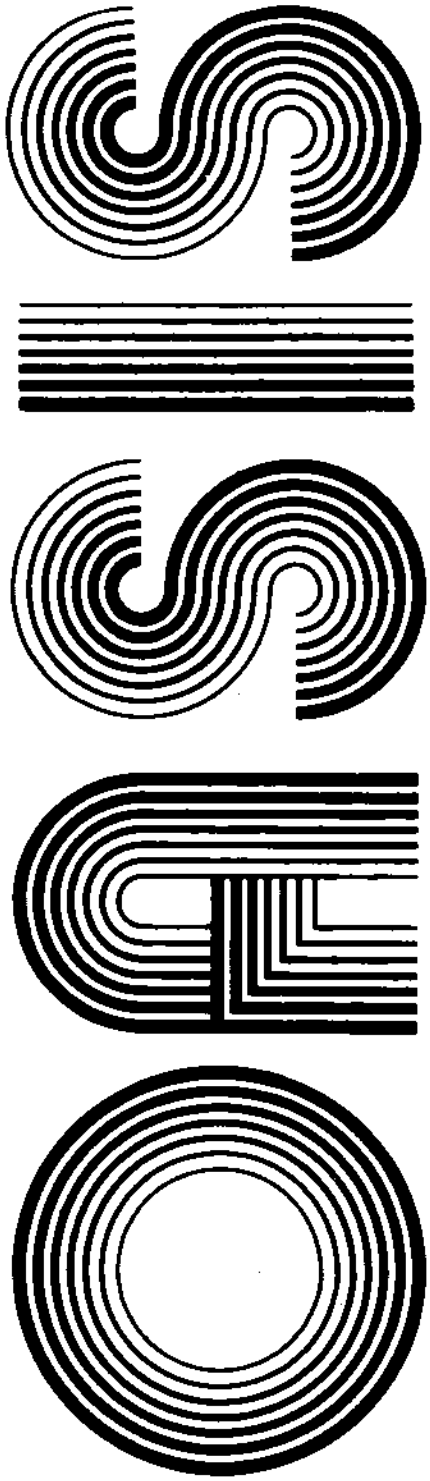

DIAGNOSTIC AND CONVERSION UTILITIES PROGRAM REFERENCE MANUAL

Second Edition

Documentation by: C. P. Williams
Software by: Timothy S. Williams

OPERATING SYSTEM SOFTWARE

MAKES MICROS RUN LIKE MINIS

From
PHASE ONE SYSTEMS, INC.
OAKLAND, CALIFORNIA

7700 Edgewater Drive, Suite 830
Oakland, California 94621
Telephone (415) 562-8085
TWX 910-366-7139

Second edition, first printing: March, 1980

PROPRIETARY NOTICE

P R E F A C E

This manual describes the diagnostic and conversion programs available as an option with the OASIS Operating System.

This manual, named DIAGREF, like all OASIS documentation manuals, has the manual name and revision number in the lower, inside corner of each page of the body of the manual. In most chapters of the manual the last primary subject being discussed on a page will be identified in the lower outside corner of the page.

Related Documentation

The following publications provide additional information useful in the use of the these utility programs:

OASIS System Reference Manual

OASIS EXEC Language Reference Manual

OASIS BASIC Language Reference Manual

OASIS MACRO Assembler Language Reference Manual

TABLE OF CONTENTS

Section **Page**

CHAPTER 1

BASCONV COMMAND

The BASCONV command translates an OASIS BASIC program written with a version of BASIC prior to 5.3H to be compatible with the OASIS BASIC compiler/interpreter. The format of the command is:

BASCONV <file-desc> [(TYPE[])]

Where:

<file-desc> Is the file description of the BASIC program to be upgraded. The file disk may be omitted, in which case a disk search will be performed. The file type may be omitted if the program's file type is BASIC.

TYPE Indicates that the input record and the resulting output record are to be displayed on the console as the conversion is taking place.

The BASCONV command renames the input file to have a file type of BACKUP before starting the conversion.

