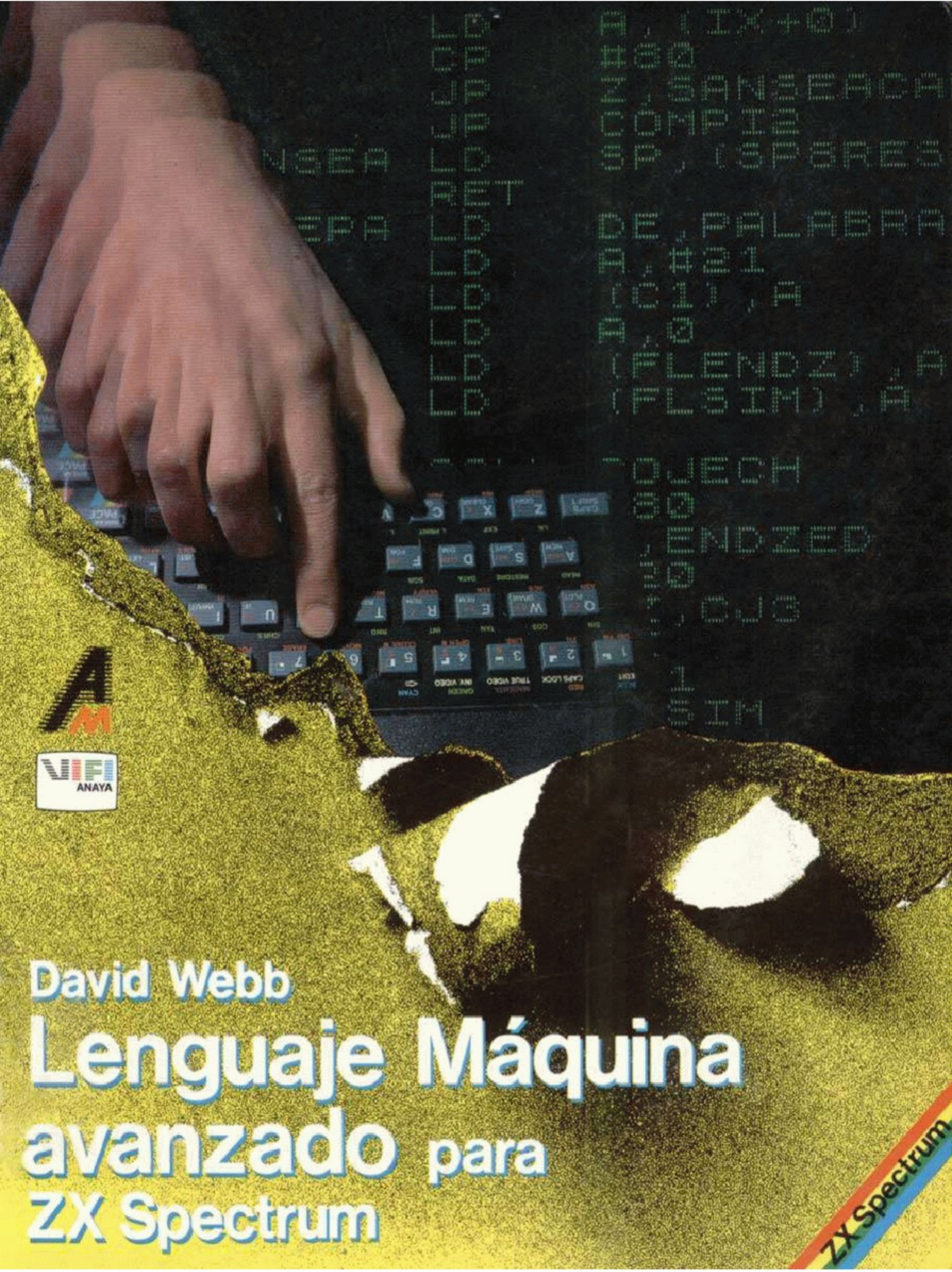


David Webb

Lenguaje Máquina avanzado para ZX Spectrum

ZX Spectrum

Lenguaje máquina avanzado para ZX Spectrum

David Webb

ANAYA MULTIMEDIA

Indice

Prefacio

Prácticamente la única rama de la programación del Spectrum que no está detalladamente estudiada por la mayoría de los libros es la de la programación avanzada en lenguaje máquina del Spectrum. Con este libro espero remediar esta situación.

Por programación "avanzada" entiendo el tipo de lenguaje de máquina de alto nivel que hay detrás de muchos de los juegos de más éxito del Spectrum. De hecho, algunas de las técnicas que se utilizan en este libro son completamente originales y diferentes de cualquier cosa que se haya visto a la hora de escribir los juegos del Spectrum. Como ejemplo, cito las rutinas en color de alta resolución y laserie de rutinas que le permiten conseguir imágenes en pantalla completa más allá del borde. Tampoco estos efectos especiales se han visto en público antes.

Es de justicia advertirle que este libro no se dirige a los principiantes, hay muchas publicaciones buenas que ya están disponibles para recién llegados al lenguaje máquina del Spectrum, y que suponen un conocimiento completo del código de instrucciones Z-80 desde el principio. Esto hace que me sea posible llevarle hasta los auténticos confines de la programación del Spectrum, llegando al grado de un arte, ampliándolos según avanzamos. Espero que disfrutará y sacará provecho de la experiencia.

Me gustaría agradecer las contribuciones de las siguientes personas:

- Mamá y papá, por 18 años de una paciencia inmensa.
- Mi editor, Fred Milgram, y todos aquellos implicados en el trabajo de editar este libro.
- John, Deb, Brian, Dermot y Nobby por su apoyo y aliento.

Finalmente dedico este libro al seis de tréboles, y esto se conoce por tener un buen triunfo.

DAVID M. WEBB
Exeter College
Oxford
Febrero de 1984

Introducción

Suposiciones a tener en cuenta para el uso de este libro

El título mismo de este libro indica que no se propone enseñar el lenguaje elemental de la máquina. Estoy suponiendo que el lector por lo menos tiene un dominio de los elementos fundamentales de, y preferiblemente cierta experiencia en, la programación del Z-80. Sin embargo, no es esencial haber aprendido ni haber practicado el lenguaje de máquina del Spectrum en profundidad, todas las peculiaridades que son específicas del Spectrum se describirán con detalle, sin suponer cualquier conocimiento previo de ellas.

Para escribir cualquier cosa que no sea un programa cortísimo en lenguaje máquina se debe utilizar un ensamblador y, por tanto, me figuro que ya tiene uno o que está a punto de comprar uno. Todos los listados en este libro están en lenguaje ensamblador, pero deliberadamente he restringido el empleo de "seudo-instrucciones" es decir, las que no están en el conjunto de instrucciones estándar del Z-80, valiéndome de las operaciones ORG, DEFB, DEFW y EQU que cualquier ensamblador que se precie debería tener.

