of subdirectory RJB1:



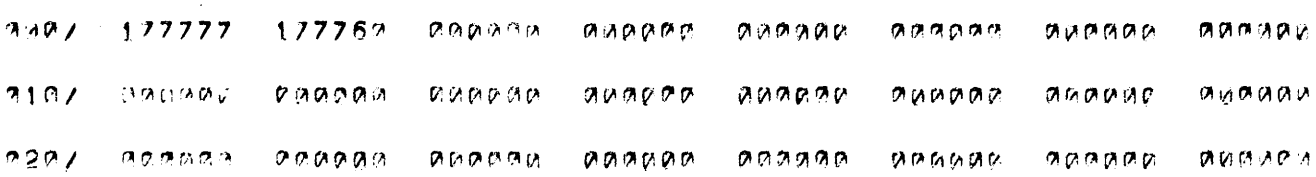
Since the range of file index entries is from 2054 through 2063, RJB1 SYS.DR storage is found in these blocks. Furthermore, since a subdirectory shares file space with its parent directory (and the parent, RJB, was created 32 blocks in length), the remaining blocks, 2064 through 2067, are the file space which is to be used by both directories RJB and RJB1.

To conclude this exercise, let us examine RJB1's SYS.DR to determine where its map directory is located, and let us examine the UFD for RJB1's SYS.DR to verify the conclusions about disk block assignments which we have made. Issuing the open and print command 2054:0<47/, we receive the following:

```
000/  <000><042>  SY  S<000>  <000><000>
004/  <000><000>  <000><000>  DR  D<207>
010/  <000><000>  <000>P  <002><000>  <004>+
014/  <007><361>  <007><357>  <011>+  D<207>
020/  <002><000>  <000><000>  <000><033>  MA
024/  P<000>  <000><000>  <000><000>  <000><000>
030/  DR  @<213>  <000><000>  <000><000>
034/  <000><014>  <004>!  <007><361>  <007><357>
040/  <011>+  @<213>  <000><000>  <000><000>
044/  <000><033>  <000><000>  <000><000>  <000><000>
```

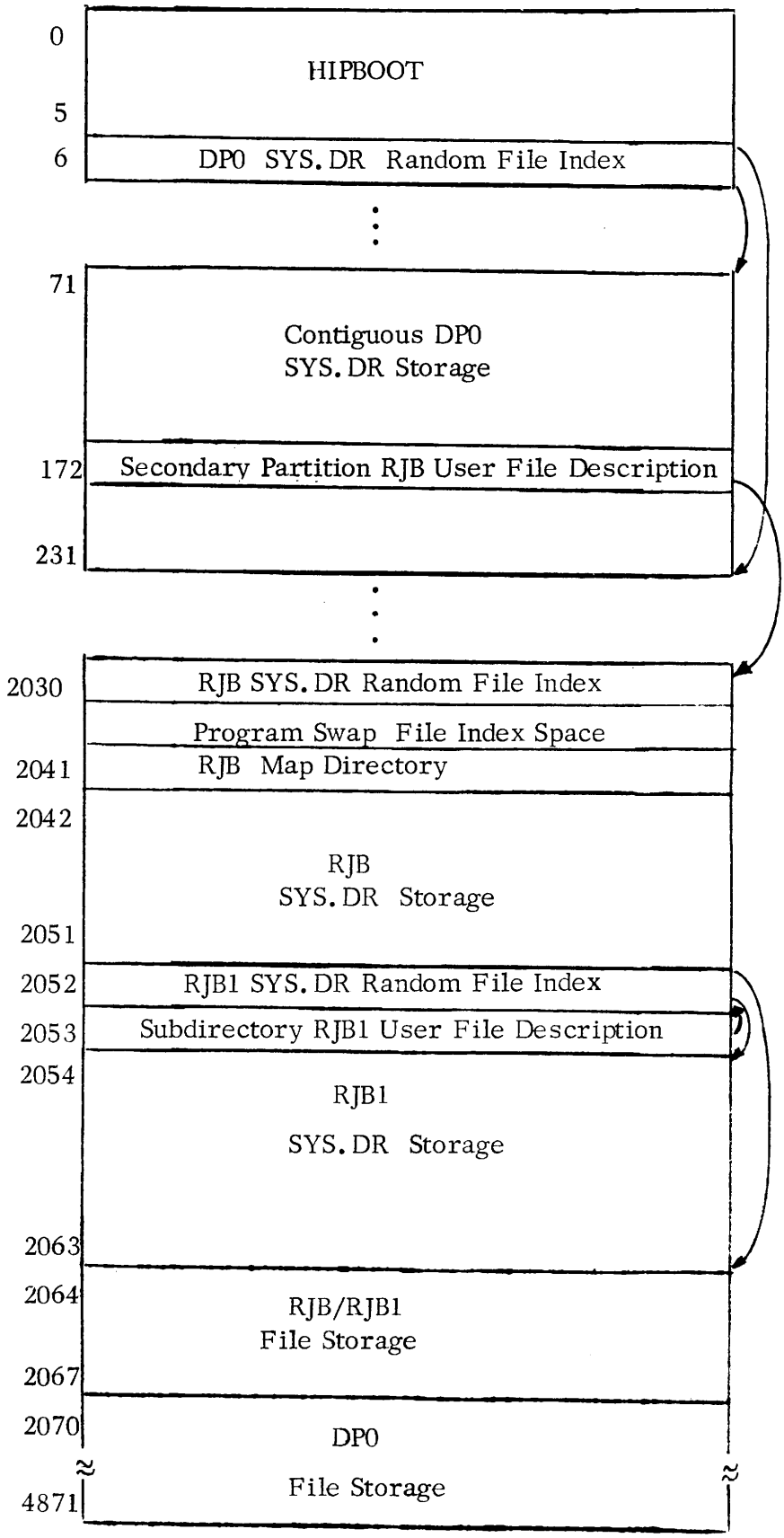
We see from this printout that the start of RJB1's SYS. DR is block 2052 (see word 13, and convert its contents to octal). Similarly, RJB1's map directory is said to have a starting address of 2041 (see word 35, and convert its contents to octal). Since subdirectories do not have their own map directories but rather they use the parent's map directory, this confirms our earlier conclusion that RJB's map directory was located in block 2041.