The BASCONV command performs almost all of the necessary changes that need to be made to a pre-existing BASIC program in order for it to be syntactically acceptable to the BASIC compiler/interpreter. BASCONV program does not handle a multi-line IF-THEN-ELSE statement. This must be changed by you. All keyword expansion, spelling changes, punctuation and syntax changes are made automatically by this command.

BASCONV cannot handle the change in the method that a BASICUSR subroutine is loaded. Since previous versions of BASIC required the USR to be specified on the BASIC command line there is nothing in the program that specifies the USR to be used. You will have to add the OPTION USR statement yourself.

The OASIS BASIC compiler/interpreter has some omitted functions: LNO, LOR, LAN, LXO, and the RPT function with a string argument (the RPT function with two numeric arguments still exists). The omitted functions can all be replaced by you with logical expressions (a new feature).

If you plan to compile your programs then you must take into account that the compiler changes variable names. This can be a serious problem for multi-module linked programs. (A reference to a variable in one program that was defined in another program may not reference the same variable if the two programs have been compiled.) To avoid this problem the programs will have to be modified to take advantage of the COMMON variable feature of the compiler/interpreter.

After a program has been upgraded with this command you should load the BASIC compiler/interpreter and then load the converted program into memory. BASIC will read in each line of code and perform syntax analysis on it before storing it in memory. Any errors in syntax not corrected by BASCONV will be reported to you at the console and you will be allowed to make the necessary correction. After the statement is acceptable to BASIC it will continue reading the rest of the program. After the program has been loaded into memory you should save it back on disk using a file type of BASICOBJ. This file type indicates that the program has already been checked for syntax and is saved in compressed format.

BASCONV Example:

```
>BASCONV SAMPLE BASIC A (TYPE

0010 FORI=1TO10
0010 FOR I=1 TO 10

0020 PRI "COUNT ="I
0020 PRINT "COUNT =";I

0030 NEXT I
0030 NEXT I

0040 IF I = 11 THEPRII;SPA(5);"OKAY"ELSPRIN"HUH"
0040 IF I=11 THEN PRINT I;SPACE(5);"OKAY" ELSE PRINT "HUH"

0050 END
0050 END

>BASIC
-LOAD SAMPLE
-INDENT
-SAVE SAMPLE
"SAMPLE.BASICOBJ:A" saved
-QUIT

>
```

CHAPTER 2

DUMPDISK COMMAND

The DUMPDISK command allows you to list the physical contents of a file on the console or printer. The format of the DUMPDISK command is:

DUMPDISK <file-desc> [(<option> ... [)]]

Where:

<file-desc> Indicates the file description of the logical file to be dumped.

DUMPDISK Options

Options for the DUMPDISK command include the following:

TYPE Indicates that the output of the dump is to be displayed on the console. This is a default option.

PRINTER[n] Indicates that the output of the dump is to be displayed on the printer. If n is not specified PRINTER1 is used.

DUMPDISK Examples

The following example illustrates the format of the output of the DUMPDISK command.

The first line of output contains the relative sector number of the data being displayed. This sector number is displayed in two formats: sector number in decimal, relative to the beginning of the disk; relative sector number broken down by physical track and sector within track.

The lines following this header display the data contained at that disk address. Each line has three sections: the first number indicates the hexadecimal address of the first byte of data displayed on the line; the next four numbers are the data from the disk displayed in hexadecimal; the last section of the line is the data from the disk displayed as its ASCII equivalent, surrounded with single quotes. When a byte of data does not have an ASCII equivalent a period (.) will be displayed.

