The Q1 Floppy Data Format and address mark
Author: Karl Wacker 26.7.2024

I had a chance to do a bit of reverse engineering on the Q1 double density data format and address mark. Please excuse any errors in this and other documents - I am recreating the information from a partial reverse engineering of the ROM listings and what I remember from when I was at Q1 at least 45 years ago!

Please pass this along to the others so that those who are trying to read the floppy disks understand the quirks of the Q1 DD format.

I hope all is well on your side of the Atlantic.

Karl

The Q1 double density format was developed before the industry standard double density address mark encoding was announced.

This and the fact that Q1 floppy disks are recorded at a slightly lower data rate so that then had less problems with the early Shugart SA800 floppy drive's bit smearing on playback.

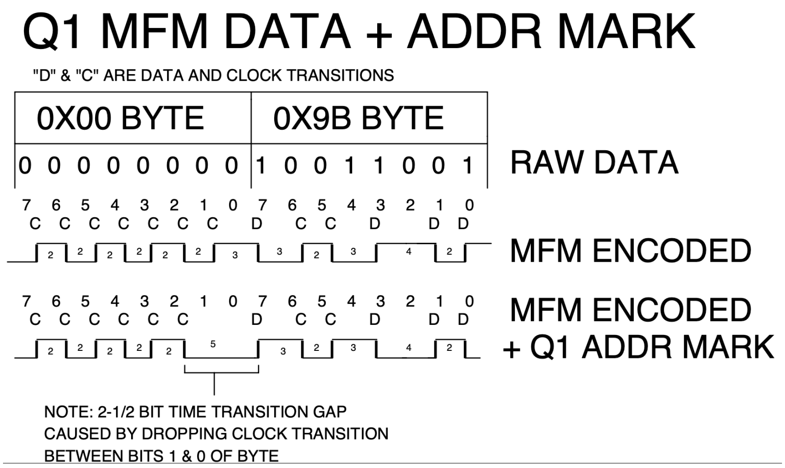
Industry standard double density is based around using a 8MHz or 16MHz crystal for write timing.

Q1 adapted a 7MHz / 14MHz crystal timing instead.

The Q1 address mark is a gap of 2-1/2 bit times between transitions on the MFM encoded data, instead of the normal 1, 1-1/2 or 2 bit times to encode the MFM data.

The data is encoded with the assumption that in the two 1/2 bit time cells for encoding, the data bit is encoded in the first 1/2 bit time of the cell, and the clock in the second 1/2 bit time of the cell.

I've included a timing diagram of the address mark encoding.



The Q1 floppy controller primarily read and writes the proprietary Q1 format.

A later document will describe the floppy formatting process

Note that the disk controller uses the WAIT* signal to "freeze" the Z80 and synchronize it to the byte transfer logic in the disk controller [there is no 'busy test bit for data transfers and that doing a I/O read from port 9 writes 0x00's to the drive w/o generating an address mark.]

The format consists of sets of header and data records.

The header record consists of the following:

1) several '0x00' bytes
2) a 0x00' byte with the Q1 address mark embedded in it
3) a "0x9E identifier byte
4) the disk track #
5) the sector #
6) the record cksum [modulo 256 suml] of 0x9e + track # + sector #
7) a 0x10 trailer byte
8) a 0x00 byte

The data record consists of

1) 5 0x00 bytes [as a spacer for timing]
2) a 0x00' byte with the Q1 address mark embedded in it
3) a 0x9B identifier byte
4) from 8 o 511 bytes of data
5) a cksum byte [modulo 256 sum} of the data
6) a 0x10 trailer byte
7) a 0x00 byte

After each data record, while the disk is being formatted, a data gap consisting of [3/32 times the sector length] bytes to allow for speed variations in the drive's rotational speed.

Reading a record:

[there are implied retry limits if an error occurs in each of the following steps]

1) copy offsets 0 & 1 to 16 & 17 in file descriptor, the desired drive is selected, and tested for a ready state. if not ready, an error code is returned,
2) a check is made of the current track # by doing a read of the next readable header and if the cksum is good, the track's # is saved. the desired track & record # are computed from # in offsets 0 & 1 in file descriptor
3) the drive is stepped to what it thinks is the correct track.
4) the next readable header's track # is read and and if incorrect, step 3 is repeated
5) the record # of the header record is compared against the desired one.
6) if it is the wrong sector #, goto step 4
7) wait [in a small time window] for an address mark and 0x9B data.
8) read the next bytes [offsets c & d in file descriptor], and then the cksum and 0x10 trailer byte
9) repeat from step 4 if there is a bad cksum or missing 0x10 byte
10) update file descriptor offsets 0&1
11) exit if done, otherwise repeat for the next record[s]

Writing a record:

[there are implied retry limits if an error occurs in each of the following steps]

verify if the file is writable [msb of Last track # in file descriptor]

Steps 1 thru 6 are the same as for reading a record
7) turn on write gate
8) read several bytes [will write them w/o address mark]
9) write 0x00 byte to generate 0x00 byte with address mark timing
10) write 0x9B byte, zero the cksum
11) write desired # bytes [offset c & d in file descriptor]and add to the cksum
12) write computed [modulo 256] cksum byte
13) write 0x10 trailer byte
14) write 0x00 [trailer byte
15) turn off write gate to floppy drive

16) update 0 & 1, a & b in file descriptor
17 exit if done otherwise repeat for the next record[s]

Karl