Sydex

AnaDisk

The Compleat Diskette Utility

Sydex

153 North Murphy Ave.
Sunnyvale, CA 94086
(408) 739-4866

AnaDisk

The Compleat Diskette Utility

Version 2.01, November, 1989

NOTICE

THIS IS NOT FREE SOFTWARE! If you paid a "public domain" vendor for this program, you paid for the service of copying the program, and not for the program itself. Rest assured that *nothing* ever gets to the originators of this product from such a sale. You may evaluate this product, but if you make use of it, you must register your copy.

We offer several inducements to you for registering. First of all, you receive the most up-to-date copy of the program that we have -- and we do update the product on a regular basis. You also receive support for **AnaDisk** -- which can be quite valuable at times. You also receive complete *printed* documentation for the product. A "do-it-yourself" update service is offered to registered users through our own BBS. And finally, we include an evaluation package of some of our other software products. Make no mistake, however--this is a fully functional version of **AnaDisk** and not "crippled" in any way. As a final inducement to you--the registered package does not contain the advertising copy...

REGISTRATION INFORMATION

The non-commercial single-user registration fee for **AnaDisk** is $25.00 US. The commercial and multisystem site fee for **AnaDisk** is $150.00. Users outside of Canada and the United States should include an additional $5.00 for international airmail. *We can accept payment only in U.S. funds.*

Send a check or company P.O. for the appropriate amount to:

Sydex
153 North Murphy Ave.
Sunnyvale, CA 94086

We can also accept VISA or Master Charge; contact us for details. On corporate orders, our terms are normally net 30. Please indicate what product you are ordering and if you have a requirement for 3.5" media. We normally ship 5.25" 360K diskettes, but will furnish 3.5" 720K media upon request.

If you are a registered user of this product and desire an update, please send $4.00 for handling and we will ship you an up-to-date copy. Please add $3.00 for printed documentation to cover duplication costs.

DISTRIBUTION NOTICE

This is "user-supported" software. You are hereby granted a license by Sydex to distribute this evaluation copy of **AnaDisk** and its documentation, subject to the following conditions:

1. **AnaDisk** may be distributed freely without charge in evaluation form only.

2. **AnaDisk** may not be sold, licensed, or a fee charged for its use. If a fee is charged in connection with **AnaDisk**, it must cover the cost of copying or dissemination only. Such charges must be clearly identified as such by the originating party. Under no circumstances may the purchaser be given the impression that he is buying **AnaDisk** itself.

3. **AnaDisk** must be presented as a complete unit, including this documentation. Neither **AnaDisk** nor its documentation may be amended or altered in any way.

4. By granting you the right to distribute the evaluation form of **AnaDisk**, you do not become the owner of **AnaDisk** in any form.

Any other use, distribution or representation of **AnaDisk** is expressly forbidden without the written consent of Sydex.

Table of Contents

Overview

AnaDisk is a utility for examining, editing and analyzing diskettes. It performs the following functions:

* Analyze diskettes for content and consistency. The density and format of diskettes are automatically determined; any format changes or media errors are noted. In the case of DOS diskettes, checks are made to ensure that a diskette complies with generally accepted DOS implementation practice.
* Search diskettes for text. Both case-sensitive and case-insensitive searches may be performed, as well as search keys with "wild card" or "don't care" positions. Both the active data areas, as well as the inactive or erased data areas may be searched.
* Examine and print data on a physical sector, as well as on a file basis. Either ASCII or hexadecimal displays may be used.
* Copy a diskette without regard to format or type.
* Modify data on a diskette.
* Repair DOS diskettes containing data errors.
* Format a diskette according to custom specifications.
* Copy an area of a diskette to a DOS file.

AnaDisk features a menu-driven "windowed" presentation. Extensive context-sensitive on-line help is available.

AnaDisk requires an IBM PC or PS/2 compatible computer for operation. Certain computers, such as the Tandy 1000, 2000 or the IBM PC Jr., are not sufficiently compatible to support **AnaDisk.**

At least 512K of memory and a hard disk are recommended for **AnaDisk** operation, although some function may be obtained with as little as 384K on a diskette-only system.

Installing AnaDisk

The software for **AnaDisk** is contained in two programs. The first, the installation program, called **ADINSTAL**, is used to determine the diskette configuration of the computer being used. This installation program modifies the second part, the file **ANADISK.EXE**, with the diskette configuration information.

```
                   AnaDisk INSTALLATION PROGRAM Ver. 2.01

                 Copyright 1989, Sydex.  All Rights Reserved.


This program installs AnaDisk.

Installation is a one-time task; you should not need to re-run this
program unless you have changed your equipment configuration.

Installation may be performed with a single diskette, or any combination
of diskettes and hard disks.  The most common installation is from a
diskette to a hard disk.  During this installation, you will be asked
to provide a copy of the AnaDisk files.  If you don't have these
files, press ESCape to exit now.  This installation procedure selects
diskette drive types and processor type.  In addition, the installed
AnaDisk is copied to a destination drive of your choice.



If you do not wish to install AnaDisk at this time, press ESCape.
Press any other key to continue...
```

Figure 1: ADINSTAL Opening Screen

AnaDisk must be installed prior to use. After installation, **AnaDisk** need not be re-installed unless the configuration of the computer being used is changed. It is possible to re-install an already installed copy of **AnaDisk**.

To install **AnaDisk**, first load DOS. Insert the **AnaDisk** distribution diskette into diskette drive **A:**, then enter the following command:

A:ADINSTAL *and press the ENTER key.*

The display in Figure 1 will appear. Press the **ENTER** key to continue with the installation process.

Prompts will appear for the source and destination disk drives. **AnaDisk** may be re-installed over itself, if need be. Otherwise, the source copy of **AnaDisk** is not modified by the installation software.

After **AnaDisk** has been read from the specified source drive, the display in Figure 2 appears. **ADINSTAL** groups PC-Compatibles into those having a PC XT-type of architecture, or those having a PC AT-type. However, this division is not always clear, particularly in the case of an XT-class PC with an add-in high-density diskette controller. These are provided by a number of vendors for enhancement purposes, and their presence requires that the PC be declared as an AT-style computer.

```
Although your computer may be an XT-type system, you may wish to force
it to be treated as an AT-class machine if you have a high-density
diskette controller.

On the other hand, if you are using a Toshiba 3100, you probably need
to indicate that it's an XT-type machine to get the diskette drives
to operate correctly.

You have a PC AT, IBM PS/2 or compatible.


Is this correct? (Y or N)
```

Figure 2: ADINSTAL Computer Selection

If permitted, **ADINSTAL** will attempt to determine the diskette configuration of the computer. In any case, a display similar to the following screen (Figure 3) is shown:

```
                         DISKETTE CONFIGURATION

 The following diskettes are present on your computer:


    NO. DRIVE   UNIT   ADAPTER     DRIVE TYPE       STEP RATE


     1.   A:     0     PRIMARY    1.2M 5.25"          6 msec.
     2.   B:     1     PRIMARY    1.44M 3.5"          6 msec.
     3.   (NONE DEFINED)
     4.   (NONE DEFINED)
     5.   (NONE DEFINED)
     6.   (NONE DEFINED)
     7.   (NONE DEFINED)
     8.   (NONE DEFINED)

     Are there any changes? (Y or N)
```

Figure 3: Diskette Configuration Display

Information contained on this display may be changed until the data reflects the actual configuration of the host computer. **DRIVE** is used by **AnaDisk** to refer to a diskette drive. While it is advisable that the drive letter be the same as that used by DOS to refer to that drive, any letter can be used. **UNIT** refers to the physical unit number of a drive. Some diskette adapters are able to access only two drives, physical units 0 and 1. Other adapters, particularly those of the XT variety, can access up to four drives, physical units 0, 1, 2 and 3.

If a second diskette adapter has been added to the host computer, it will be necessary to configure units attached to that adapter as **Secondary**. Note that, on secondary adapters, the physical unit numbering starts over again with unit 0.

The drive **step rate** refers to how quickly the positioning mechanism in a drive is able to move the read/write heads from cylinder to cylinder. In almost all instances of 5.25" drives, the default of 6 milliseconds will be satisfactory. This value sometimes needs to changed for some of the older 8" diskette drives.

AnaDisk does not explicitly support dual-speed 5.25" drives. A dual-speed drive is one that rotates the media at 300 RPM for low-density recording and 360 RPM for high-density recording. If it is desired to use a dual-speed drive, it must be declared as two separate drives, a high-density drive, and a low-density drive, using two different drive letters.

After the diskette drives have been configured for **AnaDisk**, the installed program file is written to the specified destination path and drive, and **AnaDisk** is ready for use.

Running AnaDisk

To start **AnaDisk**, simply enter the following at the DOS prompt:

ANADISK

and press the **ENTER** key. **AnaDisk** will normally adjust its video display routines to make use of the display adapter in use. However, if the display adapter is of a color type and the display monitor itself is monochrome, it will be necessary to start **AnaDisk** with the following command to get a readable display:

ANADISK M

AnaDisk has many displays and functions, but there are certain keys which *always* have the same effect during **AnaDisk** operation:

F1 is the **Help** key. It may be pressed any time additional information is required. The help displays in **AnaDisk** can be characterized as being *Context-Sensitive*. That is, only information relevant to the current operation is presented. For example, if **AnaDisk** were prompting for a file name, **F1** would produce a display that presented information about the application of the file being requested.

ESCape is the **Exit** key. It may be pressed at any time to terminate the operation in progress. Unless otherwise noted, pressing **ESCape** will cause **AnaDisk** to return to the display that preceded the current one. If **ESCape** is pressed a sufficient number of times, **AnaDisk** will exit to DOS.

When an abnormal condition has been detected in **AnaDisk**, a small window describing the condition will appear. This window is referred to as the "Alert Box":

```
═══════↑↓→← to Move═══════
Drive A: is not ready.
Please correct and press
any key to continue.
```

The Alert Box may be moved around the screen by means of the cursor keys. Up- and down-cursor move the Alert Box up or down one character row, respectively. Right- and left-cursor have a similar effect. The **Home** key moves the Alert Box to the upper left corner of the display; **End** moves it to the lower left. **PgUp** moves the Alert Box to the upper right corner of the display and **PgDn** moves it to the lower right.

After **AnaDisk** is loaded by DOS, the following display (Figure 4) appears:

```
ESC to Exit                    AnaDisk 2.01                    F1 for Help

                                   Select - ESCape to quit, F1 for help

                             MAIN MENU

        Select function with ←→ cursor keys.  ENTER (←┘) confirms
        your choice and goes to next menu.  F1 gets help, ESCape exits
        to DOS.

         SCAN  SECTOR  FILE  SEARCH  COPY  REPAIR  FAT  FORMAT  DUMP
```

Figure 4: AnaDisk Opening Display

If the **ESC**ape key is pressed, **AnaDisk** exits to DOS. Otherwise, the cursor right- and left-arrow keys are used to position to highlight, and thereby select, the desired function. After a function has been selected, pressing the **ENTER** key will cause a menu specific to the particular function to be displayed.

The functions performed by **AnaDisk** are as follows:

The Scan function reads an entire diskette and points out any problems or inconsistencies. A quick look is taken at all of the files on the diskette and checks are made to ensure that file contents are consistent with the file type. A "log" of the activity can be printed, if desired.

