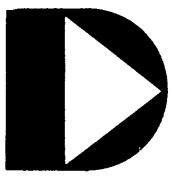
DISK OPERATING SYSTEM FOR DISKETTE DOS.C 1.1 Systems Guide

March 6, 1975

Model Code No. 50130

DATAPOINT CORPORATION



The Leader in Dispersed Data Processing

PREFACE

This DOS.C SYSTEMS GUIDE is written to describe the specific characteristics of Datapoint Corporation DOS.C, supporting 1100, 2200 and 5500 computers with the 9380-series diskette drives. This manual is intended as strictly supplementary to the DOS USER'S GUIDE, with which the reader is assumed to be familiar.

It is important to realize that the DOS characteristics documented in this manual are here precisely because they are different in the DOS.C implementation than for certain of the other Datapoint Corporation DOS; as such, programs written based on features documented herein and not documented in the DOS USER'S GUIDE will not be as easily transportable to other versions of the DOS.

CHAPTER 1. INTRODUCTION TO DOS.C

DOS.C is Datapoint Corporation's Disk Operating System supporting Datapoint 1100, 2200 and 5500 computers operating in conjunction with up to four 9380 series diskette drives.

1.1 Planning for DOS.C

The recommended configuration for a DOS.C system includes 16K of memory in the 1100 or 2200 series computers and 16K or more in 5500 series computers. Use of a single 9380-series drive is possible, but the user should at least have access to a double-drive system for backup purposes. Some consideration must be given to the question of copying files from one diskette to another. Users with 2200 and 5500 series computers (having cassette tape drives) can use cassettes if necessary as a standard exchange medium for file transfers (except for files bigger than about 450 sectors, too large to fit on a single side of a cassette). Users with Diskette 1100 series computers do not have cassette tape drives and hence must use diskettes as their file transfer medium. Since DOS.C software is distributed on diskettes by Datapoint Corporation, the user will need to have at least a two drive system to copy the software from the diskette received from Datapoint to his working diskette(s). Single drive systems should be considered only by users intending to use them as satellite systems (example: data entry stations) and planning on having at least one other system with two or more drives (for program development and file processing applications).

Another option which should be strongly considered is the High Speed, or so-called "RAM" Display option. This option can substantially increase total system throughput and responsiveness (especially in applications displaying a lot of text on the screen, such as data entry) at a very small additional cost. The RAM Display is field-installable (although less expensive when factory-installed), and is standard equipment on Datapoint 5500 and Diskette 1100 series computers.

1.2 Performance of DOS.C

Users who are currently using Datapoint computers in cassette-based systems will find substantial improvements in performance when they upgrade to DOS.C. The 9380 series diskette drives are several times faster than cassettes for ordinary sequential data transfers; random-access type operations (such as

sorting and ISAM file access) can easily be two orders of magnitude faster than is attained using tape cassettes.

Users who are currently using competitive diskette-based equipment will generally find that total system performance of Datapoint systems will exceed that which they are accustomed to. This improvement is due to the generally superior data handling techniques and file structuring as used in Datapoint's DOS. These characteristics stem from the fact that instead of employing a lower-performance cassette-style file structure as a base for the operating system, Datapoint chose instead to adapt the same advanced and dynamic disk file access techniques as used in its other DOS to the new diskette media. The result is a degree of software sophistication previously unavailable in business-oriented systems of this size.

The obvious side benefit of this DOS compatibility is that not only is virtually all of the Datapoint DOS software library available to diskette users but that most user programs which were written originally for other Datapoint DOS systems will run unmodified (except for possible file size limitations and timing differences due to the slower access times of the 9380 series disk drives) under DOS.C.

In addition to the increased speed of access to the 9380 series drives as compared to cassettes, another big advantage is that the total amount of storage available on a diskette-based system is about four times the amount usable on cassette systems, even when both cassette drives are in use.

It should be recognized that is DOS.C is not expected to eliminate the usefulness of larger capacity, higher performance disks. Many users will have applications which are too involved or too large for the 9380 series diskette drives. Users who need very large data files or very high speed random access to disk storage will find the performance they are looking for in the other versions of Datapoint DOS.

CHAPTER 2. DISK DRIVES

Datapoint DOS.C supports up to four 9380-series flexible diskette drives through their integral disk controller unit. The disk controller contains 1024 bytes of high speed, random access memory which buffers four sectors between the disk drives and the Datapoint computer, enabling greater I/O device autonomy and improved overall system performance.

2.1 Disk media

The Datapoint 9380-series flexible diskette drives use a flexible diskette for data storage. This diskette is media and format-compatible with the IBM 3741-style flexible diskette.

Data is recorded in 77 concentric circles on only one side of the diskette (as per the IBM standards for diskette data interchange). Each such circle is referred to as a track. Although each such track on the diskette actually contains 26 physical records of 128 bytes each, these are paired up by the Datapoint 9380 series diskette controller (an integral part of the diskette system) so that to the Datapoint computer each track appears to consist of 13 records (called sectors) of 256 bytes each. In Datapoint DOS documentation, unless explicitly indicated to the contrary, the term sector always refers to a 256-byte logical sector, and it is strictly incidental that this sector is broken into two physical 128 byte records for transfer to and from the diskette media.

The diskette is permanently enclosed within a durable plastic cover. This cover provides for easy insertion of the diskette into the diskette drives and provides structural rigidity for the media when it is not in use. In addition, the plastic cover provides a degree of environmental protection for the diskette and its oxide surface from damage caused by careless handling.

2.2 Loading and unloading diskettes

Upon observation of a diskette, three holes through the plastic diskette cover will be noted. Each of these holes allows one to see a portion of the oxide-coated surface of the diskette itself.

The large, round hole in the center of the cover is used by the diskette drive for the hub which clamps to the diskette and

turns the diskette within the cover.

The longer, narrower radial slot towards one edge of the enclosure is the slot through which the read/write head in the diskette drive contacts the diskette's oxide coating for data transfer operations.

The smaller round hole present on the diskette is the hole through which the index hole, a hole in the diskette proper about the diameter of a pencil lead, is sensed by the controller and used for timing and control purposes.

The reason for the description of these holes is that they provide the definitive reference for indicating the proper direction of insertion of the diskette media into the 9380 series drives. When the diskette is properly inserted, the edge of the diskette with the long slot is inserted first. The smaller hole (the one through which the index hole is sensed) will be the last of the three holes in the cover to enter the drive, and it will be positioned toward the tabletop rather than downwards toward the floor.

The diskette loading slot is covered by a long, narrow handle. A rectangular pushbutton to the side of the handle is pushed to open the handle for diskette insertion and removal. When inserting the diskette, it will meet with a spring resistance after being inserted about 3/4 into the drive. Press the diskette gently into place until the spring catches and the diskette stays in place without being held in with the finger. Be careful not to push the diskette too far into the drive, as this can cause the innermost edge of the diskette's plastic cover to be wedged between some metal projections on the diskette drive which could possibly result in damage to the diskette. After the diskette is in place, pull the door/handle to the left until it latches closed. As the door is pulled closed, the hub engages the diskette, bringing it to its rated rotational speed of 360 rpm (and then online) almost immediately.

To remove a diskette, first ensure that all input/output activity on the diskette has quiesced. (This is necessary since it is possible to open the drive door, which takes the diskette offline, in the middle of a write operation; this can result in improper data being written onto the diskette). Then press the button to the left of the door/handle. The door will open and the diskette will emerge in much the same way toast pops out of a toaster. Upon removing the diskette from the drive, it should be immediately placed in its protective paper envelope to help protect the surface from abrasive contaminants and other elements

which could damage it.

2.3 Drive numbering and switches

Diskette drives are normally installed in the cabinet starting from the left. These drives are numbered 0, 1, 2, and 3 respectively from left to right. These numbers constitute the physical drive number. In the case of DOS.C, the same number is also sometimes referred to as the DOS logical drive number, or frequently just drive number.

The main power switch for the diskette unit is located on the underside of the tabletop and to the left side of the diskette drives, positioned toward the front of the cabinet. Sliding this switch towards the rear of the diskette drive cabinet turns the diskette unit on, and sliding the switch towards the user turns the diskette unit off. This switch should normally always be left in the ON position.

There are no other controls or switches intended for use by the user on the 9380-series diskette drives.

2.4 Care and handling of diskettes

Diskettes are sturdy media which will give long and trouble-free service if they are handled with reasonable care.

1) Diskettes should always be stored in their protective paper envelopes when not inserted in a drive. These envelopes should then be stored in the protective boxes in which the diskettes are received from the manufacturer.

2) Do not force too many diskettes into one box. They should not be placed under heavy pressure, as this can warp the diskette media, possibly causing read/write errors.

3) Diskettes should not be rolled, folded or otherwise subjected to strains which could crease the media.

4) Never touch the oxide coating of the diskette through the holes in the plastic cover. The skin has oils on it which will attract and retain dust and other abrasive contaminants if these oils get onto the diskette's surface. In addition, contact between hard surfaces and the diskette oxide can scrape away the information-carrying oxide from the diskette surface; this will usually result in unrecoverable errors on the diskette.

5) Diskettes should not be subjected to strong magnetic

fields.

6) When mailing diskettes, they should either be placed between two sheets of corrugated cardboard (for rigidity and protection while going through the mails) or placed in some suitable protective carrier. Many diskette media manufacturers sell mailers specifically designed for use in sending diskettes, either singly or in multiples, through the mail.

7) Diskettes can generally be taken through airport security x-ray and metal detecting equipment without danger of damage to the information recorded on the diskette.

2.5 Preparing diskettes for use

When a diskette is first received from the media manufacturer, it contains formatting information recorded across the entire usable surface of the diskette. This information is provided to allow the controller to identify where a given sector is on the surface of the disk, and also allows the controller to verify proper head positioning in the drive mechanism. Normal reading and writing on the diskette does not destroy the formatting information contained thereon.

Diskettes cannot be used by DOS.C until they have been DOSGENed. (The DOSGEN process is described in the DOS USER'S GUIDE). Since the standard IBM file structure on diskettes is intended for data entry and not for actual computer data processing, Datapoint DOS uses its own unique file structure which is capable of more sophisticated data and file manipulation. This more sophisticated file structure is what results in the need for DOSGEN before a diskette can be used by DOS.C.

A special note regarding disks which are to be used in drive zero is appropriate. Many of the newer releases of DOS commands will use DOS FUNCTIONs (as described in the DOS USER'S GUIDE) in increasing numbers. These functions are resident on the diskette in the file SYSTEM7/SYS. When updated versions of DOS.C and associated utilities are received from Datapoint Corporation, the file SYSTEM7/SYS may also have one or more new DOS FUNCTIONs resident. Therefore, wholesale copying of DOS commands from newly received diskettes to older diskettes with older versions of SYSTEM7/SYS will frequently result in commands which either work or do not work depending on whether the older or newer version of SYSTEM7/SYS is present on the disk in drive zero. Therefore the user should generally keep his DOS commands disk more or less intact and not use a newly released diskette to supply commands to previously DOSGENed diskettes; instead, he should freshly

generate as many system diskettes (including whichever DOS commands he intends to use) as he needs.

2.6 Suggested disk organization techniques

Due to the relatively small capacity of the flexible diskette, careful consideration should be given to which files should be put on which diskettes. Users with single drives for data entry and related applications will have little choice in such matters. However, for users with multi-drive systems being used for program development, the following convention is suggested:

1) DOS system diskettes. These diskettes contain the system files (as do all diskettes for use with DOS.C) and whichever DOS commands the user intends to use. Usually this diskette will be used in drive zero during program generation, debugging and other DOS system-type functions and because of this will contain all of the DOS, DOS commands and latest set of DOS FUNCTIONs as released by Datapoint. This diskette will also frequently contain the editor scratch file, SCRATCH/TXT

2) Source program diskettes. These diskettes can be considered as library file diskettes. These diskettes contain programs, Dataform forms, and other user text files used, for example, during program generation. Once these programs are finalized, they can be copied to DOS System diskettes or User System diskettes as appropriate.

3) User system diskettes. These diskettes are similar in intent to DOS System diskettes but differ in that they are intended more for specific application use rather than for general program development and debugging. These diskettes will usually be used with Databus 1100, Scribe, DOSBASIC, or Dataform and/or have large numbers of user-written application programs on them. These disks will usually not contain the more specialized DOS commands (and other files) such as DUMP9380, REPAIR, DOS/EPT, MASSACRE, APP, CHANGE, DUMP and the like.

4) Data file diskettes. These diskettes contain primarily user data files. Typical characteristics of files on this type of diskette: non-executable, user information; SCRIBE text; other things which are user-entered (or generated) but not programs as such.

5) Scratch diskettes. These diskettes are diskettes containing no important files. These diskettes are suggested for use in transferring files from one diskette to another, and to

provide diskettes containing large unallocated areas for use as scratch files by programs using scratch files (for example, SORT and EDIT commands).

As support for the above five basic types of diskettes, the following color-coding convention is suggested for diskette labels:

red - DOS System diskettes
green - User System diskettes
blue - Text files (source programs and SCRIBE text)
yellow - Data files
grey - Scratch diskettes

For best results, users should use only diskette media provided by those manufacturers recommended by Datapoint Corporation.

CHAPTER 3. THE DUMP9380 COMMAND

The DUMP9380 command enables the programmer to inspect, record, or load physical disk sectors. DUMP9380 is intended to be a working partner of the DEBUG facility in the debugging and monitoring of programs running under the DOS.

3.1 Use

DUMP9380 can be invoked from an active DOS by keying in at the system console:

```
DUMP9380
```

Since DUMP9380 is a completely self-contained program, it can be run from an LGO cassette tape (unlike most DOS commands which rely on one or more of the DOS routines for their execution). In this mode, DUMP9380 can occasionally be useful in helping to determine the problem when the DOS will not boot up from some pack. If a user intends to use DUMP9380 in this way, he should take care to make an LGO tape and store it safely away somewhere, before he needs it.

DUMP9380 can output physical disk records (sectors) to a local printer, the cassette deck, or to the screen, and can load sectors to disk from the cassette deck.

There are two command handlers in DUMP9380. The primary command handler controls all DUMP9380 functions except the screen dump. The screen dump requires its own syntax because it is an interactive, and more flexible, facility.

All commands to DUMP9380 employ the same conceptual structure, though elements of commands may be implicit as well as explicit. The full explicit format for commands is:

```
X AAA,BBB CCC,DDD
```

where
- X is the command
- AAA is the starting cylinder number
- BBB is the starting sector on that track
- CCC is the ending cylinder number
- DDD is the ending sector on that track

The Flexible Diskette operating system uses a skewed logical

sector addressing scheme. Disk addresses used throughout DUMP9380 are the logical address unless EBCDIC mode is being used.

The command codes of the primary command handler are:

P	Print on the local printer
S	Screen dump
CD	Cassette dump
CL	Cassette load
#	Jump to D.O.S. DEBUG
^	Return to D.O.S. command interpreter
A	ASCII mode (for printer or screen dump)
E	EBCDIC mode (for printer or screen dump)
O	Octal mode (for printer or screen dump)
@	Physical drive number

The command codes of the screen dump command handler are:

^	Return to the DDUMP primary command handler
#	Jump to D.O.S. DEBUG
I	Increment the (cylinder,sector) address
D	Decrement the (cylinder,sector) address
C	Cylinder address mode
S	Sector address mode
A	ASCII display mode
E	EBCDIC display mode
O	Octal display mode

The following operating instructions discuss the commands and their applications, with some examples, in more detail.

3.2 The primary command handler

As soon as DUMP9380 has fully loaded, it displays its signon message on the screen. When the cursor appears at the lower left corner of the screen the primary command handler is ready to accept commands.

3.3 Using DUMP9380 with a local printer

P - Print on the local printer

DUMP9370 will print only to a local printer, address 0303. The 256 byte disk records (sectors) are listed 32 bytes per line, 8 lines per sector. Preceding each 8 line block of print is a short line giving the logical (or physical if in EBCDIC mode) disk address of the printed sector. One sector or the entire disk may be dumped to the printer by a P command. After the last sector is

printed the page is ejected to top of the next page.

Unless otherwise specified, the bytes are printed in octal, with a space separating each byte, except every eighth byte is delimited by a period. If the DUMP9380 command is in the ASCII mode (set with the A command) characters that are valid ASCII characters will be printed in ASCII. Lower-case ASCII alphabetic characters are indicated by a preceding underscore (_). If the DUMP9380 command is in the EBCDIC mode (set with the E command) characters that are valid EBCDIC characters will be printed in EBCDIC. Lower-case EBCDIC alphabetic characters are indicated by a preceding underscore (_).

COMMAND EXAMPLES:

```
P 000,000 000,000
```

would dump to the printer the disk records from cylinder 000, sector 000, thru cylinder 000, sector 000. In other words, print only the one sector with the disk address 000,000.

Note from the following examples that the parameter fetching subroutine will make certain assumptions about information not explicitly given.

```
P 0,0 0,14
```

would dump to the printer the disk records from cylinder 000, sector 000, thru cylinder 000, sector 014. In other words, dump to the printer all of the sectors on cylinder zero. Cylinder zero is not used by DOS.C and will be in the original EBCDIC format. Note that it is not necessary to supply leading zeros in an address.

```
P 0 0
```

would do exactly the same thing as the previous example. When only the first number is given between spaces, it is taken to be a cylinder address, with a sector address of 000 assumed for the beginning cylinder and a sector address of 014 assumed for the ending cylinder address.

```
P 4
```

would dump to the printer the disk records from cylinder 004, sector 000, thru cylinder 004, sector 014. In other words, all of the sectors on cylinder 4. When only one cylinder address is given, it is taken to be both the beginning and ending cylinder

address.

```
P 48 50,7
```

would be assumed to mean: P 048,000 050,007 .

