Ensoniq Floppy Diskette Formats
by Gary Giebler

After purchasing my first Ensoniq keyboard, I remember grabbing the disk out of the SQ-80 and running to the nearest IBM-PC computer only to be disappointed when I couldn't list the disk contents on the computer. I made the same trip to my PC when I bought my VFX-SD only to be disappointed once again. Faced with the thought of spending hours glaring at fluorescent displays pushing buttons to scroll through files, I refused to surrender. Instead, I spent hours digging through track dumps and BIOS code to determine the formats used on the disks. My efforts paid off and I now view entire directories of SQ-80, EPS, and VFX-SD disks on my PC's monitor. More importantly, I can format disks and copy files to my PC's hard disk for storage or examination - which has allowed me to discover Ensoniq's format for sequences. I have already written programs to convert SQ-80 sequences to Standard Midi Files, convert SQ-80 sequences to VFX-SD sequences, convert VFX-SD sequences to Standard Midi Files, and convert Standard Midi Files to VFX-SD sequences. By the time this article is published, I will have completed a program to convert EPS-16 sequences back and forth to Standard Midi Files. If there is any interest, I could offer a program to convert EPS-16 sequences back and forth to VFX-SD Sequences. Most commercially available music publishing software can read Standard Midi Files enabling direct transfer of sequences from the Ensoniq keyboards. (This will enable music publishers to publish sheet music direct from an Ensoniq diskette.) Also, there are literally thousands of sequences available on computer bulletin boards stored in Standard Midi Files. You can download the sequences to your computer and convert them to VFX-SD or EPS sequences. In fact, several third party vendors are already offering demo sequences and sounds on the following computer bulletin boards:

Midium (818) 764-4538 Sound Source Unlimited (818) 879-9125

This article reveals the disk formats used for each of Ensoniq's keyboards and includes information on the file directories for several of the formats. This information can be used to write software to read the files on the disks, to copy files, or even to format disks on the PC. I want to thank the engineers at Ensoniq, Alan Smith, John Senior, and Joe Friel for taking the time to verify the accuracy of the MIRAGE, EPS and VFX-SD information and for filling in the 'blanks' when necessary. However, the SQ-80 information HAS NOT been confirmed by Ensoniq so be careful when trying this at home... The software package described in this article is available for those who would rather spend their time making music instead of programming.

EPS, VFX-SD, SD-1, EPS-16 Disk Format

The EPS, VFX-SD, SD-1, and the EPS-16 PLUS share the same disk format although the directory information is slightly different. The disk contains data on both sides with 80 tracks numbered 0 - 79 on each side. Each track has ten 512 byte sec-

tors numbered consecutively from zero to nine. I will refer to the two sides of the disk by referring to the disk drive head used to read each side. The heads are numbered 0 and 1. Data is stored on both sides of each track before moving to the next track. The following examples should clarify this.

```
BLKS   TK HD SC
00-09   0  0 0-9  data is first stored on Track 0, Head 0, Sectors
0-9
10-19   0  1 0-9  data is next stored on Track 0, Head 1, Sectors
0-9
20-29   1  0 0-9  data is then stored on Track 1, Head 0, Sectors
0-9
                  this process continues until....
-1599  79  1 0-9  the last track - Track 79, Head 1, Sectors 0-9
```

Each sector is referred to as a block of data. The blocks on the EPS, EPS-16, SD-1, and VFX-SD disks are numbered from 0 - 1599. The following formula calculates the block number from the track, head, and sector number:

```
Block = (((Track x NH) + Head) x NS) + Sector
```

where NH = Number of Heads, NS = Number of Sectors per Track

Since we know that NH = 2 and NS = 10, the formula can be written:

```
Block = ((Track x 2) + Head) x 10) + Sector
```

The track, head, and sector number can be calculated from the block number as follows:

```
Track  = Integer(Block/20)
Head   = Integer((Block - (Track x 20))/10)
Sector = Block - (Track x 20) - (Head x 10)
```

"Integer" refers to the appropriate integer function for the programming language used. Sample Turbo Pascal routines are included to show how to convert back and forth between blocks, tracks, heads, and sectors.