When the DUMP is of a absolute program file the address of each line of data will be the actual address that the program is loaded into when the program is executed. A DUMP of a relocatable program file will cause the first line of data to be addressed at zero; following lines will be incremented from this base address. When the DUMP is not of a program file then the address of each line of data will be determined by making the first line of each sector zero.

```
>DUMPDISK OASIS6.DOCUMENT:A

OASIS6.DOCUMENT:MANUAL1                    08/02/77 15:47  Page 1

Sector 16  (Track=1, Sector=0)
0000: 2E534543 54202B20 5E5E4153 53454D42 '.SECT + ^^ASSEMB'
0010: 4C452043 4F4D4D41 4E440D2E 4C494E4B 'LE COMMAND..LINK'
0020: 204F4153 4953370D 1A1A1A1A 1A1A1A1A ' OASIS7.........'
0030: 1A1A1A1A 1A1A1A1A 1A1A1A1A 1A1A1A1A '................'
0040: 1A1A1A1A 1A1A1A1A 1A1A1A1A 1A1A1A1A '................'
0050: 1A1A1A1A 1A1A1A1A 1A1A1A1A 1A1A1A1A '................'
0060: 1A1A1A1A 1A1A1A1A 1A1A1A1A 1A1A1A1A '................'
0070: 1A1A1A1A 1A1A1A1A 1A1A1A1A 1A1A1A1A '................'
0080: 1A1A1A1A 1A1A1A1A 1A1A1A1A 1A1A1A1A '................'
0090: 1A1A1A1A 1A1A1A1A 1A1A1A1A 1A1A1A1A '................'
00A0: 1A1A1A1A 1A1A1A1A 1A1A1A1A 1A1A1A1A '................'
00B0: 1A1A1A1A 1A1A1A1A 1A1A1A1A 1A1A1A1A '................'
00C0: 1A1A1A1A 1A1A1A1A 1A1A1A1A 1A1A1A1A '................'
00D0: 1A1A1A1A 1A1A1A1A 1A1A1A1A 1A1A1A1A '................'
00E0: 1A1A1A1A 1A1A1A1A 1A1A1A1A 1A1A1A1A '................'
00F0: 1A1A1A1A 1A1A1A1A 1A1A1A1A 1A1A0000 '................'
  .
  .
  .

>DUMP MYPROG COMMAND

MYPROG.COMMAND:DATA                        08/24/78 09:44  Page 1

Sector 16  (Track=1, Sector=0)
1100: C3013013 0A4B0000 185FC800 00C9000A '..0..K..._......'
1110: C3101100 000000BE C3181100 00006400 '..............d.'
1120: C3201100 00000000 C32811FF 00000000 '. .......(......'
1130: C3542200 00000000 C9C9C900 00000000 '.T".............'
1140: 00000000 00000000 00000000 352E3000 '............5.0.'
1150: 00000000 00000000 0000C393 1700001E '................'
1160: 00000000 0000C300 00E3F57E FE4DD28C '...........~.M..'
1170: 2923E36F E3F1E3E5 F5E1D55D 16002178 ')#.o.......]..!x'
1180: 1319195E 2356EBD1 E3C90000 00000000 '...^#V..........'
1190: 00000000 00000000 00000000 00000000 '................'
11A0: 00000000 00000000 00000000 00000000 '................'
11B0: 00000000 00000000 00000000 00000000 '................'
11C0: 0002FFFF FFFFFFFF 0908FFFF 10FFFFFF '................'
11D0: FFFFFFFF FFFFFFFF 00000000 00000000 '................'
11E0: 00000000 0000FF00 00000000 00000000 '................'
11F0: 00000000 0000FF00 00000000 00000000 '................'
Sector 17  (Track=1, Sector=1)
1200: 00000000 0000FF00 00000000 00000000 '................'
1210: 00000000 0000FF00 00000000 00000000 '................'
  .
  .
  .
```

CHAPTER 3

FILECONV COMMAND

The FILECONV command translates OASIS indexed and direct files created with versions of the operating system prior to 5.4 to a format compatible with version 5.4 and above. The format of the command is:

FILECONV <file-desc>

Where:

<file-desc> Is the description of the indexed or direct file to be converted.

This conversion program performs an in-place transformation of the file without creating a backup of the original; therefore, it is highly advised that you make a backup copy of the file before conversion is attempted.

Due to the changes in the OASIS BASIC language capabilities (integer numbers, greater precision BCD floating point numbers, etc.) it is necessay to convert pre-existing files to the new format. The FILECONV command performs this conversion for you.

Because of the greater length of a floating point number it is possible that the converted record may not fit in the same space as the unconverted record. This situation will be evident if there is any lost data from the end of a record in the converted file. If this appears to have happened then you should: restore the old copy of the file from the backup you made; create a new file with a record length that is sufficient to hold the converted records; boot your old version of OASIS; write a copy program in BASIC that copies the smaller record length file to the larger record length file; boot up your new version of OASIS and repeat the file conversion.