The contents of logical sector 014 is of special significance to the D.O.S. because it contains the two major sets of tables: the cluster allocation tables (CATs) and the directory. A special case of the P command will dump logical sector 014 to the printer in a format that simplifies inspection of the logical sector 014 tables, especially the directory. There are two copies of each table; the master copy of the directory is from track 021 through track 040. The backup copy of the directory is from track 01 through track 020. The master CAT is on track 041, the backup CAT is on track 042, and the lockout CAT is on track 043. The special P command will dump one track at a time to the printer.

```
P Cx
```

would mean Print the Catalog. The "x" may be 0 (print the master) or 1 (print the backup). If not given, "x" is assumed to be 0 (the master copies of the CAT and directory will be printed).

The CAT is dumped in normal octal format (even if ASCII or EBCDIC mode is in force). There are sixteen directory entries on each sector of the directory, each entry being sixteen bytes long. A directory sector is dumped to the printer with four entries per line, four lines per sector. Each directory entry is delimited by a period. The format of a printed directory entry is:

```
CCC TTP LSP MSP NNNNNNNNXXX SDN
```

where CCC (octal) is the cylinder address of the file's RIB
P (octal) is the protection of the file
TT (octal) is the cluster number of the file's RIB (left-justified; the most significant three bits)
LSP (octal) is the least significant portion and
MSP (octal) is the most significant portion of the file's old logical length limit field
NNNNNNNN (ASCII) is the 8-character name of the file
XXX (ASCII) is the 3-character extension of the file
SDN (octal) is the number of the subdirectory the file is in

Note that in the NNNNNNNN and XXX fields, any characters that are not valid ASCII characters will be printed as question marks (?). Since unused directory entries are set to all octal 377, they will be printed as:

377 377 377 377 ??????????? 377

3.4 Screen Display format

S - Screen dump

DUMP9380 can display on the CRT one disk physical record (sector) at a time, in octal, ASCII, or EBCDIC, and the address of the sector displayed is controlled in a manner analogous to the display of bytes in memory by the DOS debugging facility.

A special display format is utilized to enable all 256 bytes of a sector to be displayed on the 2200 screen at one time. Below is a diagram of what a screen dump of a sector would look like; given the CYL,SEC address = 44,6 and each byte in the example sector is its location within the sector; (i.e., starting at the beginning of the sector, the bytes are (in octal) 000, 001, 002, 003,. . ., 0377:

```
044_000001002003004005006007 010011012013014015016017 020021022023024025026027
006 030031032033034035036037 040041042043044045046047 050051052053054055056057
011 060061062063064065066067 070071072073074075076077
   _100101102103104105106107 110111112113114115116117 120121122123124125126127
    130131132133134135136137 140141142143144145146147 150151152153154155156157
    160161162163164165166167 170171172173174175176177
   _200201202203204205206207 210211212213214215216217 220221222223224225226227
    230231232233234235236237 240241242243244245246247 250251252253254255256257
    260261262263264265266267 270271272273274275276277
   _300301302303304305306307 310311312313314315316317 320321322323324325326327
    330331332333334335336337 340341342343344345346347 350351352353354355356357
    360361362363364365366367 370371372373374375376377
```

Note from the diagram that:

The cylinder (044), physical sector (006), and logical sector (011) addresses are displayed in the top left corner of the screen.

Each group of 10(octal) bytes is displayed in a contiguous block of digits.

Each block of 100(octal) bytes begins at the left side of the screen, preceded by an underscore (_).

Each block of 100(octal) bytes consists of 10(octal) groups of 10(octal) contiguous bytes; 3, 3, and 2 groups to a screen line, for the three lines required to display 100(octal) bytes.

The screen displays 400(octal) bytes, which is one disk sector, 256(decimal) bytes.

To further break down the screen and enable quick location and reading of individual bytes, the first digit of every second byte is flashed on and off. Thus, each group of eight bytes is divided

into four units of two bytes.

COMMAND EXAMPLES :

S 044,011

would mean: display cylinder 44, logical sector 11 on the screen. This command can only be given to the PRIMARY COMMAND HANDLER, and after it is executed DUMP9380 will be under the control of the SCREEN DUMP COMMAND HANDLER. Note that, under the DOS.C disk organization, the sector displayed is cylinder 44, physical sector 6, logical sector 11.

3.5 The SCREEN DUMP COMMAND HANDLER

Note that as in the DOS debugging facility, the command codes entered are not displayed, the command is merely immediately executed.

^ Return to the PRIMARY COMMAND HANDLER. The screen will be rolled up, the cursor turned on, and keyed commands will be displayed as they are entered at the lower left corner of the screen.
NOTE that the SHIFT key must be depressed at the same time as the asterisk (*) key.

Jump to the DOS debugging facility. # will not work if DUMP9370 was loaded from an LGO tape.
NOTE that the SHIFT key must be depressed at the same time as the pound sign (#) key.

I Increment the cylinder or sector address and display the sector at the new address. The new disk address will be displayed at the top left corner of the screen.

If the C (Cylinder address mode) command is in force when an I command is given, the cylinder address will be incremented by one, the head and sector addresses will not change. Cylinder address wrap-around occurs at 114->000. Incrementing by cylinder address is useful for scanning quickly thru a large file by steps of 12 sectors (four DOS.C clusters) per increment.

If the S (Sector address mode) command is in force when an I command is given, the logical (or physical if in EBCDIC mode) sector address will be incremented by one. If the sector was the last on the track (014), then the

cylinder address is incremented by one and the sector address is set to zero. If the cylinder address was 114 (last on the pack), it will be set to zero. Incrementing by sector enables scanning sector by sector thru a file and inspection of the exact data on each disk record. However, sector by sector scanning will see a system sector (logical sector 014) between the last sector of a file on one track and before the first sector of that file on the next track. Files which span logical cylinders or are non-contiguous on the disk (which includes most large files) will require more detailed understanding by the user of the DOS file structure (in order to avoid incrementing out of the file's allocated space) and are usually better examined using the DOS DUMP command as described in the DOS USER'S GUIDE.

D Decrement the cylinder or sector address and display the sector at the new address. Except for the direction of address change, the D command is functionally like the I command. Cylinder address wrap-around occurs at 000->114, and sector address wrap-around occurs at 000->014.

C Cylinder address mode. This command causes subsequent I or D commands to alter the cylinder address. Optionally, a cylinder address may be keyed in before striking the C key; the current cylinder address will be replaced by the entered value before the disk record is read and displayed. The entered digits will be displayed at the lower left corner of the screen. Note that the address must be an octal address. If more than three digits are entered DUMP9380 will BEEP and the procedure must be re-begun. If the address entered is not a valid cylinder address (i.e., greater than 0114) the C command will be in force but the cylinder address will not be changed. Also note that only the eight least significant bits of the value entered will be taken for the address (a entered value 444 would be interpreted as 044).

S Sector address mode. This command causes succeeding I or D commands to alter the sector address. Optionally, a sector address may be keyed in before striking the S key. The address option is functionally similar to the C command. Sector address mode is the assumed mode of operation when the program is started.

A ASCII display mode. This command causes the bytes to be displayed in ASCII instead of OCTAL on the screen, for all bytes that have valid ASCII bit configurations. This is useful for examining text files on disk. Note that the ASCII mode will carry over to the P (print) command of the PRIMARY COMMAND HANDLER unless changed by a subsequent O command.

E EBCDIC display mode. This command causes the bytes to be displayed in EBCDIC instead of OCTAL on the screen, for all bytes that have valid EBCDIC bit configurations. This is useful for examining the index track (track zero) on a diskette and for text files on IBM formatted diskettes.

O OCTAL display mode. This command causes the bytes to be displayed in OCTAL instead of ASCII. OCTAL mode is the assumed mode of operation when the program is started.

As in the P (Print) command, there is a special case of the screen dump that provides for easy examination of a basic DOS system table, the DIRECTORY. The directory is on tracks 021 through 040, on logical sector 014 (the directory master) and tracks 1 through 020 (the directory copy). The special command S DX will display one directory sector at a time on the screen. One sector of the directory as displayed by the S D command might look like this example:

```
021    001 103 000 000 SYSTEM0 SYS 377    002 340 000 000 CAT      CMD 377
000    002 040 000 000 KILL    CMD 377    003 000 000 000 NAME     CMD 377
       002 100 000 000 CHANGE  CMD 377    003 040 000 000 AUTO     CMD 377
000    002 140 000 000 MANUAL  CMD 377    003 100 000 000 OUT      CMD 377
       002 200 000 000 SIN     CMD 377    003 140 000 000 SOUT     CMD 377
       002 240 000 000 LGO     CMD 377    003 200 000 000 APP      CMD 377
       002 300 000 000 SAPP    CMD 377    003 240 000 000 ASM      CMD 002
       034 300 000 000 EDIT    CMD 377    134 040 364 001 MASTER   TXT 001
```

Note from the diagram that the actual cylinder and sector address is displayed at the upper left corner of the screen, and that the display format for each directory entry is the same as the format of directory entries printed by the special P C (Print the Catalog) command. The octal number below the sector address is the relative directory page address; 000->017 for the sixteen directory pages (sectors).

COMMAND EXAMPLES:

S D

is the basic format for the screen directory dump. This command can be given only to the PRIMARY COMMAND HANDLER; there is no provision for going from the SCREEN DUMP COMMAND HANDLER directly to the screen directory dump format. The command above would cause the first sector of the directory master to be displayed.

S D146

Optionally, an octal number may be given immediately following the D. It must be less than or equal to 377 and is taken to be the PHYSICAL FILE NUMBER of a file which will be on the directory page that will be displayed. This provides a quick means of looking at the directory entry for a file if its PFN is known. Note that the DOS CAT command follows each filename/extension by the file's PFN in parentheses. Note further that once a file's directory entry is accessed, one need only look at the first two bytes of the entry to see where the file physically begins on the disk. By setting the last digit of the second byte (byte 1 of the entry) to zero, the resultant two-byte physical disk address is the DOS.C physical disk address of the RIB of the file. The most significant three bits of the second byte (which is the least significant byte) are the DOS.C cluster number, and the low order five bits (always zero for the primary RIB) are the sector number on that track. Note that since DOS.C only keeps four clusters per track, these clusters can be conceptually numbered 0, 2, 4 and 6; the actual two-bit cluster number is kept in the top two bits of the three bits allotted for the cluster number.

3.6 SCREEN DIRECTORY DUMP Commands

As in the regular SCREEN DUMP COMMAND HANDLER, commands may be given to DUMP9380 while it is in the screen directory dump mode. The commands are not displayed as they are entered, since they are executed immediately. There are only four commands that can be given while in the screen directory dump mode:

^ Leave the screen dump mode. The program returns to the PRIMARY COMMAND HANDLER.

I Increment the directory page address by one. To keep the disk address pointers within the directory, the sector address is never changed from 014 and the cylinder address wraps around at 040->021.

D Decrement the directory page address by one. Wrap-around occurs at 021->040.

F Flip to the alternate copy of the directory. The contents of each directory master page should be exactly matched by the contents of its corresponding directory copy page. The F command is provided so this can be visually checked. Even single byte differences may be readily detected by easily flipping back and forth between the master and the copy page.

3.7 Cassette operations

CD - Cassette Dump
CL - Cassette Load

DUMP9380 can write to the front cassette deck the contents of specified disk sectors, and can read DUMP9380 tapes from the front deck to load specified sectors.

COMMAND EXAMPLES:

```
CD 000,000 002,014
```

would mean: dump the sectors from cylinder 000, sector 000, thru cylinder 002, sector 014 to the cassette in the front deck. In other words, dump the first three cylinders of the disk to cassette. The CD command will dump from one sector to ten cylinders, in contiguous sectors. The disk addresses given (explicitly or implicitly) must be from lesser to greater (e.g. CD 40,0 36,12 would be invalid because the second address is less than the first address). If any fault is found in the addresses given, the message:

```
PARAMETER ERROR
```

will be displayed and the machine will BEEP. If the command is correct, the message:

```
FRONT DECK SCRATCH ?
```

will be displayed. A reply of "Y" will cause the cassette dump to proceed, while a reply of "X" will cause an exit to the PRIMARY COMMAND HANDLER. When the front deck is ready, the cassette dump will rewind the tape and begin dumping the specified sectors to tape as individual 256-byte records. When all of the sectors have been written, the tape is rewound and checked sector by sector against the sectors on disk. If the tape data does not match the disk data exactly, the cassette dump will abort with the message:

```
TAPE/DISK VERIFY FAILURE
```

and exit to the PRIMARY COMMAND HANDLER. If the tape is correct, it is rewound and control is returned to the PRIMARY COMMAND HANDLER.

CD 0 1

is interpreted to mean CD 000,000 001,014.

Refer to the discussion of the P (print) command for more examples of explicit and implicit addresses in commands.

CL 0 1

means: load the disk sectors addressed 000,000 thru 001,014 from the front cassette. Not more than ten cylinders may be specified to be loaded from a cassette. The cassette load read routines expect to find records of exactly 256 bytes on the tape for at least as many records as there are sectors to be loaded. If a record that does not meet the specifications is encountered before the last sector has been loaded, the cassette load will abort with the message

BAD DUMP TAPE

and return control to the PRIMARY COMMAND HANDLER. It is not necessary that the records on the tape be written to the same disk addresses as from which they were read. Therefore, the CD and CL commands provide a means of moving sectors from place to place on one disk, or from one disk to another. It is not necessary that a CL read all of the records that may be on a cassette, only that there are at least as many records on the cassette as there are sectors to be loaded. When the specified sectors have been loaded, the tape is rewound and the tape records are re-read and matched against the loaded sectors on the disk. If the data on the tape does not match the data on the disk, the cassette load routine will abort with the message:

TAPE/DISK VERIFY FAILURE

and exit to the PRIMARY COMMAND HANDLER. If everything is correct, the cassette load routine rewinds the front tape and returns control to the PRIMARY COMMAND HANDLER.

3.8 Physical and logical drive numbers

When DUMP9380 begins execution it assumes that it is to deal with the pack in drive zero. The @ command instructs DUMP9380 to deal with the pack in the specified physical drive.

COMMAND EXAMPLE:

```
@ 1
```

would mean: succeeding commands will refer to the pack in physical drive 1. The @ 1 command will remain in force until another @ command addresses a different physical drive. Note that the address parameter for the @ command consists of one and only one digit.

DUMP9380 is (as its name implies) a very low-level, physical device oriented dump command. DUMP9380 always works by physical drives.

3.9 Error messages

Some of the error messages produced by DUMP9380 and their meanings are explained below.

PARAMETER ERROR

This occurs if an invalid command and/or disk address is given to the PRIMARY COMMAND HANDLER. Note that all disk addresses must be expressed in octal.

SO MUCH ?

This occurs if a command is given to dump more than 10 cylinders to the printer. Note that one cylinder will fill several printer pages, and ten cylinders would represent a good-sized file. Respond "N" if you really don't want the printer to print out that many pages of paper. Otherwise, "Y" will cause the printing to proceed.

CASSETTE TOO SMALL

This occurs if a command is given to dump too many cylinders (more than 10) to cassette.

TAPE/DISK VERIFY FAILURE

This occurs during the tape-against-disk check phase of a cassette dump or cassette load if the data on the tape does not match exactly the data on disk. The tape is rewound and the dump or load should be retried.

BAD DUMP TAPE

This occurs if a tape record is read that does not conform to the DUMP9380 tape record format. If it occurs during a cassette load, no data from the bad tape record is written to disk.

DISK NOT ON LINE

This message is self-explanatory.

DISK PROTECTED

This occurs if the disk is protected and a cassette load command is given. Nothing will be written to the disk as long as the READ ONLY indicator is on.

C.R.C. ERROR

This occurs if a hardware read or write error persists after three attempts to accomplish the read/write unless the read error occurs during a printer dump command (so that data on bad sectors can be hard-copy recorded and examined). If a C.R.C. error occurs during a printer dump, the machine will beep.

BEEP (Audio signal)

See C.R.C. ERROR

SECTOR NOT FOUND

This occurs if the disk controller SECTOR NOT FOUND status bit is set. This usually occurs as a result of the formatting information on a disk being incomplete or incorrect, but could also indicate a software or hardware malfunction.

CHAPTER 4. THE REPAIR COMMAND

The purpose of REPAIR is to repair a malfunctioning or non-functioning DOS pack. The performance of the DOS is directly related to the correctness of disk-resident system tables. Errors in these tables can cause DOS difficulties ranging from occasional mysterious losses of data to complete inability of the DOS to function on the pack. REPAIR finds, identifies to the operator, and attempts to correct errors in the system tables.

REPAIR, once activated by an operator, is capable of seeking errors and determining corrective measures on its own. However, there are operator interfaces which exist to give a human operator the power to monitor and control the program's progress. The program constantly displays on the screen information about what it is doing. There are several points in the program's execution which always occur which require operator response. Finally, if errors are discovered the operator will be asked if the error should be corrected on disk. Thus, the operator has control over any changes made to disk and may suppress any correction suggested by the program.