Su ensamblador debe poder calcular hacia delante y hacia detrás los saltos relativos y trabajar con etiquetas de preferiblemente seis o más carácteres de longitud.

En el encabezamiento de cada listado hay un grupo de comentarios que le informan de cualquier parámetro que los registros deban contener al entrar en la rutina. También le informan del contenido de los registros a la salida y de aque-

llos que permanecen inalterados. A no ser que se indique lo contrario, puede suponer que los registros alternativos AF', BC', DE', HL', el puntero de la pila SP, los registros de índices IX e IY y el registro del vector de interrupción I son todos conservados por la rutina.

Al mismo tiempo, a no ser que se indique lo contrario, debe suponer que los registros A, F, B, C, D, E, H y L se destruyen todos al llamar a la rutina. El contador de programa, por supuesto, se conserva en la pila por un CALL.

La gran cantidad de comentarios explicativos que se incorporan en todas las rutinas salvo en las más sencillas de este libro están para su provecho, con la esperanza de que pueda aprender gracias al ejemplo. Son, por supuesto, completamente no-funcionales, y pueden omitirse cuando introduzca los listados en su ordenador exactamente como omitiría las sentencias BASIC REM para ahorrar memoria.

En este libro, cualquier valor numérico estará expresado por defecto en decimal, a no ser que esté precedido por un signo "#" o de la abreviatura "Hex" para hexadecimal, o por la palabra "binario" cuando se utiliza la base dos.

Ahora posee los conocimientos necesarios para utilizar el resto de este libro. Sólo un consejo, se pretende que el que emplee el libro lo lea de "principio a fin", ya que muchos de los programas posteriores contienen referencias al material publicado anteriormente en el libro.

1

Direccionamiento de la pantalla

Empiezo este capítulo con la explicación de lo que es, con frecuencia, una fuente de confusión. En todo este libro diré que la pantalla del Spectrum consta de 24 LINEAS, conteniendo cada una de ellas ocho FILAS de *pixels* (puntos gráficos), en vez de 24 filas de ocho líneas de *pixels*. De esta forma vemos que el área de texto de la pantalla tiene 24 x 8 = 192 filas.

Una vez aclarado este punto técnico, continuaré con una discusión de cómo se calcula la dirección del texto de cualquiera de las 768 (24 x 32) CELDILLAS de la pantalla.

No puedo olvidarme de mencionar el detalle de que el archivo de pantalla está definido de una forma poco usual en la memoria. Un rápido POKE con este programa le mostrará lo que quiero decir.

```
10 REM DEMOSTRACION DE DISPOSICION DEL TEXTO EN MEMORIA
20 FOR A=0 TO 6143
30 POKE 16384+A,255
40 NEXT A
50 PAUSE 0
```

De hecho el archivo de pantalla reside en las direcciones #4000 hasta #57FF de la siguiente manera. Cada fila tiene 32 columnas y cada columna tiene un ancho de 8 *pixels*. Puesto que hay 8 bits en un octeto, cada columna de cada fila está representada por un octeto. Los 32 octetos de cada fila son, como puede figurarse, almacenados consecutivamente en la memoria, leyéndose de

izquierda a derecha. Lo primero que se almacena (dirección #4000) es la fila 0 de la línea 0. Luego viene la fila 0 de la línea 1, y así sucesivamente hasta la fila 0 de la línea 7. Luego, en vez de encontrar la fila 0 de la línea 8, tenemos la fila 1 de la línea 0, hasta la fila 1 de la línea 7. El modelo continúa hacia abajo hasta la fila 7 de la línea 6, y luego la fila 7 de la línea 7. En este punto, han sido ocupados 2K de la memoria, y nos encontramos que toda la tercera parte de la pantalla tiene su zona homóloga en memoria a modo de mapa.

El modelo arriba descrito se repite para los tercios de en medio y de la parte de abajo de la pantalla, consumiendo cada uno 2K de RAM. Una consecuencia bastante buena de tener los tres tercios de la pantalla en bloques separados de la memoria es que podemos llevar a cabo comandos SAVE... SCREEN$ parciales. Los números requeridos para esto son los siguientes:

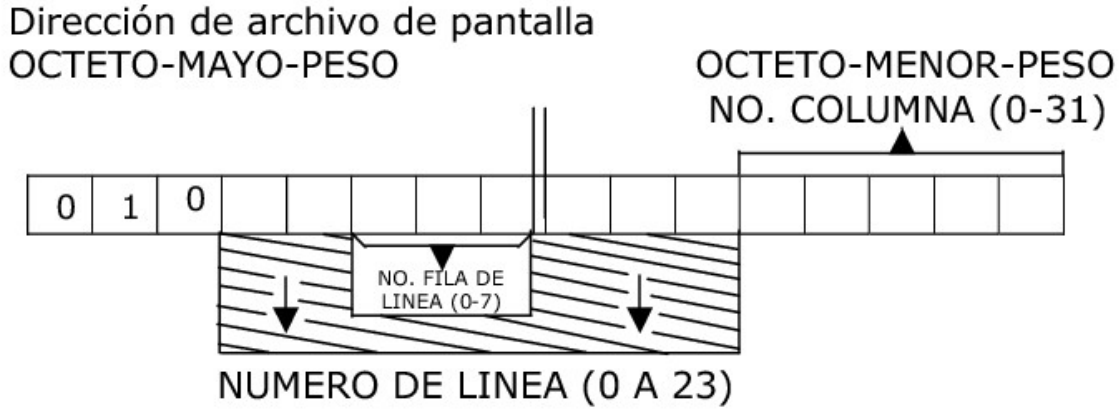
	DIRECCION DE COMIENZO
TERCIO DE ARRIBA	16384
CENTRAL	18432
DE ABAJO	20480

LONGITUDES
2K = 2.048 OCTETOS
4K = 4.096 OCTETOS
6K = 6.144 OCTETOS

Así para SALVAR los dos tercios de abajo de la pantalla (de una longitud de 4K)

```
SAVE "(NOMBRE)"CODE 18432,4096
```

Puesto que el archivo de pantalla está contenido en 8K de RAM desde #4000, los tres bits de más a la izquierda de cualquier dirección contenida en él son siempre 010. La configuración completa está compuesta como se ve en este diagrama:



La ventaja de este esquema es que podemos atravesar las direcciones de las ocho filas de cualquier celdilla de la pantalla simplemente con incrementar

el octeto de mayor peso de la dirección original, en vez de añadir 32 a la dirección entera, como haría falta en un esquema "normal". Esto permite conseguir un toque adicional de velocidad en las rutinas impresoras de una sola celdilla.

