SOURCE
https://erichizdepski.wordpress.com/2019/09/15/ensoniq-mirage-dsk-8/

Mirage Disk File Format

I put this here just in case someone is really bored.
Ensoniq Mirage Disk Format The Mirage has a format similar to the SQ-80 keyboard. 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 configured 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)

TK SC
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
Bit 0 Not Used
Bit 1 Sound 1 Lower
Bit 2 Sound 1 Upper
Bit 3 Sound 2 Lower
Bit 4 Sound 2 Upper
Bit 5 Sound 3 Lower
Bit 6 Sound 3 Upper
Bit 7 Not Used

SHORT SEQUENCE DIRECTORY BYTE
Bit 0 Seq. 1
Bit 1 Seq. 2
Bit 2 Seq. 3
Bit 3 Seq. 4
Bit 4 Seq. 5
Bit 5 Seq. 6
Bit 6 Seq. 7
Bit 7 Seq. 8

LONG SEQUENCE DIRECTORY BYTE
Bit 0 Seq. 1
Bit 1 Seq. 2
Bit 2 Seq. 3
Bit 3 Not Used
Bit 4 Not Used
Bit 5 Not Used
Bit 6 Not Used
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. — Gary Giebler
I just joined the fdutils list, hoping to figure out how to make use of fdutils to create a command-line utility to format, read, and write 3.5 DD disks for the Ensoniq Mirage on Linux. The Mirage is a strange, wonderful instrument - it's an 8-bit sampler from the 80's cursed by a limited amount of storage on a very strangely arranged 3.5" DD disk.

Mirage disks are formatted this quite weird way:

o single-sided
o 80 tracks
o each track contains five type 3 sectors (1024 b) + one type 2 sector

(512 b)

So the real problem here is that the each track contains 2 sector types. I have successfully *read* these disks, by dumping out each sector. Here is a bash loop that dumps out the Mirage disk data:

```
do
  fdrawcmd read 0 $i 0 0 3 6 0x1b 0xff \
           length=5120 rate=2 need_seek track=$i \
           > $REPERTOIRE/d$i.dmp 2> /tmp/mirage-data.err
  fdrawcmd read 0 $(($i)) 0 5 2 6 0x1b 0xff \
           length=512 rate=2 need_seek track=$i \
           > $REPERTOIRE/s$i.dmp 2> /tmp/mirage-small.err
  cat /tmp/mirage-data.err | xargs echo
  cat /tmp/mirage-small.err| xargs echo
done
```

So we have gotten to the point where we can use some rustic "fdrawcmd" invocations in a script to dump out the sample data. I have examined the results. THIS DOES WORK. So at least we understand the disk format to the point where we can dump out the wavesamples and other data using a script.

What I'd like to do now is to be able to read, write, and format these disks (cp, fdformat, etc.). I understand that the out-of-the-box fdutils will probably not do what is needed for this disk format that has 2 sector sizes on each track.

The question is this - what is the best way for me to proceed? Do I need to write a program that does "fdrawcmd"-type stuff, or can I do some driver/kernel level work to accomodate the Mirage format, and then use the standard Linux utilities??

Please advise - I am ready to do some hard work on this and just need some direction.
You should be fine using overlapping sectors to set up the layout. For the format size code in the command, use the smallest sector you need on the track. For larger sectors pad out the size as multiples of this smallest sector size, using a valid header for the first sector, and dummy values for the padding.

i.e. to write a track containing 3 sectors of sizes 1024, 512 and 256, use (something roughly like):

```
cat > track_data
00 00 01 03
00 00 99 00
00 00 99 00
00 00 99 00
00 00 02 02
00 00 99 00
00 00 03 01
^D

fdrawcmd format 0 2 7 0x2b 0xf6 track=0 < track_data
```

The formatted track still contains 7 sectors at this point, but once you write data to each, the data from the larger sectors will overwrite the dummy headers leaving only the 3 sectors we want. You didn't mention the density or sectors sizes you need, so you'll need to tweak the above for your own requirements.

Just pointing out a minor error: fdrawcmd takes binary input, not ASCII. So your track data file would need to be filled with binary charachers, as follows (where echo is bash's or zsh's builtin echo command):

```
echo -ne '\x00\x00\x01\x03'  >track_data
echo -ne '\x00\x00\x99\x00' >>track_data
echo -ne '\x00\x00\x99\x00' >>track_data
echo -ne '\x00\x00\x99\x00' >>track_data
echo -ne '\x00\x00\x02\x02' >>track_data
echo -ne '\x00\x00\x99\x00' >>track_data
echo -ne '\x00\x00\x03\x01' >>track_data

fdrawcmd format 0 2 7 0x2b 0xf6 track=0 < track_data
```

```
Mirage SYSEX data transfer format (load samples via MIDI)
Related to MIDI Sample Dump Standard.
Download Program Data from Mirage
---------------------------------
11110000    0xf0    System Exclusive
00001111    0x0f    Ensoniq code
00000001    0x01    Mirage code
000N0011    0x 3    N=0: lower N=1: higher
[ receive program data ]


Upload Program Data to Mirage
-----------------------------
11110000    0xf0    System Exclusive
00001111    0x0f    Ensoniq code
00000001    0x01    Mirage code
000N0101    0x05    N=0: lower N=1: higher
[ send program data ]

Program Data Format
-------------------
Offset  Len Description
0   1   Sound Rev Level
1   24  Wavesample Control Block #1
25  24  Wavesample Control Block #2
49  24  Wavesample Control Block #3
73  24  Wavesample Control Block #4
97  24  Wavesample Control Block #5
121 24  Wavesample Control Block #6
145 24  Wavesample Control Block #7
169 24  Wavesample Control Block #8
193 32  Segment List, Wave Sample #1
225 32  Segment List, Wave Sample #1
257 32  Segment List, Wave Sample #1
289 32  Segment List, Wave Sample #1
321 32  Segment List, Wave Sample #1
353 32  Segment List, Wave Sample #1
385 32  Segment List, Wave Sample #1
417 32  Segment List, Wave Sample #1
449 32  Segment List, spare
481 36  Parameter Block, Program #1
517 36  Parameter Block, Program #2
553 36  Parameter Block, Program #3
589 36  Parameter Block, Program #4

Total Length: 695 Bytes
```