REPAIR consists of three phases; the Cluster Allocation Table and Directory check phase, the Retrieval Information Blocks check phase, and the Cluster Allocation Table regeneration phase. In general terms, the program progresses from simple error analysis to quite involved error analysis during its execution. Beginning with the cylinders-to-be-locked-out information supplied by the Lockout CAT on disk and supplemented the operator, each program phase progresses according to information developed or verified during preceding checks.

The amount of interface and systems expertise required of the operator ranges from almost zero to very much, and is directly proportional to two things: how badly the pack is damaged and whether the operator wants to try to save files with errors. If the operator merely permits REPAIR to delete every file found to be in error, the result would be guaranteed to be error-free disk-resident system tables, and the operator would not need to understand any details of the DOS. Sometimes, however, it will be easier for the operator to take notes and refer to the appropriate DOS documentation in order to save a file, rather than delete the file and then have to re-create it.

REPAIR is a completely self-contained program and does not

require a working DOS to run. REPAIR can be executed as a COMMAND from the DOS or from an LGO cassette. REPAIR carries its own copies of the standard basic DOS I/O routines (DR$, DW$, KEYIN$, DSPLY$), the DOS interrupt handler and the DOS DEBUG$ routine.

4.1 Applications of REPAIR

There are three general classes of errors that can cause a DOS to work improperly:

1. Errors in the data within a file. Example: An incorrectly written object code record in a program object file will make the program unloadable and thus unexecutable.

2. Errors in the DOS system files. Example: If one of the DOS system files were inadvertently damaged, as by being partially overwritten, then sooner or later some part of the DOS would not function properly.

3. Errors in the disk system tables. Example: The Cluster Allocation Table is overwritten.

Far and away the most commonly occuring class of error is class 3. (Incidentally, the most common error is the one given for the example: a destroyed C.A.T.). Also, class 1 and class 2 errors most often occur because of previously existing class 3 errors.

REPAIR will not find or fix class 1 or 2 errors. Once those errors have occurred the file with the error should be reloaded to disk. If the user is interested in fixing these kinds of errors he should refer to later sections in this chapter and other appropriate DOS documentation.

REPAIR can fix almost all class 3 errors, and thus can fix almost all of the problems that commonly occur with a disk pack.

4.2 When to use REPAIR

There are three times to run the REPAIR program:

1. Regular disk-pack checking. It never hurts to run REPAIR after every few hours of disk use, in order to catch errors that may be developing that haven't been noticed.

2. Unexplained strange things start happening. If you ever see the message:

FAILURE IN SYSTEM DATA

it is time to run REPAIR. If other error messages are displayed by the DOS, such as:

RECORD FORMAT ERROR

and there seems to be no reason that the error should have occured, REPAIR may find the reason. If files or records in files disappear or get scrambled, it is probably a good idea to run REPAIR to see if errors have developed in the system tables.

3. The DOS will not run at all. Many times if the DOS will not "boot" it is because 1) The CAT has been destroyed - specifically, the auto-execute PFN (the last byte in the sector) is not 000 (REPAIR will always reset the auto-execute PFN to 000 when it writes the regenerated CAT to disk); or 2) The directory (one or more sectors) has been destroyed; or 3) One or more of the RIBs for the system files have been destroyed.

4.3 Understanding REPAIR

This chapter is divided into three major sections for three levels of reference:

1. Minimal operator interface (Section 7.4).

The first major section is for users who wish to use REPAIR to make their pack work again as quickly and with as little effort as possible. To use REPAIR, one does not have to understand very much about the DOS or the structure of the data on disk.

2. Medial operator interface (Section 7.5).

The second major section is a rather comprehensive discussion of the various messages and options provided by the REPAIR program, and is for users who wish to be able to take advantage of the file-saving options available with REPAIR.

3. Maximal operator interface (Section 7.6).

The third major section discusses a varitey of things that can go wrong on a disk pack and how REPAIR can be used to deal with those problems. This section is for users who are interested in understanding the DOS system disk data structure for its own sake, with emphasis, of course, on problems that can occur.

4.3.1 Preliminary reading

At absolute minimum, anyone who wants to use the REPAIR program must understand some basic DOS concepts. The REPAIR user must have a concept of what a DOS FILE is, and should be acquainted with the use of the OPERATOR COMMANDS (entered at the DOS system console) and FILE NAMES. The user must understand the concept of FILE DELETION. The user must also know what DRIVE NUMBER means.

If possible, the REPAIR user should read and understand the DOS Advanced Programmer's Guide (Part IV of the DOS USER'S GUIDE), particularly section 3.1, Disk Structure. To use and understand REPAIR to the maximum extent, the user should understand terms such as: cylinder, sector, cluster allocation table, directory, directory page, directory MASTER and COPY, directory entry, retrieval information block, segment descriptor, and cluster.

4.4 Minimal Operator Interface

This section is for those who wish to use REPAIR to make their pack work again as quickly and with as little effort as possible. To use this section requires no knowledge of the DOS beyond the concept of files. It does require the ability to read through and understand the following step-by-step instructions.

In the most ultimately simple case, the user will not want to lock out any cylinders (a cook-book process -- you don't have to know what a cylinder is), and the REPAIR program will not find any errors. The main structure of the following example is built on such a case: however, places in the example where there may be variations are noted and where in the manual to find explanations of the variations is also noted.

4.4.1 Executing REPAIR

If REPAIR is catalogued on the disk (as REPAIR/CMD), and if the DOS is capable of loading and executing it, the fastest and easiest way to get REPAIR started is by simply keying the command at the system console:

```
REPAIR
```

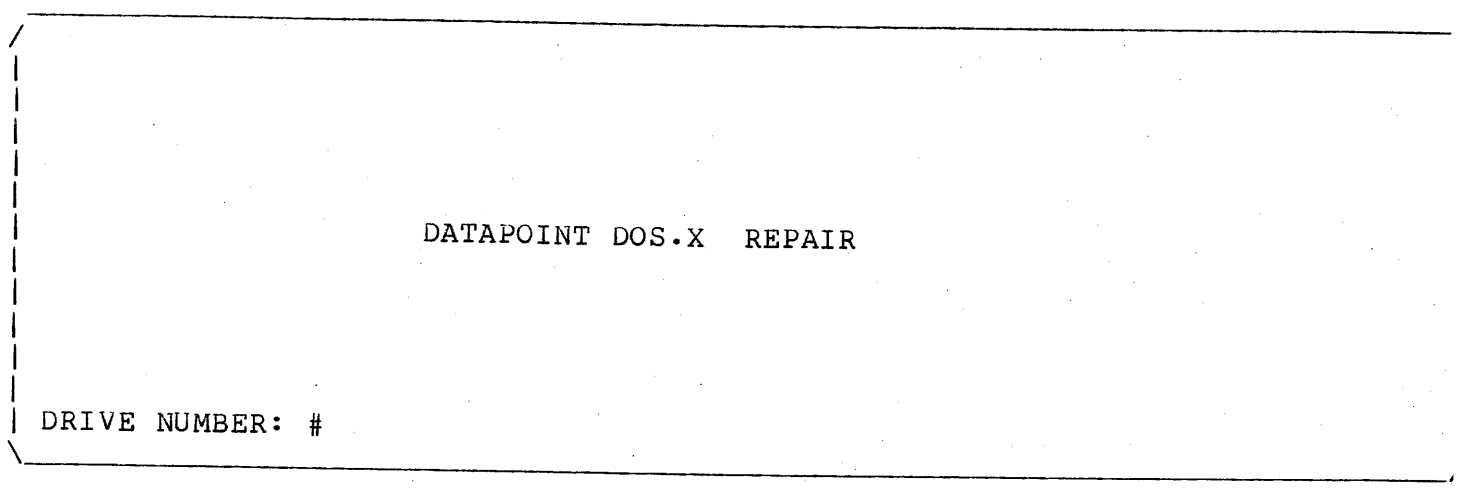
REPAIR may also be executed by placing a LOAD-AND-GO (LGO) tape of REPAIR in the back cassette deck and pressing RESTART.

In either case, the pack to be checked must be placed in a drive connected to the computer and brought on line.

In the following examples, a pictogram of the state of the CRT display will be given followed by a brief explanation and instructions for the operator.

Note that a pound sign (#) in one of the bottom two lines of the pictogram represents the cursor position. The cursor will be flashing when the operator is required to respond to the information on the screen.

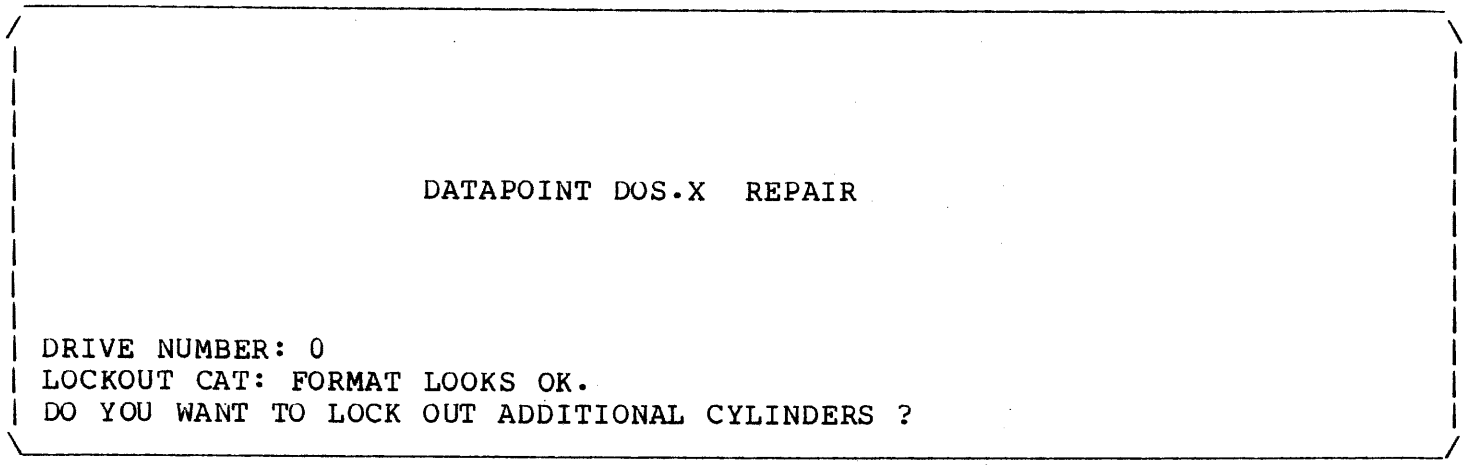
4.4.2 Sign-on and drive number specification



The screen appears as above when REPAIR has been loaded and execution has begun.

The operator must enter the logical number of the drive holding the disk pack that is to be REPAIRed.

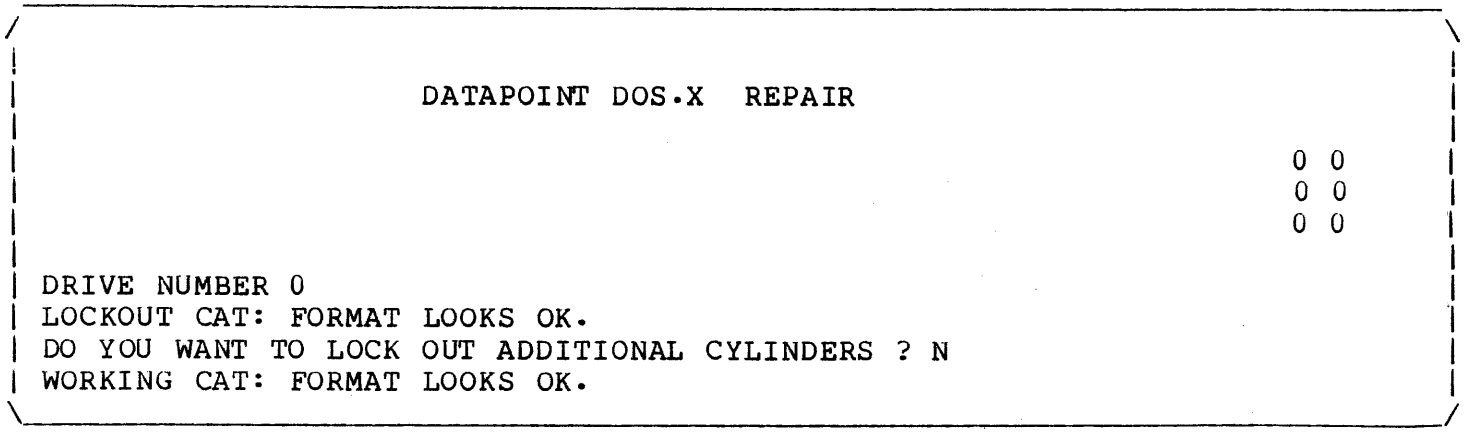
4.4.3 Cylinder Lockout



The screen may appear as above when REPAIR is ready to accept cylinder lockout. Cylinder lockout is a way of reserving disk space from DOS use. If cylinders are to be locked out, there will generally be a sticker or label on the case of the disk pack with the numbers of the cylinders to be reserved. If there are cylinders to be locked out refer to Section 4.7.

If no cylinders are to be locked out, enter "N".

4.4.4 Directory check monitor



The screen appears as above when the cylinder lockout option has not been taken and the CAT and Directory check phase has

begun. Specifically, note the vertical numbers at the right center of the screen: these numbers monitor the cycling of the directory check. If something besides these numbers appears on the screen after the cylinder lockout is completed, refer to Section 4.8.

No operator response is required.

4.4.5 Directory Errors

```
DIRECTORY ENTRY COPY: DELETE INCOMPLETE



          M A S T E R :                          C O P Y :

0 3 3 0                          3    3 3 3 3                          3    0 0
0 0 6 0                          7    7 7 7 7                          7    0 2
2 6 4 1 C A T          C M D 7   7    7 7 7 7 C A T          C M D 7   7    6 0

ENTER: 1=MASTER->COPY, 2=DELETE BOTH, 3=NO CHANGE: #
```

The screen will appear as above (in general -- specific words will vary) if REPAIR finds an error in the directory.

For explanation of the messages refer to section 4.5.5. To delete the erroneous file enter the number corresponding to DELETE BOTH (in this example, 2=DELETE BOTH).

4.4.6 Retrieval Information Blocks check

```
          RIB MASTER:              (PFN 000)        RIB COPY:
                                       *
                                       *
                                       *
                                       *
                                       *
                                       *                          0 0
                                                                  0 0
                                                                  0 0
```

The screen appears as above during the Retrieval Information Blocks check. The vertical numbers at the right of the screen monitor the cycling of the RIB check. The column of asterisks is displayed only while a RIB is actually being checked. If a pack does not have many files on it the asterisks will not appear during most of the RIB check.

No operator response is required.

4.4.7 Retrieval Information Blocks errors

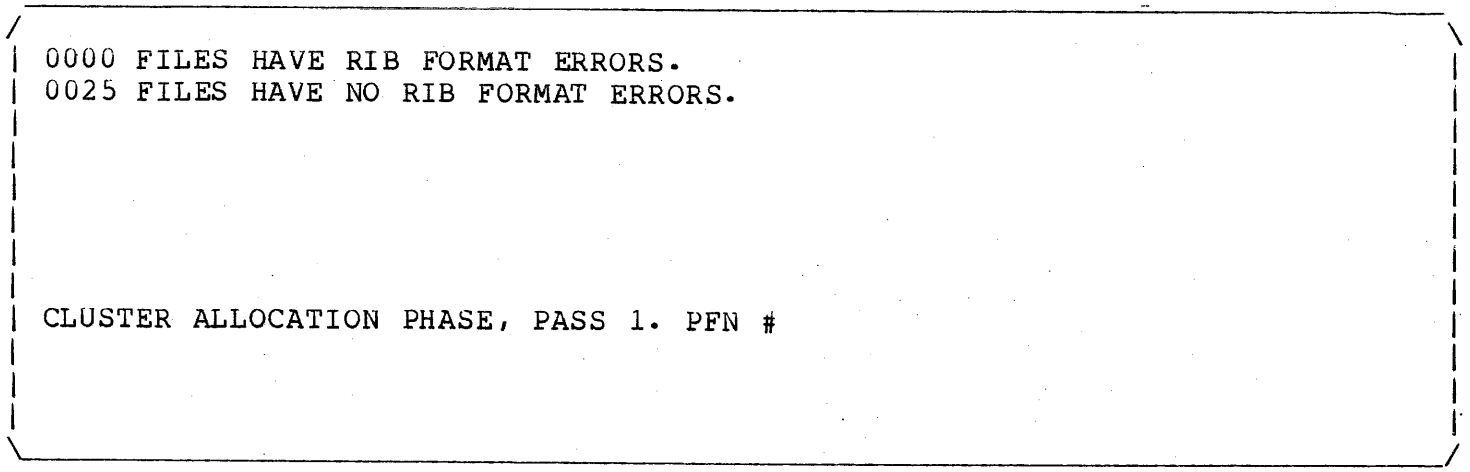
```
PFN ERROR                    LRN ERROR     * PFN ERROR                    LRN ERROR
4TH BYTE NOT 0377                          * 4TH BYTE NOT 0377
1ST SEG.DES. ERROR                         * 1ST SEG.DES. ERROR
MULTIPLE ALLOCATION 00001                  * MULTIPLE ALLOCATION 00001
CYL.ADR.OVERFLOW             CYL.ERROR     * CYL.ADR.OVERFLOW             CYL.ERROR
RIB TERMINATOR ERROR                       * RIB TERMINATOR ERROR
                      0 3 2 0                                3
                      0 0 1 2                                7
                      1 0 0 3 S Y S T E M 0   S Y S 7

DELETE THE FILE ? #
```

The screen will appear as above (in general - specific words will vary) if REPAIR finds an error in a RIB.