Ahora, le oigo preguntar: "¿Cuál es la forma más fácil para calcular la dirección de una celdilla?" Bueno, podría hacer cosas mucho peores que echar mano de esta rutina, llamada DF-LOC por localización de archivo de pantalla (Display File LOCation). Hay que tener en cuenta que la rutina devolverá una dirección lógica para cualquier número de línea que se introduzca en el registro B, tanto si está en el intervalo de 0-23 como si no. Esto puede ser útil si, por ejemplo, quiere trabajar hacia abajo desde el principio lógico de un bloque de caracteres 2 x 2, la línea superior del cual está fuera de la parte superior de la pantalla. Sólo tiene que introducir B = 255 (para —1, el número de la línea por encima de la pantalla) y llamar a DF-LOC como siempre.

```
                  10 ;ENTRADA: B=LINEA,C=COLUMNA
                  20 ;SE CONSERVAN : BC,DE
                  30 ;SALIDA: HL=DIRECCION EN EL ARCHIVO DE PANTALLA, A=L
87A1 78           40 DF_LOC  LD          A,B
87A2 E6F8         50         AND         #F8
87A4 C640         60         ADD         A,#40
87A6 67           70         LD          H,A
87A7 78           80         LD          A,B
87A8 E607         90         AND         7
87AA 0F          100         RRCA
87AB 0F          110         RRCA
87AC 0F          120         RRCA
87AD 81          130         ADD         A,C
87AE 6F          140         LD          L,A
87AF C9          150         RET
```

Para convertir la rutina en una especie de PRINT AT (IMPRIME EN) puede añadir la línea

LD (# 5C84), HL

antes de la sentencia RET, para cargar en la variable del sistema DF-CC la siguiente dirección, que será utilizada por alguna rutina de impresión.

Antes de seguir con una discusión del archivo de atributos, daré una rutina para borrar el archivo de pantalla, CLS-DF. Trabaja llenando el primer octeto con un cero y luego utilizando la poderosa instrucción LDIR para "copiar" el contenido de dicho octeto al que está encima de él, repitiendo 17FF Hex, veces para llenar todo el archivo de pantalla con ceros. Esta técnica debe utilizarse siempre que se necesite llenar un bloque de memoria con un octeto en particular.

Observe que el empleo de la instrucción:

```
LD      (HL),L
```

(ya que L = 0)

es más rápido y ocupa menos memoria que:

```
LD      (HL),0
```

```
                    10 ;SE CONSERVA : A
                    20 ;SALIDA:BC=0,DE=#5800,HL=#57FF
                    30 ;
873B  210040        40 CLS_DF  LD          HL,#4000
873E  01FF17        50         LD          BC,#17FF
8741  75            60         LD          (HL),L
8742  54            70         LD          D,H
8743  1E01          80         LD          E,1
8745  EDB0          90         LDIR
8747  C9           100         RET
```

Obviamente y de la misma manera que el comando parcial SAVE SCREEN$, descrito anteriormente en este capítulo, podría adaptar la rutina para borrar sólo una parte de la pantalla. Algunos números útiles son estas direcciones y longitudes hexadecimales.

	DIRECCION		**VALOR PARA BC**
TERCIO SUPERIOR	#4000	1 TERCIO	#07FF
CENTRAL	#4800	2 TERCIOS	#0FFF
INFERIOR	#5000	PANTALLA ENTERA	#17FF

Así que, para borrar los dos tercios inferiores de la pantalla, utilice las líneas

```
LD    HL,#4800
LD    BC,#0FFF
```

Los atributos del archivo de pantalla del Spectrum son aquellos octetos que son responsables de los colores de INK (tinta) y PAPER (papel) y del estado de BRIGHT (brillo) y FLASH (parpadeo) en cada celdilla de carácter de la pantalla. Por tanto, hay 768 octetos en el archivo de atributos y están dispuestos lógicamente como 24 grupos de 32 octetos, uno para cada columna, leyéndose de izquierda a derecha a través de la pantalla.

La siguiente rutina encontrará la dirección de los atributos de cualquier celdilla de la pantalla y se llama ATTLOC, por localizador de atributo (ATTribute LOCator).

```
                    10;ENTRADA : B=LINEA, C=COLUMNA
                    20;SE CONSERVAN: BC,DE
                    30 ;SALIDA : HL=DIRECCION EN EL ARCHIVO DE ATRIBUTOS, A=L
88B5  78            40 ATTLOC  LD          A,B
88B6  CB2F          50         SRA         A
88B8  CB2F          60         SRA         A
88BA  CB2F          70         SRA         A
88BC  C658          80         ADD         A,#58
88BE  67            90         LD          H,A
88BF  78           100         LD          A,B
88C0  E607         110         AND         7
88C2  0F           120         RRCA
88C3  0F           130         RRCA
88C4  0F           140         RRCA
88C5  81           150         ADD         A,C
88C6  6F           160         LD          L,A
88C7  C9           170         RET
```

Observe el empleo de SRA A para extender el signo del valor en A si se desplaza hacia la derecha. Esto permite que la rutina devuelva, como en el caso de DF-LOC, una dirección lógica dada en cualquier número de línea en B (alcance —128 hasta + 127).

La extensión lógica de la rutina para conseguir la función ATTR (Y, X) es el añadir la instrucción

```
LD    A,(HL)
```

antes de RET, devolviendo el atributo en el registro A.

He incluido un par de rutinas para conversión de direcciones entre el archivo de pantalla y el de atributos, que pueden ser útiles si tiene uno pero no el otro. La primera rutina, DF-ATT encontrará la dirección del atributo al cubrir cualquier octeto en el archivo de pantalla, sin tener en cuenta si está en la fila cero de una línea o no.

```
                  10  ;ENTRADA: HL=DIRECCION EN EL ARCHIVO DE PANTALLA
                  20  ;SE CONSERVAN:HL,BC
                  30  ;SALIDA: DE=DIRECCION DE ATRIBUTOS, A=D
8781 7C           40  DF_ATT   LD          A,H
8782 0F           50           RRCA
8783 0F           60           RRCA
8784 0F           70           RRCA
8785 E603         80           AND         3
8787 F658         90           OR          #58
8789 57          100           LD          D,A
878A 5D          110           LD          E,L
878B C9          120           RET
```

La rutina opuesta es ATT-DF, que encuentra la dirección de la primera fila de una celdilla en el archivo de pantalla, dada la dirección de sus atributos.

