

mvac7 blog

13 febrero 2017

Sprites MSX · La OAM

Cuando tengamos definidos una serie de patrones de Sprites, podemos visualizarlos desde el lenguaje Basic con la instrucción [PUT SPRITE](#), pero otra forma que podemos utilizar, sobretodo si trabajamos en Ensamblador o con C, se basa en escribir en una zona concreta de la VRAM llamada Object Attribute Memory (OAM).

En esta tabla se guarda la información de visualización de los planos de Sprites. Utiliza 4 Bytes para cada uno de los 32 planos disponibles, dispuestos en orden consecutivo y podemos definir la siguiente información:

1. Coordenada Y
2. Coordenada X
3. Número de patrón.
4. Color + EC (bit 7 Early Clock)

En los modos gráficos G1, G2 y MC (screen 1, 2 y 3), del TMS9918, esta se encuentra en la posición 1B00h, mientras que en los modos gráficos del [V9938](#) esta tabla se ubicará en diferentes posiciones de la VRAM (ver Tabla 1).

Modo	VRAM
G1, G2 y MC	1B00h
G3	1E00h
G4 y G5	7600h
G6 y G7	FA00h

Tabla 1. Tabla de Atributos de Sprite (OAM).

En los nuevos modos gráficos del V9938, el valor de color no se utiliza, ya que se necesita 16 bytes por plano de Sprite. Para ello se dispone de una extensión de la OAM. Como en los sprites del TMS9918, además de la información de color, también se incluye un bit para el Early Clock, otro para la combinación de colores (OR) y uno más para la colisión de Sprites (ver post: [sprites del V9938](#)). Esta tabla se encuentra en diferentes posiciones de la VRAM, dependiendo del modo de pantalla (ver tabla 2).

Modo	VRAM
G3	1C00h
G4 y G5	7400h
G6 y G7	F800h

Tabla 2. Tabla de atributos de colores de Sprite.

Un detalle a tener en cuenta, es que cuando trabajemos con Sprites de 16x16, al asignar el número de patrón a un plano, tendremos que multiplicar su valor por 4, ya que es la forma interna de trabajar del VDP. Esto es debido a que se componen de 4 sprites de 8x8.

Si programamos en ensamblador, la BIOS del MSX contiene funciones que nos pueden facilitar algunas tareas. La función CALATR (0087h), nos devolverá en HL la dirección de la VRAM donde se encuentra la información del OAM, del número de plano indicado en A. Además, se ajusta al modo de pantalla que estemos utilizando, independientemente de la generación de MSX que se trate.

En el desarrollo de juegos, la visualización de Sprites se puede trabajar de varias formas: Podemos tocar los Sprites directamente en la VRAM cuando se requiera, como haríamos en Basic con Put Sprite, o se puede crear un buffer en RAM, que volcaremos a la VRAM en la interrupción de VBlank. Esta segunda puede ser la mejor opción si utilizamos muchos Sprites, sobretodo por que el acceso a RAM siempre será más fácil y rápido que a la VRAM.

Desde Basic también puede ser útil acceder directamente a la OAM con VPOKE, para casos que solo tengamos que modificar un parámetro de un plano de Sprite, como por ejemplo si queremos que se desplace horizontalmente solo necesitaríamos tocar el valor para la coordenada X.

Ejemplos

Acceso a la OAM desde Basic:

```

10 REM EJEMPLO ACCESO AL OAM
20 SCREEN 1,2:REM screen 1 con sprites de 16x16
30 IX=BASE(8)::REM OAM para screen 1,2 y 3
40 PAT=BASE(9):REM tabla de patrones de sprite
50 FOR BC=PAT TO PAT+(2*32):VPOKE BC,255:NEXT
60 REM PUTSPRITE 0,(100,80),1,0
61 VPOKE IX,80:REM coordenada Y
62 VPOKE IX+1,100:REM coordenada X
63 VPOKE IX+2,0:REM n patron * 4 en sprites de 16
64 VPOKE IX+3,1:REM color negro
70 REM PUTSPRITE 1,(200,110),8,1
71 VPOKE IX+4,110:REM coordenada Y
72 VPOKE IX+5,200:REM coordenada X
73 VPOKE IX+6,(1*4):REM n patron * 4 en sprites de 16
74 VPOKE IX+7,8:REM color rojo

```

Descargar ejemplo 1: [OAME.BAS](#)

Referencias:

- Wikipedia [V9938](#)
- MSX Resource Center>BASE (Direcciones de las tablas VRAM)
- Portar Specifications>VDP>[Sprites](#)
- MSX Assembly Page>[MSX BIOS calls](#)



Etiquetas: desarrollo, MSX

Enlaces a esta entrada

Crear un enlace

[Entrada más reciente](#) • • • • • [Página principal](#) • • • • • [Entrada antigua](#)