****

TURBO PASCAL LISTINGS

```
Function Block (Trk, Hed, Sct : Word) : Word ;
begin
  Block := (((Trk SHL 1) + Hed) * 10) + Sct ;
end ;

Procedure GetTrkHedSct (Block : Word; var Trk, Hed, Sct : Byte) ;
var Temp : Word ;
begin
  Sct  := Block MOD 10 ;
  Temp := Block DIV 10 ;
  Hed  := Block MOD 2  ;
  Trk  := Temp  DIV 2  ;
end ;
```

Sample Calls:

```
FirstDirectoryBlock := Block(0,0,3) ;
GetTrkHedSct(FirstDirectoryBlock,Track,Head,Sector) ;
```

****

Since IBM-PC disks are formatted with nine sectors per track (numbered from 1 - 9), the standard DOS functions can't be used to read the Ensoniq disks. However, by setting up the proper parameter table and making calls directly to the BIOS, you can use the BIOS to read the disks. If you don't know what the BIOS is, there are several good books on the subject. You can probably pick one up at your local book store. I don't intend to go into the details of calling the BIOS - such a discussion belongs in a computer magazine. However, it is important for you to understand that it takes special programming to read these disks, and you probably should not attempt this unless you really know what you are doing.

A software program for IBM-PC compatible computers is available from Giebler Enterprises. This program will read, copy, format, and display EPS, EPS-16 Plus, SD-1, and VFX-SD diskettes on the PC. Individual files or entire diskettes can be copied to a hard disk drive for storage or examination. (Great for sending your latest "sure hit" to friends over modems.) The disk copy feature formats the disk while copying and can be used to make multiple copies of Ensoniq diskettes which could be quite useful for third party sound or sequence developers. Just select the correct disk file on the hard disk and make as many copies as you need. The program will also display SQ-80 directories although the program can't format SQ-80 diskettes at this time. Disk labels including directory listings can be printed for the diskettes. As a special introductory offer, anyone who purchases Version 1.0.2 of the software will be granted unlimited software upgrades for a reasonable ($5.00) handling charge. Version 1.0.2 will also use free space on your hard disk drive to copy diskettes (EPS, VFX-SD, SD-1, EPS-16 PLUS) without the repetitive and tedious disk-swapping normally required with the Ensoniq keyboards. An IBM-PC or compatible with a 3 1/2" diskette drive is required. During the introductory offer, the software is available for only $18.00 (free shipping in the U.S.) New York residents - add appropriate sales tax. Contact: Giebler Enterprises, 8038 Morgan Road, Liverpool, New York, 13090-2009.

EPS & EPS-16 PLUS Sector Information

BLK	TK	HD	SC	Sector Information
0	0	0	0	Unused - Repeating 2 byte pattern of 6D B6 (hex)
1	0	0	1	Device ID Block (similar to VFX-SD)
2	0	0	2	Operating System Block
3	0	0	3	Main Directory (1st sector)

4	0	0	4	Main Directory (2nd sector)
5	0	0	5	File Allocation Block
6	0	0	6	File Allocation Block
7	0	0	7	File Allocation Block
8	0	0	8	File Allocation Block
9	0	0	9	File Allocation Block
10	0	1	0	File Allocation Block
11	0	1	1	File Allocation Block
12	0	1	2	File Allocation Block
13	0	1	3	File Allocation Block
14	0	1	4	File Allocation Block
15...1599				Unused - Repeating 2 byte pattern of 6D B6 (hex)