Finally, printing block 2041, the RJB/RJB1 map directory, we see the following:



As discussed earlier, map directories are contiguously organized files, and each bit in each word corresponds to the usage of each block in the file it describes. Thus the first 29 blocks of RJB are in use, since the first 29 bits are set; "60" in the second word indicates that the last four blocks are free, and this confirms our conclusion that blocks 2064 through 2067 comprise the unused file space in this partition.

This concludes our examination of directories and partitions. The following disk block map summarizes our observations about the sample disk space which has been the object of this illustration.



DISK BLOCK MAP ILLUSTRATION

APPENDIX B

RDOS HASHING ALGORITHM

The most convenient method for determining a file's UFD position within a SYS.DR is to obtain an unsorted list of all entries within the SYS.DR. However, if the system has crashed and no recent list is available, the hashing algorithm must be applied to a file name to determine the relative position within SYS.DR's random file index of the disk block address containing the file. Although this method is foolproof, it is quite tedious and can be avoided in most instances by adhering to the practice of generating frequent unsorted lists of SYS.DRs.

The first step in the RDOS revision 02 and higher hashing algorithm is to consider the file name and its two-character extension as a series of 16 bit integers. The first two characters, packed left to right, become the first integer, etc. If there is an extension, it is calculated as an integer separate from the file name (e.g., A.DR is calculated as two integers, "A" and "DR"). Having created this series of up to 6 integers, add up the series (discarding any overflow digits) to obtain a 16-bit sum S. Divide S by the frame size of the master device. The following disk units and their frame sizes were given in Appendix A:

Unit Type	Frame Size (in octal)
4047	141
4048	301
4057	1405

The quotient Q obtained by dividing S by the frame size (F) will include a remainder R where R is in the range 0 to F-1 inclusive. This remainder R indicates the relative position within a SYS.DR random file index where the address of the disk block containing the UFD is found. Merely examine this disk block to find the file's UFD.

It is conceivable that a block indicated by the hashing algorithm has overflowed, i.e., that it was already filled with file UFDs when an attempt was made to insert a given file's UFD into the block. If hash overflow has occurred, merely add one frame size to R to determine the relative position of the block address within SYS.DR's random file index. If this block overflowed too, add two frame sizes. Continue the process until the relative position of the block address within SYS.DR is found. In the normal course of events, SYS.DR will not need to be extended more than one frame.

To illustrate the hashing algorithm, let us derive the hash position for file name RJB.DR. This file name is illustrated in Appendix A, pages A-6, A-9, and A-10.

The first step in the algorithm is to convert RJB.DR into a series of integers:

R	J	051112
B	null	041000
D	R	042122

Adding up the three integers, we obtain the sum 154234:

$$\begin{array}{r} 051112 \\ 041000 \\ 042122 \\ \hline S = 154234 \end{array}$$

Division of S by the frame size (F), 141_8, yields a quotient with a remainder of 101_8:

$$\begin{array}{r} 10721\ R101 \\ \hline 141_8 \overline{)154234_8} \\ 141 \\ \hline 1323 \\ 1247 \\ \hline 544 \\ 302 \\ \hline 242 \\ 141 \\ \hline 101 \end{array}$$

The remainder, 101, is the relative position within SYS.DR's random file index which contains the address of the disk block containing RJB.DR's UFD. Examining such an index on page A-8, we find disk block address 172 in displacement 101. Page A-10 confirms that our answer is correct, since block 172 does indeed contain the UFD for file RJB.DR.

****

DATA GENERAL CORPORATION
PROGRAMMING DOCUMENTATION
REMARKS FORM

DOCUMENT TITLE _____

DOCUMENT NUMBER (lower righthand corner of title page) _____

TAPE NUMBER (if applicable) _____

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

General Comments and Suggestions for Improvement of the Publication.

FROM: Name: _____ Date: _____
Title: _____
Company: _____
Address: _____
_____

cut along dotted line

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: Programming Documentation

FOLD UP SECOND FOLD UP

STAPLE