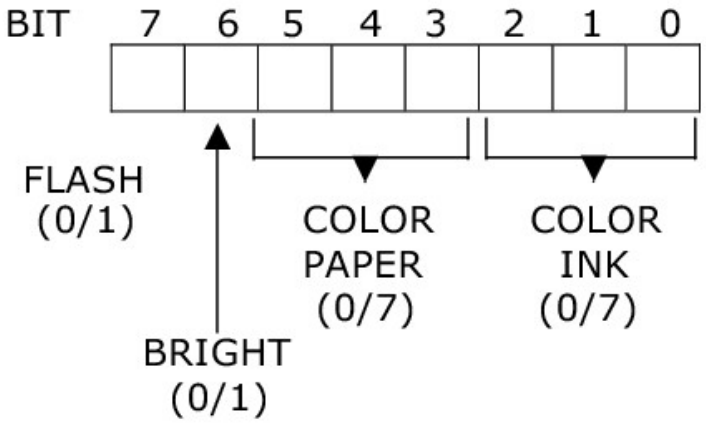
```
                  10  ;ENTRADA : HL=DIRECCION ATRI.
                  20  ;SE CONSERVAN : HL, BC
                  30  ;SALIDA: DE=DIRECCION DEL A.P. ,  A=D
8769 7C           40  ATTDF    LD          A,H
876A E603         50           AND         3
876C 07           60           RLCA
876D 07           70           RLCA
876E 07           80           RLCA
876F F640         90           OR          #40
8771 57          100           LD          D,A
8772 5D          110           LD          E,L
8773 C9          120           RET
```

Para completar la serie de rutinas de "Localización", he incluido una de propósitos múltiples que devuelve la dirección de una celdilla en el archivo de pantalla, la almacena en una variable llamada DFCC, devuelve la dirección de sus atributos, y, finalmente, los propios atributos, en el acumulador. He llamado a esta rutina LOCATE (LOCALIZA).

```
10 ;ENTRADA: B=LINEA, C=COLUMNA
20 ;CONSERVA:BC
30 ;SALIDA: HL=DIRECCION A.P. , DE= DIRECCION ATRI. ,A=ATR(B,C)
```

```
                    40 ;DF_CC ES MODIFICADO
                    50 ;
5C84                60 DFCC   EQU          #5C84
                    70 ;
8831 78             80 LOCATE LD           A,B
8832 E618           90        AND          #18
8834 67            100        LD           H,A
8835 CBF4          110        SET          6,H
8837 0F            120        RRCA
8838 0F            130        RRCA
8839 0F            140        RRCA
883A F658          150        OR           #58
883C 57            160        LD           D,A
883D 78            170        LD           A,B
883E E607          180        AND          7
8840 0F            190        RRCA
8841 0F            200        RRCA
8842 0F            210        RRCA
8843 81            220        ADD          A,C
8844 6F            230        LD           L,A
8845 5F            240        LD           E,A
8846 1A            250        LD           A,(DE)
8847 22845C        260        LD           (DFCC),HL
884A C9            270        RET
```

Como ya te he dicho, los atributos de cada celdilla le indican los colores de su INK (tinta) y PAPER (papel) y su estado de BRIGHT (brillo) y FLASH (parpadeo). La configuración de bits asociada con esto se muestra en el diagrama



Por tanto, la configuración para FLASH 1, BRIGHT 0, INK 3, PAPER 6 sería 10110011 o Hex B3.

Podemos borrar el archivo de atributos llenándolo con cualquier octeto, utilizando la siguiente rutina, CLSATT, que funciona de una forma muy similar a CLS-DF.

```
                    10 ;ENTRADA: A=ATRIBUTO DE PANTALLA
                    20 ;SE CONSERVA :  A
                    30 ;SALIDA: BC=0,  DE=#5B00, HL=#5AFF
                    40 ;
8765 210058         50 CLSATT LD           HL,#5800
8768 01FF02         60        LD           BC,#2FF
876B 77             70        LD           (HL),A
876C 54             80        LD           D,H
876D 1E01           90        LD           E,1
876F EDB0          100        LDIR
8771 C9            110        RET
```

Por tanto, para volver a poner los atributos a su condición inicial (FLASH 0, BRIGHT 0, PAPER 7, INK 0), haga:

```
LD    A,#38
CALL  CLSATT
```

También he incluido una combinación de CLS-DF y CLSATT que borra el archivo de pantalla y pone los atributos a un valor dado. La rutina se llama CLS por razones obvias.

```
                    10  ;ENTRADA: A=ATRIBUTO DE PANTALLA
                    20  ;SE CONSERVA: A
                    30  ;SALIDA: BC=0, DE=#5B00, HL=#5AFF
                    40  ;
8789 210040         50  CLS   LD          HL,#4000
878C 010018         60        LD          BC,#1800
878F 75             70        LD          (HL),L
8790 54             80        LD          D,H
8791 1E01           90        LD          E,1
8793 EDB0          100        LDIR
8795 77            110        LD          (HL),A
8796 01FF02        120        LD          BC,#2FF
8799 EDB0          130        LDIR
879B C9            140        RET
```

Mi última observación sobre el direccionamiento de la pantalla es el control del color del borde. El color del borde del Spectrum usualmente está contenido como bits 3, 4 y 5 de la variable del sistema BORDCR, en la dirección #5C48.

Sin embargo, al alterar esta dirección a través del lenguaje máquina no se obtiene ningún efecto en el color del borde, únicamente cambia el valor contenido en #5C48. Para cambiar el borde, tenemos que poner a cero el bit 0 del bus de direcciones y luego dar salida al número del nuevo color por el bus de datos. Para no afectar a ningún otro dispositivo conectado al puerto del usuario, ponemos a uno los 7 bits restantes del octeto de menor peso del bus de datos, obteniendo 1111 1110 en binario, o FE en Hex.

Por tanto, para cambiar el color del borde al color rojo (valor 2) utilizamos la secuencia

```
LD    A,2
OUT   (#FE),A
```

Debo señalar que el color del borde sólo utiliza los bits 0, 1 y 2 del bus de datos. De hecho el bit 3 controla las salidas MIC y EAR y el bit 4 el altavoz. Alterando el estado de éstas (complementándolas), se provoca el envío de un clic a las clavijas MIC y EAR o se oye en el altavoz.

Es una buena práctica de programación el "enmascarar" aquellos bits que no son necesarios para alterar el color del borde, de forma que se mantenga su estado y no se oigan clics extraños en el altavoz. Si almacenamos el último valor enviado al puerto #FE en la variable BORD, y luego cambiamos el color del borde, será aplicable este segmento de programa.

```
LD      A,(BORD)
AND     #0F8
OR      (NUEVO VALOR DE BORDE)
OUT     (#FE),A
LD      (BORD),A
```

Si tiene un par de registros de sobra tales como HL, entonces es ligeramente mejor y más rápido utilizar:

```
LD      HL,(BORD)
LD      A,(HL)
AND     #0F8
OR      (NUEVO VALOR DE BORDE)
OUT     (#0FE),A
LD      (HL),A
```

Antes de concluir con este tema, una peculiaridad interesante del lenguaje de máquina del Z-80 es que, al contrario de otras instrucciones, es realmente más rápido el empleo de datos inmediatos que el de un registro como el número de puerto cuando se da salida desde el acumulador. O sea:

```
OUT     (#FE),A      ;TARDA 11 T-ESTADOS MIENTRAS QUE
OUT     (C),A        ;TARDA 12 T-ESTADOS
```

Esto representa bastante diferencia cuando se necesita una salida a muy alta velocidad, como se verá más adelante en este libro.