Sectors provides a diskette editing function, operating on a sector-by-sector basis. Diskette data can be printed, displayed or changed.

Files provides a facility to examine file data on the basis of file name, rather than physical diskette sector addresses. It is possible to "walk" the directory tree and print or display file data.

Search provides a facility to search for data on a diskette. A number of search key values can be specified and these may contain "wild-card" or "don't care" values. The results of the search can be displayed or printed.

Copy provides a disk-to-disk copying function. A "true" copy of an entire diskette is produced within the limits of the PC diskette adapter hardware. It is not necessary to pre-format the target diskette, but it is advisable to write-protect the source diskette.

Repair scans a DOS diskette for data errors and attempts to mitigate their effect by "moving" the contents of erroneous sectors to other areas of the diskette.

FAT provides a DOS **F**ile **A**llocation **T**able editor.

Format supplies a custom diskette formatting capability. This is a feature intended for advanced users.

Dump copies specified areas of a diskette to a DOS file. The diskette being copied need not be a DOS diskette.

As is the case throughout **AnaDisk**, help may be obtained by pressing **F1**.

Scan Operation

When the **Scan** function is selected, the display shown in Figure 5 appears. The choice for each item in the menu can be selected by positioning the highlighted area using the right- and left-cursor keys. The up- and down-cursor keys select the desired menu item. When the desired selections have been made, the **ENTER** key is pressed to begin processing **Scan** mode.

```
ESC to Exit                         AnaDisk 2.01                          F1 for Help

                                     Select - ESCape to quit, F1 for help

                                Scan DISKETTE

    Select choices with ↔ cursor keys.  Move between lines with ↑↓
    keys.  ENTER (←┘) begins execution.  F1 gets help, ESCape
    goes back to the opening menu.

        Diskette Unit                              A:  B:

        Printer output                                 NO  YES

        Pause after anomalies                          NO  YES
```

Figure 5: Scan Menu

The **Scan** function reads an entire diskette and attempts to discover data errors or inconsistencies. **AnaDisk** first determines the layout of cylinders, tracks and sectors, then attempts to classify the diskette according to operating system type. If the diskette appears to have a DOS format, a validity check is made of the File Allocation Table (FAT). Finally, every sector of the diskette is read, and errors and format changes noted.

If **Pause after anomalies** on the **Scan** menu is selected, some information is presented by opening an additional window which overlays the display and temporarily suspends **AnaDisk** operation until acknowledged from the keyboard. If **Pause after anomalies** is not selected, important information is written in the **ANALYSIS** window, but operation of **AnaDisk** is not suspended.

If **Printer output** is selected, a running log of analysis information is printed on the default printer (the DOS **PRN:** device).

A normal **Scan** function operating display is shown in Figure 6. The "thermometer bar" across the top of the screen indicates where **AnaDisk** is positioned on the diskette. The small arrow on the display points down if the first side of a diskette is being accessed, or up for the second side. In addition, position is also shown in the small window in the upper-left part of the display. A progress message is displayed in the small window in the upper-right part of the display.

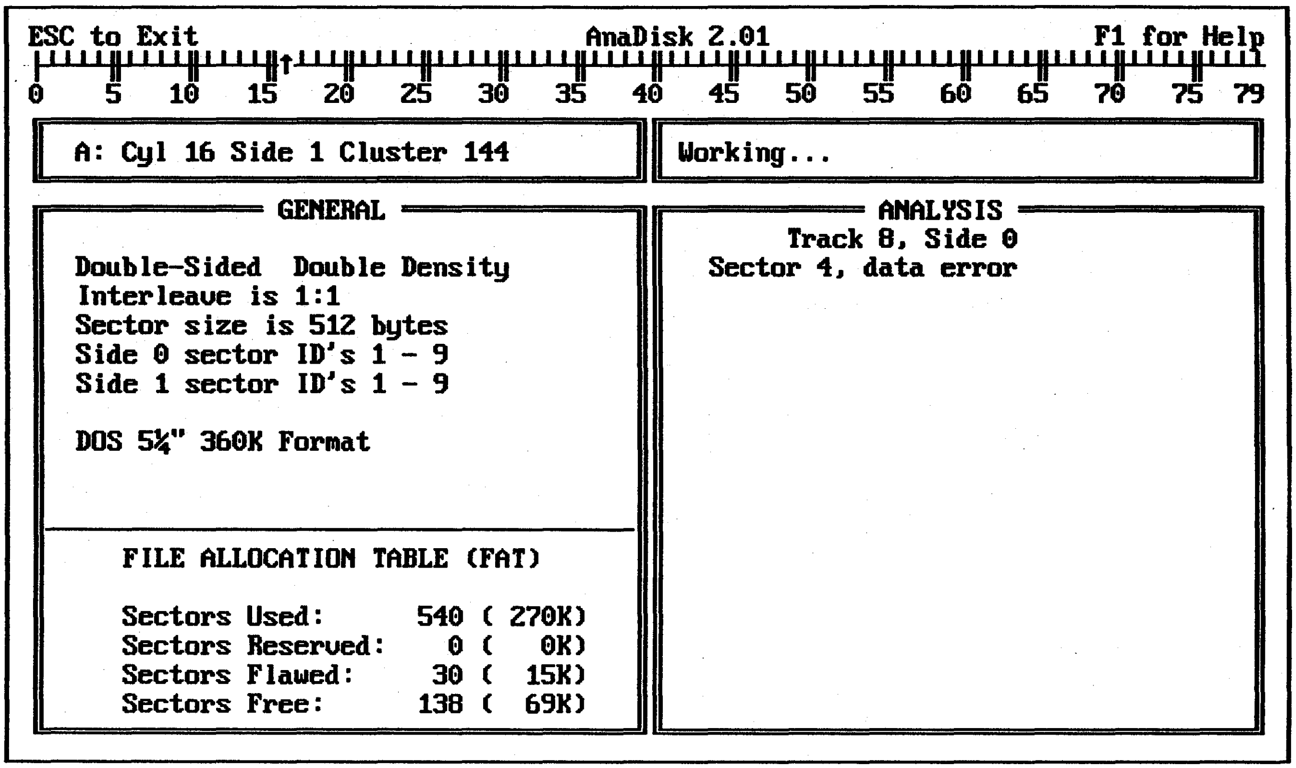


Figure 6: Scan function operating display.

The window labeled **GENERAL** contains information about the static or unchanging aspects of the diskette under examination, such as number of sectors per track and File Allocation Table information. The window labeled **ANALYSIS** describes events of a dynamic nature, such as read errors and anomalies in diskette structure. This window uses a scrolling display; that is, the oldest information is replaced by the newest.

For DOS diskettes with readable file allocation information, the lower part of the **GENERAL** window presents the data contained in the File Allocation Table (**FAT**) on the diskette.

One of the items of information that **AnaDisk** displays during **Scan** operation is shown in Figure 7. This shows information contained in the first sector of a DOS diskette, which will be used in performing file structure analysis of the diskette.

```
BOOT SECTOR INFORMATION

System Name.................IBM  3.3
Bytes/Sector................512
Sectors/Cluster.............2
Reserved Sectors............1
Number of FATs..............2
Root Directory Entries......112
Sectors on Diskette.........720
Media Byte..................fd
Sectors/FAT.................2
Sectors/Track...............9
Number of Sides.............2
Other Reserved Sectors......0
```

Figure 7: Scan function Boot Sector display.

If the number of sides declared by the DOS Boot Sector does not match that actually detected by **AnaDisk**, one of the following messages appears:

```
↑↓→← to Move
This is a double-sided DOS
format, but only one side
has data.  Press any key
to resume...
```

```
↑↓→← to Move
This is a single-sided DOS
format, but both sides
have data.  Do you want to
check both sides (Y or N)?
```

The **Scan** function can detect media errors. A few of the more common error messages displayed in the **ANALYSIS** window are:

`Data Error`

Meaning: Information has been read from the diskette, but internal checks made by the controller indicate that the data transferred is suspect.

`ID but no Data Found`

Meaning: The marker identifying the beginning of a sector is present, but not the marker that signifies that data follows. No data is transferred by the diskette controller.

`Sector Missing`
`Gap in Addresses`

Meaning: In DOS diskettes, sectors are numbered consecutively. This message indicates that one or more sectors could not be found in the normal numbering sequence. This message can also be indicative of some copy-protection schemes.

No Data on Track

Meaning: The diskette track may be blank or be written in some other recording mode, such as single-density or those modes used by Apple or Commodore. It is not possible to read this track with the standard PC-style diskette adapter.

Edit Sectors Operation

When the **Edit Sectors** operation is selected from the Main Menu, the display shown in Figure 8 appears.

The **Edit Sectors** function provides a facility to inspect, change and print diskette data on a sector-by-sector basis, as contrasted with **Examine Files**, which implements inspection of data within a specified DOS file.

The **Edit Sectors** sub-menu provides for the selection of a diskette unit.

```
ESC to Exit                      AnaDisk 2.01                      F1 for Help

                                  Select - ESCape to quit, F1 for help

                              EDIT SECTORS

    Select choice with ←→ cursor keys.  ENTER (←┘) begins
    execution.  F1 gets help, ESCape goes back to the opening menu.


        Diskette Unit                            A:  B:
```

Figure 8: Edit Sectors menu

Like the **Scan** function, **Edit Sectors** begins operation by determining the diskette type and drive characteristics. After this is done, the track and side containing the sector(s) to be inspected or modified can be selected.

The right- and left-cursor keys are used to select the track; the up- and down-cursor keys are used to select the head or side containing the sector. The "thermometer" line shows this position change correspondingly. When the **ENTER** key is pressed, the track is read for analysis and a track map is displayed as shown in Figure 9.

```
ESC to Exit                        AnaDisk 2.01                         F1 for Help
0    5   10   15   20   25   30   35   40   45   50   55   60   65   70   75  79

 A: Cyl 1 Side 0 Cluster 5              Select sector

                              TRACK MAP
 Select sector ↑↓, ENTER starts edit, Change track ↔, Exit ESC

 CYL HEAD SECT SIZE FLAGS
   1    0    1  512
   1    0    2  512
   1    0    3  512
   1    0    4  512
   1    0    5  512
   1    0    6  512
   1    0    7  512
   1    0    8  512
   1    0    9  512
```

Figure 9: Edit Sectors track map

The desired sector is selected from the displayed track map by use of the up- and down-cursor keys. When the sector to be edited is highlighted, the **ENTER** key is pressed to display the data for that sector. If either the right- or left-cursor key is pressed, the next or previous track is analyzed, and the track map for that track displayed. If the **ESCape key** is pressed, **AnaDisk** returns to the track selection display.

