

SPRITE FLICKER ROUTINE

The TMS9918a VDP can only display 4 sprites on a line. If your program has more than 4 sprites on a line, there is a “flicker” routine that can display all the sprites by rapidly flickering them. This can be used on standard XB or XB256, and it can be compiled. This was adapted from the flicker routine by Tursi. I have modified Tursi's code to be a little more compact. Also, instead of flickering all 28, it will only flicker the necessary sprites, which helps speed things up. If you are using 5-8 sprites it will rotate 8 sprites. 9-12 will rotate 12, 13-16 will rotate 16, and so on up to 28. The routine knows how many sprites to rotate by looking at the value in >837A, which is the highest number sprite in motion. It will always flicker the sprites, even when there are less than 5 on a line.

If you are using standard XB, first CALL INIT, then CALL LOAD("DSKn.HWFLICKER.OBJ") to load the flicker routine.

XB256 has no room in low memory for the flicker routine. It must be loaded into high memory and embedded in the XB program. This is easy to do. Type OLD DSKn.HMFLICKER to load a short XB program that has the flicker routine embedded in the code. After this, you can enter XB program lines, merge them, or paste them in using Classic99. But do not do a NEW because that will erase the flicker routine. When you save the program, the flicker routine will be saved along with the program.

After the program line that creates the highest number sprite, you can activate/deactivate the flicker routine.

CALL LINK("FLICK") to turn on the flicker routine.

CALL LINK("FLICKX") to cancel the flicker routine.

Before activating the flicker routine, be sure the highest number sprite is set into motion. If your sprites are stationary, you still must set the highest number sprite in motion, but with zero velocity. The line 100 CALL SPRITE(#12,65,16,92,120,0,0) sets sprite #12 in motion, even though the velocities are zero. As usual in XB, keep the sprite numbers as low as possible. i.e. start with sprites #1,#2,#3 instead of #28,#27,#26.

If you just want to compile but without testing in XB, you can add CALL LINK("FLICK") to the program as described above. (This is built into RUNTIME10 of the compiler, so you do not have to convert FLICK to lower case.)

The program below creates 18 sprites, sets them into motion, then starts the flicker routine.

```
5 CALL MAGNIFY(2)
10 CALL CLEAR
20 CALL LOAD(-31806,64) !turn off auto sprite motion
30 A=1
32 VM=-2
40 CALL SPRITE(#A,64+A,2,100,A*8,VM,0)
50 A=A+1 :: IF A=19 THEN 60
52 VM=VM+1 :: IF VM=3 THEN VM=-2
54 GOTO 40
60 CALL LOAD(-31806,0) !enable auto sprite motion
65 CALL LINK("FLICK")
70 GOTO 70
```

Since the highest sprite is #18, only the sprites from 1 to 20 will be included in the flicker routine. After the flicker routine is started up, sprites can be modified or deleted as necessary and the routine will still flicker sprites 1 to 18.

I don't know why my notes say this, and it may not be an issue or not, but there may be problems with memory usage if your program uses:

CALL LINK("CRAWL") The Starwars text crawl

CALL LINK("CHSETD") to load the lower case definitions

Speech

Disk access