As the file conversion takes place the record keys (indexed files) or record numbers (direct files) of the records being converted are displayed on the screen.

FILECONV Example:

```
>COPY DATAFILE.MASTER:A = BACKUP
DATAFILE.MASTER:A copied to DATAFILE.BACKUP:A

>FILECONV DATAFILE MASTER A
00015

>
```

CHAPTER 4

FILT8080 COMMAND

The FILT8080 command translates INTEL assemble language mneumonics into an OASIS mneumonics (ZILOG). The format of the command is:

FILT8080 <file-desc>

Where:

<file-desc> Indicates the file description of the file to be "filtered". The file name must be specified but the file type may be omitted, indicating a file type of ASSEMBLE.

When this command is executed the input file will be renamed to have a file type of BACKUP (similar to TEXTEDIT) and the translated output will have the same name as the original input file description.

This program recognizes all of the standard op-codes used in the INTEL, TDL, MICROSOFT, CP/M, etc. assemblers but may not translate the assembler directives (also called pseudo op-codes). Once the translation has been done the system text editor can be used to translate the directives to the equivilent directive used in MACRO.

* (CP/M is a registered trademark of Digital Research.)

FILT8080 Example:

Before

```
; Positive integer division
DIVIDE:  PUSH    B
         MVI     B,8
DD04:    SLAR    D
         RAL
         SUB     E
         JM      .+6
         INR     D
         JMPR    .+3
         ADD     E
         DJNZ    DD04
         POP     B
         RET
```

After

```
; Positive integer division
DIVIDE:  PUSH    BC
         LD      B,8
DD04:    SLA     D
         RLA
         SUB     E
         JP      M,.+6
         INC     D
         JR      .+3
         ADD     E
         DJNZ    DD04
         POP     BC
         RET
```

CHAPTER 5

GETFILE COMMAND

The GETFILE command provides a method of transferring a file from a non-OASIS compatible disk to an OASIS disk. Once this has been done the file (probably a program) can be manipulated with the various utilities for whatever purpose. The format of the GETFILE command is:

GETFILE <fn>.<ext>:<fd> (<disk type>[])

Where:

fn Indicates the name of the file on the non-OASIS disk.

ext Indicates the file extension of the file on the non-OASIS disk.

fd Indicates the file disk that the file is to be transferred to on the OASIS system.

GETFILE Disk Types

CPM Indicates that the non-OASIS disk is a CP/M format disk.

When the command is executed the program will ask you to specify the drive that the source disk is in:

Enter physical drive code of CP/M disk?

Enter a value from 1 to 8 that corresponds to the drive number of the disk. When this has been done the program will access the disk and transfer the designated file to the OASIS disk specified. The destination file will have a file name equal to the source file name, and a file type equal to the source file extension.

Only ASCII sequential files may be transferred with this command.

* CP/M is a registered trademark of Digital Research.

CHAPTER 6

INTELHEX COMMAND

The INTELHEX command converts an object file that conforms to the INTEL standard to an object file conforming to the OASIS object file requirements. The format of the command is:

INTELHEX <file-desc>

Where:

<file-desc> Is the file description of the INTEL HEX object file to be converted. The file type may be omitted, in which case the default type of HEX is used.

The resulting file created by the INTELHEX command is a file with the same file name as the input file but has a file type of OBJECT.

After the INTELHEX command has converted an object file to the OASIS format the LINK command can be used to create a command program with absolute code (not relocatable). See the RELOCATE command for information on converting absolute programs to relocatable programs.

