of course, must be an even number.) However, the trail speed uses V7 to decide when to speed up, as previously described. This will have to be adjusted if you program a shorter trail, or the trail will never speed up. With imagination, you might want to program a variable-length trail, but I leave this as an exercise to the reader.

When the computer scores, the trail erases in reverse. If you change the size of the trail, you'll also need to change locations 0316, 031C, and 033A to reflect the changes. These locations are well marked in the program listing. 0316 and 031C use a value of the trail length minus two, while 033A uses the value of the exact length. To be exactly correct, the trail is always one bip shorter (two bytes of storage) than I have said it is, since a bip is being taken off each time a new bip is being created. So while the trail is 80 bips long in memory, it is 79 bips long on display.

Locations 02CA-02F6 determine if a target was hit. The trail XY coordinates must first be masked to the effective range of the target with logical "AND" instructions. (This is only in the wrap-around version, where the trail may hit a target while having different XY coordinates.) VX=0 and VX=40 are at the same screen location. 40 AND 3F will equal zero. Remember, the target XY coordinates were limited to this range earlier in the program. The routine here simply loops through all the possible XY coordinates (9) of the target after a hit is detected. If any of these match those of the trail, then a target was hit and the user scores. If not, the computer scores. The result of the test is indicated by V0=1, a temporary flag, rather than simply exiting from the test. This allows the target's XY coordinates to be easily reset in order to erase the target. (On second thought, the target could have been erased first, then the check made. This proves my theorem than any program longer than zero bytes can be shortened. The most efficient size for any program must therefore be equal to zero. But it's only a theory.)