Sectors having errors are marked in the track map with an **E** in the **FLAGS** column; those having a Deleted Data Address Mark are marked with the letter **M.**

When the desired sector has been selected, the editing display appears as shown in Figure 10. Initially, sector data is displayed in an ASCII representation, with undisplayable control characters shown as dots. The **F8** key may be used to toggle between this ASCII display and a mixed hexadecimal-ASCII display, as illustrated in Figure 11. The current position within the sector data is indicated by a highlighted reverse video block, and may be moved by use of any of the cursor keys. Data on the display may be changed by simply entering new data at the current position. *However, data is not written to the diskette until Alt-F2 is pressed.* If data is changed, the notation **ALTERED** appears in the menu window.

```
ESC to Exit                    AnaDisk 2.01                   F1 for Help

  F1   HELP          F2   Read Sect     F3   Print       F4   Disk Print    F5   Shift←
  F6   Shift→        F7   NOT           F8   Alpha       F9   Read Last     F10  Read Next
  F2S  Diag. Read    F3S  What File?    F4S  Save        F5S  Recall        F6S  Fill->End
  F9S  Track - 1     F10S Track + 1     F2A  Write       F3A  Reformat      ESC  QUIT

  Clust 3 Cyl 1 Side 0 Sect 9, Size 512                            0000(0)

 0000  RNT=0••1170 GOSUB 14310 'TOGGLE CAPSLOCK ON••1180 GOSUB 11700 'W
 0040  HAT KIND OF VIDEO ADAPTER••1190 GOSUB 11850 'display Soft-SHARE
 0080  logo••1200 GOSUB 12120 'REQUEST RUNTIME OPTIONS••1210 START$="".
 00c0  •1220 FINISH$=""••1230 S=1••1240 ROWLINES=10••1250 PAGESIZE=50••
 0100  1260 FREELMT=10••1270 LINECNT=0••1280 PRVLNUM=-32768!••1290 CURL
 0140  INE!=0••1300 '••1310 IF RUNTYPE<3 GOTO 1470••1320 IF RUNTYPE>4 G
 0180  OTO 1470••1330 READ RESCNT••1340 DIM RESWRD$(158) ' RESWRD$(153)
 01c0      FIXED IF COMPILED••1350 DIM WORDFLAG(158) ' WORDFLAG(153)
```

Figure 10: ASCII Sector display

```
 0000  52 4e 54 3d 30 0d 0a 31 31 37 30 20 47 4f 53 55   RNT=0→→1170 GOSU
 0010  42 20 31 34 33 31 30 20 27 54 4f 47 47 4c 45 20   B 14310 'TOGGLE
 0020  43 41 50 53 4c 4f 43 4b 20 4f 4e 0d 0a 31 31 38   CAPSLOCK ON→→118
 0030  30 20 47 4f 53 55 42 20 31 31 37 30 30 20 27 57   0 GOSUB 11700 'W
 0040  48 41 54 20 4b 49 4e 44 20 4f 46 20 56 49 44 45   HAT KIND OF VIDE
 0050  4f 20 41 44 41 50 54 45 52 0d 0a 31 31 39 30 20   O ADAPTER→→1190
 0060  47 4f 53 55 42 20 31 31 38 35 30 20 27 64 69 73   GOSUB 11850 'dis
 0070  70 6c 61 79 20 53 6f 66 74 2d 53 48 41 52 45 20   play Soft-SHARE
 0080  6c 6f 67 6f 0d 0a 31 32 30 30 20 47 4f 53 55 42   logo→→1200 GOSUB
 0090  20 31 32 31 32 30 20 27 52 45 51 55 45 53 54 20    12120 'REQUEST
 00a0  52 55 4e 54 49 4d 45 20 4f 50 54 49 4f 4e 53 0d   RUNTIME OPTIONS→
 00b0  0a 31 32 31 30 20 53 54 41 52 54 24 3d 22 22 0d   →1210 START$=""→
 00c0  0a 31 32 32 30 20 46 49 4e 49 53 48 24 3d 22 22   →1220 FINISH$=""
```

Figure 11: Mixed Hexadecimal-ASCII display.

Several function key combinations will specialized functions, as follows:

F1 obtains a brief Help display.

F2 Re-reads the current sector. This function may be needed if data in the sector display has been modified, and it is desired to bring back the original data.

F3 Prints the current sector on the DOS printer (**PRN:**). A mixed hexadecimal-ASCII format is used.

F4 Writes a mixed hexadecimal-ASCII representation of the sector data to a standard DOS file. If the DOS file exists prior to the **F4** operation, the data is appended to the end of the file. Otherwise, a new file is created.

F5 Shifts the data from the current (highlighted) position to the end of the sector one bit left. Bits shifted out of the high-order positions of the current byte are lost; low order positions at the end of the current sector are filled with zero.

F6 Shifts the data from the current (highlighted) position to the end of the sector one bit right. Bits shifted out of the low-order positions of the last byte in the sector are lost; high-order bit positions at the current byte are filled with zero. The shift-right and -left functions are useful in reconstructing information in a sector which has become garbled because of a data read error.

F7 Performs the one's complement Boolean operation on sector data from the current (highlighted) position to the end of the sector.

F8 Toggles the display mode from ASCII to mixed ASCII-hexadecimal.

F9 Reads the sector immediately preceding the current one. If the current sector is the first on the track, the last sector on the previous track is read.

F10 Reads the sector immediately following the current one. If the current sector is the last on the track, the first sector on the next track is read.

Shift-F2 Performs a "diagnostic read" of the current track. A diagnostic read involves reading the data field of the first sector and continues with all fields until 16,384 bytes have been read. ID fields, gap bytes and CRCs are read indiscriminately; no attempt to re-synchronize the data discrimination logic is made after the first sector has been read. This function allows viewing of raw data and may be used to determine the value of a data field whose address ID field has been corrupted, or data contained within inter-sector gaps. See the Diskette Tutorial section for more information.

Shift-F3 Identifies the file of which the sector is a part. Note that is function is available only for DOS diskettes.

Shift-F4 Saves the contents of the sector in an internal buffer. Each time **Shift-F4** is depressed, the previous contents of the save buffer are lost.

Shift-F5 Recalls the contents of the save buffer to the current display. Note that the contents of the buffer are not written to the diskette.

Shift-F6 Fills the sector from the current position to the end of the sector with the value at the current position.

Shift-F9 Analyzes the preceding track and displays the track map for it. If the current track is the first on the diskette, it is re-read.

Shift-F10 Analyzes the following track and displays the track map for it.

Alt-F2 Writes the contents of the editing display back to the diskette.

Alt-F3 First reads all sectors on the current track, then re-formats the current track with a fresh format pattern, then re-writes the sectors back to the track. A display requesting confirmation appears before the track is reformatted.

ESCape Returns to the track map for this track.

In addition to this information, the menu window also shows the current sector, the position within the sector in both decimal and hexadecimal and the following flags, if applicable:

> **ERR** The sector could not be read without error. Before an attempt is made to read a sector, the data is set to all zero. If the error is one that results in no data being transferred, the display will show all zero.
>
> **CTL MK** A Deleted Data ID Address Mark was detected on the current sector. DOS makes no use of this feature and never writes this flag, but **AnaDisk** does detect its presence and reports it.

Examine Files

The **Examine Files** function provides for examination of data contained in DOS files. After the **Examine Files** function has been selected from the Main Menu, the diskette unit to be examined is selected from a subsidiary menu similar to that used for **Edit Sectors.**

The main **Examine Files** display presents a list of file names contained in the root directory of a diskette. The cursor up- and down-arrow keys are used to select the highlighted file or directory name to be examined; the file name display can be "paged" by means of the Page Up (**PgUp**) or Page Down (**PgDn**) keys. Figure 12 shows a sample display.

In the window displaying file names, the meanings of the column labels are as follows:

NAME is the name of the file.

ATTRIB show flags for the file attributes. These are as follows:

A is the "archive" flag. This flag is used by back-up utilities and is set whenever the file is created or modified. Most back-up utilities clear this flag once the file has been backed up.

D is the "directory" flag. When this flag is present, the name shown is the name of a directory, rather than a file.

V is the "volume label" flag. When this flag is present, the name shown is the name of the volume label for this diskette. This name can be changed by means of the DOS **LABEL** utility.

H is the "hidden" flag. When this flag is present, the file name will not be shown in a DOS directory listing.

R is the "read only" flag. When this flag is set, DOS will not permit the file to be written to. This flag is modified by means of the DOS **ATTRIB** command.

S is the "system" flag. This flag is normally only associated with DOS files loaded at system "boot" time. Files flagged with this attribute are also implicitly read-only and hidden (**R** and **H** attributes).

To examine the files in a particular subdirectory, highlight the subdirectory name and press **ENTER**. To go back to the parent directory of a subdirectory, position to the **..** entry and press **ENTER**.

Several functions are available on the **Examine Files** menu:

F1 obtains a brief Help display.

F2 causes the **Edit Sectors** display to be activated, using the first sector of the current (highlighted) file. All functions on the **Edit Sectors** menu operate normally, with the following exceptions:

F9 moves one sector back *within the file.*

F10 moves one sector forward *within the file.*

Shift-F9 and **Shift-F10** do nothing.

```
ESC to Exit                    AnaDisk 2.01                         F1 for Help

        F1 Help              F2 Examine Sector            F3 Display File
        F4 Show Deleted      F5 Print ASCII               F6 Print Hex
        F7 Edit FAT          ENTER Select Directory       ESC Exit

  Attributes:  Archive  Directory  Volume label  Hidden  Read only  System

 Directory: \

          NAME        ATTRIB    TIME       DATE      LENGTH   CLUS

       MKCHARS.C        A    14:17:21   11/16/89      1195      2
       TEXT2DAT.C       A    14:17:21   11/16/89       993      4
       IMGCPY.C         A    14:17:21   11/16/89      6058      5
       SAVECMOS.C       A    14:17:22   11/16/89      4443     11
       PUTCMOS.C        A    14:17:22   11/16/89      5010     16
       CGEN2ASM.C       A    14:17:24   11/16/89      1127     42
       CGEN2HP.C        A    14:17:24   11/16/89      8594     44
       PRUNE.C          A    14:17:24   11/16/89      4003     53
```

Figure 12: Examine Files display

ESCape returns to the **Examine Files** menu, not to the track map.

In addition, the relative byte offset within the file is displayed, in addition to the cylinder, side and sector.

F3 displays the currently highlighted file as ASCII text. The cursor-up, cursor-down, **PgUp**, **PgDn**, **Home** and **End** keys may be used to navigate within a file. Data past the end-of-file point is displayed following the notation:

```
>>> END-OF-FILE <<<
```

F4 toggles the file directory display between **Active** and **Deleted** or erased files. A deleted file always has a "sigma" as the first character of the name. Figure 13 shows a deleted files display.

F5 causes the currently highlighted file to be printed as ASCII text. Data past the end-of-file point is also printed, with the end-of-file shown as in the **F3** function, above.

F6 causes the currently highlighted file to be printed as a mixed ASCII-hexadecimal representation.

ENTER changes the display to the highlighted directory. To return to the parent of the current directory, highlight the .. entry and press **ENTER.**

```
ESC to Exit                    AnaDisk 2.01                       F1 for Help

 F1 Help          F2 Examine Sector F3 Display File    F4 Show Active
 F5 Print ASCII   F6 Print Hex      ENTER Select Dir.  ESC Exit

         Attributes:     Archive  Directory  Volume label
                         Hidden   Read only  System

  Directory: \

         NAME       ATTRIB    TIME      DATE      LENGTH   CLUS

      σNADISK.EXE      A   15:16:01  05/16/89     71313     68
      σRCHECK.EXE      A   11:55:20  06/29/89      5084    302
      σNSTALL.EXE      A   10:36:05  09/07/89         0      0
```

Figure 13: Deleted files display.