INTELHEX Example:

```
>LIST SORT HEX A

:10010000214601360121470136007EFE09D2190140
:100110002146017EB7C20001FF5F16002148011988
:10012000194E79234623965778239EDA3F01B2CAA7
:100130003F0156702B5E712B722B732146013421C7
:07014000470134C30A01006E
:1001480050064001E0032001400070038032C01BB
:0401580064000180BE
:0000000000

>INTELHEX SORT

>LIST SORT OBJECT (OBJ

10 P 00 005C SORT      01 0100
15 T 00 0100 214601360121470136007EFE09D21901
15 T 00 0110 2146017EB7C20001FF5F160021480119
15 T 00 0120 194E79234623965778239EDA3F01B2CA
15 T 00 0130 3F0156702B5E712B722B732146013421
0C T 00 0140 470134C30A0100
15 T 00 0148 050064001E0032001400070038032C01
09 T 00 0158 64000180
02 Z 02 1A1A

>
```

CHAPTER 7

KILL COMMAND

The KILL command removes a file's entry in the directory without de-allocating its disk space. The format of the command is:

KILL <file-desc>

Where:

<file-desc> Is the file whose directory entry is to be killed. The complete file description must be specified as no disk searching is performed.

It is possible that, during program development, a disk file is created that uses another file's disk space (file collision). When this situation arises it is virtually impossible to recover both files as at least one of the files has been overwritten.

After you have determined which file you wish to save you should use the KILL command to erase the other file's directory entry then use the REPAIR command to de-allocate the file's disk space that does not collide with the other file. (If you use the ERASE command mis-allocation will result because the file's entire disk space is de-allocated, even the portion that belongs to the other, good file.)

KILL Example:

```
>REPAIR A
File 1 DATA.FILE
File 2 TEST.DATE
*** Collision with file 1 at sector 52 ***

>KILL DATA.FILE:A

>REPAIR A (FIX
File 1 TEST.DATA
Sector 40 is allocated but not used
Sector 44 is allocated but not used
Sector 48 is allocated but not used

>SHOW DISK A

A(2) Label: "DATADISK",
     Capacity: 497K bytes (77-1-26),
     Available: 308K bytes (61%),
     Largest area: 295K bytes,
     1 files in use (out of 80).
```

CHAPTER 8

MEMTEST COMMAND

The MEMTEST command allows you to diagnose memory for most of the types of errors that might occur. The format of the MEMTEST command is:

MEMTEST [<start-addr> <end-addr>]

Where:

<start-addr> Indicates the starting address of the memory to be tested.

<end-addr> Indicates the last address of the memory to be tested. When the <start-addr> is specified this parameter must be specified.

When <start-addr> and <end-addr> are not specified all of memory will be tested.

When the MEMTEST command begins execution it will display on the console terminal the range of memory being tested. As the program completes a cycle of testing a counter is incremented and displayed on the terminal.

To exit from this program you must use the System Cancel-key.

When the MEMTEST detects an error in memory storage the following information is displayed:

* Memory address in hexadecimal
* The contents that should have been stored in the memory address. This value is displayed in hexadecimal and binary.
* The contents that was stored in the memory address. This value is displayed in hexadecimal and binary.

Caution: It is possible to destroy the resident operating system with this command. If you think this may have happened then reboot the system.

When no starting and ending addresses are specified the test will be perfomed in two parts. Part I is a non-destructive test of the volatile portions of memory. This is performed by determining where the volatile portions are, moving the data in these areas to a temporary location, testing the area, and moving the data back to its original location. Testing of this area is performed only once with the test ending when the end of the region is encountered or when the first memory error is detected and reported.

Part II is a destructive test of the user area of memory. This part of the test executes until the program is stopped by the System Cancel-key.

MEMTEST Examples:

```
>MEMTEST A000H AFFFH
A000H-AFFFH
A400H,4CH(01001100),04H(00000100)
5

>MEMTEST

Part I  - Non Destructive Test of System Regions

    a. NUCLEUS-  1100-3AFFH
    b. MEMTEST-  3B00-3DFFH
    C. Protected (Drivers etc.)- C900H-FFFFH
       (Note- may get error when end of memory reached)
       D000H,00H(00000000),FFH(11111111)

Part II - Destructive Test of User Regions

3E00H-C8FFH
99

>
```

The first example illustrates the display of an error encountered in the program's fourth iteration of the memory test. The second example illustrates the default testing size for a system with 48K of memory and no errors detected up through the 99th iteration.

Note: The actual address of the default testing vary from computer type to computer type.