SD-1 & VFX-SD Sector Information

BLK	TK	HD	SC	Block Information
0	0	0	0	Unused - Repeating 2 byte pattern of 6D B6 (hex)
1	0	0	1	Device ID Block (similar to EPS)
2	0	0	2	Operating System Block
3	0	0	3	Main Directory (1st sector) Points to Sub-Directories 1 - 4
4	0	0	4	Main Directory (2nd sector)
5	0	0	5	File Allocation Block
6	0	0	6	File Allocation Block
7	0	0	7	File Allocation Block
8	0	0	8	File Allocation Block
9	0	0	9	File Allocation Block
10	0	1	0	File Allocation Block
11	0	1	1	File Allocation Block
12	0	1	2	File Allocation Block
13	0	1	3	File Allocation Block
14	0	1	4	File Allocation Block
15	0	1	5	Sub-Directory 1 (1st sector)
16	0	1	6	Sub-Directory 1 (2nd sector)
17	0	1	7	Sub-Directory 2 (1st sector)
18	0	1	8	Sub-Directory 2 (2nd sector)
19	0	1	9	Sub-Directory 3 (1st sector)
20	1	0	0	Sub-Directory 3 (2nd sector)
21	1	0	1	Sub-Directory 4 (1st sector)
22	1	0	2	Sub-Directory 4 (2nd sector)
23...1599				Unused - Repeating 2 byte pattern of 6D B6 (hex)

DEVICE ID BLOCK (Block 1)

The Device ID Block contains the following 40 byte pattern (repeated to fill the entire block on a newly formatted disk). The keyboards only read the first occurrence of the pattern. In fact, they overwrite the rest of the block with unused data when storing files. Except for changing the disk label on the EPS-16, you shouldn't need to write to this block.

00 80 01 00 00 0A 00 02 00 50 00 00 02 00 00 00 06 40 1E 02
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 49 44
(All values in Hexadecimal)

The EPS-16 PLUS has a disk name stored in the first occurrence of the above pattern. For the disk name 'DISK000', the first pattern would appear as follows:

```
00 80 01 00 00 0A 00 02 00 50 00 00 02 00 00 00 06 40 1E 02
00 00 00 00 00 00 00 00 00 00 FF 44 49 53 4B 30 30 30 49 44
                               ( D  I  S  K  0  0  0 )
```

Byte	Description
1	Peripheral Device Type
2	Removable Media Device Type
3	Various Standards Version #
4	Reserved for SCSI
5-6	Number of Sectors per Track (10 Sectors)
7-8	Number of Read/Write Heads (2 Heads)
9-10	Number of Cylinders (80 Tracks)
11-14	Number of Bytes per Block (512 Bytes)
15-18	Number of Blocks on Diskette (1600 Blocks)
19	SCSI Medium Type
20	SCSI Density Code
21-30	Reserved for later use
31-38	EPS-16 Disk Label (preceded by FF)
39-40	Device ID Signature = "ID"

SD-1 & VFX-SD OPERATING SYSTEM BLOCK (Block 2)

The Operating System Block for the VFX-SD contains the following 30 byte pattern repeated to fill the entire block:

```
00 00 06 29 00 00 00 00 00 01 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 4F 53
(All values in Hexadecimal)
```

The first four bytes represent the number of free blocks remaining on the diskette and changes as files are stored. Bytes 9 and 10 are used to indicate that the diskette is for the VFX-SD family instead of the EPS. The last two bytes are the ASCII characters "OS". The remainder of the block after the first occurrence of the pattern fills with unused data when files are stored on the diskette.

EPS & EPS-16 OPERATING SYSTEM BLOCK (Block 2)

The System Information Block for the EPS contains the following 30 byte pattern repeated to fill the entire block. MA, MI, RM, and RI are 00 except for the first occurrence of the pattern:

```
00 00 06 31 MA MI RM RI 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 4F 53
(All values in Hexadecimal)
```

The first four bytes represent the number of free blocks remaining on the diskette and changes as files are stored. If the operating system is not stored on the disk, MA, MI, RM, and RI are all 00. If the operating system is stored on the disk, MA is the major revision level, MI is the minor revision level, and RM & RI are the minimum internal ROM Revision level for the operating system stored on the disk. For the EPS rev. 2.40, MA = 02, MI = 28, RM = 01, and RI = 00 (hex). Bytes 9 and 10 are 00 for the EPS and EPS-16. The last two bytes are the ASCII characters "OS".