It is possible to display and print data belonging to deleted files, if the file has not already been overwritten. If an attempt is made to display or print an overwritten file, the following message will be displayed:

```
==========↑↓→← to Move==========
 This deleted file has
 been overwritten.  Press
 ESCape to quit display,
 Any other key to proceed...
```

If the **ESCape** key is pressed, **AnaDisk** returns to the file directory display. If, however, any other key is pressed, **AnaDisk** displays information as if the file were not overwritten. To gather data for display, **AnaDisk** searches (forward) for the next unallocated area and displays it. Depending on the method by which the file was created, however, *this may not actually reflect data contained in the file beyond the first allocation unit or cluster.*

Search

AnaDisk's **Search** function will examine a diskette for a match on any one of a series of search key values, referred to simply as **keys**. The search can extend over the entire diskette, the active file area, or just the deleted (inactive) area. Up to five 65-character keys may be specified; **AnaDisk** will search for all keys simultaneously and report a match on <u>any</u> key.

Keys may contain what are known as "**don't care**" or "**wild card**" values, which will match *any* character. To obtain the most efficient performance, a key should not begin with a "don't care" value. In addition, the search may be instructed to ignore the case of alphabetic data; e.g., the letter **A** will match both **a** and **A**.

The **Search** menu is shown in Figure 14. The **Source for search value** selection offers two alternatives:

> **KEYBOARD** specifies key entry from the keyboard. After all key values are entered, they may be saved to a file for later recall.
>
> **OLD VALUES** specifies that the keys entered for the preceding search should be used. This option is invalid for the first search of a **AnaDisk** session.

Search alpha case-sensitive specifies whether the search should consider the case of alphabetic characters. If **NO** is selected, the search considers both upper- and lower-case alphabetic characters to be equivalent. Thus, **DOG** will match both **DOG** and **dog** if the search is *not* case-sensitive.

```
ESC to Exit                         AnaDisk 2.0                          F1 for Help

                                       Select - ESCape to quit, F1 for help

                           SEARCH DISK FOR DATA

    Select choices with ↔ cursor keys.  Move between lines with ↑↓
    keys.  ENTER (◄─┘) confirms.  ESCape returns to the main menu.

        Diskette Unit                              A:  B:

        Source for search values              KEYBOARD  OLD VALUES

        Search alpha case-sensitive                      NO  YES

        Search what part of the disk?       ALL  ACTIVE  DELETED

        Print hits without pausing                       NO  YES

        Masked Search                                    NO  YES
```

Figure 14: Search Menu.

Search what part of the disk? offers three alternatives. The first, **ALL**, specifies that the entire disk should be searched without regard to file boundaries or use of a sector. **ACTIVE** specifies that only the data areas belonging to files or their directories be searched; non-active (deleted or free) data areas will not be searched. **DELETED** specifies that only the deleted or unallocated areas of the disk should be examined.

Print hits without pausing instructs **AnaDisk** to record all search matches on the printer without stopping for verification. Normally, **AnaDisk** opens a window when a match is found, as shown in Figure 16, and requests further direction.

Masked Search refers to the optional specification of an 8-bit hexadecimal quantity which determines which bits in a byte are to be taken into consideration when a comparison is made. Each bit in the mask quantity that contains a "0" represents a "don't care" position. That is, bits in these positions will have no effect on the outcome of a comparison. Conversely, those bit positions containing "1" bits will be considered in comparisons. A mask quantity of all zeroes would represent no significant positions and so will not be allowed by **AnaDisk**.

```
ESC to Exit                       AnaDisk 2.0                        F1 for Help

                                   Enter Search Key - F1 for help

                          SPECIFY SEARCH VALUE

    Enter search value 1, 65 characters maximum.  Alt-X for Don't Care,
    Alt-V for verbatim values,  TAB changes hex/ASCII mode.  Press
    ENTER (◄┘) when done.  ◄┘ alone for last value.

                          ──── ASCII ────
   Magic Value█
                          ──── HEX ────
   4d 61 67 69 63 20 56 61 6c 75 65
```

Figure 15: Search Value Specification

When search keys are to be entered from the keyboard, the display in Figure 15 appears. Key values up to 65 characters may be entered; "don't care" values may be specified by the key combination **Alt-X**. "Don't care" values will appear as highlighted inverted question marks.

Data entry may be performed in either ASCII or hexadecimal modes. Initially, data entry begins in ASCII mode; the **Tab** key is used to alternate between modes and may be used at any time. Regardless of the mode used to enter search values, an appropriate representation appears in both the ASCII and the hexadecimal parts of the display as the values are entered.

Special non-printing characters can be entered in ASCII mode if first preceded by **Alt-V**, or *verbatim* key sequence. Unless it is part of a verbatim sequence, **ESCape** terminates data entry and return is made to the Main Menu.

Both the cursor-left and the backspace keys may be used to erase the most recently typed character. When a key has been completely entered, pressing the **ENTER** key will cause it to be stored and a prompt for the next key will appear. After the last key has been entered, pressing **ENTER** as the first character of the line will signal the end of key specification.

Up to five search keys may be specified; **AnaDisk** will report matches on **any** of the specified keys.

If use of a Search Mask has been indicated, the following is displayed:

```
↑↓→← to Move
Please enter hex
search mask value:

      █
```

The search mask value can then be entered. If a value of 0 is given, the following message is displayed:

```
↑↓→← to Move
A search mask value of
0 matches everything. Press
any key to respecify the
mask value.
```

A new search mask may then be entered, or, **ESCape** can be pressed, and a mask value of 255 (FF hex) or all 1's will be used.

After obtaining the search keys, **AnaDisk** reads each diskette track and searches it for a match on each key value in succession. If a matching value crosses (or "straddles") a track boundary, **AnaDisk** will report the value as occurring on the lower track. If a match is located, **AnaDisk** will then report the match if it occurs within the area of the disk that has been specified for searching, i.e., **ALL**, **ACTIVE** or **DELETED**.

It is important to observe that **AnaDisk** searches the diskette in physical sector order; that is, the first sector of the first side of the first cylinder to the last sector of the last side of the last cylinder, which is not necessarily the way data is organized within a given file.

Unless **Print hits without pausing** has been selected, each match is reported as shown in Figure 16. The value of the matching key is also displayed. If any positions of the key were entered in hexadecimal mode, the key will be displayed in hexadecimal notation. Otherwise, the key will be displayed in ASCII.

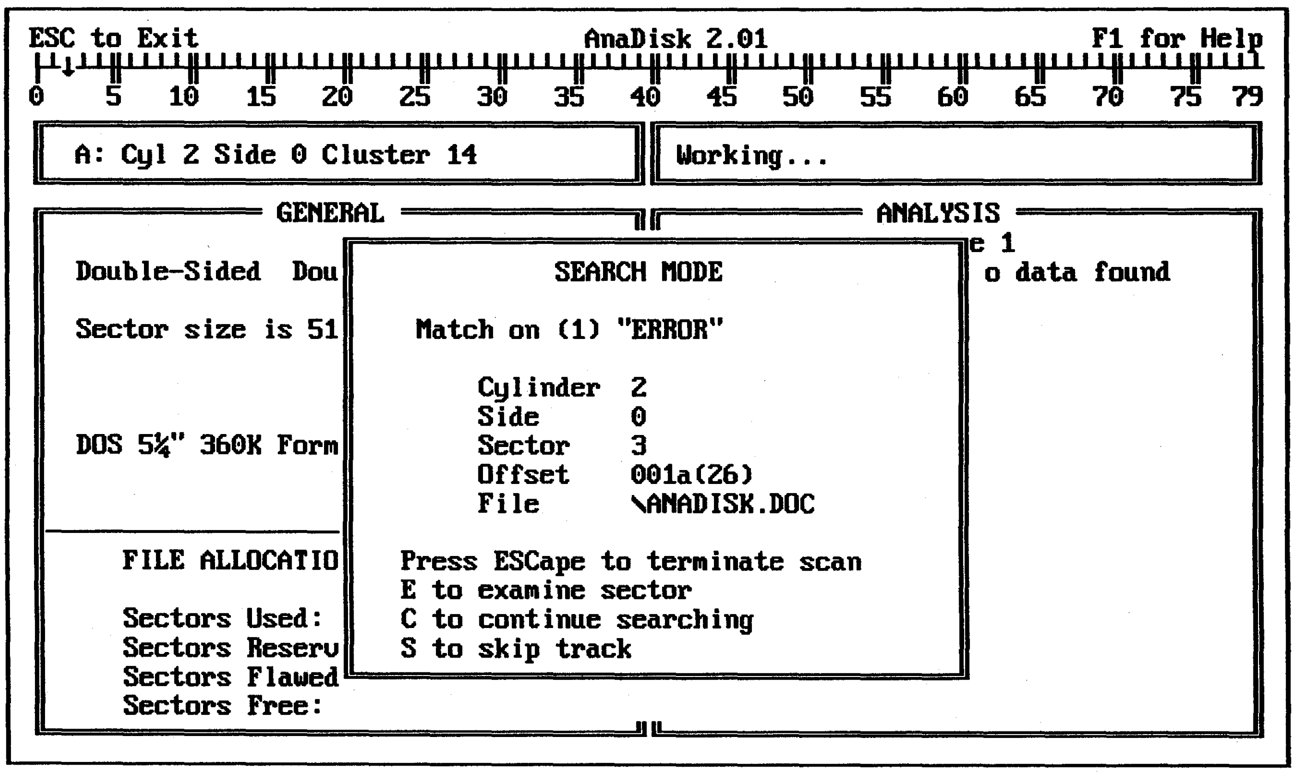


Figure 16: Search match display.

When a match "hit" is displayed, four actions are possible, depending on the character entered from the keyboard:

1. If **ESCape** is pressed, the search terminates and **AnaDisk** exits to the main menu display.

2. If **E** is pressed, the sector in which the match occurred is displayed with the **Edit Sectors** display. If **ESCape** is pressed while in this display, **AnaDisk** exits to the match screen (Figure 16).

3. If **C** is pressed, the search continues with the next character after the data causing the match.

4. If **S** is pressed, the remainder of the track is skipped, and searching resumes with the data on the next track.

```
ESC to Exit                    AnaDisk 2.0                    F1 for Help

                            Select - ESCape to quit, F1 for help

                          COPY DISKETTE

   Select choices with ←→ cursor keys.  ENTER (←┘) begins
   execution.  F1 gets help, ESCape goes back to the opening menu.

      Source Diskette Unit                      A:  B:

      Destination Diskette Unit                 A:  B:
```

Figure 17: Copy Diskette Menu.

Copy Diskette

The **Copy Diskette** function performs a diskette-to-diskette copy operation. Within the limits of PC hardware, an exact copy of the original is made. This is the *only* instance in which **AnaDisk** can be used to **write** a diskette.

It is strongly recommended that ALL original diskettes used with AnaDisk be write-protected! It is most important, however, to observe this precaution when performing the **Copy Diskette** function; a brief mental lapse or distraction could result in the original diskette being written.