For explanation of the messages refer to section 4.4.7.

To delete the erroneous file enter "Y".

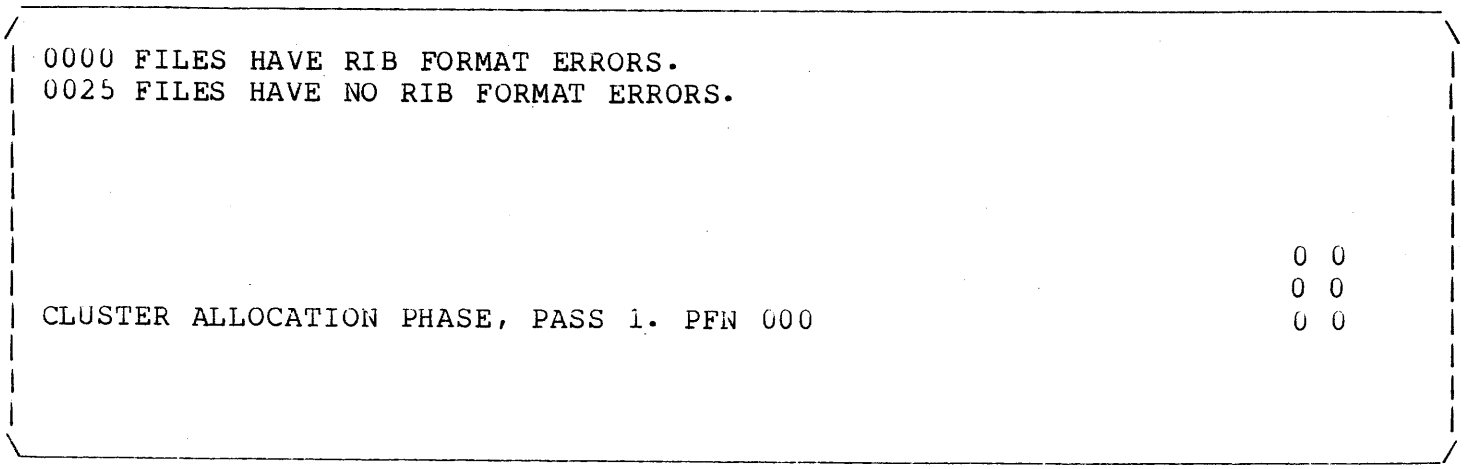
4.4.8 End of RIB check



The screen appears as above when the RIB check phase is finished. The messages at the top of the screen are a summary of the information accumulated during the RIB check phase. The message near the bottom of the screen is notification to the operator that REPAIR is ready to begin the cluster allocation phase.

To proceed, depress the <ENTER> key. If no RIB format errors remain on the disk, pressing <ENTER> is not required.

4.4.9 Cluster allocation phase, Pass 1



The screen appears as above during the first pass of the cluster allocation phase. The vertical numbers at the right of the screen are the pass cycle monitor.

No operator response is required.

4.4.10 Cluster allocation phase, Pass 2

```
 0000 FILES HAVE RIB FORMAT ERRORS.
 0025 FILES HAVE NO RIB FORMAT ERRORS.




                                                              0 0
                                                              0 0
 CLUSTER ALLOCATION PHASE, PASS 1. PFN 000                    0 0
 CLUSTER ALLOCATION PHASE, PASS 2. PFN 000
```

The screen appears as above during the cluster allocation phase, pass 2. The bottom message is displayed and the cycle monitor numbers at the right of the screen are restarted when pass 2 begins.

No operator response is required.

4.4.11 Cluster allocation phase, pass 3

```
  0000 FILES WITH ALLOCATION CONFLICTS.
 00000 CLUSTERS IN THOSE FILES.




                                                              0 0
                                                              0 0
 CLUSTER ALLOCATION PHASE, PASS 1. PFN 000                    0 0
 CLUSTER ALLOCATION PHASE, PASS 2. PFN 000
 CLUSTER ALLOCATION PHASE, PASS 3. #
```

The screen appears as above at the end of the cluster allocation phase, pass 2. The messages at the top of the screen are a summary of the information gathered during cluster allocation phase pass 1 and 2. The message at the bottom of the screen indicates that REPAIR is ready to begin the cluster allocation phase pass 3.

To proceed, depress the <ENTER> key. If no allocation conflicts are present, it is not necessary to press <ENTER>.

4.4.12 Cluster allocation conflicts

```
PFN 200                                    PFN 220
 0 0 3 0                          3         0 0 3 0                          3
 0 0 6 0                          7         0 0 6 0                          7
 3 0 4 1 S I N          C M D 7             3 6 4 1 S O U T          C M D 7
# OF CLUSTERS IN FILE: 00001               # OF CLUSTERS IN FILE: 00002
# OF CONFLICTING FILES: 002                CONFLICTING FILE # 001
# OF CONFLICTING CLUSTERS: 00001           # OF CONFLICTING CLUSTERS: 00001
# OF CORRECT PFN/LRN: 00004 OF 00006       # OF CORRECT PFN/LRN: 00000 OF 00006


ENTER: DELETE FILE: 1=LEFT, 2=RIGHT, 3=BOTH; 4=NO CHANGE: #
```

The screen will appear as above (in general, specific words will vary) if REPAIR finds that two or more files are trying to use the same space on disk.

For explanation of the messages refer to 7.5.12.

To delete the files in error enter "3".

4.4.13 Cluster Allocation Table replacement

```
 00000 CLUSTERS IN THOSE FILES.




                                                                  0 0
                                                                  0 0
 CLUSTER ALLOCATION PHASE, PASS 1. PFN 000                        0 0
 CLUSTER ALLOCATION PHASE, PASS 2. PFN 000
 CLUSTER ALLOCATION PHASE, PASS 3.

 WRITE NEW C.A.T. TO DISK ? #
```

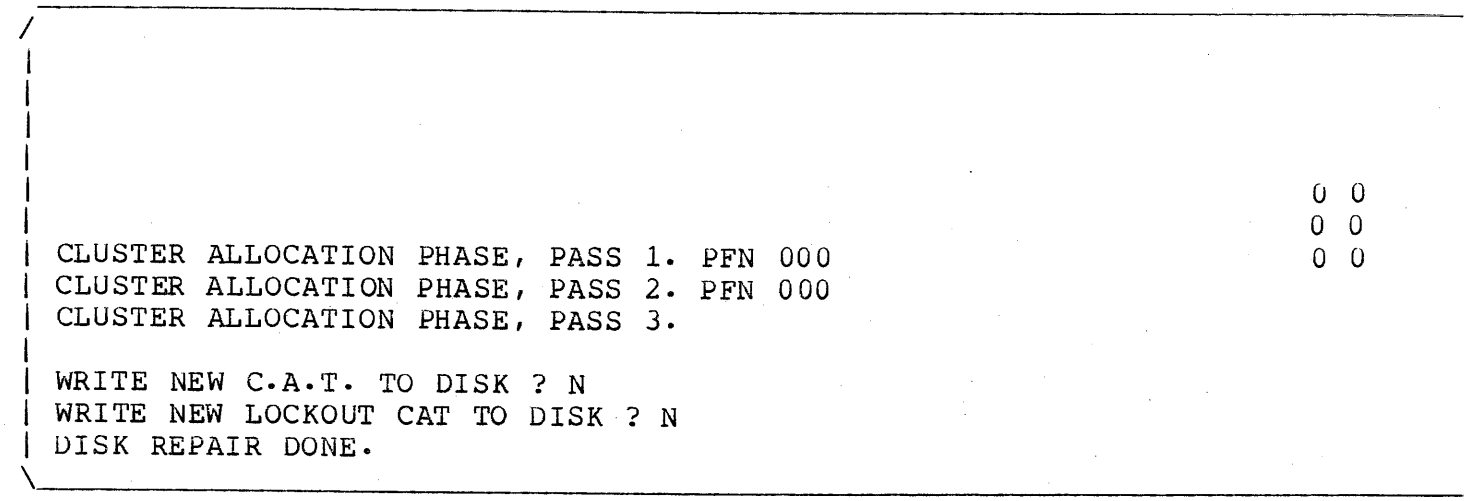
The message on the last line of the screen above will appear when REPAIR has completed all of its checks and is ready to write the regenerated CAT to disk.

To overwrite the CAT on disk enter Y. To prevent overwrite of the CAT on disk enter N. If no errors have been discovered by REPAIR, the operator should enter N.

Then a message will appear asking if the Lockout CAT is to be written back to the disk as well.

To overwrite the Lockout CAT on disk (making any additional cylinders locked out during the primary stages of REPAIR permanent) enter Y. To prevent overwrite of the Lockout CAT on disk enter N. If no errors in the Lockout CAT have been discovered by REPAIR and no additional cylinders were locked out, the operator should enter N.

4.4.14 Termination of REPAIR



The message on the last two lines of the screen above will appear when the REPAIR program is finished checking the disk. The REPAIR program does not attempt to re-load the DOS when it is finished. To get any other program running on the computer again the operator must press the RESTART key.

No operator response is required.

4.5 Medial Operator Interface

This section is a rather comprehensive discussion of the various messages and options provided by REPAIR, and is for users who wish to be able to take advantage of the file-saving options available with REPAIR. To use this section will require that the user gain an understanding of whatever error(s) REPAIR finds that the user wishes to repair. For example, if the only errors on the user's pack are in the directory, it is not necessary that the user study to understand about Retrieval Information Blocks or Cluster Allocation.

This section follows the section numbering scheme of the previous section, Minimal Operator Interface.

When a facet of REPAIR operation is discussed more appropriately elsewhere, the discussion is not repeated in this section, but the user is referred to the section containing the discussion.

Discussions in this section will occasionally refer to pictograms of the state of the display in other sections.

4.5.1 Executing REPAIR

If REPAIR is cataloged on the disk (as REPAIR/CMD), and if the DOS is capable of loading and executing it, the fastest and easiest way to get REPAIR started is by simply keying the command at the system console:

REPAIR

REPAIR may also be executed by placing a LOAD-AND-GO (LGO) tape of REPAIR in the rear cassette deck and pressing RESTART.

In either case, the pack to be checked must be placed in a drive connected to the computer and brought on line.

Note that a pound sign (#) in one of the bottom two lines of the pictogram represents the cursor position. The cursor will be flashing when the operator is required to respond to the information on the screen.

4.5.2 Sign-on and drive number specification

Refer to the pictogram in 4.4.2.

After the operator has entered the number of the drive holding the pack to be REPAIRed, REPAIR will wait for that drive to come ready before proceding to do cylinder lockout.

4.5.3 Cylinder lockout

Section 4.7 is a discussion with examples of the cylinder lockout process.

Cylinders are locked out because they give read/write errors or because by system design they are to be reserved for some special use.

4.5.4 Directory check monitor

Refer to the pictogram in 4.4.4.

The directory check monitor is the means by which REPAIR indicates its progress to the operator. Specifically, the directory check monitor constantly displays the disk address of the current directory entry being checked. This display is in

the form of two vertically displayed octal numbers at the right of the screen. The first number is a directory sector number indicator, and the second number is the buffer page address of the directory entry being checked.

If the directory check monitor stops and no other messages are displayed, then the REPAIR program was loaded to memory improperly or something is wrong with the hardware.

4.5.5 Directory errors

The directory is a table of entries for files on the pack. Their are two copies of the directory, the MASTER and the COPY. There are 16 pages to each copy of the directory, each page holds entries for up to 16 files. (One disk physical sector is one directory logical page). Thus, the directory has a MASTER and a COPY entry for up to 256 files.

The REPAIR program checks the directory one file at a time. That is, the MASTER and the COPY of a directory entry are checked at the same time.

If an error in the MASTER or the COPY entry or both is detected, REPAIR will display:

1. A brief error description at the top of the screen,

2. The MASTER and COPY entry across the lower center of the screen,

3. An option message near the bottom of the screen.

The error description will indicate whether the error is in the MASTER or the COPY entry or both, and will define the type of error.

Note that although directory entries for a file may have several types of errors at the same time, REPAIR will only deal with one error type at a time.

The directory entries are displayed under their respective headings; MASTER: and COPY: . The first four bytes and the last byte of each entry are always displayed in vertical octal. The 5th thru 15th bytes (being the file name and extension) of each entry are displayed in ASCII except for bytes in those fields which cannot be displayed in ASCII on the CRT display; those bytes will be converted to vertical octal.

The option message at the bottom of the screen will enable the operator, by selecting and entering a digit, to correct the MASTER entry with information from the COPY entry, to correct the COPY entry with information from the MASTER entry, to delete both entries (and thus the file), or to make specific changes to one or both entries, or to make no change at all to either entry.

Below are examples of the various directory errors that may occur and discussions of the respective messages. The first example is the most complete; the other directory error routines work basically the same way but their examples are not as expanded.

Note that for the examples concerning the directory MASTER, the same messages (transposing the words COPY and MASTER) apply to the directory COPY.

4.5.5.1 Delete errors

Delete errors include those where the directory entry master is deleted and the copy is not deleted, or the directory entry master is partially deleted.

4.5.5.1.1 One entry deleted

```
DIRECTORY ENTRY MASTER: DELETED


          M A S T E R :                          C O P Y :

3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3    0 3 3 0                              3    0 0
7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7    0 0 6 0                              7    0 2
7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7    2 0 4 1 C A T              C M D 7    6 0

ENTER: 1=COPY->MASTER, 2=DELETE BOTH, 3=NO CHANGE: #
```

The screen will appear as above if REPAIR finds a file for which the directory MASTER entry is deleted (filled with 0377's) but the directory COPY is not.

The operator has three options:

1. Enter "1" to copy the COPY entry to the MASTER entry,

thus saving the file's name in the directory;

2. Enter "2" to delete both entries, and thus the file;

3. Enter "3" to take no action on the file's entries and continue the directory check.

```
DIRECTORY ENTRY MASTER: DELETED



          M A S T E R :                          C O P Y :

3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3    0 3 3 0                              3     0 0
7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7    0 0 6 0                              7     0 2
7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7    2 0 4 1 C A T              C M D 7     6 0

ENTER: 1=COPY->MASTER, 2=DELETE BOTH, 3=NO CHANGE: 1
*** ARE YOU SURE ? *** #
```

The message on the last line of the screen above will appear when the operator has selected and entered one of the digits given in the option message. REPAIR will always make sure the operator entered what he intended to before proceeding to carry out the funciton.

To carry out the function selected enter "Y". If "N" is entered the option message will be re-displayed.

```



          M A S T E R :                          C O P Y :

0 3 3 0                            3    0 3 3 0                              3
0 0 6 0                            7    0 0 6 0                              7     0 0
2 0 4 1 C A T              C M D 7    2 0 4 1 C A T              C M D 7     0 2
                                                                             6 0
ENTER: 1=COPY->MASTER, 2=DELETE BOTH, 3=NO CHANGE: 1
*** ARE YOU SURE ? *** Y
DONE.
```

The screen will appear as above if the operator has replied "1" to the message above and replied "Y" to the message "*** ARE YOU SURE ? ***". When REPAIR does an entry to entry copy, both entries are re-displayed to show the operator the results of the copy, and the message: "DONE." is displayed at the bottom of the screen.

No further operator response is required.

```
DIRECTORY ENTRY MASTER: DELETED


          M A S T E R :                          C O P Y :

3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3   3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7   7 7 7 7 7 7 7 7 7 7 7 7 7 7 7   0 0
7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7   7 7 7 7 7 7 7 7 7 7 7 7 7 7 7   0 2
                                                                  6 0
ENTER: 1=COPY->MASTER, 2=DELETE BOTH, 3=NO CHANGE: 2
*** ARE YOU SURE ? *** Y
DONE.
```

The screen will appear as above if the operator has replied "2" to the message above and replied "Y" to the message "*** ARE YOU SURE ? ***". When REPAIR deletes the entries from the directory, the entries are re-displayed to show the operator that the delete has been accomplished, and the message: "DONE." is displayed at the bottom of the screen.

No further operator response is required.

```
          M A S T E R :                          C O P Y :

3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3    0 3 3 0                          3   0 0
7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7    0 0 6 0                          7   0 2
7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7    2 0 4 1 C A T          C M D 7   6 0

ENTER: 1=COPY->MASTER, 2=DELETE BOTH, 3=NO CHANGE: 3
*** ARE YOU SURE ? *** Y
```

The screen will appear as above if the operator has replied "3" to the message above and replied "Y" to the message "*** ARE YOU SURE ? ***". REPAIR will make no change to the entries and will resume the directory check.

No further operator response is required.