Once again, the keyboards appear to read only the first occurrence of each pattern. These patterns are for formatted disks without any files stored. Once files are stored, the information beyond the first occurrence of the patterns may change.

Ensoniq EPS & EPS-16 Directory Entries

The EPS and EPS-16 use the main directory to store file directory entries and sub-directory entries. When a sub-directory (File type 2) is created, the first entry in the sub-directory is set to File type 8 to point back to the parent directory. Each directory and sub-directory can hold 39 entries and there is no limit to the number of sub-directories that can be created. However, in most practical applications you would run out of disk space long before filling the directory.

Ensoniq VFX-SD & SD-1 Directory Entries

The VFX-SD & SD-1 use the Main Directory only to store the location of the four sub-directories. The directory entries and file information are always written into the sub-directories. Each sub-directory can hold 39 entries for a total of 156 files per disk (numbered 0 - 155). If the Sequencer Operating System is on the disk, it is stored starting at Track 1, Head 0, Sector 3 (Block 23). The directory entry for the Sequencer Operating System is stored in the last location of Sub-Directory 4 (directory entry # 155). Each directory entry contains 26 bytes of data as described below:

Directory Entry Format (EPS, VFX-SD, SD-1, EPS-16)

Byte	Information
01	Type-dependant Information (reserved on EPS)
02	File Type - see list of types
03-14	File Name (EPS 12 bytes) (VFX-SD 11 Bytes followed by 00)
15-16	File Size (in blocks)
17-18	Number of Contiguous Blocks
19-22	Pointer to First Block Location
23	File Number 0 - 59 for each VFX-SD file type, (reserved on EPS) (Multi File Index on EPS-16)
24-26	File Size (24 bit Byte Count) (VFX-SD), (reserved on EPS)

The File Number for the VFX-SD determines the bank number and position of the file when displayed on the keyboard. The first byte of a directory entry for some of the VFX-SD and SD-1 file types has the following definitions:

File Type	Definition & Possible Values
6 Programs	Bank # (0-9)
30 Programs	00 = Banks 0-4
	01 = Banks 5-9

```
10 Presets     00 = Bank A
               01 = Bank B
30 Sequences   00 (00 hex) = No Programs Stored  Banks 0-4
               01 (01 hex) = No Programs Stored  Banks 5-9
               16 (10 hex) = 30 Programs Stored  Banks 0-4
               17 (11 hex) = 30 Programs Stored  Banks 5-9
               32 (20 hex) = 60 Programs Stored  Banks 0-4
               33 (21 hex) = 60 Programs Stored  Banks 5-9
               (Bank Numbers are for Sequences AND Programs)
60 Sequences   00 (00 hex) = No Programs Stored
               16 (10 hex) = 30 Programs Stored  Banks 0-4
               17 (11 hex) = 30 Programs Stored  Banks 5-9
               32 (20 hex) = 60 Programs Stored
               (Bank Numbers are for Programs Only)
               The VFX-SD currently doesn't allow 30 Programs

Operating Sys  00 (hex) = VFX-SD Sequencer Operating System File
               FF (hex) = SD-1   Sequencer Operating System File
```

