As V5 decreases, the amount of time spent in the time loop is also decreased, and the trail speed increases - right on through the ceiling (or, as is more likely, one of the four walls, and 20 points for you-know-who!)

The next section, beginning at 0272, controls the target placement. First a random number is set into VE and the number is tested to see if it's equal to zero (for a decision whether to display a new target on the screen). The percentage of time a target stays where it is may be changed with a new mask at line 0272. Remember, though, you may only use numbers with the CXKK instruction that contain unbroken bit positions (speaking binarily) from right to left. If you are confused by all this, write down - in binary - the following numbers: 01, 03, 05, 07, 0F, 3F, 5F, 7F, FF, and you'll see what I mean. The number 05, for example, won't work because it contains binary "holes" (o's) which defeat it's ability to mask the random number. Changing 0272, then, to CE1F will permit the target to remain on the screen a longer time. As the target is sometimes off, however, this will also result in its remaining off for a longer time. Thus, by trying to make the game easier, you may make it harder. (I told you computers weren't fair.)

If the decision is to display a target, the "target on" flag (VA) is tested to see if there is an old target already being displayed. (Replace the instruction at 027A with a 127C; GO TO 027C instruction; and you'll see why the flag is necessary.)

Lines 027E - 0298 choose new random XY coordinates for the target and limit these to the screen edges. This keeps the target off the borders if there are any, and eliminates target wrap-around, if there aren't any borders. It also keeps the target in a testable range for checking to see if it was hit, although this could be done in other ways. (Masking the X coordinate by a logical "AND" with 3F, for instance.)

The program then displays the target, checking to see if it ran into anything. Obviously, the trail can be anywhere