Diskettes can be copied using either one diskette drive, in which case diskette "swaps" will be required, or with two drives. If a two-drive copy is performed, the drives being used must both be capable of supporting the format being copied. It is not possible, for example, to copy a 3.5" 720K diskette to a target in a 5.25" 360K drive. Should this be attempted, the following message appears:

```
═══════↑↓→← to Move═══════════
 The source diskette type
 is incompatible with the
 destination type.  Press
 any key for Main Menu...
```

Because the diskette drives specified are unsuited for the particular copy operation, **AnaDisk** exits to the Main Menu in order that a new drive selection can be made.

The operation of the **Copy Diskette** function is straightforward. The source diskette is analyzed, track by track, and the data read from the source is accumulated in PC main memory. When no more data can be stored, the accumulated information is written to the destination diskette. Each track of data that has been written is verified by reading the data back. The process continues until there is no more data to be copied. Progress and error messages are displayed in the **ANALYSIS** window.

Not all diskettes can be copied faithfully by **AnaDisk**, particularly diskettes containing copy protection information written by specialized equipment, or diskettes that have been physically modified (e.g., a hole burned by laser on a particular track). Nor can diskettes that have been written by some non-PC compatible systems, such as the Apple Macintosh, be copied using **AnaDisk**.

Repair

The **Repair** function will operate with DOS diskettes only. The diskette is scanned for data errors; any part of a file or subdirectory containing errors is moved to a free area on the diskette and the original sectors marked as flawed in the file allocation table. The data as read from the sectors containing the error is copied as read; note that this data may contain errors. The sectors containing the original data are left undisturbed.

After the **Repair** function has been selected from the Main Menu, the diskette unit containing the diskette to be scanned is selected. A printer "log" of the repair activity may also be selected, if desired.

```
ESC to Exit                          AnaDisk 2.01                          F1 for Help
0    5   10   15   20   25   30   35   40   45   50   55   60   65   70   75  79

A: Cyl 7 Side 1 Cluster 198                 Working...

                GENERAL                                  ANALYSIS
                                                   Track 7, Side 1
Double-Sided  Hig  ↑↓→← to Move                  D. adding sector 115
Interleave is 1:1  Data error detected.  Press     from 1 to 115
Sector size is 51  E to edit data, M to move,      data error
Side 0 sector ID'  ESC to ignore error.            is in
Side 1 sector ID'                                 DEF

DOS 5¼" 1200K For

   FILE ALLOCATION TABLE (FAT)

   Sectors Used:        772 ( 386K)
   Sectors Reserved:      0 (   0K)
   Sectors Flawed:        0 (   0K)
   Sectors Free:       1599 ( 799K)
```

Figure 18: Repair Diskette Display.

AnaDisk then enters **Scan** mode. When a sector read error is detected, the display shown in figure 18 appears. If **E** is pressed, the sector containing the error will be displayed with the **Sector Edit** display. It may then be possible to correct the error by rewriting the sector. If **M** is pressed, the data from the diskette will be moved *as read* to a free area of the diskette and the File Allocation Table (**FAT**) adjusted accordingly. If **ESC**ape is pressed, the error is left as-is, and the scan for errors continues.

If the error occurs in a immovable area of the diskette, such as the root directory or FAT, the **M** option will not be displayed. In this case, it will be necessary to reconstruct the data manually.

Before attempting a **Repair** operation, it is **strongly** suggested that a backup copy of the diskette be made with the **Copy** function. Additionally, it should be noted that attempting to repair a diskette in a 1.2M drive which was written using a 360K drive may result in a diskette that is unreadable in a 360K drive. **AnaDisk** will request confirmation before proceeding with a repair of this type.

FAT Editor

AnaDisk provides a facility for displaying and editing the DOS File Allocation Table, or **FAT**, on a diskette. After **FAT** edit mode is selected from the Main Menu, the drive containing the diskette to be edited is selected. If the diskette being examined has a valid File Allocation Table, a display similar to that shown in Figure 19 appears.

The FAT describes how the groups of sectors, or **Clusters**, are put together. Every file directory entry contains the number of the first cluster for a file. If a file is less than one cluster in length, the entry for that cluster will contain the value 4095, signifying the end of the cluster list. If the file is larger than one cluster, however, each cluster in the FAT will "point" to the next one, forming a chain of clusters.

Certain FAT values have certain meanings. The first value in the FAT is simply used to identify the FAT type; the second always has the value 4095. A value of 4087 indicates that a block contains an error--this is almost always set by the FORMAT program. A value of 0 specifies a "free" or unallocated cluster.

```
ESC to Exit                         AnaDisk 2.01                        F1 for Help

       F1 Help           F2 Edit Cluster     F3 What File?     F4 Write FAT
       F5 Enter value    F6 Re-read FAT      F7 Hex            ESC Exit

       4088-4095 End-of-file     4087 Bad Cluster     4080-4086 Reserved

    Cluster 2(0002) = 3(3)

 00000    4089  4095     3     4     5 -    6     7     8     9    10
 00010      11    12    13    14    15 -   16    17    18    19    20
 00020      21    22    23    24  4095 -   26    27    28    29    30
 00030      31    32    33    34    35 -   36    37    38    39    40
 00040      41    42    43    44    45 -   46    47    48    49    50
 00050      51    52    53    54    55 -   56    57    58    59    60
 00060      61    62    63    64    65 -   66    67    68    69    70
 00070      71    72    73    74    75 -   76    77    78    79    80
 00080      81    82    83    84    85 -   86    87    88    89    90
 00090      91    92    93    94    95 -   96    97    98    99   100
 00100     101   102   103   104   105 -  106   107   108   109   110
 00110     111   112   113   114   115 -  116   117   118   119   120
 00120     121   122   123   124   125 -  126   127   128   129   130
```

Figure 19: FAT Editor Display

The cursor may be moved about the editing display, the up- down- right- and left-cursor keys and the **PgUp** and **PgDn** keys perform the navigation. In addition, the following function keys have special meaning:

F1	Produces a help display.
F2	Produces a **Sector Edit** display for the currently highlighted cluster.
F3	Identifies the file to which the currently highlighted cluster belongs.
F4	Rewrites the modified FAT to a diskette. **AnaDisk** requests confirmation before writing, however.
F5	Allows a new value for the currently highlighted cluster to be entered.
F6	Reads the FAT from a diskette. Any changes that were made to the displayed FAT are lost.
F7	Toggles between a hexadecimal and decimal display.
ESC	Returns to the main menu. Any changes made to the displayed FAT are lost.

AnaDisk performs a few checks on the value entered when the **F5** function is selected. The value must lie within the range of allocatable clusters for the diskette and not be already allocated to some other file.

Custom Format Design

AnaDisk provides a custom format design facility which may be used to produce formatted diskettes for non-DOS computers or to design a rudimentary "copy-protection" method. **AnaDisk**'s custom formatting function is not intended for use by beginners; a detailed knowledge of diskette structure is necessary for effective use of this feature.

After the **FORMAT** function has been selected from the Main Menu, the diskette drive containing the diskette to be formatted is selected. A display similar to that shown in Figure 20 then appears. Because of checking that is performed by **AnaDisk**, the simplest operation can be obtained if the following sequence of steps is observed:

To "lay out" a format, first select the sector size using the **F2** key; each depression of this key cycles through the allowable sector sizes from 128 to 8,192 bytes. Next, set the recording mode (**FM** or single-density, **MFM** or double-density) using the **F5** key. Either a **high**- or **low**-density data rate should be next selected using the **F6** key.

The number of sectors per track are then set using the **F3** key. If too many sectors are specified for the given recording mode and density, the sector count will be adjusted to the largest number of sectors that will fit on the track.

The actual starting cylinder is set with **F4**. Note that a high-density 1.2M 5.25" is considered to possess 80 cylinders, regardless of density. The cylinder increment between formatted cylinders is set with **F8**. The total number of cylinders formatted is set with **F9**. Note that this is the total count of cylinders formatted, and *not* the highest cylinder number. The side or sides to be formatted is set with **F7** and may be side 0, side 1 or both sides. The **F10** key is used to duplicate the previous editing display line into the currently highlighted line.

By default, sector ID address headers written are as follows:

ID Cylinder: Actual cylinder
ID Head: Actual side
ID Sector: Starts with 1, increments by 1
ID Length: Actual sector length

These defaults should suffice for most "normal" applications. However, the values for each ID address field may be altered considerably.

```
ESC to Exit                          AnaDisk 2.0                            F1 for Help

      F1 Help           F2 Size  512       F3 10 Sectors      F4 Start Cyl. 0
      F5 MFM            F6  low-density    F7 Both Sides      F8 Step 1
      F9 80 Cyls.       F10 Duplicate      <-┘ BEGIN FORMAT   ESC Exit

                                   ENTER TERM
       F2S Cylinder    F3S Cyl. Count F4S Side       F5S Sector    F6S Constant
       +   Add         -   Subtract   *   Multiply   /   Divide    .   End

  NO.    ┌──CYLINDER──┐   ┌───SIDE───┐    ┌──SECTOR──┐     ┌───LENGTH──┐
   1.    CYL*2            HD              SCT-1             2
   2.    CYL*2            HD              SCT-1             2
   3.    CYL*2            HD              SCT-1             2
   4.    CYL*2            HD              SCT-1             2
   5.    CYL*2            HD              SCT-1             2
   6.    CYL*2            HD              SCT-1             2
   7.    CYL*2            HD              SCT-1             2
   8.    CYL*2            HD              SCT-1             2
   9.    CYL*2            HD              SCT-1             2
  10.    CYL*2            HD              SCT-1             2
```

Figure 20: Custom Format Display

Each entry in the ID address table shown in the lower part of the display can take on the following mathematical representation:

value1 **op** *value2* **op** *value3* **op** *value4*

Where each *value* is either a constant or one of the following special values:

Shift-F2 is the number of the actual cylinder currently being formatted; shown as **CYL** on the editing display.

Shift-F3 is the count of the cylinders that have been formatted up to this point; shown as **CNT** on the editing display.

Shift-F4 is the side (0 or 1) currently being formatted; shown as **HD** on the editing display.
Shift-F5 is the number of the current sector on this track, starting with 1; shown as **SCT** on the editing display.

Shift-F6 is used to specify a fixed numeric value between 0 and 255.

op can be one of the following:

+	for addition
-	for subtraction
*	for multiplication
/	for division

If a period (.) is entered for **op**, the expression is terminated at that point. Expressions are evaluated from left-to-right; results are limited to 8 bits (decimal 255). Division by zero results in zero with no diagnostic. Overflow is ignored.

For example, suppose it is desired to produce a single-sided format where the ID address cylinder field begins with 39 and ends with 0 at the innermost track. The expression for the **CYLINDER** field for every sector in the editing display would be:

39-CYL

It should be observed that the **Custom Format** function only writes the *format* pattern to a diskette. Other data, such as system tables must be written using other means, such as the **Sector Edit** function. It should be further observed that all sectors written by the **Custom Format** are of the same size, regardless of the length specifier in the ID Address header.

Dump Operation

The **Dump** operation writes a specified area of a diskette to a DOS file. After selecting the **Dump** option from the Main Menu, the diskette drive containing the diskette to be read, the range of cylinders and sides to be written to a specified DOS file are selected.