Ensoniq EPS, EPS-16, SD-1 & VFX-SD File Types

00 (00) = Unused (or Blank)
01 (01) = Eps Operating System
02 (02) = Sub-Directory
03 (03) = EPS Individual Instrument File
04 (04) = EPS Bank of Sounds
05 (05) = EPS Sequence File
06 (06) = EPS Song File
07 (07) = EPS System Exclusive File
08 (08) = Pointer to Parent Directory
09 (09) = EPS Macro File
10 (0A) = SD-1 or VFX-SD 1 Program File
11 (0B) = SD-1 or VFX-SD 6 Program File
12 (0C) = SD-1 or VFX-SD 30 Program File
13 (0D) = SD-1 or VFX-SD 60 Program File
14 (0E) = SD-1 or VFX-SD 1 Preset File
15 (0F) = SD-1 or VFX-SD 10 Presets File
16 (10) = SD-1 or VFX-SD 20 Presets File
17 (11) = SD-1 or VFX-SD 1 Sequence/Song File
18 (12) = SD-1 or VFX-SD 30 Sequence/Songs File
19 (13) = SD-1 or VFX-SD 60 Sequence/Songs File
20 (14) = SD-1 or VFX-SD System Exclusive File
21 (15) = SD-1 or VFX-SD System Setup File
22 (16) = SD-1 or VFX-SD Sequencer Operating System
23 (17) = EPS-16 Plus Bank File
24 (18) = EPS-16 Plus Effect File
25 (19) = EPS-16 Plus Sequence File
26 (1A) = EPS-16 Plus Song File
27 (1B) = EPS-16 Plus Operating System

(Values in parenthesis are in hexadecimal.)

EMPTY DIRECTORY BLOCKS

The first sector of an empty directory or sub-directory contains all zeros. The second sector contains all zeros except for the last two bytes of the sector. Those two bytes contain: 44h 52h which are the ASCII characters 'D' and 'R' respectively.

EMPTY FILE ALLOCATION BLOCK

An empty file allocation block contains all zeros except for the last two bytes of the sector. Those two bytes contain: 46h 42h which are the ASCII characters 'F' and 'B' respectively. Each File Allocation Block contains 170 three-byte entries. Each block on the disk has a corresponding entry in the file allocation blocks. Each zero entry indicates that the corresponding block is unused. A value of one indicates the end of a file. A value of two indicates a bad block on the diskette. Each non-zero entry points to the next block in a file. The first 15 entries for an EPS or EPS-16 disk and the first 23 entries for a VFX-SD disk are set to one. See the example below for clarification.

READING A FILE USING THE FILE ALLOCATION BLOCKS

To read a file from the disk, the system must first read the directory entry for the file to locate the beginning of the file. The directory entry points to the first block of the file. If part of the file is contiguous, the system would start reading the file at the first block and continue until all the contiguous blocks (as specified in the directory entry) have been read. After reading the last contiguous block, the system would read the corresponding entry in the file allocation block and read the block the entry points to. This process would continue until the file allocation entry equals one signifying the end of the file. Refer to Alan Smith's two articles in the Transoniq Hacker for more information on reading Ensoniq Dos files. (Issue #45, page 11; Issue #70, page 9)

To write a file to the disk, the system must first check to see if the file name is already being used. If so, the system should prompt the user to see if the existing file should be deleted. If so, the system should delete the file and then determine if there is enough free blocks on the disk to hold the new file. On the VFX-SD, the system must also make sure there is an available file number for that file type. If so, the system should read the file allocation table (FAT) from the disk. If the disk has enough contiguous free blocks to hold the file, the system should store the file as one contiguous file. This speeds up access to the file since the system doesn't have to keep looking at the FAT to locate the next block. However, the FAT is maintained and updated even if the file is contiguous. If there isn't enough contiguous free blocks, the system should locate the first free block and start storing the file. After writing each block, set the FAT entry for that block to point to the next free block. This process would continue until the last block was written to the disk. Set the FAT entry for the last block to 01 to indicate the end of the file. Write the updated FAT back to the disk. Subtract the file size (in blocks) from the number of remaining free blocks and rewrite the system information block on the disk. Finally, write the directory entry to the disk.