CHAPTER 9

RECOVER COMMAND

The RECOVER command recreates an ASCII sequential file's directory entry. When, for whatever reason, a disk's directory loses it's integrity, the RECOVER command might allow you to recover a file on the disk, restoring it's directory entry. The format of the command is:

RECOVER <file-desc>

Where:

<file-desc> Indicates the explicit file description (filename, filetype, filedisk) of the file to be recovered. This command operates on sequential format files only.

Upon execution of this command the program will ask you:

Enter match string (LF = end of line, continue):

In response to this question you must enter the text of the first line(s) of the file to be recoverd. This text may be up to 254 characters in length. To indicate the end of a record without indicating end of entry use the line-feed character (on some terminals you may have to enter a CTRL/J for the line-feed character). Enter the text exactly as you remember it, using the proper case mode for each character.

The program will then search the designated disk for this string of characters. If no match can be found the message "End of disk" is displayed and the command is exited.

When a match is found the command will display the first sector of the file found on the screen, followed by the sector number of the disk that it was found on. Then it will ask you:

Is this the correct file (Y/N)?

You may reply Y or N to this question depending upon whether or not it is the correct file. If you are unsure reply with an N and the program will continue searching the disk for another match.

When you reply with a Y the program will display the file on the screen in its entirety. Upon completion of the display a directory entry will be created for the file in your account. This program does not attempt to allocate space for the file and therefore you may end up with a mis-allocation on the disk. When this happens the REPAIR program will probably correct it. The purpose of the RECOVER command is to recover a file whose directory entry was lost, not one that was erased, although it can be used for such.

It is suggested that you use the FILELIST command with option * to find out if there are other versions of the file still in the directory and what sector they start on. It may also help if you had a similar FILELIST of the disk before the file was lost (of course this is rarely anticipated).

RECOVER Example:

The following example is an attempt to recover the script file that begins the generation of this manual.

```
>RECOVER OASIS SYSREF A
Enter match string (LF = end of line, continue):
.SIZE 5,72,1,58



.SIZE 6,88,1,60
.CASE M
.TITLE SYSTEM REFERENCE MANUAL
.SKIP 16
.CENTER &OASIS&
.SKIP 2
.CENTER &System Reference&
.CENTER &Manual&
.EJECT
.NOFILL
First printing: February, 1978
Revised:          April, 1978
                   June, 1978
                Oct



Sector number: 252
Is this the correct file (Y/N)? N
End of disk

>
```

CHAPTER 10

RELOCATE COMMAND

The RELOCATE command converts two absolute command files (one program - two copies with different origin addresses) into a relocatable program. The format of the command is:

RELOCATE <fn1> <fn2> [<fn3>]

Where:

<fn1> Is the program name of an absolute command.

<fn2> Is the program name of the same command but origined at a different address.

<fn3> Is the name to be used for the relocatable command generated. When this parameter is not specified <fn1> will be used, replacing the original absolute program.

When the RELOCATE command is executed the two absolute programs are compared, any differences are assumed to be relocation differences. Using the information gathered from the comparison a relocatable program is generated.

This command is quite useful when the program source code is not available. This is frequently the case when converting from another operating system.

RELOCATE Example:

```
>RELOCATE PROG1 PROG2 GAME

>
```

CHAPTER 11

REPAIR COMMAND

The REPAIR command provides a means of determining and, optionally correcting a disk's file allocation. The format of the command is:

REPAIR <fd> [(<option> [)]]

Where:

<fd> Indicates the file disk to be checked or fixed.

REPAIR option

FIX Indicates that an attempt is to be made to correct any errors discovered by the program.

The REPAIR command searches through the disk's directory sequentially, displaying the file name and owner id number for each file in the directory. During this process the allocation of the file is saved and checked against previous file's allocation. Collisions between files are noted (not corrected) and mis-allocated files are noted (not corrected). A sequential file whose length does not match it's directory entry is noted (not corrected).

At the end of the REPAIR process any mis-allocated disk space is noted and may be corrected if the FIX option was specified.

Note: Any errors detected by this program should be analyzed for their cause before they are repaired. If it is determined that the error was caused by a system program then the symptoms should be reported to your OASIS distributor and you should not use that program until a correction is made. Keep in mind that most frequent cause of a disk error is a loss of power during disk access or a static charge.