4.5.5.1.2 Delete incomplete

```
DIRECTORY ENTRY MASTER: DELETE INCOMPLETE

          M A S T E R :                          C O P Y :

3 3 3 3 3 3                    3    0 3 3 0                          3   0 0
7 7 7 7 7 7                    7    0 0 6 0                          7   0 2
7 7 7 7 7 7 T          C M D 7      2 0 4 1 C A T          C M D 7   6 0

ENTER: 1=COPY->MASTER, 2=DELETE BOTH, 3=NO CHANGE: #
```

The screen will appear as above if REPAIR finds a file for which the directory MASTER entry is partially deleted (partially filled with 0377's) but the directory COPY is not.

The operator options and REPAIR actions are the same as for <u>One entry deleted</u>, see 4.5.5.1.1 above.

4.5.5.2 RIB Address errors

RIB Address errors include invalid RIB addresses or unequal.

4.5.5.2.1 RIB Address Invalid

```
DIRECTORY ENTRY MASTER: R.I.B. ADDRESS INVALID


          M A S T E R :                          C O P Y :

0 3 3 0                          3     0 3 3 0                          3     0 0
0 0 6 0                          7     0 0 6 0                          7     0 2
0 0 4 1 C A T          C M D 7   2     2 0 4 1 C A T          C M D 7   6     0 0

ENTER: 1=COPY->MASTER, 2=DELETE BOTH, 3=NO CHANGE: #
```

The screen will appear as above if REPAIR finds a directory MASTER entry with an invalid RIB address.

In this example, the RIB address of the directory MASTER (the area in the box) is invalid because the cylinder address is 000.

The RIB address is the first byte and the top two digits of the second byte of a directory entry. The first byte is the cylinder address and to be valid must be an octal number in the range 001 thru 0114 inclusive. The top two digits of the second byte define the cluster number and to be valid must be one of the following:

00, 10, 20, 30

The operator has three options:

1. Enter "1" to copy the COPY entry RIB address to the MASTER entry RIB address;

2. Enter "2" to delete both entries, and thus the file;

3. Enter "3" to take no action on the file's entries and

resume the directory check.

```
DIRECTORY ENTRY MASTER: R.I.B. ADDRESS INVALID


          M A S T E R :                          C O P Y :

0 3 3 0                          3    0 3 3 0                          3    0 0
0 0 6 0                          7    0 0 6 0                          7    0 2
0 3 4 1 C A T          C M D 7   2    2 3 4 1 C A T          C M D 7   6    0

ENTER: 1=COPY->MASTER, 2=DELETE BOTH, 3=NO CHANGE: 1
*** ARE YOU SURE ? *** #
```

The message on the last line of the screen above will appear if the operator has replied "1" to the message above.

To carry out the function selected enter "Y".

```
DIRECTORY ENTRY MASTER: R.I.B. ADDRESS INVALID


          M A S T E R :                          C O P Y :

0 3 3 0                          3    0 3 3 0                          3    0 0
0 0 6 0                          7    0 0 6 0                          7    0 2
0 3 4 1 C A T          C M D 7   2    2 3 4 1 C A T          C M D 7   6    0

ENTER: 1=COPY->MASTER, 2=DELETE BOTH, 3=NO CHANGE: 1
MOVE ENTIRE ENTRY ? #
```

The message on the last line of the screen above will appear if the operator replied "Y" to the message "*** ARE YOU SURE ? ***".

Enter "N" to have REPAIR copy the R.I.B. address (only) from the COPY entry to the MASTER entry. Enter "Y" to have REPAIR copy the entire COPY entry to the MASTER entry.

The "MOVE ENTIRE ENTRY ? " option is given to give the operator the ability to correct many types of errors in an

errornous entry at one time, rather than correct each error as it is found. If the operator can recognize a severly destroyed entry the first time he sees it, this option can enable him to repair the directory more quickly.

4.5.5.2.2 RIB Addresses not equal

```
DIRECTORY ENTRY MASTER & COPY: R.I.B. ADDRESSES NOT EQUAL


          M A S T E R :                          C O P Y :

0 3 3 0                          3    0 3 3 0                          3    0 0
0 0 6 0                          7    0 0 6 0                          7    0 2
2 3 4 1 C A T          C M D 7   3    3 3 4 1 C A T          C M D 7   6    0

ENTER: 1=MASTER->COPY, 2=COPY->MASTER, 3=DELETE BOTH, 4=NO CHANGE: #
```

The screen will appear as above if REPAIR finds a file with directory entries with RIB address that are both valid but not equal.

In this example, the RIB address in the MASTER is 002,300 and in the COPY is 003,300.

The operator has four options:

1. Enter "1" to copy the MASTER entry RIB address to the COPY entry RIB address;

2. Enter "2" to copy the copy entry RIB address to the master entry RIB address;

3. Enter "3" to delete both entries, and thus the file;

4. Enter "4" to take no action on the file's entries and resume the directory check.

If it is not obvious by visual inspection of the directory entries which is in error, the operator should note the RIB address as given by each directory entry, and enter "4". If REPAIR later discovers PFN and LRN errors in the actual RIBs for the file (see 4.5.7), then the operator can be fairly sure the

directory MASTER entry for the file is in error, since only the directory MASTER entry is used to determine the RIB address of a file for the RIB check phase of REPAIR.

If the operator wants to make very sure which, if either, of the directory entries is correct, he can use the DUMP or DUMP9370 commands to look at the file after REPAIR has finished execution.

When it is determined which directory entry for the file has the correct RIB address, the operator can execute REPAIR again, this time entering "1" or "2" as appropriate to correct the erroneous directory entry.

If neither entry is correct, and it would be easier to modify the directory entries for the file than to delete them and re-create the file, refer to the available DOS documentation (both in the DOS SYSTEM GUIDE and the DOS USER'S GUIDE) for details on ways to modify the directory sectors on disk.

4.5.5.3 File protection not same

```
DIRECTORY ENTRY MASTER & COPY: FILE PROTECTION NOT SAME

ENTRY MASTER: WRITE PROTECTION
  ENTRY COPY: NO PROTECTION
          M A S T E R :                          C O P Y :

0 3 3 0                        3    0 3 3 0                        3    0 0
0 0 6 0                        7    0 0 6 0                        7    0 2
2 3 4 1 C A T          C M D 7    2 0 4 1 C A T          C M D 7    6 0

ENTER:1=DELETE ENTRIES,2=NO CHANGE;PROTECTION: 3=NONE,4=DELETE,5=WRITE: #
```

The screen will appear as above if REPAIR finds a file with directory entries with protection not the same.

In this example, the directory MASTER entry has WRITE protection indicated for the file, while the directory COPY entry has no protection indicated for the file. Note: where the bits for both WRITE and DELETE protection are set, WRITE protection has precedence, since WRITE protection implies DELETE protection.

The protection indication is in the bottom two bits (bottom digit) of the second byte of a directory entry. If the upper bit

of the two is set on (the digit is 2) then the directory entry indicates that the file is DELETE protected. If the bottom bit is set on (the digit is 1 or 3) then the directory entry indicates that the file is WRITE protected. If neither of the two bits is set on (the digit is 0) then the directory entry indicates NO protection for the file, that is, that the file is unprotected.

The operator has five options:

1. Enter "1" to delete both entries, and thus the file;

2. Enter "2" to take no action on the file's entries and resume the directory check;

3. Enter "3" to set both entries to indicate NO protection;

4. Enter "4" to set both entries to indicate DELETE protection;

5. Enter "5" to set both entries to indicate WRITE protection.

4.5.5.4 Name-Extension not equal

```
DIRECTORY ENTRY MASTER & COPY: NAME-EXTENSION NOT EQUAL


          M A S T E R :                          C O P Y :

0 3 3 0                          3    0 3 3 0                          3    0 0
0 0 6 0                          7    0 0 6 0                          7    0 2
2 0 4 1 C A T X X X X X C M D 7       2 0 4 1 C A T          C M D 7    6 0

ENTER: 1=MASTER->COPY, 2=COPY->MASTER, 3=DELETE BOTH, 4=NO CHANGE: #
```

The screen will appear as above if REPAIR finds a file with directory entries that do not have the same NAME/EXTENSION.

The NAME/EXTENSION of a directory entry is located in bytes 5 through 15 inclusively. The NAME/EXTENSION of a directory entry (and the file) is the normal means by which the file is identified and manipulated, especially from the DOS system console.

Note that REPAIR does not seek or identify as erroneous files with NAME/EXTENSIONS that contain non-ASCII characters, since by DOS rules non-ASCII characters are perfectly legal in the NAME/EXTENSION field.

The operator has four options:

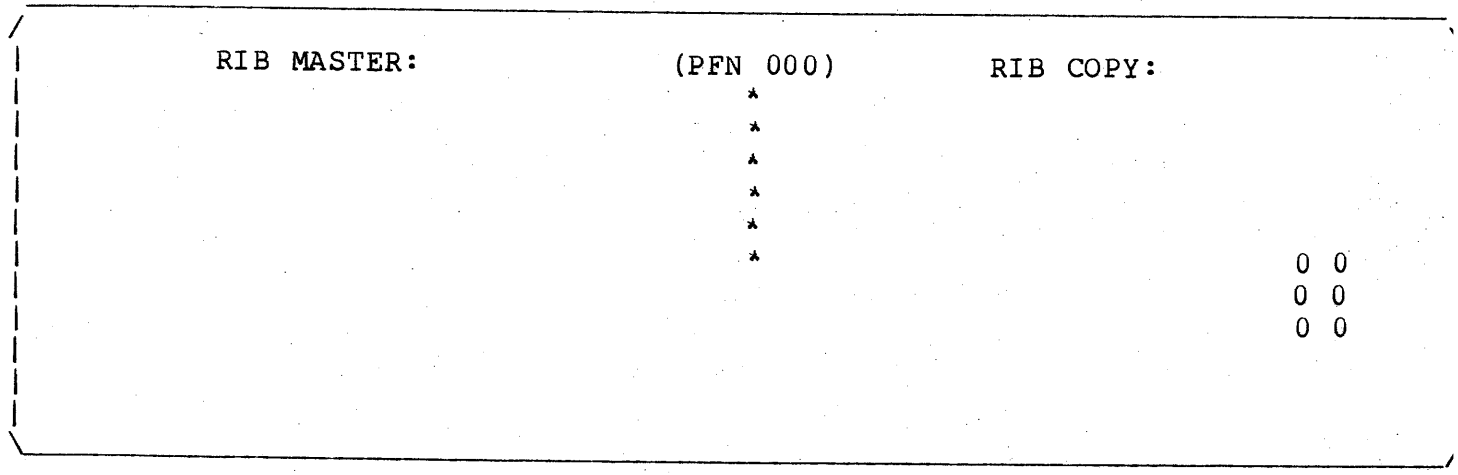
1. Enter "1" to copy the MASTER entry NAME/EXTENSION to the COPY entry NAME/EXTENSION;

2. Enter "2" to copy the COPY entry NAME/EXTENSION to the MASTER entry NAME/EXTENSION;

3. Enter "3" to delete both entries, and thus the file;

4. Enter "4" to take no action on the file's entries and resume the directory check.

4.5.6 Retrieval Information Blocks check



The screen appears as above during the Retrieval Information Blocks check.

REPAIR checks Retrieval Information Blocks (RIBs) for all files in the directory with a valid RIB address, in the order of the files' occurrence in the directory. The three-digit octal number after "PFN" in the top line will indicate the actual Physical File Number currently being checked. The two vertically-displayed octal numbers at the right of the screen provide the same information as the directory check monitor described in 4.5.4.

There are two RIBs for each file, a MASTER and a COPY. The RIB MASTER is the very first record in the file and the RIB COPY is the second record in the file. Each RIB uses one full 256-byte disk sector. Refer to the appropriate DOS manuals for a description of the structure of the RIBs.

If REPAIR detects any errors in the RIB MASTER a message describing the class of error will be displayed in the portion of the screen under the heading "RIB MASTER:". If REPAIR detects any errors in the RIB COPY a message will be displayed in the portion of the screen under the heading "RIB COPY:".

The PFN indicator and the cycle monitor numbers are incremented and displayed for each entry in the directory. The column of asterisks is displayed only while the RIBs for a file are actually being checked.

4.5.7 Retrieval Information Blocks errors

```
        RIB MASTER:                    (PFN 000)          RIB COPY:
PFN ERROR                    LRN ERROR     * PFN ERROR                    LRN ERROR
4TH BYTE NOT 0377                          * 4TH BYTE NOT 0377
1ST SEG.DES. ERROR                         * 1ST SEG.DES. ERROR
MULTIPLE ALLOCATION 00001                  * MULTIPLE ALLOCATION 00001
CYL.ADR.OVERFLOW             CYL.ERROR     * CYL.ADR.OVERFLOW             CYL.ERROR
RIB TERMINATOR ERROR                       * RIB TERMINATOR ERROR
```

The screen will appear as above if REPAIR finds errors in the RIB MASTER or COPY for a file. Note that all of the messages given in the example above will not necessarily appear. The pictogram above shows the screen as it would appear while the RIB check was in progress. The next pictogram shows the state of the screen when the RIB check has finished and has displayed the file's directory MASTER entry and is ready for operator response.

Below is a discussion of each of the messages in the screen above. In the above pictogram all possible messages are shown in their respective positions for both the RIB MASTER and the RIB COPY. Note that since the RIB MASTER and COPY have the same formats, (indeed, normally they are exact duplicates of each

other, except for their Logical Record Number [LRN]) they can have the same errors.

There are two types of errors that a RIB may have: simple and complex. If REPAIR finds only one simple error in only one of the RIBs then the operator will be given the option of having REPAIR correct the error. If multiple simple errors or any complex errors are detected then the errors are too serious for REPAIR to cope with, and will only give the operator a choice between deleting the file or making no change at all. Even with multiple or complex errors the file may be saveable though, refer to 7.6.7.

PFN ERROR

This message is displayed if the first byte of the RIB is not the file's Physical File Number (PFN). This is a simple error and is correctable under the conditions given above.

LRN ERROR

This message is displayed for the RIB MASTER if the Logical Record Number (LRN) is not zero, and for the RIB COPY if the LRN is not one. This is a simple error and is correctable under the conditions given above.

4TH BYTE NOT 0377

This message is displayed if the 4th byte of the RIB is not 0377. When the DOS object code loader loads a program into memory it skips over disk records with a 0377 in the 4th byte: since the RIBs of a file are not part of the object code of a file their fourth byte should always be 0377 so the loader will not attempt to load them to memory. This is a simple error and is correctable under the conditions given above.

1ST SEG.DES. ERROR

Expanded: First Segment Descriptor Error. This message is displayed if the first segment descriptor of the RIB does not point to itself. Since the RIBs are the first two records in any file, they will always be in the first cluster. The first segment descriptor must point to the beginning of the file, which is the cluster where the RIBs are.

MULTIPLE ALLOCATION 00001

This message is displayed if REPAIR discovers that, according to the RIB's SEGMENT DESCRIPTORS, two or more SEGMENTS of the file overlap. Specifically, SEGMENT DESCRIPTORS identify CLUSTERS on the disk which belong to the given file. If one or more of these CLUSTERS is indicated as belonging to more than one SEGMENT, then there is MULTIPLE ALLOCATION of CLUSTERS. The five digit octal number indicates how many CLUSTERS are multiply allocated.

CYL.ADR.OVERFLOW

This message is displayed if REPAIR discovers a SEGMENT DESCRIPTOR which indicates that a SEGMENT overruns the physical end of the disk. Of course it is not actually possible for a file to extend beyond the upper limit of the disk space, but it is possible for a SEGMENT DESCRIPTOR to erroneously indicate this. For example, a SEGMENT DESCRIPTOR might say, in effect: "This SEGMENT begins at the last CLUSTER on the disk and extends for ten CLUSTERS".

CYL.ERROR

This message is displayed if REPAIR discovers a SEGMENT DESCRIPTOR with a cylinder address that is either 0 (always reserved) or greater than decimal 76 (the last cylinder on a disk).