File Allocation Table Example

End File	End File	End File	Next Blk	Empty	Empty
00 00 01	00 00 01	00 00 01	00 00 23	00 00 00	00 00 00

The first three entries in the above FAT indicate that the end of

the file has been reached after reading the corresponding block. The next entry indicates the system should read block 23 (hex) after reading the corresponding block. The next two entries indicate that the corresponding blocks are empty and available for use. Again, refer to Alan Smith's articles for more information on the file allocation tables.

SQ-80 Disk Format

The SQ-80 has a completely different format than the keyboards mentioned above. The disk contains data on both sides with 80 tracks numbered 0 - 79 on each side. However, each track has five 1024 byte sectors numbered consecutively from zero to four followed by one sector of 512 bytes with a sector ID of five. I will refer to the two sides of the disk by referring to the disk drive head used to read each side. The heads are numbered 0 and 1. Data is stored on both sides of each track before moving to the next track. However, when switching tracks, the system remains on the same head as the previous track. The following examples should clarify this.

```
TK HD SC
 0  0 0-5  data is first stored on Track 0, Head 0, Sectors 0-5
 0  1 0-5  data is next stored on Track 0, Head 1, Sectors 0-5
 1  1 0-5  data is then stored on Track 1, Head 1, Sectors 0-5
 1  0 0-5  data is then stored on Track 1, Head 0, Sectors 0-5
 2  0 0-5  data is then stored on Track 2, Head 0, Sectors 0-5
           this process continues until....
79  0 0-5  the last track - Track 79, Head 0, Sectors 0-5
           (Track 79, Head 1, Sectors 0-5 were written first)
```

SQ-80 Sector Information

Finding your way around an SQ-80 diskette is not an easy task. Some of the files (and the directory) are stored on the smaller sectors, while others are stored on both! All files on the SQ-80 are stored in fixed locations eliminating the need for a file allocation table. The format allows for ten (10) large data files (any combination of one-sequence, all-sequence, or system-exclusive files). Each file occupies 64 contiguous large sectors (sectors 0-4) of the diskette. In addition, each of the ten files occupies one small sector (sector 5) located on the same track and head as the last large sector. (Simply set the sector number to five after reading the last large sector). This small sector is only used to store header information for the all-sequence files. It is not used for the other two types.

The format also allows for 40 program bank files. These files occupy four large sectors each starting at Track 64, using the remaining large sectors on the diskette. The diskette directory occupies the first four small sectors of the diskette as shown below. The remaining small sectors are used to store the 128 individual programs (sounds) with each program occupying one sector. However, the program files are not on alternating sides. The first 64 are stored on head 0, and the last 64 are stored on head 1. The ten program locations which conflict with the small sectors occupied by the large data files are stored in the small sectors starting at Track 66, Head 0, Sector 5. (I'm not making this up - this is really the way it works!)