2

El desarrollo de una rutina de impresión

Las rutinas de impresión en la ROM del Spectrum son terriblemente lentas y aburridas de utilizar. Esto es una consecuencia del uso de la instrucción RST # 10 para gran cantidad de funciones de impresión diferentes. Una vez llamada, la rutina tiene que decidir, entre otras cosas, si está imprimiendo en el área de INPUT, en la parte superior de la pantalla o sobre la impresora, si está intentando cambiar el color de INK o del PAPER o ejecutando algún control más, tal como una función TAB o AT, y si está imprimiendo un carácter "normal", uno semigráfico o un carácter gráfico definido por el usuario. Además de todo esto, debido a la naturaleza del BASIC, la rutina también pasa tiempo realizando una serie de comprobaciones de error de las que podemos prescindir en un programa en código-máquina.

Por tanto, es esencial que desarrollemos nuestra propia rutina de impresión hecha a la medida, y esto es lo que voy a hacer en este capítulo.

El proceso de imprimir un carácter puede desglosarse en 3 etapas. Primero localizamos la dirección de los datos del carácter (los 8 octetos cuyos bits definen el carácter), luego copiamos estos datos en la celdilla deseada de la pantalla, y finalmente cambiamos los atributos de esa celdilla según se necesite.

Sin duda estará familiarizado con el concepto de dejar ciertos atributos de una celdilla tal y como están con el empleo de INK 8, PAPER 8, FLASH 8 y BRIGHT 8 en el momento de imprimir un nuevo carácter en esa celdilla. Estas funciones BASIC se realizan fácilmente en lenguaje máquina, donde se conoce la operación como enmascaramiento (MASKING) de los bits individuales del "viejo" octeto de atributo en el momento de imprimir un carácter "nuevo".

Utilizaremos una variable de un octeto, ATT, para contener los nuevos atributos para el carácter que se va a imprimir, y un segundo octeto, MASK, para contener la máscara para los viejos atributos. Para cada bit de los atributos viejos que necesitamos conservar, ponemos a uno el bit correspondiente de MASK. Supongamos que sólo queremos que se enmascare el bit BRIGHT (por ejemplo, BRIGHT 8). Entonces, consultando el bit de atributo en el capítulo 1, vemos que BRIGHT ocupa el bit 6. Por lo que si ponemos a uno el bit 6 de nuestra máscara obtenemos 0100 0000 o Hex 40. Por tanto, hacemos la variable MASK igual a Hex 40.

Si examinamos los 8 valores que representan detalladamente el juego de colores de INK (tinta) y PAPER (papel), encontramos que están asignados a los colores de una forma extremadamente lógica. Todos los colores son combinaciones de los tres colores primarios, azul, rojo y verde, y se han asignado tres bits de estos colores para cada INK y PAPER. Esto le hace más sencilla la tarea al conocido chip ULA, que, por decirlo de alguna manera, debe enviar estos bits a los cañones de electrones, azul, rojo y verde que "disparan" sobre los pixels de su televisor en color.

El color azul está asignado al más bajo de los tres bits (valor 1), luego el rojo (valor 2) y luego el verde (valor 4). De esta forma los bits de INK (tinta) y PAPER (papel) aparecen como se indica a continuación:

BIT-MAYOR-PESO BIT-MENOR-PESO

VERDE	ROJO	AZUL

y la estructura del atributo completo (y de la máscara) es la siguiente:

Flash Bright «---------- Paper ----------»«---------- Ink ----------»

7	6	5	4	3	2	1	0
		V	R	A	V	R	A

Siempre que se necesita un color primario para fabricar otro color, su bit está puesto a uno. Por tanto, cian, que es una mezcla de verde y azul, tiene la estructura binaria:

V	R	A
1	0	1

= decimal 5

El blanco es la combinación de los tres colores primarios, por lo que tiene la estructura

V	R	A	
1	1	1	= decimal 7

mientras que el negro es la ausencia total de color y por consiguiente se representa por

V	R	A	
0	0	0	= decimal 0

Por cierto, no hay diferencia entre el negro brillante (con BRIGHT 1) y el negro oscuro (BRIGHT 0). Puede verificar que esto es así haciendo:

```
;BORDER 0; PAPER 0; BRIGHT 1; CLS
```

Al introducir esta línea, los colores del borde y área de texto no se distinguirán.

Una de las ventajas de tener los valores de los colores asignados tan lógicamente es que puede enmascarar los colores primarios o una combinación de ellos, en vez de estar restringido a tener que enmascarar los tres bits, que es todo lo que ofrecen INK 8 y PAPER 8 en el BASIC.

Después de obtener los valores para ATT y MASK, estamos preparados para crear el nuevo octeto de atributo para una celdilla. Cargando los viejos atributos en el acumulador, la forma más rápida de realizar la operación es:

```
XOR     ATT
AND     MASK
XOR     ATT
```

El nuevo octeto de atributo está ahora preparado para ser colocado en el archivo de atributos. Verá que aparece un fragmento de programa así en la rutina de impresión siguiente.

Almacenamos la dirección base de los datos del carácter que se va a utilizar en la variable de dos octetos BASE. Esta debería apuntar a la fila cero del primer carácter de su juego. He previsto sitio para hasta 256 caracteres, y puesto que cada uno necesita 8 octetos, un juego completo necesitará 2K de memoria. En el caso, poco probable, de que se necesiten más de 256 caracteres, necesitará dos octetos para representar cada carácter, y debe utilizar el octeto de mayor peso para indicar qué valor de BASE se necesita, y a continuación llamar a la misma rutina de impresión.

El Spectrum tiene las estructuras en bits de 96 de sus caracteres en ROM, que van desde el ESPACIO hasta el símbolo de copyright. Estos datos ocupan los últimos 768 octetos de la ROM, desde la dirección #3D00.

La variable del sistema CHARS de la que se habla en el manual del Spectrum "contiene la dirección del juego de caracteres menos 256". Esto puede

parecer un poco extraño, hasta que se dé cuenta de que el primer carácter, un espacio, está representado numéricamente por #20, o por 32 en decimal. Ahora bien 32 x 8 = 256, así que poniendo en CHARS la dirección del juego de caracteres menos 256, el Spectrum puede encontrar la dirección de un carácter con sólo multiplicar su código por 8 (filas) y sumarlo a CHARS.