RIB TERMINATOR ERROR

This message is displayed if REPAIR discovers a RIB that has an incorrect terminator. The logical end of a RIB is indicated by either the actual physical end of the disk record or a pair of 0377's. An 0377 in the first byte of a SEGMENT DESCRIPTOR but a non-0377 in the second byte defines a TERMINATOR ERROR.

```
PFN ERROR                              *
                                       *
                                       *
                                       *
                                       *
                                       *
                    0 2 3 0                              3
                    0 0 6 0                              7
                    2 0 4 1 C H A N G E      C M D 7


DELETE THE FILE ? #
```

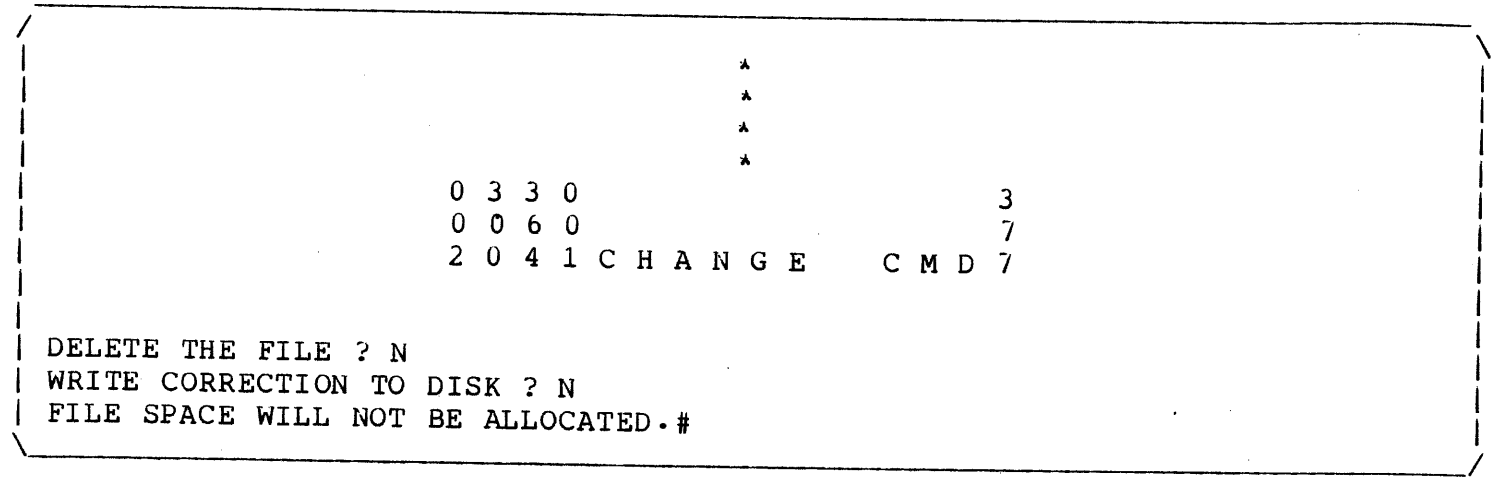
The screen will appear as above when REPAIR has completed the RIB check for a file whose RIBs had only one simple error. Note that the screen is rolled up one line so that the heading containing the PFN is no longer displayed. However the directory MASTER entry for the file, containg the NAME/EXTENSION for the file, is displayed under the error message area.

To simply have REPAIR delete the file enter "Y". To attempt to save the file enter "N".

```
                                       *
                                       *
                                       *
                                       *
                                       *
                    0 2 3 0                              3
                    0 0 6 0                              7
                    2 0 4 1 C H A N G E      C M D 7


DELETE THE FILE ? N
WRITE CORRECTION TO DISK ? #
```

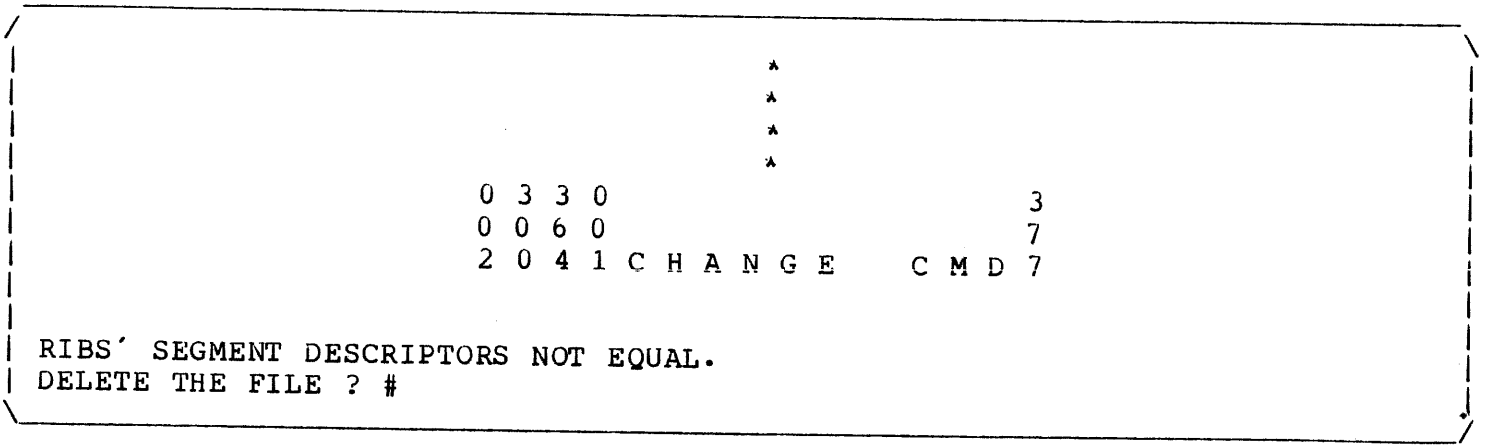
The message on the last line of the screen above will appear if the operator replied "N" to the message above.

Enter "Y" to have REPAIR write the correct RIB information to the RIB in error and resume the RIB check.



The message on the last line of the screen above will appear if the operator replied "N" to the message above. REPAIR will not allocate space (by setting the appropriate bits in the CAT) for a file if there is any uncorrectable error in either of the RIBs.

REPAIR will wait until the operator depresses the ENTER key before resuming the RIB check.



The messages on the last two lines of the screen above will appear if the RIB MASTER and COPY for a file individually have no format errors but do not describe the same segments for the file.

Enter "Y" to delete the file from the directory. Enter "N" to make no change to the file and resume the RIB check.

NOTE that whether or not the file is deleted REPAIR will not

allocate any space on disk for the file (refer to the pictogram and discussion above). The consequence of this is that, although the file will still be accessable, the space it occupies in marked as available for allocation to some other file. As a result, the remains of the file on disk are almost certain to be overwritten by some other file sooner or later.

4.5.8 End of RIB check

Refer to the pictogram in 4.4.8.

When all of the RIBs for all of the files on the disk have been checked REPAIR will count the number of files with uncorrected RIB format errors and the number of files with no RIB format errors and will display the counts on the screen. The files that do have RIB format errors will not be allocated space on disk and will not be processed in the CLUSTER ALLOCATION PHASE (below).

4.5.9 Cluster allocation phase, Pass 1

Refer to the pictogram in 4.4.9.

The CLUSTER ALLOCATION PHASE of REPAIR re-constructs in memory the CAT from the information in the RIBs of files that have no RIB format errors. The CLUSTER ALLOCATION PHASE consists of three passes. The first pass makes one pass through all the files on the pack with no RIB format errors and builds in memory two CATs: one for all files that have no space (cluster allocation) conflicts with other files and a second for files which do have cluster allocation conflicts.

4.5.10 Cluster allocation phase, Pass 2

Refer to the pictogram in 4.4.10.

The second pass of the CLUSTER ALLOCATION PHASE makes another pass thru all the files allocated to the first CAT and finds any that may have conflicts with space allocated to the second CAT, and removes those files' space allocation from the first CAT and allocates their space to the second CAT.

4.5.11 Cluster allocation phase, Pass 3

Refer to the pictogram in 4.4.11

The third pass of the CLUSTER ALLOCATION PHASE does an analysis on any two files with disk space (cluster allocation)

conflicts with each other and displays the results of the analysis on the screen (see 4.5.12 below). If no files have cluster allocation conflicts REPAIR displays the CLUSTER ALLOCATION TABLE REPLACEMENT message (see 4.5.13 below).

4.5.12 Cluster allocation conflicts

```
PFN 200                                    PFN 220
 0 0 3 0                          3         0 0 3 0                          3
 0 0 6 0                          7         0 0 6 0                          7
 3 0 4 1 S I N          C M D 7             3 0 4 1 S O U T        C M D 7
# OF CLUSTERS IN FILE: 00001                # OF CLUSTERS IN FILE: 00002
# OF CONFLICTING FILES: 002                 CONFLICTING FILE # 001
# OF CONFLICTING CLUSTERS: 00001            # OF CONFLICTING CLUSTERS: 00001
# OF CORRECT PFN/LRN: 00004 OF 00006        # OF CORRECT PFN/LRN: 00000 OF 00006


ENTER: DELETE FILE: 1=LEFT, 2=RIGHT, 3=BOTH; 4=NO CHANGE: #
```

The screen appears as above (in general, specific words will vary) if REPAIR finds two files with cluster allocation conflicts - that is, if two files have, according to their respective RIBs, been allocated in whole or in part the same clusters on disk.

The possible combinations of file cluster allocation conflicts is myriad. One file may have conflicts with only one other file. One file may have conflicts with many other files. Many files may have conflicts with many files in different combinations of numbers.

REPAIR handles any possible combination of files with cluster allocation conflicts by dealing with only two files at a time. As in the above example, the directory MASTER entry (and some additional information) for a file is displayed on the left of the screen, and the directory MASTER entry (and some additional information) for a file that has cluster allocation conflicts with it is displayed on the right of the screen.

As long as the file on the left of the screen is not deleted, all of the files that have cluster allocation conflicts with it will be displayed in turn on the right of the screen. When all of the cluster allocation conflicts with the file on the left of the screen have been dealt with, then the next file with cluster allocation conflicts will be displayed on the left of the

screen, and all files that have cluster allocation conflicts with it will displayed in turn on the right of the screen, and so on until all files that have cluster allocation conflicts have been dealt with.

The information displayed for the two files having cluster allocation conflicts is to guide the operator in deciding among the four options given by REPAIR. For explanation of the messages see 4.5.12.1 following.

REPAIR corrects a cluster allocation conflict by deleting one of the files involved. If many files are involved in cluster allocation conflicts then the operator will probably want to enter "4" after each display so that he can accumulate the information necessary to decide which files should be deleted and which should be retained (that is, REPAIR will be executed twice, once to gather all the information about cluster allocation conflicts and once to actually delete files).

Specifically, the operator has four options:

1. Enter "1" to delete the file indicated on the left of the screen;

2. Enter "2" to delete the file indicated on the right of the screen;

3. Enter "3" to delete both of the files;

4. Enter "4" to take no action on either of the files and resume the CLUSTER ALLOCATION PHASE, PASS 3.

4.5.12.1 Cluster allocation phase, Pass 3 Messages

The explanations below describe the information given in the respective messages and how the operator can interpret the information.

4.5.12.1.1 Right side of display

PFN 000

This message gives the PHYSICAL FILE NUMBER of the file whose directory MASTER entry is displayed immediately below it.

The PFN is a means of identifying the file besides the NAME/EXTENSION given in the directory entry. Additionally, the

PFN of a file tells the file's relative location in the directory (refer to the appropriate DOS manuals for a discussion of the directory). This information can be useful, especially with a relatively new pack, in indicating which files are older and which are newer.

DIRECTORY MASTER entry

The directory entry for a file provides the fundamental means of identifying the file on the disk. The directory entry contains information as follows:

The physical disk address of the beginning of the file is given in the first byte and the higher two digits of the second byte. The first byte is the cylinder address and the top three bits of the second byte are the cluster number (00, 10, 20 and 30 are valid). Since the RIBs are the first two records in the file, this address points to the file's RIBs.

The protection of the file is given in the bottom digit of the second byte. 1 or 3 = write protection, 2 = delete protection.

The old logical record number limit field is given in the third (LSP) and fourth (MSP) byte of the file, as a 16-bit binary number. This field is currently unused by DOS.C, which normally sets it to zero when a file is created.

The NAME/EXTENSION of the file is given in the 5th through 12th bytes and the 13th through 15th bytes respectively.

The last byte of the directory entry is the number of the DOS subdirectory on that logical drive containing the file.

OF CLUSTERS IN FILE: 00000

This message gives the number of clusters in the file as a 5-digit octal number.

Besides giving the operator an indication of the size of the file, it can be compared to the number of clusters in the file involved in cluster allocation conflicts (below), to give a relative indication of what percent of the file may be in error.

OF CONFLICTING FILES: 000

This message gives the number of files (in octal) that conflict with the file displayed on the left of the screen.

If the number is very large, and the file not very important to the operator, then the operator may decide to delete the file rather than look at all of the files that have cluster allocation conflicts with it.

OF CONFLICTING CLUSTERS: 00000

This message gives the number (in octal) of clusters that are in conflict for the entire file. If the file has conflicts with many files then this number will almost always be larger than the corresponding number on the right side of the display.

The number of conflicting clusters for a file can give the operator a quantitative indication of possible damage to the file.

OF CORRECT PFN/LRN: 00000 OF 00000

This message gives the number of records in the file that have the correct DOS header information in them (being the PFN in the first byte of the physical record and the LRN in the second and third bytes of the record) for the clusters that are in conflict with other files, and the number of records in the clusters that are in conflict. Both of the numbers are in octal.

If a record in a contested cluster does not have the correct PFN/LRN information, then it has probably been overwritten by a record of a file that also claims the cluster.

This message gives the operator an indication of actual damage to the file. If the number of correct PFN/LRN is high, then there is little damage to the file and the RIB for the file is probably correct. If the number of correct PFN/LRN is very low, then the file has probably been overwritten by another file and/or the file's RIB is incorrect.

4.5.12.1.2 Right side of screen

PFN 000

Same as for left side of screen.

DIRECTORY MASTER entry

Same as for left side of screen.

OF CLUSTERS IN FILE: 00000

Same as for left side of screen.

CONFLICTING FILE # 000

This message provides a counter (in octal) to help the operator keep track of which file among several that he is dealing with which the file on the left of the screen has cluster allocation conflicts. This number can never exceed the # OF CONFLICTING FILES: 000 count.

OF CONFLICTING CLUSTERS: 00000

This message gives the number (in octal) of clusters that are in conflict between the files indicated on the left and right of the screen.

OF CORRECT PFN/LRN: 00000 OF 00000

This message gives the number of records in the file that have the correct DOS header information in them (PFN and LRN) for the clusters that are in conflict with the file indicated on the left of the screen, and the number of records in the clusters that are in conflict. Both of these numbers are in octal.

Refer to the discussion under this message for the left side of the screen.

4.5.13 Cluster allocation table replacement

Refer to the pictogram in 4.4.13.

The CLUSTER ALLOCATION TABLE that will be written to disk is a combination of the CAT for files that had no cluster allocation conflicts and the CAT for files that had cluster allocation conflicts but that the operator did not wish to delete. The allocation for files with cluster allocation conflicts is retained so that if a new file is created it will not take space that is being used by one of the un-deleted but erroneous files, thus compounding cluster allocation conflicts.

Files that will still exist in the directory but will not have space allocated to them will be:

Files with an invalid RIB address in their directory MASTER entry;
Files with any uncorrected error in either RIB

The reason disk space is not allocated to these files is

that if REPAIR cannot find the RIB for a file or if the RIB has uncorrected errors then REPAIR has no way of knowing where the file's clusters should be located. Any files of this class are best transferred to cassette (if possible) and KILLed before any new data is loaded to disk.

4.5.14 Termination of REPAIR

Refer to the pictogram in 4.4.14.

When the REPAIR program has finished execution it goes into a dead loop; that is, it executes a JUMP to self. This is so that the processor will be "locked up" by the REPAIR program until the operator takes some specific action, such as putting a LGO program cassette in the rear deck and depressing RESTART. REPAIR does not assume the DOS is loadable so it does not return to the DOS. If the auto-restart tab were punched out of the cassette in the rear deck and REPAIR executed a HALT instruction upon completion, then the computer would attempt to load and execute the cassette in the rear deck, which the operator may not wish to happen.

4.6 Maximal Operator Interface

This section discusses a variety of things that can go wrong on a disk pack and how REPAIR can be used to deal with those problems. This section is for users who are interested in understanding the DOS disk data structure for its own sake, with emphasis, of course, on problems that can occur.

To use this section requires that the user have a copy of and understand the use of either of the DOS commands, DUMP or DUMP9380. The ability to use the Assembler will be mandatory in some cases.

This section assumes that the REPAIR program is used as an error-finding tool, and that the user, with the aid of one of the disk dump programs and special programs he may create, can fix errors that develop on the disk. A specific example is the case of a file with bad RIBs. REPAIR can tell the operator that the file's RIBs contain errors. Either disk dump program can be used to determine the magnitude of the damage to the RIBs, and, if necessary, where the file's records actually are on disk. If necessary, the user can create a simple Assembly language program to re-create the file's RIBs on disk. Sometimes it will be less effort to re-create a file's RIBs than to re-create the file itself.

This section follows the section numbering scheme of the two previous sections, MINIMAL and MEDIAL OPERATOR INTERFACE.

When a facet of REPAIR operation is discussed more appropriately elsewhere, the discussion is not repeated in this section, but the user is referred to the section containing the discussion.

4.6.1 Executing REPAIR

Refer to the discussion in 4.5.1.

4.6.2 Sign-on and drive number specification

Refer to the discussion in 4.5.2.

4.6.3 Cylinder lockout

Refer to the discussion in 4.5.3.

If the user is not sure whether cylinders have been locked out on a disk (and the Lockout CAT and backup have both been destroyed), either of the disk dump programs can be used to look at the cylinders on disk.

Cylinders that have been reserved for special use can generally be recognized by the formatting of their sectors. Sectors that have not been used by the normal DOS routines will not have the special DOS header information in the first three bytes. The first byte is the PFN (Physical File Number) of the file, and the second and third bytes give the LRN (Logical Record Number) of the record in the file. For records that have been written by the normal DOS, each cluster will have the same first byte, and the second and third bytes will be incremented by one (LSP, MSP).

4.6.4 Directory check monitor

Refer to the discussion in 4.5.4.

If a page in the directory has been accidentally overwritten by a record from a file, then REPAIR will find many errors in that directory page. If while executing REPAIR the operator notices that there are quite a few errors in the directory, he can note the directory page address as shown by the directory check monitor. (The left number of the directory check monitor is the physical sector number of the directory page). Using either of the disk dump programs the operator can look at the bad

directory page(s).