TK	HD	SC	
0	0	5	Directory (First Sector)
0	1	5	Directory (Second Sector)
1	1	5	Directory (Third Sector)
1	0	5	Directory (Last Sector)
2	0	5	Program # 1
3	0	5	Program # 2
4	0	5	Program # 3
5	0	5	Program # 4
7	0	5	Program # 6 (Program # 5 stored at 66,0,5)
8	0	5	Program # 7
			Continues until....
65	0	5	Program # 64
2	1	5	Program # 65
3	1	5	Program # 66
			Continues until....
65	1	5	Program # 128
0	0	0	Large Data File # 1 (First Sector)
6	0	3	Large Data File # 1 (Last Sector)
6	0	4	Large Data File # 2 (First Sector)
6	0	5	Large Data File # 1 (Small Sector)
12	1	2	Large Data File # 2 (Last Sector)
12	1	3	Large Data File # 3 (First Sector)
12	1	5	Large Data File # 2 (Small Sector)
19	1	1	Large Data File # 3 (Last Sector)
19	1	2	Large Data File # 4 (First Sector)
19	1	5	Large Data File # 3 (Small Sector)
25	0	0	Large Data File # 4 (Last Sector)
25	0	1	Large Data File # 5 (First Sector)
25	0	5	Large Data File # 4 (Small Sector)
31	0	4	Large Data File # 5 (Last Sector)
31	0	5	Large Data File # 5 (Small Sector)
32	0	0	Large Data File # 6 (First Sector)
38	0	3	Large Data File # 6 (Last Sector)
38	0	4	Large Data File # 7 (First Sector)
38	0	5	Large Data File # 6 (Small Sector)
44	1	2	Large Data File # 7 (Last Sector)
44	1	3	Large Data File # 8 (First Sector)
44	1	5	Large Data File # 7 (Small Sector)
51	1	1	Large Data File # 8 (Last Sector)
51	1	2	Large Data File # 9 (First Sector)
51	1	5	Large Data File # 8 (Small Sector)
57	0	0	Large Data File # 9 (Last Sector)
57	0	1	Large Data File #10 (First Sector)
57	0	5	Large Data File # 9 (Small Sector)
63	0	4	Large Data File #10 (Last Sector)
63	0	5	Large Data File #10 (Small Sector)
64	0	0	Program Bank # 1 (Sector 1 of 4)
64	1	0	Program Bank # 2 (Sector 1 of 4)
65	1	0	Program Bank # 3 (Sector 1 of 4)
65	0	0	Program Bank # 4 (Sector 1 of 4)
			Continues until ....
79	0	0	Program Bank # 40 (Sector 1 of 4)

Ensoniq SQ-80 Directory Sectors

The SQ-80 does NOT use its directory for the 128 individual programs stored on the disk. These files are stored in a fixed location on the disk and must be read directly to obtain the names of the sounds. Banks of sounds and the ten 64K files are listed in the regular directory but the position of the directory entries is fixed. This eliminates the need for a file allocation table since the files are in fixed locations. The first 10 entries are for the ten large data (64K) files for sequences, all sequences, and system exclusive files. The next 40 entries in the directory are for the 40 bank files.

Ensoniq SQ-80 Directory Entries

Each Entry contains 13 bytes of data.

Byte	Information
01	File Type - see list of types
02-11	File Name (10 Bytes)
12	Type-dependant Information
13	Type-dependant Information

The SQ-80 File Names can contain some unusual characters since the keyboard used some non-ACSII characters. See Joe Slater's article (Issue #70, page 13) to find out the meaning of these characters. In fact, Joe's entire series of articles on the ESQ-1 are valid for the SQ-80 as well.

Ensoniq SQ-80 File Types

00 (00) = Unused (Blank)
01 (01) = Operating System
02 (02) = Program Bank File
03 (03) = All Sequence File
04 (04) = One Sequence File
05 (05) = System Exclusive File
06 (06) = Single Program File

Mirage Disk Format

The Mirage has a format similar to the SQ-80 keyboard mentioned above. However, the disk only contains data on one side of the disk with 80 tracks numbered 0 - 79. Like the SQ-80, each track has five 1024 byte sectors numbered consecutively from zero to four followed by one sector of 512 bytes with a sector ID of five. The following examples should clarify this.

TK	SC	SIZE	
0	0-4	1024	data is first stored on Track 0, Sectors 0-4
0	5	512	data is next stored on Track 0, Sector 5
1	0-4	1024	data is next stored on Track 1, Sectors 0-4
1	5	512	data is next stored on Track 1, Sector 5
			this process continues until...
79	5	512	the last track - Track 79, Sector 5

The diskette may contain the Operating System, six sounds config-

ured as 3 lower-half keyboard sounds and 3 upper-half keyboard sounds and either eight short sequences or three long sequences. The first 11K of the Operating System is stored on both small and large sectors from Track 0, Sector 0, to Track 1, Sector 5. The remaining 5k of the Operating System is stored only on small sectors (Sector 5) from Track 2 to Track 10. The configuration parameters are stored on Track 11, Sector 5. The directory and the sequences are only stored on the small sectors (Sector 5) and the sound files are only stored on the large sectors (Sectors 0-4).