En la rutina PRINT 1 observará que he inicializado nuestra variable BASE como #3C00, de forma que los valores normales CODE para el juego de caracteres del Spectrum son aplicables. También he puesto ATT inmediatamente antes de MASK para que las dos puedan accederse con una sola instrucción LD.

```
                    10 ; ENTRADA : A=CODIGO CAR.
                    20 ;SE CONSERVA: C
                    30 ;SALIDA: B=0, DE=DIRECCION ATRIBUTO
                    40 ;
8A8F 003C           50 BASE   DEFW         #3C00
8A91 0040           60 DFCC   DEFW         #4000
8A93 38             70 ATT    DEFB         #38
8A94 00             80 MASK   DEFB         0
                    90
                   100 ;CONSTRUCCION DE DIRECCION DE DATOS DE CARACTER
8A95 6F            110 PRINT1 LD           L,A
8A96 2600          120        LD           H,0
8A98 29            130        ADD          HL,HL
8A99 29            140        ADD          HL,HL
8A9A 29            150        ADD          HL,HL
8A9B ED5B8F8A      160        LD           DE,(BASE)
8A9F 19            170        ADD          HL,DE
                   180 ;TOMAR DIRECCION DEL ARCHIVO DE PANTALLA
8AA0 ED5B918A      190        LD           DE,(DFCC)
8AA4 0608          200        LD           B,8
                   210 ;IMPRIMIR CARACTER FILA POR FILA
8AA6 7E            220 NXTROW LD           A,(HL)
8AA7 12            230        LD           (DE),A
8AA8 23            240        INC          HL
8AA9 14            250        INC          D
8AAA 10FA          260        DJNZ         NXTROW
                   270 ;CONSTRUIR DIRECCION DE ATRIBUTO
8AAC 7A            280        LD           A,D
8AAD 0F            290        RRCA
8AAE 0F            300        RRCA
8AAF 0F            310        RRCA
8AB0 3D            320        DEC          A
8AB1 E603          330        AND          3
8AB3 F658          340        OR           #58
8AB5 57            350        LD           D,A
8AB6 2A938A        360        LD           HL,(ATT)
                   370 ;H=MASCARA, L=ATRIBUTO
                   380 ;TOMAR EL ANTERIOR ATRIBUTO
8AB9 1A            390        LD           A,(DE)
                   400 ;CONSTRUIR UNO NUEVO
8ABA AD            410        XOR          L
8ABB A4            420        AND          H
8ABC AD            430        XOR          L
                   440 ;REEMPLAZAR ATRIBUTO
8ABD 12            450        LD           (DE),A
                   460 ;FINALMENTE PONER DFCC A LA SIGUIENTE POSICION IMPRESION
8ABE 21918A        470        LD           HL,DFCC
8AC1 34            480        INC          (HL)
8AC2 C0            490        RET          NZ
8AC3 23            500        INC          HL
8AC4 7E            510        LD           A,(HL)
8AC5 C608          520        ADD          A,8
```

```
8AC7 77          530        LD          (HL),A
8AC8 C9          540        RET
```

En la rutina anterior he hecho la suposición de que ya se había puesto DF-CC a la dirección correcta del archivo de pantalla, utilizando la rutina LOCATE del capítulo 1 o de otra forma. Observará que se actualiza a la siguiente posición de impresión cada vez que se imprime un carácter.

Como ejemplo para mostrar PRINT 1 en acción, he aquí una rutina para imprimir el juego de caracteres de la ROM (códigos #20 hasta #7F). Necesitará la rutina LOCATE del capítulo 1.

```
                  10 ;DEMOSTRACION DE PRINT1
                  20 ;PONER DFCC A (0,0)
87F3  010000      30         LD          BC,0
87F6  CDF687      40         CALL        LOCATE
                  50 ;PONER BASE SENALANDO AL JUEGO DE CARACTERES  ROM
87F9  21003C      60         LD          HL,#3C00
87FC  22FC87      70         LD          (BASE),HL
                  80 ;AHORA IMPRIMIR DESDE EL CODIGO #20 HASTA EL #7F
                  90 ;RECORDAR QUE PRINT1 CONSERVA EL REGISTRO C
87FF  0E20       100         LD          C,#20
8801  79         110 LOOP    LD          A,C
8802  CD0288     120         CALL        PRINT1
8805  0C         130         INC         C
```

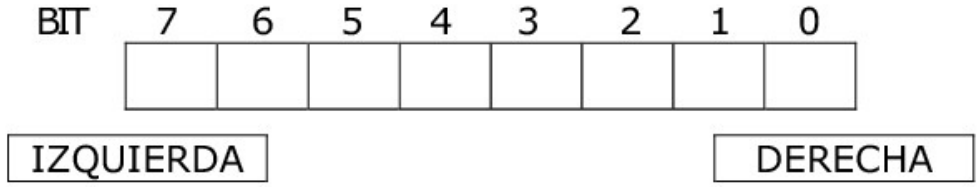
3

Dibujar y trazar

Habrá muchas ocasiones en que necesite dibujar estrellas y posiciones en mapas o trazar rayos láser. Aquí voy a desarrollar una rutina para que pueda dibujar en cualquier parte de la pantalla, y una para trazar, utilizando coordenadas absolutas, una línea entre dos puntos cualesquiera de la pantalla. Las rutinas son ligeramente más rápidas que las del BASIC en ROM, puesto que he quitado muchas comprobaciones de error engorrosas.

El proceso que se necesita para dibujar un punto en la pantalla puede desglosarse en 4 etapas. Primero, encontramos la dirección del octeto del archivo de pantalla que contiene el bit que representa nuestro *pixel* (punto gráfico) de "destino" en la pantalla. Luego encontramos la dirección de los atributos de la celdilla donde se encuentra dicho *pixel*. A continuación cambiamos los atributos de acuerdo con nuestras variables estándar ATT y MASK, y finalmente realizamos el dibujo en sí, observando cuidadosamente si se necesita INVERSE 1 u OVER 1.

Debo comentar en este punto que, dado un octeto del archivo de pantalla que representa una fila de una celdilla, el bit más a la izquierda (BIT 7) representa el *pixel* más a la izquierda, mientras que el bit más a la derecha (BIT 0) representa el *pixel* más a la derecha. Un error frecuente de la programación es pensar que el bit 0 representa el primer *pixel* (el de más a la izquierda). La forma más fácil para recordar esto es con este diagrama:



Verá que la rutina PLOT encuentra la dirección del atributo convirtiendo la dirección del archivo de pantalla. Esto es mucho más fácil y rápido que volver a nuestras coordenadas originales y calcular la dirección en función de ellas.