If the damage is only to one copy of the directory (the usual case) then REPAIR can recover the directory. However, the operator may wish to use either disk dump command to look at the directory to see if, by examining the data there, he can determine if an error in a user program has caused the directory to be overwritten. Clues to such events can be gleaned by noting the first byte of the record (which would be a file PFN), for example.

4.6.5 Directory errors

Refer to the discussion in 4.5.5 and 4.6.4 above.

4.6.6 Retrieval Information Blocks check

Refer to the discussion in 4.5.6.

4.6.7 Retrieval Information Blocks errors

Refer to the discussion in 4.5.7.

The following discussion gives an example of how the user can go about re-constructing one or both damaged RIBs for a file. Only the case of complex errors (errors in the segment descriptors) is considered because REPAIR is capable of correcting simple errors.

Complex RIB errors can come in infinitely many kinds and combinations. The REPAIR diagnostics will describe specific errors, but if the user is considering fixing a RIB he must examine the RIB himself and determine what is wrong with it and how to correct it. Sometimes this will involve examining records on disk and determining whether or not the records belong to the file and how they should be organized in the RIBs SEGMENT DESCRIPTORS. Because of the potential complexity of this operation the current version of REPAIR does not attempt the analysis necessary to re-construct a RIB with complex errors.

4.6.7.1 A simple case

A relatively simple-to-fix case might go like this:

1. REPAIR would find a file with simple and complex errors in the RIB MASTER.

2. The user would use either disk dump program to look at the RIBs and determine that the RIB MASTER had somehow been completely destroyed, but the format of the RIB COPY seemed to be correct.

3. Using the information in the segment descriptors of the RIB COPY, the user would determine that the COPY was correct.

4. The user could then use the DUMP9380 CASSETTE DUMP command to dump the RIB COPY to cassette, then use the DUMP9370 CASSETTE LOAD command to load the record to the RIB MASTER.

5. The user would run REPAIR again. This time REPAIR would find that the RIB MASTER for the file had one simple error, namely, that the LRN was incorrect. REPAIR could correct this error.

6. The original error is thus corrected.

4.6.7.2 A complex case

The worst case of RIB damage could be corrected in the following manner:

1. REPAIR would find a file with simple and complex errors in both the RIB MASTER and COPY.

2. The user would use either disk dump program to look at the RIBs and determine that the RIBs had somehow been completely destroyed, but that the file following the RIBs was not damaged. (This can happen when a program incautiously uses DOS logical file 0).

3. Using either disk dump program the user would locate and map all of the file's SEGMENTS on disk.

4. From the information about the file's segments, the user would re-construct the file's RIBs, and write a program to write the RIBs to disk.

5. As a check on the above, REPAIR would be run to insure that the new RIBs for the file did not indicate an allocation conflict with another file.

6. The error is thus corrected.

4.6.8 End of RIB check

Refer to the discussion in 4.5.8.

4.6.9 Cluster allocation phase, Pass 1

Refer to the discussion in 4.5.9.

4.6.10 Cluster allocation phase, Pass 2

Refer to the discussion in 4.5.10.

4.6.11 Cluster allocation phase, Pass 3

Refer to the discussion in 4.5.11

4.6.12 Cluster allocation conflicts

Refer to the discussion in 4.5.12.

The user may wish to use either disk dump program to inspect the actual data on disk before deleting one or both of two files with cluster allocation conflicts.

For a file with cluster allocation conflicts, one of five things may be true:

1. The file may have correct RIBs and all correct records. (That is, the error is in the file(s) having the cluster allocation conflict with this file).

2. The file may have incorrect RIBs.

3. The file's space has been erroneously allocated to another file.

4. Another file has erroneously been allocated the file's space.

5. Any combination of (2), (3) and (4) above.

Either disk dump program can be used to look at the RIBs of files with cluster allocation conflicts. From the information given by the SEGMENT DESCRIPTORS either disk dump program can be used to look at where the file's records should be on disk. If the records for the file are where they should be according to

the RIB, then the file possibly has no errors.

Note: For files such as Indexed DATABUS 7 files (physically random access as opposed to ISAM access files), all of the space allocated to a file will not necessarily be used. In these cases, some (frequently many) of the records in the file will never have been written and in such cases the PFN/LRN of these unwritten records being incorrect does not necessarily indicate file allocation problems. The user will need to be aware of the structure of the files being examined.

From the information gathered by examination of the actual data on disk, the user can determine whether a file has errors and if so, whether corrections should be made, and if so, what corrections. In some cases the user may want to change a file's RIB to relocate the file on disk. This, of course, would require careful study of the real allocation of space on the disk and regeneration of the file's RIBs.

4.6.13 Cluster allocation table replacement

Refer to the discussion in 4.5.13.

4.6.14 Termination of REPAIR

Refer to the discussion in 4.5.14.

4.7 All about Cylinder Lockout with REPAIR

This chapter describes the mechanics of locking out cylinders. To accomplish this with the REPAIR program does not require an understanding of the cylinder concept.

Any cylinders that are reserved (locked out) on a disk should be recorded on a sticker or label on the cover of the disk. In addition, the cylinders to be locked out are recorded internally on the disk itself in the Lockout CAT and its backup. Note that the example that follows indicates cylinders larger than the maximum number present under DOS.C, but the concept is what is being illustrated rather than being a precise, letter-for-letter example. The list of cylinders to be locked out might look something like:

FLAGGED CYLINDERS (or TRACKS):
40-50
167
200-202

The following example shows how a list of cylinders as above would be locked out in the REPAIR program.

```
                    DATAPOINT DOS.X  REPAIR


DRIVE NUMBER: 0
LOCKOUT CAT: FORMAT LOOKS OK.
DO YOU WANT TO LOCK OUT ADDITIONAL CYLINDERS ? #
```

The screen appears as above when REPAIR is ready to accept cylinder lockout instructions. The instructions serve as <u>additional</u> cylinders to be locked out. REPAIR will not normally allow cylinders which have been previously locked out to be "unlocked". When REPAIR finishes execution and optionally rewrites the Lockout CATs, the cylinders locked out will be those originally locked out plus those specified by the operator in the following steps.

To lock out cylinders, the operator must enter "Y".

```
                    DATAPOINT DOS.X  REPAIR


DRIVE NUMBER: 0
LOCKOUT CAT: FORMAT LOOKS OK.
*** ARE YOU SURE ? *** #
```

REPAIR will make sure the operator wants to lock out cylinders before accepting cylinder numbers to be locked out.

To lock out cylinders, the operator must enter "Y".

```
                    DATAPOINT DOS.X  REPAIR




 DRIVE NUMBER: 0
 LOCKOUT CAT: FORMAT LOOKS OK.
 *** ARE YOU SURE ? *** Y
 CYLINDER NUMBER<S> <1-202>: #
```

The screen will appear as above when REPAIR is ready to accept the first cylinder(s) to be locked out.

If the operator were locking out the cylinders listed above, he would enter 40-50 and press ENTER.

```
                    DATAPOINT DOS.X  REPAIR




 DRIVE NUMBER: 0
 LOCKOUT CAT: FORMAT LOOKS OK.
 *** ARE YOU SURE ? *** Y
 CYLINDER NUMBER<S> <1-202>: 40-50
 CYLINDER NUMBER<S> <1-202>: #
```

The screen appears as above when REPAIR has accepted the previous cylinder lock out and is ready for the next cylinder number(s).

According to the above sample list, the operator would now enter 167.

```
DRIVE NUMBER: 0
LOCKOUT CAT: FORMAT LOOKS OK.
*** ARE YOU SURE ? *** Y
CYLINDER NUMBER<S> <1-202>: 40-50
CYLINDER NUMBER<S> <1-202>: 167
CYLINDER NUMBER<S> <1-202>: #
```

The screen appears as above when REPAIR has accepted the previous lock-out and is ready to accept the next cylinder number(s).

According to the above list the operator must now enter 200-202.

```
DRIVE NUMBER: 0
LOCKOUT CAT: FORMAT LOOKS OK.
*** ARE YOU SURE ? *** Y
CYLINDER NUMBER<S> <1-202>: 40-50
CYLINDER NUMBER<S> <1-202>: 167
CYLINDER NUMBER<S> <1-202>: 200-202
CYLINDER NUMBER<S> <1-202>: #
```

The screen appears as above when REPAIR has accepted the previous lock-out and is ready to accept the next cylinder number(s).

According to the above example the operator has no more cylinders to lock out. At this point then, the operator would merely depress the ENTER key to signal REPAIR that no more cylinders are to be locked out. REPAIR would proceed immediately to the cluster allocation table and directory check phase.

4.8 CAT errors and directory read/write errors

This section describes messages displayed by the REPAIR program when it discovers an error (of any kind) in the CLUSTER ALLOCATION TABLEs (CATs) or a read or write error in the directory.

These errors are the first type of error checked for by REPAIR. A format (logic) error in one or more of the CATs is not fatal (will not cause REPAIR to abort), but will be noted to the operator. An uncorrectable read or write error in any of the CATs or the directory is fatal, because the disk is in very serious trouble if hardware errors occur in any of these tables.

REPAIR does not consider a read error in the CAT or DIRECTORY fatal until either an attempt to clear the error by writing back to disk has failed or the operator has instructed REPAIR not to attempt the write. A write error to the CAT or DIRECTORY is always fatal.

There is a working (MASTER) and a backup (COPY) version of the Working CAT, the Lockout CAT, and the directory.

The examples that follow are given in the sequence of their potential occurrence in REPAIR execution.

Important notice: Similar sequences are used for errors in the Lockout CAT and its backup as for the Working CAT and its backup. In this chapter, only the Working CAT sequence is used as an example, since both are directly comparable. The messages for both sequences are largely identical to save space. The user can tell at any time whether the messages refer to the Working or Lockout CATs by looking for the header "LOCKOUT CAT:" or "WORKING CAT:" more or less directly preceding the message.

4.8.1 Cluster allocation table read errors

Note that although this example concerns the CAT MASTER, the same messages (substituting the word COPY for MASTER) apply to the CAT COPY.

```
                    DATAPOINT DOS.X  REPAIR




DRIVE NUMBER: 0
LOCKOUT CAT: FORMAT LOOKS OK.
DO YOU WANT TO LOCK OUT ADDITIONAL CYLINDERS ? N
WORKING CAT:
C.A.T. MASTER READ ERROR
WRITE TO DISK TO ATTEMPT TO CLEAR ERROR ? #
```

The messages on the last two lines of the screen above will appear when the REPAIR program has detected a read error in the CAT MASTER. Notice how in this case, the header "WORKING CAT:" implies that the read error has occurred in the Working CAT MASTER as opposed to the Lockout CAT MASTER.

To have REPAIR attempt to clear the read error enter Y; otherwise enter N.

```




DRIVE NUMBER: 0
LOCKOUT CAT: FORMAT LOOKS OK.
DO YOU WANT TO LOCK OUT ADDITIONAL CYLINDERS ? N
WORKING CAT:
C.A.T. MASTER READ ERROR
WRITE TO DISK TO ATTEMPT TO CLEAR ERROR ? Y
READ ERROR CLEARED.
```

The message on the last line of the screen above will appear when the operator has replied "Y" to the message above and the attempt to clear the read error was successful.

No further operator response is required.

```
DRIVE NUMBER: 0
LOCKOUT CAT: FORMAT LOOKS OK.
DO YOU WANT TO LOCK OUT ADDITONAL CYLINDERS ? N
WORKING CAT:
C.A.T. MASTER READ ERROR
WRITE TO DISK TO ATTEMPT TO CLEAR ERROR ? N
READ ERROR UNCORRECTABLE.
THE PACK IS NOT FIXABLE.
```

The messages on the last two lines of the screen above will appear if the operator replies "N" to the message above or if the write to disk did not clear the read error. The REPAIR program will not accept any further commands. To get any other program running on the computer the operator must press the RESTART key.

No operator response is required.

```
                    DATAPOINT DOS.X  REPAIR


DRIVE NUMBER: 0
LOCKOUT CAT: FORMAT LOOKS OK.
DO YOU WANT TO LOCK OUT ADDITIONAL CYLINDERS ? N
WORKING CAT: FORMAT LOOKS OK.
THE C.A.T. MASTER HAS DEVELOPED A READ ERROR
THE PACK IS NOT FIXABLE.
```

The messages on the last two lines of the screen above will appear if a read error occurs when REPAIR reads the CAT MASTER for the second time during the CAT check. This read error is automatically considered fatal because it is evidence of a transient hardware error in the CAT.

No operator response is required.

4.8.2 Cluster Allocation Table is destroyed

Note that although this example concerns the CAT MASTER, the same messages (transposing the words COPY and MASTER) apply to the CAT COPY.

```
                    DATAPOINT DOS.X  REPAIR




DRIVE NUMBER: 0
LOCKOUT CAT: FORMAT LOOKS OK.
DO YOU WANT TO LOCK OUT ADDITIONAL CYLINDERS ? N
WORKING CAT:
THE C.A.T. MASTER IS DESTROYED
WRITE C.A.T. COPY INTO C.A.T. MASTER ? #
```

The messages on the last two lines of the screen above will appear when the REPAIR program has discovered that the CAT MASTER is destroyed but the CAT COPY appears to be valid.

To have REPAIR copy the CAT COPY into the CAT MASTER, enter "Y". Otherwise, enter "N".

```




DRIVE NUMBER: 0
LOCKOUT CAT: FORMAT LOOKS OK.
DO YOU WANT TO LOCK OUT ADDITIONAL CYLINDERS ? N
WORKING CAT:
THE C.A.T. MASTER IS DESTROYED
WRITE C.A.T. COPY INTO C.A.T. MASTER ? Y
DONE.
```

The message on the last line of the screen above will appear when the operator has replied "Y" to the message above and the write to the CAT MASTER was successful. REPAIR will proceed to check the directory (see 4.4.4).

No operator response required.

```
DRIVE NUMBER: 0
LOCKOUT CAT: FORMAT LOOKS OK.
DO YOU WANT TO LOCK OUT ADDITIONAL CYLINDERS ? N
WORKING CAT:
THE C.A.T. MASTER IS DESTROYED
WRITE C.A.T. COPY INTO C.A.T. MASTER ? Y
DISK WRITE ERROR FOR C.A.T. MASTER.
THE PACK IS NOT FIXABLE.
```

The messages on the last two lines of the screen above will appear if a write error occurs when REPAIR tries to write to the CAT MASTER. The REPAIR program will not accept any further commands. To get any other program running on the computer the operator must press the RESTART key.

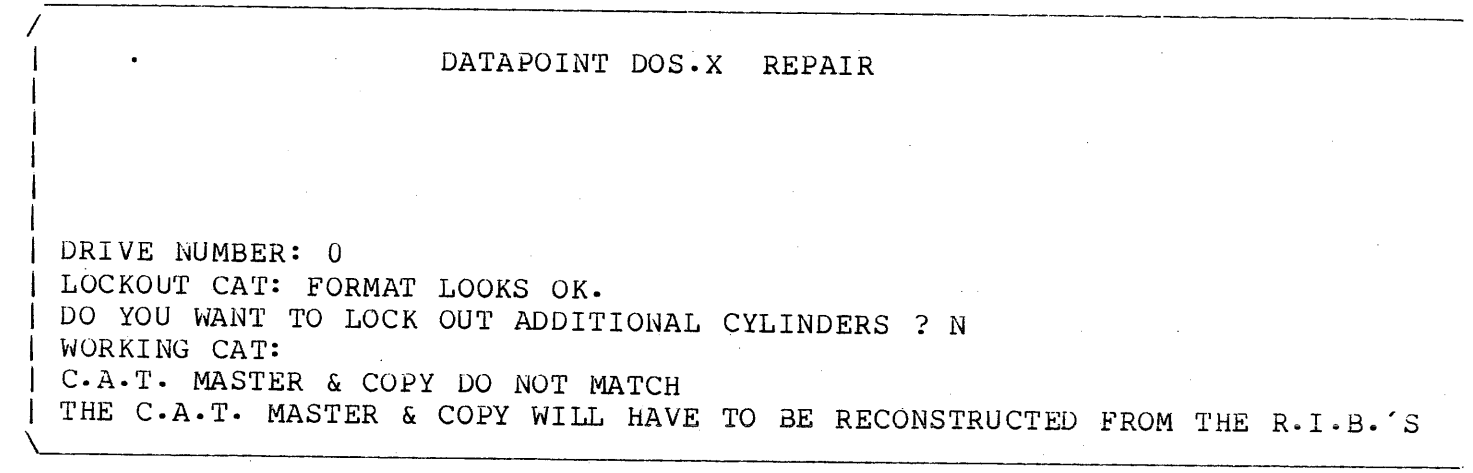
No operator response is required.

```
DRIVE NUMBER: 0
LOCKOUT CAT: FORMAT LOOKS OK.
DO YOU WANT TO LOCK OUT ADDITIONAL CYLINDERS ? N
WORKING CAT:
THE C.A.T. MASTER IS DESTROYED
THE C.A.T. MASTER & COPY ARE DESTROYED
THE C.A.T. MASTER & COPY WILL HAVE TO BE RECONSTRUCTED FROM THE R.I.B.'S
```

The messages on the last three lines of the screen above will appear if REPAIR discovers that both copies of the CAT are destroyed. After the messages are displayed REPAIR will proceed

to check the directory (see 4.4.4).