If this program reports a collision between two files the recommended method of fixing the problem is to copy (not backup) the files from the disk with the error to an empty disk. This will remove the collision problem but one or both of the files may not contain valid information. Again, the cause of the problem should be determined before the data and programs are used.

REPAIR Example:

```
>REPAIR A (FIX

File 1 CSIFUNCT.SYSREF5
File 2 MEMORY.SYSREF5
File 3 SET.SYSREF5
File 4 ANSIFORM.SYSREF5
  .
  .
  .
File 45 INDSIZES.SYSREF5
Sector 252 is used but not allocated
Sector 256 is used but not allocated
Sector 428 is allocated but not used
Sector 440 is allocated but not used
Sector 708 is used but not allocated
Sector 1196 is used but not allocated

>
```

CHAPTER 12

SECTOR COMMAND

The SECTOR command provides a means of determining the disk sectors used by a file. The format of the command is:

SECTOR <file-desc>

Where:

<file-desc> Is the file to be analyzed. The complete file description is required--no directory search will be performed if the file disk is omitted.

It is sometimes necessary to know the sector numbers used by a file. Since indexed, direct, absolute, and relocatable files require contiguous disk space it is easy to determine the sector numbers used by these files (use the asterisk option of the FILELIST command). On the other hand, a sequential file is a linked file that does not require contiguous disk space. The SECTOR command is the only easy way to determine the sector numbers used by this type of file.

When the SECTOR command is executed the sector numbers used by the file will be displayed on the screen, in the sequence that they are used by the file, four sector numbers per line. The last sector number used in a sequential file has a forward link of zero.

SECTOR Example:

```
>SECTOR HELP EXEC S
1340    1341    1342    1343
1344    1345    1346    1347
1348    1349    1350    1351
1352    1353    1354    1355
1356    1357    0

>SECTOR DATA FILE A
96      97      98      99
100     101     102     103
500     501     502     503
696     697     698     699
700     701     702     703
704     705     706     707
708     709     710     711
712     713     714     715
716     717     718     719
720     721     722     723
724     725     726     727
728     729     730     731
732     733     734     735
736     737     738     739
740     741     742     743
744     745     746     747
832     833     834     835
836     837     838     839
864     865     0
```

CHAPTER 13

SEEK COMMAND

The SEEK command performs a test of a disk drive's sector seek capability. The format of the command is:

SEEK [<fd>]

Where:

fd Is the file disk that the seek test is to be performed on.

When the SEEK command is executed a random, valid sector number is generated and displayed on the console. The specified drive is then instructed to seek and read that sector. Any errors reported by the disk drive will be reported to the operator on the screen. The error report will consist of the error number only. For a definition of these error number see the appendix "Error Messages" in the OASIS System Reference Manual.

After the sector is read the operation (generate sector number and seek it) is repeated until the operator cancels the program with the System Cancel-key.

SEEK Example:

```
>SEEK A
245,7
1200

>
```

CHAPTER 14

VERIFY COMMAND

The VERIFY command allows you to verify the readability of a disk quickly. The format of the command is:

VERIFY <fd>

When the VERIFY command is executed the specified disk is read one track at a time. As each track is read the track number is displayed on the console. Any disk errors detected are displayed on the console and the track is reread until the retry count is exhausted. At this time the track number is incremented and the verify process continues.

If the retry count is exhausted and the track was not successfully read then the message "unrecoverable" is displayed.

It is advisable to verify a disk whenever time permits, especially after a disk has been initialized (INITDISK command) or copied (BACKUP command). If the VERIFY command reports any unrecoverable errors on a disk or consistently reports recoverable errors on the same track of a disk then you should replace the disk with a new one, copying as much of the data as is possible.

VERIFY Example:

```
>VERIFY A

Track: 36 4,4,4
Track: 55 7,7,7,7,7,7,7,7 Unrecoverable!
Track: 76

>
```

The above example illustrates the verification of a disk that has a "soft" data error on track 36 (able to read the data after three tries) and a "hard" sector address error on track 55 (could not successfully read the data after eight tries--the current retry count). All other tracks on the disk were successfully read on the first attempt.