Each sector written to the file is optionally preceded by an 8-byte header record of the following form:

ACYL	ASID	LCYL	LSID	LSEC	LLEN	COUNT

ACYL	Actual cylinder, 1 byte
ASID	Actual side, 1 byte
LCYL	Logical cylinder; cylinder as read, 1 byte
LSID	Logical side; or side as read, 1 byte
LSEC	Sector number as read, 1 byte
LLEN	Length code as read, 1 byte
COUNT	Byte count of data to follow, 2 bytes. If zero, no data is contained in this sector.

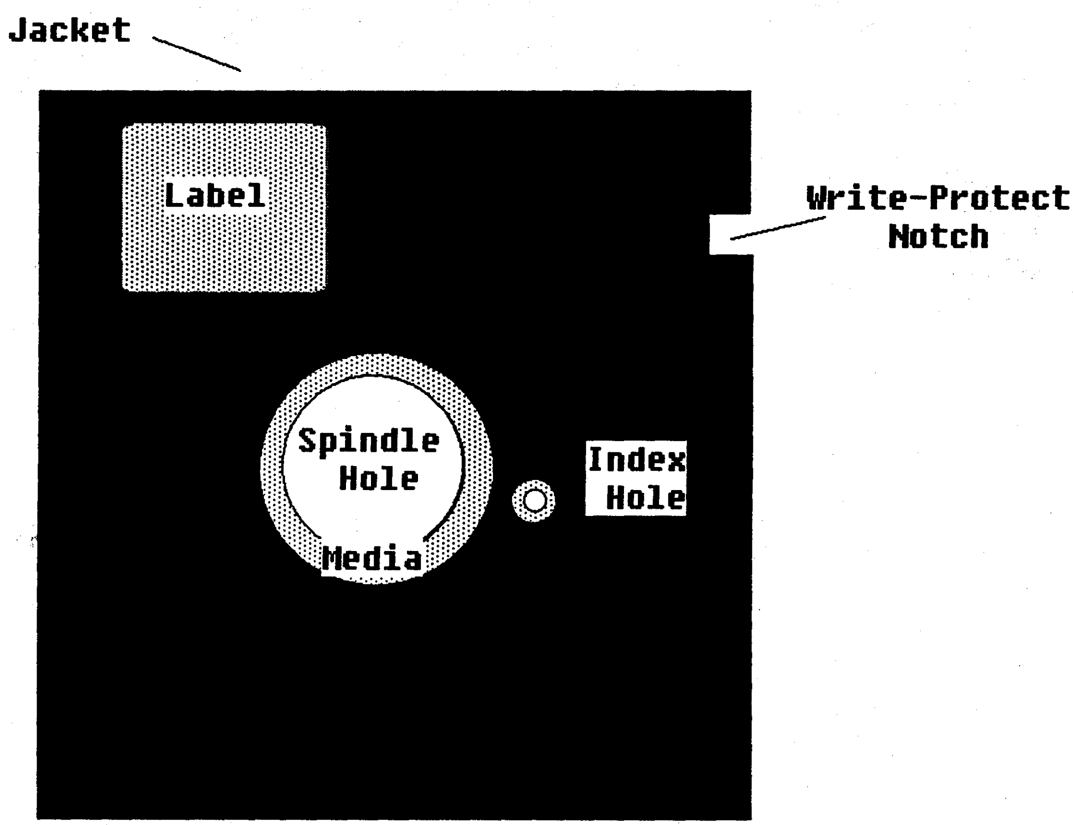
All sectors occurring on a side will be grouped together; however, they will appear in the same order as they occurred on the diskette. Therefore, if an 8 sector-per-track diskette were scanned which had a physical interleave of 2:1, the sectors might appear in the order 1,5,2,6,3,7,4,8 in the DOS dump file.

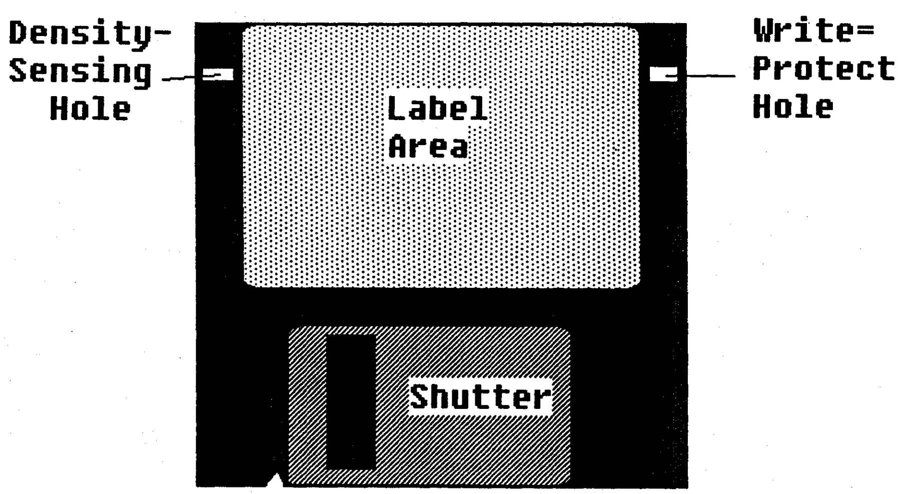
After the last specified cylinder has been written to the DOS file, **AnaDisk** returns to the Main Menu.

Looking Forward

We believe that **AnaDisk** Version 2.0 represents a major improvement over earlier verisons. Our list of features yet to be added to AnaDisk includes an intelligent file unerase, save and load data to a DOS file--and yes, a hard disk version!

All of which is made possible by your support of the Shareware concept. We hope we will continue to be deserving of that support. If there is any way in which we can assist you with this product, please call or drop us a line.


Jacket
Label
Write-Protect
Notch
Spindle
Hole
Media
Index
Hole


Density-
Sensing
Hole
Label
Area
Write=
Protect
Hole
Shutter

A Short Course on Diskettes

A diskette is made of much the same stuff as audio cassette and VCR tapes, enclosed in a black plastic envelope, or, in the case of 3.5" diskettes a blue or gray plastic container. If you really want to see what the business end of a diskette really looks like, take a junked 5.25" diskette and cut the black envelope open along one edge (or all edges, if you're really inquisitive). You'll see that the black plastic envelope has a fabric-like liner which is in contact with a rather flimsy brown ring of plastic, somewhat smaller, but similar in shape to a 45 rpm phonograph record. If you take the brown disk out of the envelope, it will immediately become apparent to you why these things are sometimes called 'floppy' disks.

We're talking about 5.25" diskettes here, by the way. The 3.5" has a hard plastic envelope and a metal hub, but otherwise is similar in operation to its larger cousin.

Have you ever spilled coffee on one of these things? If you've been keeping your only copy of your Great American Novel on the resulting mess, you can salvage the information this way -

1. Cut the black envelope open and carefully remove the brown disk.

2. Throw away the black envelope - it's ruined anyway because the cloth "wiper" inside is fouled with whatever was spilled on the diskette.

3. Carefully wash the brown disk under cold water, preferably with no soap. Take care not to wrinkle or fold the disk. Dry it carefully with a soft lint-free cloth.

3. Get a junked diskette or a new blank one and carefully cut the top edge (the edge that faces you when the diskette is in a drive) of the black envelope off to open it. Replace the disk inside with your just-washed disk.

4. You can optionally tape the edge of the envelope back up with plastic package-sealing tape (ordinary mending tape is too messy).

5. Copy your resurrected diskette to a new blank one.

If you've done in a 3.5" diskette, you're probably out of luck - most of the 3.5" jackets are fused shut. You can probably get one open, but there's no easy way to get one neatly closed again...

As we mentioned, the "business end" of a diskette is made up of much the same stuff that comprises normal audio or video tape. There is one difference, however. Where tape is coated only on one side with brown magnetic material, a diskette is coated on both sides. The actual plastic part is clear - it's the magnetic material that gives it the color. What this double-coating does is allow both sides of the diskette to be recorded, giving rise to the terminology "single-sided" or "double-sided". In actuality, *all* diskettes are coated on both sides, whether or not they're labeled "double-sided"; it's just that only one surface on a "single-sided" diskette has been certified to be error-free. However, there are many people who save money by recording both sides of a "single-sided" diskette...

Do you need to tell a high-density 1.2M diskette from a ordinary 360K diskette? Almost all 360K diskettes have a reinforcing ring around the central large hub hole; none of the 1.2M high-density diskettes do.

So much for physical appearances. How does information get recorded on the diskette? Well, to begin with, data is recorded on a series of circular concentric *tracks* on each side. On a phonograph record, recording is done as a very long spiral from the outer edge to the inside. The circular tracks on a diskette are not connected however, so if you start "playing" one back, you will eventually come back to where you started without moving to another track -- sort of like a "stuck" record.

To get from track to track, the record/playback head is attached to a positioning device, powered by some sort of motor, which allows "stepping" from one track to another. If both sides of the disk are to be used, a set of heads on the other side of the diskette is present, but both sets of heads move together as they're connected mechanically to the same positioning motor.

By convention, recording of data generally starts on the outer tracks of a diskette, just like a phonograph record. However, since it is fastest not to move the heads when recording or reading back data (head motion is sometimes called "seeking"), the track on one side is first used, then the track on the other side, then the heads are moved one track, and so on - not at all like playing a phonograph record!

If you take a look at a diskette (with or without its envelope), you'll see a much smaller hole, called the *index hole* next to the large "hub" hole. This is used to indicate where the start of each track is as an aid to organizing data. A small photo-transistor located in your diskette drive detects the passage of the hole once each revolution - usually about 300 times per minute. You may also notice a notch in the side of the diskette envelope. This notch is present to indicate that the diskette may be written on - if covered up, the diskette is said to be *write-protected*; that is, it may not be written on.

By the way, this 300 revolution-per-minute speed results in the media passing the read/write head at about 4 miles per hour - the speed of a brisk walk. By contrast, a "hard" disk moves past its heads at about 40 miles per hour.

3.5" diskettes don't have an index hole per se, but the metal hub ring does have a peculiar hole pattern that will lock it to the motor spindle in only one position; the net effect is the same as that of the index hole on a 5.25".

Next, let's define a few terms and look at how your computer manages data on a diskette.

Getting Down to Bits...

So far, we've talked about how a diskette is constructed and generally used, but to go much further, we'll need to settle on a working vocabulary -

A *disk* is just what you'd expect - that thing that holds the data or information you're interested in. We define it here for completeness' sake.

A *side* of a disk is either the top or bottom. A disk is said to be *single-sided* if only one side is used for storing information, and *double-sided* if both sides are used. Sometimes we refer to a side by the *head* used to read or write on it - the distinction is the same.

A *track* is a circle of information read or written by a drive on a single side - either top or bottom. There is much confusion over this term. We will use "track" as being synonymous with "side" or "head", though some erroneously use it to mean both sides of a diskette (or *cylinder*). Normally, a disk contains 40 tracks per side (or 80 tracks in the case of the very dense 1.2 Mb or 1.44 Mb PC-AT or PS/2 diskettes). The 40 track diskettes are recorded so that 48 tracks would form a "doughnut" one inch thick; the 80 track diskettes are recorded so that 96 tracks would be required to form a one inch doughnut. Sometimes we refer to track packing density as 48 TPI (tracks per inch) or 96 TPI. This means that a little less than a one inch "doughnut" is used from the two inches available on a 5 1/4" diskette.