La rutina decide si hay que utilizar OVER o INVERSE consultando el estado de dos banderas (*flags*) en una variable de un octeto llamada PFLAG. Esto es equivalente a la variable del sistema BASIC del mismo nombre de la dirección 23697 (5C91 Hex). Indicamos OVER 1 poniendo a uno el bit 1 de PFLAG, e INVERSE 1 poniendo a uno el bit 3.

Observe que estoy utilizando un sistema nuevo de coordenadas en la pantalla, que hace que sea más fácil y más rápido calcular las direcciones. El ángulo superior izquierdo es (0,0) mientras que el inferior derecho (contando las líneas de INPUT) es (255,191). He aquí la rutina:

```
                    10 ;ENTRADA : H=X, L=Y
                    20 ;SE CONSERVA : HL
                    30 ;DE=DIRECCION DEL PIXEL EN EL ARCHIVO DE PANTALLA
                    40 ;A=(DE), C=(PFLAG)
                    50 ;ES QUINA SUPERIOR IZQ UIERDA = (0,0)
                    60 ;
                    70 ;BIT   1 (PFLAG)=OVER
                    80 ;BIT   3 (PFLAG)=INVERSE
                    90 ;
                   100 ;
8BAE 38             110 ATT    DEFB          #38
8BAF 00             120 MASK   DEFB          0
8BB0 00             130 PFLAG  DEFB          0
                    140 ;
                    150 ;ENCONTRAR LA DIRE CCION   DEL BYTE REQUERIDO EN EL A.P.
8BB1 7D             160 PLOT   LD            A,L
8BB2 E6C0           170        AND           #C0
8BB4 1F             180        RRA
8BB5 37             190        SCF
8BB6 1F             200        RRA
8BB7 0F             210        RRCA
8BB8 AD             220        XOR           L
8BB9 E6F8           230        AND           #F8
8BBB AD             240        XOR           L
8BBC 57             250        LD            D,A
8BBD 7C             260        LD            A,H
8BBE 07             270        RLCA
8BBF 07             280        RLCA
8BC0 07             290        RLCA
8BC1 AD             300        XOR           L
8BC2 E6C7           310        AND           #C7
8BC4 AD             320        XOR           L
8BC5 07             330        RLCA
8BC6 07             340        RLCA
8BC7 5F             350        LD            E,A
                    360 ;LA DIRECCION SE ALMACENA EN DE
8BC8 D5             370        PUSH          DE
                    380 ;ENCONTRAR DIRECCION DE ATRIBUTO
8BC9 7A             390        LD            A,D
8BCA 0F             400        RRCA
8BCB 0F             410        RRCA
8BCC 0F             420        RRCA
8BCD E603           430        AND            3
8BCF F658           440        OR            #58
8BD1 57             450        LD            D,A
8BD2 ED4BAE8B       460        LD            BC,(ATT)
```

```
                470 ;CAMBIAR ATRIBUTO
8BD6  1A        480         LD              A,(DE)
8BD7  A9        490         XOR             C
8BD8  AO        500         AND             B
8BD9  A9        510         XOR             C
8BDA  12        520         LD              (DE),A
                530 ;RECUPERAR DIRECCION  DE A.P.
8BDB  D1        540         POP             DE
8BDC  7C        550         LD              A,H
8BDD  E607      560         AND             7
8BDF  47        570         LD              B,A
8BE0  04        580         INC             B
                590 ;B TIENE (NUMERO BIT DE DESTINO)+1
8BE1 3EFE       600         LD              A,#FE
                610 ;ROTAR UNA VENTANA HASTA EL BIT DE DESTINO
8BE3  0F        620 PLOOP   RRCA
8BE4  10FD      630         DJNZ            PLOOP
8BE6  47        640         LD              B,A
8BE7  3AB08B    650         LD              A,(PFLAG)
8BEA  4F        660         LD              C,A
                670 ;TOMAR UN BYTE DEL A.P.
8BEB  1A        680         LD              A,(DE)
                690 ;COMPROBAR OVER1
8BEC  CB49      700         BIT             1,C
8BEE  2001      710         JR              NZ,OVER1
8BF0  A0        720         AND             B
                730 ;COMPROBAR INVERSE1
8BF1  CB59      740 OVER1   BIT             3,C
8BF3  2002      750         JR              NZ,INV1
8BF5  A8        760         XOR             B
8BF6  2F        770         CPL
8BF7  12        780 INV1    LD              (DE),A
8BF8  C9        790         RET
```

La parte final de la rutina, la que realiza propiamente el dibujo, merece una explicación más detallada. Pasando por alto los otros 7 bits de nuestro octeto, ya que siempre conserva su estado al final, vamos a examinar el comportamiento del bit "destino",

```
AND     B
```

hace que el destino sea cero si se selecciona OVER 0, OVER 1 hace que se omita la instrucción,

```
BIT     3,C
JR      NZ,INV1
```

provoca un salto hacia el final si se había seleccionado INVERSE 1, dejando el bit como estaba si se había seleccionado OVER 1, o a cero (PAPER) si se había seleccionado OVER 0.

Finalmente, "restringiendo" nuestras selecciones a INVERSE 0,

```
XOR     B
CPL
```

que puede ser considerado con más claridad como

```
XOR         B
XOR         #FF
```

deja el bit complementado en el caso de OVER 1 o puesto a uno en el caso de OVER 0. Ahora se reemplaza el octeto en el archivo de pantalla.

* * * *

Tras esta rutina PLOT, he decidido que merecería la pena hacer un ejercicio para desarrollar nuestra propia rutina DRAW en código máquina. Aunque utiliza el mismo algoritmo que el de la ROM del Spectrum, la rutina funcionará algo más deprisa debido al empleo de código "optimizado" y a que hay menos riesgo de error.

Utilizaré coordenadas absolutas como parámetros para la rutina, en vez de las relativas que se emplean en el BASIC de Spectrum. Esto es en gran parte un asunto de preferencia personal, y además es muy sencillo modificar la rutina para que efectúe trazado relativo. Como con la rutina PLOT, las coordenadas se asignan como sigue:

Para poder discutir el algoritmo del trazado, vamos a suponer que la línea va desde (X1, Yl) a (X2, Y2), ambos inclusive. Antes de continuar, la rutina dibuja el primer punto de la línea. Ahora se necesita algo de preparación para decidir en qué dirección hay que trazar.

Si (X2-X1) es positivo, estaremos trazando hacia la derecha. Si es negativo, entonces hacia la izquierda. De manera similar, si (Y2-Y1) es positivo, estaremos trazando hacia abajo, y en caso contrario, la dirección será hacia arriba.

