RLL encoding

The encoding scheme called run length limited (RLL) is useful for squeezing the largest possible amount of data onto a hard disk drive. To understand how encoding schemes work, let's look at the three most common ones used today: frequency modulation (FM), used on older floppy drives; modified frequency modulation (MFM), used on current floppy disk drives and many hard disk drives; and 2,7 RLL (used on most RLL hard disk drives).

Data on a magnetic disk is recorded as a series of pulses and silences. In the FM encoding scheme, each 1 or 0 is represented by a pattern consisting of pulses and silences. For example, a pulse followed by a silence is a 0, while two consecutive pulses is a 1. The pulse that's always there is called the clock pulse. Because there is a clock pulse in every bit, it's easy for the controller to keep pace with the data as it comes in (a process known as 'clock extraction').

Fig A shows why this technique is called FM. Twice as many pulses occur per unit of time during a string of 1s than during a string of 0s, and the average (for an even mix of 1s and 0s) is 1.5 pulses per bit.

The constraint that determines how much data you can get on a disk is simple: there must be enough space between pulses so that they don't run together. FM encoding always leaves room for two pulses per bit, in case that bit is a 1. The maximum number of bits you can have, therefore, is always half the maximum number of pulses you can fit in. There is, however, a way to use fewer pulses to represent the same data. This is the idea behind MFM (see Fig B).

In MFM, the encoding rule is as follows: a 1 is represented by a silence followed by a pulse, while a 0 is represented by one of two patterns: a pulse followed by a silence if no pulse oc-

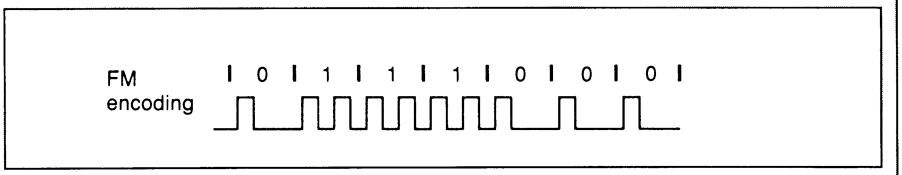

Fig A In FM encoding, each bit is represented either by a pulse and a silence (0) or by two consecutive pulses (1)

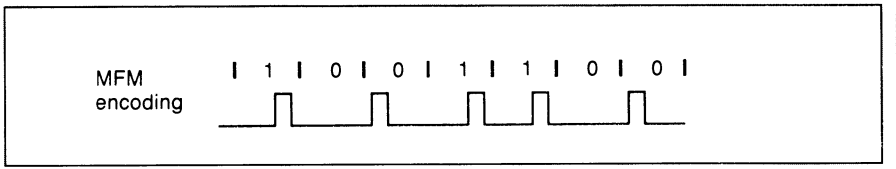

Fig B In the MFM encoding scheme, all pulses are separated by at least one silence. Since the amount of data that can fit on a disk depends on the closeness of successive pulses, MFM allows twice the data density of FM encoding

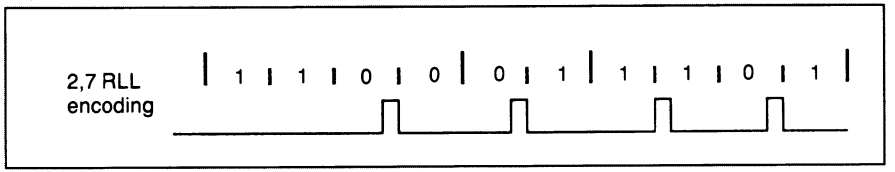

Fig C Here's how a sample bit pattern is encoded in the 2,7 RLL scheme. Each code group is 4 to 8 half-bits long and is encoded from a code group of 2 to 4 data bits. The length of the pattern corresponds to the length of the original data, but the pulses are guaranteed to maintain the required minimum and maximum spacings

curred at the end of the previous bit, or by two silences If a pulse did occur at the end of the previous bit.

The MFM scheme guarantees that there will always be at least one silence between pulses (so that they can be packed more tightly without running together), but no more than three (so that a clock can still be recovered). This pattern yields an average of 0.75 pulse per bit (assuming that 50 per cent of the 0s are represented by each of the two possible patterns), and it therefore lets you pack the bits twice as closely together. For this reason, when MFM floppy disks first came out, they were called double-density disks.

ST506 hard disk drives originally used MFM encoding. Is there another encoding scheme that could increase the density still further? To answer this question, let's review the schemes just discussed in terms of run lengths, the minimum and maximum numbers of consecutive silences in each encoding scheme.

FM allows a minimum run length of 0 (it's possible to have no silences between pulses) and a maximum run length of 1 (there's always a clock pulse after a silence). So, one way to describe FM is as 0,1 run-length-limited encoding, or 0,1 RLL for short.

Similarly, MFM always has at least one silence between the pulses, but more than three — making it 1,3 RLL. It's the minimum run length that determines how tightly data can be packed onto the disk, while the maximum run length determines how accurate the controller must be at timing when the pulses come in (so that it can generate a clock to go with the data).

The encoding scheme we know simply as RLL is usually 2,7 RLL (see Fig C and Table A). It uses a more complex set of rules to determine the pulse pattern for each bit based on the values of the preceding bits, but the principle is the same: there are fewer pulses, but their precise positions convey more information about the original data pattern.

Data bits to be encoded	2,7 RLL encoding (0 = silence, 1 = pulse)
0 0	1 0 0 0
0 1	0 1 0 0
1 0 0	0 0 1 0 0 0
1 0 1	1 0 0 1 0 0
1 1 0 0	0 0 0 0 1 0 0 0
1 1 0 1	0 0 1 0 0 1 0 0
1 1 1	0 0 0 1 0 0

Table A The 2,7 RLL scheme encodes groups of 2 to 4 bits into pulse patterns. Note that there are always at least two, and no more than seven, silences between pulses regardless of the combination of bits encoded