A *cylinder* refers to both the top and bottom tracks taken together, or, more generally, all of the diskette than can be accessed without moving the read/write heads.

The *index hole* is the small hole punched in the disk about 1/4 inch away from the large center hole.

A *sector* is a block of information present on the diskette--it is the smallest amount of information that can be read or written on the diskette. On the PC, this is generally 512 characters. A track can hold about nine sectors - but more about this later.

From time to time, we will refer to disk *addresses*. Diskette sectors need something to tell them apart, like a name, but convention dictates that a number be used for each, rather like a house address in your neighborhood.

A *bit* is the smallest unit of information that a computer can handle - and has only one of two values--1 or 0.

A *byte* for our purposes is a grouping of 8 bits and can have 256 different values. A byte also represents exactly one character of data or text.

The world of disks is a hard, practical one and we must live with the following realities -

Rule 1: Your computer writes at most 1 bit at a time and this bit must be 1 or 0.

Rule 2: Diskette drives are physical objects with small differences between them. One of the more interesting is that diskettes are generally read and written with the disk spinning at *approximately* 300 revolutions per minute. That one drive turns at *exactly* the same speed as another is only a possibility, not a certainty.

Rule 3: Index holes are almost never *exactly* the same in size, and anyway, two drives wouldn't see them as such.

So what are some of the problems that these rules cause? Consider that we want to record nine sectors on a track (glossary above!). We record these sectors, starting at the index hole, running one right after the other, thus:

```
| Sector 1 | Sector 2 | Sector 3 | Sector 4 | Sector 5 |
|                                |
The index hole is here.          Watch this space!
```

Now, suppose we change sector 3, but we use another drive and it spins *slightly* faster than our original...

```
| Sector 1 | Sector 2 |    Sector 3 |ctor 4 | Sector 5 |
|                                   |
Index hole is here                  It got clobbered!
```

You'll note that we've just clobbered the start of sector 4 because it took us the same amount of time to write the sector, but what we did write took up more of the track than the original. Think of two sprinters with spray paint cans painting lines on a track by holding down the spray nozzle for exactly one second. The faster runner will paint a longer line, but the same amount of paint (or data) will have been used by both.

What if we left a few extra bytes between sectors to take up any reasonable difference in speed? Good idea--and we should also stick some *filler* bytes after the index hole because of Rule 3. So far, so good. But how do we tell the filler bytes from real data? After all, we need to know where our data is. We could put a special data pattern at the start of each sector that would be really different from the slack bytes. We then might get something that looks like this--

```
|FFFFFFM Sector 1 |FFFFFM Sector 2 |FFFFFM Sector 3 |FFFFFF...

|                  |                      |
Index hole here    This is a filler byte  This is a sector marker

        F - Filler Byte
        M - Sector Mark Byte
```

Then, if we over-write filler bytes, who cares? But now, the only way to find any sector is to wait for the index hole to come around and then to look for our sector marks until we find the sector that we are looking for. Why not put each sector's number or *address* with each sector mark? Then all we would have to do is to find a sector mark with the correct address. So, we get something that looks like this -

```
|FFFFFFFFMAA Sector 1 |FFFFFFFMAA Sector 2 |FFFFFFFMAA Sector 3 |FFFFF...

|                        |                   |
Index hole here          Filler byte         Mark and sector address
```

Okay, we just about have it all now. The filler bytes we talked about are referred to collectively as *gap* bytes, the sector marks are divided into two kinds for electrical reasons - the first is called an *ID Address Mark* and signals the start of a sector address. The second is called an *Data Address Mark* and signals the start of the actual data in a sector. The primary reason that two marks are used is to avoid continually rewriting (and taking the chance of clobbering) the sector address information.

One additional item is added to each sector data and address, called a *Cyclical Redundancy Check* or *CRC*. This is a rather involved type of checksum of all of the information in the data or address fields. When the disk controller reads a sector, it calculates its own CRC from what it reads and compares it with the CRC that was written when the data was first stored. If the two match, then we can be pretty sure that what we read was the same as what was written; we have a problem otherwise.

A short digression is appropriate here. There is another method, not in common use today, to demarcate sector boundaries, called *hard sectoring*. This is done by having not just a single index hole, but many - one for each sector. The index hole itself is represented by three closely-spaced holes. Before the advent of large-scale integrated circuits, this resulted in a considerable hardware reduction. This is rare today, however. *All* diskette recording methods use one or the other scheme.

So why can't **AnaDisk** read Apple diskettes? Apple diskettes are indeed soft-sectored, but hail from a time when LSI disk controllers chips where $50 each in quantity. Steve Wozniak, the designer of much of the Apple hardware, designed a diskette controller where the 6502 microprocessor in the Apple would do most of the work in handling diskette read and write data encoding. A 5- or 6-out of-8 bit scheme was used, which is peculiar to Apple computers. The rest of the world tends to use variations on the IBM diskette convention. Ironically, Wozniak's collection of inexpensive IC's on the original Apple disk controller has been collected in a single LSI chip called the "IWM" or "Integrated Woz Machine" in later Apple and Macintosh computers. If you're interested, Jim Sather's *Understanding the Apple II* contains one of the best expositions on the mechanics of Apple disk handling to date.

Back to the world according to IBM. Let's look at a real diskette. The following is a description of what is normally present on a single track of a diskette -

	Index Hole	
	80 Gap Bytes	
	12 More Gap Bytes	
	4 Byte ID Address Mark	(Start of Track Mark)
	50 Gap bytes	
***	12 More Gap Bytes	
	4 Byte ID Address Mark	(Start of Sector Mark)
	1 Byte Cylinder Number	
	1 Byte Side Number	(0 or 1)
	1 Byte Sector Number	(This sector's address)
	1 Byte Length Code	(Says how long the sector is)
	2 Byte Address CRC	
	22 Gap Bytes	
	12 More Gap Bytes	
	4 Byte Data Address Mark	
	512 Bytes Data	(Finally!)
	2 Byte Data CRC	
	50 Gap Bytes	
	*... Repeat from *** for as many sectors as will fit ...*	
	Gap bytes to Index Hole, as many as will fit	

It's obvious that there's a lot of overhead just to store 512 bytes of data - about 112 bytes worth. But it's all done in the name of reliability - and it works.

A couple of comments before we move on. You'll notice that the disk layout talks about "Gap Bytes" and "More Gap Bytes". The difference between them is that the "Gap Bytes" have one value and were made to be written over if necessary. The "More Gap Bytes" have a different value and are not to be written over; their chief function is to allow the electronics some time to prepare for the next ID Address Mark.

Another item is that two different Data Address Marks are possible - one simply referred to as a "normal" Data Address Mark; the other, a "Deleted" Data Address Mark. The Deleted mark is not used by the PC in normal circumstances; but some copy protection schemes do use it, just because most PC software is ignorant of its existence.

Which brings us to the next topic...

Copy Protection, or, Ways to Skin A Cat...

In the beginning, there was no copy protection. A user could copy whatever he wanted and give it to whomever he wished, if he didn't get caught. But then there were few personal computers and little incentive to copy, except for purposes of creating backup diskettes.

Then along came Apple and IBM, and the personal computing world was never to be the same. Because there also came copy protection...

Okay, how do you "copy protect" something? Basically, there are two types of copy-protection schemes -

1. Make the diskette different enough in some aspect that the normal system copying utilities can't handle the format. Copy errors result, and the copy is worthless.

2. Hide some sort of difference that won't be detected by the system copy utilities. You think you've made a good copy, but it won't run.

Here are just a few ways in which this can be accomplished:

1. Use a Deleted Data Address Mark somewhere on the disk. DOS COPY and DISKCOPY won't see it, so it won't be transferred to the new copy. (Type 2).

2. A DOS diskette has 40 tracks - but most diskette drives can write at least 41 - and DOS won't see this extra track. (Type 2).

3. Leave out a sector somewhere on the disk (Type 1).

4. Make one sector either longer or shorter than 512 bytes (Type 1).

5. Leave off an ID Address Mark somewhere. The Data Address mark will still be present, but DOS won't find the sector (Type 1).

6. Hide something in the Gap Bytes; DOS won't see it. (Type 2).

7. "Jumble" the order of sectors on a track. Instead of numbering them 1, 2, 3,....9, number them 1, 9, 3, 2,... DOS won't know the difference and will still write them as 1, 2, 3,... (Type 2).

8. Stick in an extra sector somewhere. DOS doesn't look for it. (Type 2).

9. Goof up the address information somewhere on an unused portion of the diskette. (Type 1).

10. Burn a tiny hole somewhere on the diskette with a laser. Try to write over the hole - if no errors result, the diskette's a copy. (Type 1 or 2).

11. Record ID's and sector addresses on a track, but leave out the data. By doing this, you can fit about 100 sectors on a track and thoroughly confuse anything trying to copy it.

And on it goes... The publishers get clever and develop a new scheme; the hackers see a challenge and break the scheme. With the advent of networks and the proliferation of fixed disks, however, copy protection is slowly being abandoned by most publishers. On the other hand, some copy-protection schemes are darned effective and require drastic measures to defeat - Central Point's **COPYIIPC** program actually alters the copy protection programs for some copies.

AnaDisk will produce true copies of *some* copy-protected diskettes - it is impossible to accommodate all copy-protection schemes without special hardware. The DIAGNOSTIC READ function available under the **Edit Sectors** function of **AnaDisk** will allow you to examine all the data recorded on a track, however - including address marks and gap bytes.

High-Density Variations...

What we've mostly talked about up to this point is the garden-variety, double-sided, double-density diskette holding about 360,000 bytes of information. But there exist diskettes which hold much more than this, referred to variously as high-density or quad-density.

So what's the trick?

One way to get higher density is to pack more tracks on a diskette by making the tracks narrower. So, instead of 48 tracks per inch, you get, say, 96. This would give you 737,280 bytes in the same space as before; the PC can support this format in DOS 3.2 and later, but in general, 720K diskettes in a 5.25" form factor are not very common.

Another way is to squeeze bits closer together on a track radially. The PC AT high density drive gets 15 sectors on a track, rather than 9, and also uses the narrow track scheme to get 80 tracks. Multiply it out -

15 sectors X 80 tracks X 2 sides X 512 bytes per sector

and you get 1,228,800 bytes or 1.2 megabytes.

The IBM Personal System/2 machines normally record 9 sectors per track on 80 tracks on a 3.5" micro-floppy diskette for 720,000 bytes. The higher-powered PS/2 models will also "squeeze" 18 sectors per track on the same disk drive for 1,474,560 bytes. Quite a bit of storage for such a small package!

You may have heard that the high-density drives on a PC AT are capable of writing a "normal" 360K diskette, but that they don't do a good job of it. Quite true; consider that the disk drive is writing a track only half as wide as a "normal" drive. This results in a lower read-back signal level and, therefore, is much less reliable than a "normal" drive. A 360K diskette read back on a 1.2M byte high-density drive is perfectly all right, though. The trick to producing readable 360K diskettes on a high-density drive is to first degauss or "bulk erase" the diskette using a video- or audio recording tape bulk eraser, then format the diskette on the high-density drive. It is recommended by some sources that you should first format the 360K diskette on a 360K drive, but that's exactly what you *don't* want to do.