4.8.3 Cluster Allocation Table copies do not match

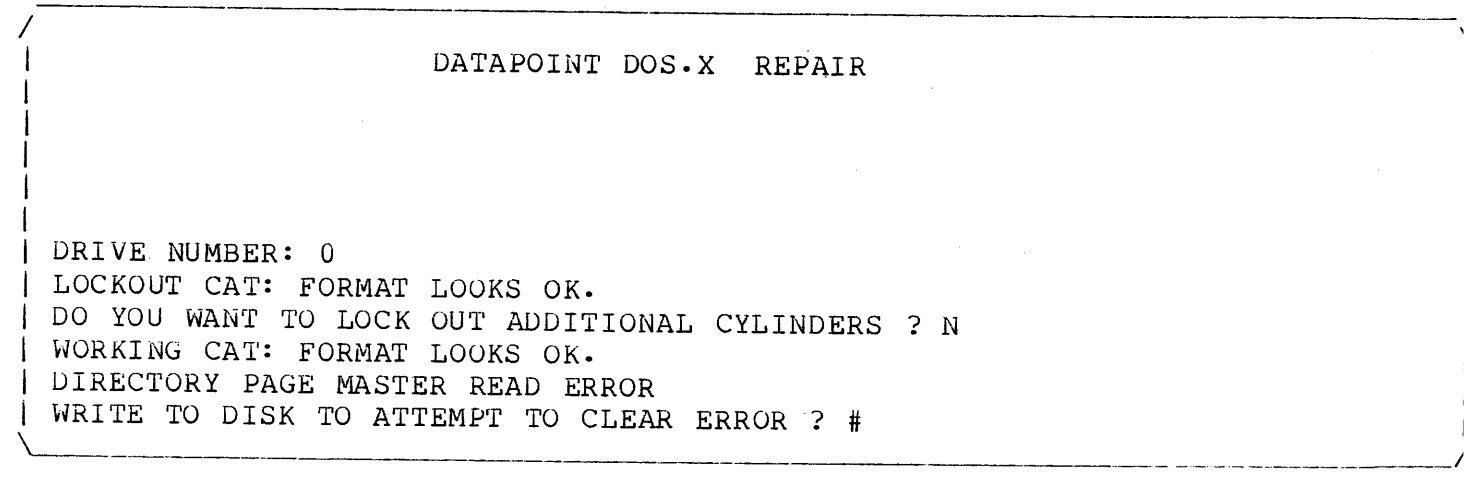


The messages on the last two lines of the screen will appear when REPAIR has discovered that the CAT MASTER and COPY versions do not agree with each other. Since it is not possible for REPAIR to choose which version is correct at this point, it will proceed to check the DIRECTORY (see 4.4.4).

No operator response is required.

4.8.4 Directory read errors

Note that although this example concerns the directory MASTER, the same messages (transposing the words COPY and MASTER) apply to the directory COPY.



The messages on the last two lines of the screen above will appear when REPAIR has detected a read error in the directory MASTER.

To have REPAIR attempt to clear the read error enter "Y", otherwise enter "N".

```
DRIVE NUMBER: 0
LOCKOUT CAT: FORMAT LOOKS OK.
DO YOU WANT TO LOCK OUT ADDITIONAL CYLINDERS ? N
WORKING CAT: FORMAT LOOKS OK.
DIRECTORY PAGE MASTER READ ERROR
WRITE TO DISK TO ATTEMPT TO CLEAR ERROR ? Y
WRITE COPY PAGE TO MASTER PAGE ? #
```

The message on the last line of the screen above will appear if the operator has replied "Y" to the message above.

To have REPAIR copy the directory COPY page to the directory MASTER page enter "Y", otherwise enter "N". If "N" is entered the directory check will continue.

```
DRIVE NUMBER: 0
LOCKOUT CAT: FORMAT LOOKS OK.
DO YOU WANT TO LOCK OUT ADDITIONAL CYLINDERS ? N
WORKING CAT: FORMAT LOOKS OK.
DIRECTORY PAGE MASTER READ ERROR
WRITE TO DISK TO ATTEMPT TO CLEAR ERROR ? Y
WRITE COPY PAGE TO MASTER PAGE ? Y
DONE.
```

The message on the last line of the screen above will appear when the write to the directory MASTER has been successful. The directory check will continue.

No further operator response is required.

```
DRIVE NUMBER: 0
LOCKOUT CAT: FORMAT LOOKS OK.
DO YOU WANT TO LOCK OUT ADDITIONAL CYLINDERS ? N
WORKING CAT: FORMAT LOOKS OK.
DIRECTORY PAGE MASTER READ ERROR
WRITE TO DISK TO ATTEMPT TO CLEAR ERROR ? Y
WRITE COPY PAGE TO MASTER PAGE ? Y
DIRECTORY PAGE MASTER WRITE ERROR
THE PACK IS NOT FIXABLE.
```

The messages on the last two lines of the screen above will appear if the operator replied "Y" to the message above and REPAIR detected a write error when it attempted to write to the directory MASTER. The REPAIR program will not accept any further commands. To get another program running on the computer the operator must press the RESTART key.

No operator response is required.

```
DRIVE NUMBER: 0
LOCKOUT CAT: FORMAT LOOKS OK.
DO YOU WANT TO LOCK OUT ADDITIONAL CYLINDERS ? N
WORKING CAT: FORMAT LOOKS OK.
DIRECTORY PAGE MASTER READ ERROR
WRITE TO DISK TO ATTEMPT TO CLEAR ERROR ? Y
WRITE COPY PAGE TO MASTER PAGE ? Y
DIRECTORY PAGE MASTER READ ERROR
THE PACK IS NOT FIXABLE.
```

The messages of the last two lines of the screen above will appear if the operator replied "Y" to the message above and REPAIR detected a read error when it attempted to re-read the directory MASTER page it had just written.

No operator response is required.

```
DRIVE NUMBER: 0
LOCKOUT CAT: FORMAT LOOKS OK.
DO YOU WANT TO LOCK OUT ADDITIONAL CYLINDERS ? N
WORKING CAT: FORMAT LOOKS OK.
DIRECTORY PAGE MASTER READ ERROR
WRITE TO DISK TO ATTEMPT TO CLEAR ERROR ? Y
WRITE COPY PAGE TO MASTER PAGE ? Y
DIRECTORY COPY PAGE HAS DEVELOPED A READ ERROR
THE PACK IS NOT FIXABLE.
```

The messages on the last two lines of the screen above will appear if the operator replied "Y" to the message above and REPAIR detected a read error when it attempted to re-read the directory COPY to compare it against the directory MASTER page just written and re-read.

No operator response is required.

```
DRIVE NUMBER: 0
LOCKOUT CAT: FORMAT LOOKS OK.
DO YOU WANT TO LOCK OUT ADDITIONAL CYLINDERS ? N
WORKING CAT: FORMAT LOOKS OK.
DIRECTORY PAGE MASTER READ ERROR
WRITE TO DISK TO ATTEMPT TO CLEAR ERROR ? Y
WRITE COPY PAGE TO MASTER PAGE ? Y
DIRECTORY PAGE MASTER & COPY DO NOT MATCH
THE PACK IS NOT FIXABLE.
```

The messages on the last two lines of the screen above will appear if the operator replied "Y" to the message above but the directory page MASTER and COPY did not match after the page copy had been made. This error is automatically considered fatal because it is evidence of a hardware error in the directory.

No operator response is required.

```
DRIVE NUMBER: 0
LOCKOUT CAT: FORMAT LOOKS OK.
DO YOU WANT TO LOCK OUT ADDITIONAL CYLINDERS ? N
WORKING CAT: FORMAT LOOKS OK.
DIRECTORY PAGE MASTER READ ERROR
WRITE TO DISK TO ATTEMPT TO CLEAR ERROR ? Y
DIRECTORY PAGE COPY ALSO GIVES READ ERROR
WRITE TO DISK TO ATTEMPT TO CLEAR ERROR ? #
```

The messages on the last two lines of the screen above will appear if the operator replies "Y" to the message above and REPAIR detects a read error in the directory COPY page.

To have REPAIR attempt to clear the read error entery "Y", otherwise enter "N". If the write is successful REPAIR will continue with the directory check.

```
DRIVE NUMBER: 0
LOCKOUT CAT: FORMAT LOOKS OK.
DO YOU WANT TO LOCK OUT ADDITIONAL CYLINDERS ? N
WORKING CAT: FORMAT LOOKS OK.
DIRECTORY PAGE MASTER READ ERROR
WRITE TO DISK TO ATTEMPT TO CLEAR ERROR ? Y
DIRECTORY PAGE COPY READ ERROR
WRITE TO DISK TO ATTEMPT TO CLEAR ERROR ? Y
READ ERROR UNCORRECTABLE.
THE PACK IS NOT FIXABLE.
```

The messages on the last two lines of the screen above will appear if the operator replied "N" to the message above or if the write to disk did not clear the read error.

No operator response is required.

CHAPTER 5. DISK ORGANIZATION UNDER DOS.C

This chapter describes the logical organization of the data on the disk when operating under DOS.C and how it relates to the general DOS file concepts as described in the Advanced Programmer's Guide section of the DOS USER'S GUIDE. In this chapter it is assumed that the user is familiar with the basic DOS file structuring.

5.1 Radius spiraling and sector skewing

Under DOS.C, the sectors on the diskette are logically renumbered to allow substantially increased performance over what would be possible otherwise. Program loading, in particular, goes about three times faster than would be possible if this were not done. This renumbering of the sectors on the track is referred to as sector skewing. This sector skewing takes the form of placing logically sequential sectors about four sectors apart on a track of the diskette. Thus logical sector zero would appear in physical sector zero; logical sector one would appear in physical sector five; logical sector two would appear in physical sector ten; and so forth.

In addition to rearranging the order of the sectors on a track of the diskette, the starting points (logical sectors zero on each track) do not line up along a straight-line radius as do the physical sectors zero. Instead, the starting point for numbering sectors on a track spirals inwards. Therefore, the logical radius line (sectors zero, for example) forms a spiral on the diskette surface, and hence the term radius spiraling. The intention behind radius spiraling is twofold: one, it allows for head seek time between adjacent tracks while rapidly scanning through a data file (in addition to the processing time lag provided by the normal sector skewing) and secondly allows searching the directory (which is along a logical radius of the diskette, as will be described later) about three times faster than would otherwise occur. Together with sector skewing, radius spiraling aids in achieving much higher overall system performance than is obtainable on most competitive diskette based systems.

5.2 Size of a diskette

There are 77 tracks on a diskette, each of which contains thirteen 256-byte sectors. This yields a total of 1001 sectors, or a grand total of 256,256 bytes of storage capacity. The first track on the diskette (the one nearest the edge of the diskette) is not used by DOS.C, in order to help provide compatibility with IBM equipment. Additionally, the logical last sector on each track (sector 12 if one counts starting at 0) is not used by DOS.C for data, for reasons which will be described in subsequent sections. Subtracting these two unallocatable areas results in a total allocatable file space of 912 sectors. About ninety sectors of these are used by the DOS for its system files, leaving about 825 sectors for user files, a user file capacity of over 200,000 bytes. This constitutes about twice the capacity of a tape cassette on each diskette. Due to the higher data storage efficiency attained by Datapoint software, most users will find that the total number of records stored on a Datapoint format diskette will be as large as, and in most cases substantially larger than, the number achieved on competitive systems.

5.3 Cluster mapping

Under DOS.C, each track of the diskette consists of 4 clusters of three sectors each. This implies that one cluster or three sectors is the smallest allocatable unit of space on a diskette, and that all files are multiples of three sectors in length.

In the cluster allocation table, the four clusters on each track are represented by the low-order four bits of each byte. As in other Datapoint DOS, a one bit represents that the corresponding cluster is allocated and a zero bit indicates that the corresponding cluster is unavailable for allocation. The high-order four bits of each byte in the CATs are reserved for future use in DOS.C, and are currently always set to zero.

5.4 Segments under DOS.C

The use of a three sector cluster has numerous advantages on the diskette. One which should be immediately apparent is that the amount of space wasted due to always allocating an integral number of clusters is reduced to only an average of one and a half sectors per file. Perhaps a less obvious advantage results from the manner in which the Datapoint DOS allocate disk space to files. During space allocation, the DOS will allocate the first

contiguous, maximum-size segment it can find as an initial (or secondary for that matter) allocation. Since a segment consists of 32 clusters (there are five bits of cluster number information in each segment descriptor), this results in files being initially allocated 96 sectors, assuming that the space on the diskette is sufficiently unfragmented to allow such an allocation. Making this initial allocation smaller than the 192 and 240 sectors as used in some of the other Datapoint DOS allows for several scratch files to be opened on a diskette which already has a few files on it, as each newly opened file will take a smaller bite out of that portion of space remaining unallocated. Making the full segment size much smaller than 96 sectors quickly increases the amount of overhead required to index through the file (since the number of RIB accesses required increases) and decreases performance.

5.5 Maximum file size

Under DOS.C, the maximum file size attainable depends upon the amount of space used on the diskette for system files, but using the current size of DOS.C as an example indicates that at least 800 sectors should be available for user file allocation on a normal data file diskette. Users having only a single diskette drive and therefore having several programs on the diskette in addition to the DOS will have a corresponding reduction in the maximum size of data files they may have. Users whose files exceed the capacity of one diskette will need to segment their files at the user program level much as they would do on a cassette system when a file exceeded the capacity of a single cassette.

5.6 Cluster Allocation Table and Directory

Under DOS.C, the use of four three-sector clusters per track results in one unused sector per track. This restriction arises from the facts that (1) all clusters under Datapoint DOS must contain the same number of sectors and that no cluster may span a track boundary; and (2) a 13-sector cluster is not practical because it results in excessive amounts of wasted space at the end of each file on the diskette. Since these 76 sectors on the diskette (remember that track zero is not used) are not available for allocation as file space, they are partially put to use for storage of DOS system tables: four cluster allocation table sectors and thirty two directory sectors. These system tables are positioned in the following manner:

Track	0 - Unused; for IBM compatibility
Tracks	1-16 - Directory copy, for backup purposes

Tracks 17-32 - Primary DOS directory
Track 33 - Working Cluster Allocation Table
Track 34 - Working Cluster Allocation Table backup
Track 35 - Lockout Cluster Allocation Table
Track 36 - Lockout Cluster Allocation Table backup
Tracks 37-76 - Reserved for future DOS use

Again recall that each of the above sectors is in logical sector 12 of the track indicated.

In the directory, the RIB address field is organized in the same manner as a DOS.C standard physical disk address. The most significant byte is the most significant byte of the physical disk address, exactly as would be in the D register on entry to DRS. The less significant byte contains the least significant byte of the physical disk address, except that since the primary RIB is always the first sector in a cluster the least significant bits of this byte are used as protection indicators for the file (rather than a sector number within the cluster indicated by the most significant two bits).

The field in each 16-byte directory entry that was used as a maximum file length limit field in the old DOS 1.2 is not used by DOS.C and under DOS.C is always initially set to zeros.

The last byte of each directory entry is the subdirectory number, corresponding to the number (in binary) that is displayed by the SUR command immediately following the name of each of the indicated subdirectories on the indicated drives.

In the Cluster Allocation Tables, bytes 239-254 are used for the Directory Mapping bytes. These sixteen bytes each contain either an 0377 or the number of files currently allocated in the corresponding one of the sixteen directory sectors. These bytes are updated automatically by the DOS whenever a file is created or deleted, and are updated by the DOS occasionally if they are found to be inaccurate. The purpose of these directory mapping bytes is to provide improved speed of directory lookups and to allow faster creation of files. They are of the greatest benefit to users who have several drives in their system where relatively few files exist on each drive. The intention is to eliminate the need to read in directory sectors while looking for a file if those sectors are known to not contain any active directory entries, and likewise when looking for an empty slot for use by a new file to eliminate having to read sectors known to have all sixteen directory entries in use.

CHAPTER 6. INTERNAL DOS PARAMETERIZATION

This section describes the DOS-dependent details of the parameterization of DOS.C system routines.

6.1 Physical disk address format

Under DOS.C, physical disk addresses are represented in a two-byte format in a manner quite similar to that used under the other DOS. The most significant byte (which is traditionally placed in the D register) is the cylinder number, just like for DOS.A and DOS.B. The less significant byte (usually placed in the E register) has its most significant two bits representing a cluster number within the track (all combinations of these two bits are valid since there are four clusters per track) and the least significant two bits representing the sector number within the specified cluster. Because there are only three sectors per cluster, only binary values 00, 01 and 10 are valid for these low-order bits. (The only exception to this rule is that a least significant PDA byte of 0303 permits access to the unallocatable sector on each track, that sector used for system table sectors). (For compatibility reasons, the most significant three bits can be considered the cluster number, yielding clusters numbered 0, 2, 4, and 6). This format has several sizeable advantages over the format used by old DOS 1.2 from the standpoint of uniformity among various system data items, easier internal address conversions (between, for example, segment descriptors in the RIB and physical disk addresses), and easier internal computation of sequential sectors within the current segment.

The unused bits of the least significant physical disk address byte (that is, the center four bits) should always be set to zero.