TK	SC	
2	0	Sound # 1, Lower Half, Parameters (1 Sector)
2	1	Sound # 1, Lower Half, Data (64 Sectors)
15	0	Sound # 1, Upper Half, Parameters (1 Sector)
15	1	Sound # 1, Upper Half, Data (64 Sectors)
28	0	Sound # 2, Lower Half, Parameters (1 Sector)
28	1	Sound # 2, Lower Half, Data (64 Sectors)
41	0	Sound # 2, Upper Half, Parameters (1 Sector)
41	1	Sound # 2, Upper Half, Data (64 Sectors)
54	0	Sound # 3, Lower Half, Parameters (1 Sector)
54	1	Sound # 3, Lower Half, Data (64 Sectors)
67	0	Sound # 3, Upper Half, Parameters (1 Sector)
67	1	Sound # 3, Upper Half, Data (64 Sectors)
20	5	Short Sequence # 1 (4 Sectors)
35	5	Short Sequence # 2 (4 Sectors)
55	5	Short Sequence # 3 (4 Sectors)
24	5	Short Sequence # 4 (4 Sectors)
28	5	Short Sequence # 5 (4 Sectors)
39	5	Short Sequence # 6 (4 Sectors)
43	5	Short Sequence # 7 (4 Sectors)
59	5	Short Sequence # 8 (4 Sectors)
12	5	Long Sequence # 1 (16 Sectors)
35	5	Long Sequence # 2 (16 Sectors)
55	5	Long Sequence # 3 (16 Sectors)

Mirage Directory Sectors

The directory information for the Mirage is contained in three bytes which are stored in three sectors of the diskette. Each sector contains 512 copies of the directory byte for that sector. This was done because there wasn't any buffer space to read a whole directory sector. Therefore, the Mirage would read (or write) the sector and would use the last byte of the sector as the valid value for the directory byte. Track 32, Sector 5 contains the Sound Directory Byte. Track 33, Sector 5 contains the Short Sequence Directory Byte and Track 34, Sector 5 contains the Long Sequence Directory Byte. The directory bytes are defined as follows:

SOUND DIRECTORY BYTE		SHORT SEQUENCE DIRECTORY BYTE		LONG SEQUENCE DIRECTORY BYTE	
Bit 0	Not Used	Bit 0	Seq. 1	Bit 0	Seq. 1
Bit 1	Sound 1 Lower	Bit 1	Seq. 2	Bit 1	Seq. 2
Bit 2	Sound 1 Upper	Bit 2	Seq. 3	Bit 2	Seq. 3
Bit 3	Sound 2 Lower	Bit 3	Seq. 4	Bit 3	Not Used
Bit 4	Sound 2 Upper	Bit 4	Seq. 5	Bit 4	Not Used
Bit 5	Sound 3 Lower	Bit 5	Seq. 6	Bit 5	Not Used
Bit 6	Sound 3 Upper	Bit 6	Seq. 7	Bit 6	Not Used
Bit 7	Not Used	Bit 7	Seq. 8	Bit 7	Not Used

If the Sound or Sequence exists, the appropriate bit is set to one (1). If not, the bit is cleared (0).

As the various formats suggest, Ensoniq has come a long way since the Mirage single-sided drives. I would like to suggest that they consider the possibility of using the higher density 1.44 megabyte diskette drives in the future. The drives (and diskettes) are not that much more expensive, but they would make a big difference especially with the EPS-16. The lower density diskettes could still be used with the higher density drives giving us the option of which diskettes we want to buy. If new formats are introduced, I will try to include their formats in future issues of the Hacker. If you have questions concerning the formats, please feel free to contact me. I have used the information in this article to restore files on diskettes which had been 'trashed' by some glitch in the keyboards. If you have a disk you thought was forever lost, you may be able to recover the files using this information. In my next article, I will cover the format used for the VFX-SD Sequencer.