DOS Diskettes - FATS, Clusters and Directories

Up to this point, we've discussed how data gets on and off a diskette from a very low-level standpoint. On top of all of what we've discussed, there is DOS.

What DOS provides is a way of *structuring* the information on a diskette so it can be managed in a very systematic way. As a sort of comparison, we can talk about the mechanics of writing on paper (use a pen or pencil) and how to organize what we've written (memos, novels, filing cabinets). DOS organizes logical groups of information into something called a *file*, which is probably no news to you. What may be news are the *structures* that DOS uses to get from a sector to the logical concept of a file.

If DOS is to organize things on a diskette, it should have some means of determining whether a sector contains data belonging to a file, or if a sector is "empty" or available to receive data for a new file. DOS does this by means of a list of sectors called a *FAT* for File Allocation Table.

To get an idea of how a FAT works, suppose that we have a diskette with a grand total of 10 sectors on it. We make a little table of 10 entries, one for each sector. If a sector is unused, we write a 0 in the table entry for that sector. If a sector contains part of the data for a file, we write the number of the *next* sector in the file in that sector's entry; we *point* to the next sector in the file. If a sector is the last sector in a file, we write 99 in that entry. If a sector cannot be used because of a defect in the diskette, we write 88 in that entry.

This business of using one table entry to point to another is called a *linked list* or a *chain*. The general concept is used widely in computer systems programming.

Let's take a file made up of 4 sectors, and let the order in which the sectors were written be 8, 3, 4, 2. Further, let's suppose that sector 6 is defective. Our little 10-entry FAT would look like this -

Slot	1	2	3	4	5	6	7	8	9	10
	0	99	4	2	0	88	0	3	0	0

Notice that if we know that the file starts with sector 8, we can thread our way through the table to get all of the sectors for our file in the correct order. But how do we know that the file starts at sector 8?

Enter the *Directory*. A directory is a just a special file which contains a list of file names and the starting sector of each. Other information gets placed in the directory, such as the creation date of a file, but the important items are the name and the starting sector. Since we need to know where the FAT and (root) directories are, they occur in the same place for all diskettes of a given format.

The notion of a *cluster* lets us keep the size of the FAT down on larger diskettes; essentially what is done is to group adjacent sectors up in "bunches" and to assign a number to the whole bunch, instead of to individual sectors. A bookkeeping trick, mainly.

Some things should be obvious about this system that DOS implements.

The first is that if you inadvertently mess up a FAT, you're in big trouble because you don't know how the sectors fit together on a diskette. Sure, the directory will tell you the *first* sector of every file, but that's all. DOS maintains two copies of the FAT on a diskette to avoid such a problem, but doesn't quite know how to make good use of the second FAT in case of an error. a diskette that is a hopeless case for DOS.

The second obvious thing is that if you lose a directory, it still may be possible to resurrect the files on a diskette from the FAT, but the names of the files will be unknown. The DOS utility CHKDSK handles this job nicely, and "invents" new file names.

Something worth mentioning at this point is that *Subdirectories* are simply files of file names - the sectors belonging to them are present as FAT entries, just as those for any other file would be. In fact, under DOS 2.0, it was possible to treat a subdirectory in some cases precisely the same as a file. This created a few operational problems and this "feature" has been carefully disabled in DOS 3.

We mentioned that there are two copies of the FAT on every DOS diskette. The first of these copies starts on the second sector of a diskette (Cylinder 0, Side 0, Sector 2). A FAT is immediately recognizable because the first 3 bytes have the hexadecimal patter FxFFFF, where *x* depends on the diskette type. What makes a FAT hard to unravel is that each table entry is 12 bits long and stored in Intel "flipped" format. Suppose we have two FAT entries, 123 hex and 789 hex. They would be stored in three consecutive bytes thus:

23 91 78

Appendix II shows the format of a directory entry. Note that exactly 16 of them will fit in a 512-byte sector.

If you're interested in further exploring the area of DOS disk organization, check out Peter Norton's *Programmer's Guide to the IBM PC*. Here, we've covered just about everything you need to find your way around on a diskette:

* Physical construction of a diskette.

* How data is recorded.

* How DOS organizes the data.

There are lots of little things left to discover; copy a DOS diskette and use **AnaDisk** to explore!

Appendix 1 - Disk Track Formats

FM (SINGLE-DENSITY) TRACK FORMAT

40 Bytes hex *FF*
6 Bytes *00*
1 Byte hex *FC* (Index Mark)
26 Bytes hex *FF* or *00*

The following is repeated for each sector:

6 Bytes *00*
1 Byte hex *FE* (ID Address Mark)
1 Byte cylinder number
1 Byte side number (*00* or *01*)
1 Byte sector number
1 Byte sector length code: *00* = 128, *01* = 256, *02* = 512...
2 Bytes ID CRC
11 Bytes hex *FF* or *00*
6 Bytes *00*
1 Byte hex *FB* (Data Address Mark)
Sector Data
2 Bytes Data CRC
27 Bytes hex *FF* or *00*

After the last sector, the following is written to the index hole:

Repeated hex *FF* or *00*

Some older formats leave off the initial 73 bytes for the Index Address mark.

MFM (DOUBLE-DENSITY) TRACK FORMAT

80 Bytes hex *4E*[1]
12 Bytes *00*
3 Bytes hex *F6*
1 Byte hex *FC* (Index Mark)
50 Bytes hex *4E*

The following is repeated for each sector:

12 Bytes *00*
3 Bytes hex *F5*
1 Byte hex *FE* (ID Address Mark)
1 Byte cylinder number
1 Byte side number (*00* or *01*)
1 Byte sector number
1 Byte sector length code: *01* = 256, *02* = 512, *03* = 1024...
2 Bytes ID CRC
22 Bytes hex *4E* or *00*
12 Bytes *00*
3 Bytes hex *F5*
1 Byte hex *FB* (Data Address Mark)
Sector Data
2 Bytes Data CRC
54 Bytes hex *4E*[2]

After the last sector, the following is written to the index hole:

Repeated hex *4E*

[1] On some older formats and on older Sony format 3.5" diskettes, this Index preamble is omitted.

[2] The length of this gap is adjusted for the number and size of sectors written on the track.

Appendix 2 - DOS Directory Entry Format

A DOS directory entry is 32 bytes in length; 16 of them fit exactly in one 512 byte sector.

A directory entry has the following format:

- 8 bytes, ASCII file name[1]
- 3 bytes, ASCII file name extension
- 1 byte, file attribute:[2]
  - 1 Read-only
  - 2 Hidden
  - 4 System
  - 8 Volume Label
  - 16 Subdirectory
  - 32 Archive
- 10 bytes, Unused, set to 0
- 2 bytes, time of creation[3]
- 2 bytes, date of creation
- 2 bytes, beginning cluster number
- 4 bytes, file size in bytes

[1] When a file is deleted, the first byte of this field is set to hex *E5*. Newly formatted diskettes have this byte set to 0; DOS will terminate a directory search when a leading 0 byte in the filename field is found.

[2] Attributes may be combined for a single file. The Archive bit is set when a file is created and every time it is modified. Volume labels take a directory entry; they appear as zero-length files. The System attribute implies Hidden, Read-only and also signifies that the file may not be moved or fragmented.

[3] The time and dates of file creation are in a peculiar encoded format. The encoding for the date is:

- 7 bits, year beyond 1980
- 4 bits, month of year
- 5 bits, day of month

The encoding for the time is:

- 5 bits, hours
- 6 bits, minutes
- 5 bits, seconds divided by 2

Appendix 3 - Boot Sector for DOS Diskettes

The first sector of a DOS diskette contains a description of the diskette and a short program to do either of the following:

* Load DOS from the diskette (System Diskette)

* Display a message saying that this diskette isn't a system diskette (Non-system Diskette.

The program on sector 0 is loaded at location *0:7C00* and executed when the system is booted.

DOS BOOT SECTOR LAYOUT

Offset	Length	Contents
0	3	Either *E9 xx xx* or *EB xx 90* (A jump to the start of the boot)
3	8	ASCII System ID, e.g. "IBM 3.3 "
11	2	Bytes per sector
13	1	Sectors per cluster
14	2	Reserved sectors at start of disk (Almost always 1 - for sector 0)
16	1	Number of FAT copies - always 2.
17	2	Number of root directory entries. (Usually 64, 112 or 224)
19	2	Total sectors on this disk. (720 for 360K)
21	1	FAT ID byte (usually *Fx*)
22	2	FAT length in sectors
24	2	Sectors per track
26	2	Sides on this diskette
28	2	Number of "special" reserved sectors (Usually 0)
30	492	The boot program

It can be seen that the total number of sectors can be 65,535 at most; this propagation of a 16-bit sector number throughout DOS is what leads to the 32 megabyte upper limit on hard disk size. (65,535 x 512 bytes) = 32 Mb. Some vendors offer attempts to get around this limitation by increasing the size of a sector, either logically (with software) or physically, thus increasing the total amount of storage that can be represented. This deficiency is carried over to OS/2, but not to DOS 4.0, which uses 32-bit sector numbers.

The Sydex Product Line

AnaDisk -- The compleat diskette utility. Nothing like it anywhere else; scan, edit, repair and copy just about any kind of diskette. $25.00 ($150.00 site) registration fee.

Con>Format -- Concurrent "background" diskette formatter. Features "pop-up" operation and "hot key" activation. You've got to see it to believe. Supports all current DOS formats. $15.00 ($50.00 site) registration fee.

22DISK -- Transfer files, format, examine and erase files on "foreign" CP/M diskettes on your PC. Includes tips on supporting 8" and 5.25" single-density diskettes. Contains definitions for over 200 different formats. $25.00 ($100.00 site) registration fee.

22NICE -- A CP/M 2.2 emulation package. Supports the NEC V-series chips or performs emulation by software for both the 8080 and Z80 processors. Includes terminal emulation and diskette handling for common CP/M systems. Includes **22DISK**. $40.00 ($150.00 site) registration fee.

TeleDisk -- Turn any diskette into a compressed data file and vice-versa. Allows you to send and receive entire diskettes via modem. Even works with some "copy-protected" diskettes. $20.00 ($65.00 site) registration fee.

COPYQM -- Mass diskette duplicator. Format, copy and verify multiple diskettes from a single master. Implements "no keyboard" interaction mode and drive "round robin" servicing. Supports all standard DOS formats. $15.00 ($50.00 site) registration fee.

FORMATQM -- Mass diskette formatter - format a box of diskettes at a single sitting. Implements "no keyboard" interaction mode and drive "round robin" servicing. Supports all standard DOS formats. $10.00 ($40.00 site) registration fee.

...And, soon to be released:

InterDisk - The universal Diskette Interchange utility. Features user-programmable media conversion for those formats not covered by **22DISK**. Available 1Q90.

Information on any of these products can be obtained from Sydex by calling (408) 739-4866 (voice) or by leaving a request on our Bulletin Board System (408) 738-2860 (1200-9600 bps, N81).