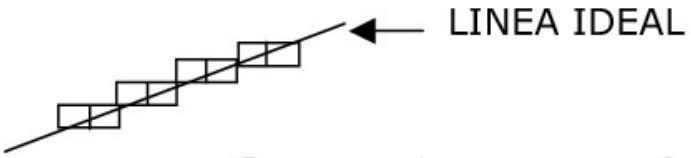
La rutina carga los cambios de unidad en X e Y relativos a la dirección a lo largo de cada eje en los registros D y E, respectivamente. Por ejemplo, si estuviéramos trazando hacia arriba y hacia la derecha, entonces el cambio en X sería positivo (D = 1) y en Y negativo (E = –1). Así la rutina produce:

```
DE=#01FF
```

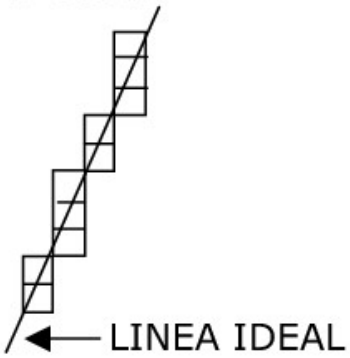
Ahora necesitamos considerar exactamente cómo puede formarse una línea a base de pasos de una unidad horizontal, vertical o diagonalmente entre puntos en una rejilla. Pensando un momento, observamos que, a menos que los dos puntos de los extremos de la línea estén en una diagonal de *pixel,* necesitaremos combinar una mezcla de pasos "rectos" y diagonales para trazar la línea. La rutina define la pareja BC como sigue:

```
B=ABS(X2-X1)    C=ABS(Y2-Y1)
```

Si B es mayor que C, entonces necesitaremos una mezcla de pasos horizontales y diagonales, de esta forma:



Mientras que si B es menor que C, se requiere una mezcla de pasos VERTICALES y diagonales, tales como éstos:



La rutina decide si necesitamos pasos verticales u horizontales, y almacena el valor necesario de DE, como se explicó anteriormente, en la variable VHSTP. La dirección de los pasos diagonales está almacenada en DIASTP.

Los valores en B y C están almacenados de manera que B sea mayor o igual que C. Estamos ahora a punto para empezar a trazar la línea que tendrá (B-C) pasos rectos y C diagonales. Para poder asegurarse que los pasos rectos y diagonales están distribuidos por igual, se utiliza el procedimiento siguiente.

Se copia B en H, y luego se divide por dos y se copia en L. Ahora, entrando en el bucle, se suma C a L, y si el resultado es mayor o igual que B, entonces se reduce por B, y se toma un paso diagonal. De no ser así, se toma un paso recto. Se dibuja un punto, se decrementa el contador H, y el bucle se repite hasta que la línea esté acabada. Se puede considerar este bucle como que C se suma continuamente a sí mismo y se toma un paso diagonal cada vez que el resultado pasa un nuevo múltiplo de B. La razón de que se inicialice L como 1/2 B es sencillamente para asegurar que la línea sea recta desde el principio.

He aquí por fin la rutina DRAW; los valores de salida más útiles son las coordenadas del último punto dibujado, en HL. Como HL contiene también las coordenadas del primer punto de una línea a la entrada en la rutina, vemos que se puede usar sin alteración de las llamadas a la rutina DRAW, para trazar líneas hacia, y después desde, el mismo punto. Esta facilidad será utilizada muchísimo en el programa de demostración que sigue a la rutina.

```
10 ;ENTRADA : H=X1, L=Y1, B=X2, C=Y2
20 ;DE ESTE MODO SE DIBUJA DESDE (H,L) HASTA (B,C) INCLUSIVE
30 ;NO SE CONSERVA NADA
40 ;SALIDA:DE ES LA DIRECCION DEL ULTIMO PIXEL DIBUJADO
50 ; QUE ESTA EN (H,L).
60 ;B ES EL MAYOR
70 ;Y C EL MENOR DE ABS(X2-X1) Y ABS(Y2-Y1).
```

```
                        80 ;
                        90 ;
8DFF  0100              100 DIASTP    DEFW      1
8E01  0100              110 VHSTP     DEFW      1
                        120 ;
8E03  C5                130 DRAW      PUSH      BC
                        140 ;PLOT(X1,Y1)
8E04  CD048E            150           CALL      PLOT
8E07  C1                160           POP       BC
8E08  110101            170           LD        DE,#101
                        180 ;DE CONTIENE LA DIRECCION DE LOS
                        190 ;PASOS X E Y
8E0B  78                200           LD        A,B
                        210 ;IR A LA IZQ (-1),O A LA DER (+1)?
8E0C  94                220           SUB       H
8E0D  3004              230           JR        NC,X2X1
8E0F  15                240           DEC       D
8E10  15                250           DEC       D
8E11  ED44              260           NEG
8E13  47                270 X2X1      LD        B,A
                        280 ;B CONTIENE EL NO. DE PASOS EN X
8E14  79                290           LD        A,C
8E15  95                300           SUB       L
                        310 ;SUBIR (-1) O BAJAR (+1)?
8E16  3004              320           JR        NC,Y2Y1
8E18  1D                330           DEC       E
8E19  1D                340           DEC       E
8E1A  ED44              350           NEG
8E1C  4F                360 Y2Y1      LD        C,A
                        370 ;C CONTIENE EL NO. DE PASOS EN Y
                        380 ;COMPROBAR QUE ESTA LINEA NO ES UN PUNTO
8E1D  B0                390           OR        B
8E1E  C8                400           RET       Z
8E1F  79                410           LD        A,C
8E20  B8                420           CP        B
8E21  E5                430           PUSH      HL
                        440 ;ALMACENAR LA DIRECCION DE UN PASO EN DIAGONAL
8E22  62                450           LD        H,D
8E23  6B                460           LD        L,E
8E24  22FF8D            470           LD        (DIASTP),HL
                        480 ;DECIDIR ENTRE PASOS VERTICALES Y
                        490 ;HORIZONTALES DEPENDIENDO DE
                        500 ;CUAL ES EL MAYOR DE B Y C
8E27  2E00              510           LD        L,0
8E29  3804              520           JR        C,BBC
8E2B  65                530           LD        H,L
8E2C  6B                540           LD        L,E
8E2D  48                550           LD        C,B
8E2E  47                560           LD        B,A
                        570 ;ALMACENAR EL PASO V/H
                        580 ;
8E2F  22018E            590 BBC       LD        (VHSTP),HL
                        600 ;
                        610 ;B ES AHORA >=C. LA RUTINA TOMA B-C PASOS RECTOS
                        620 ;Y C DIAGONALES
8E32  60                630           LD        H,B
8E33  78                640           LD        A,B
8E34  CB3F              650           SRL       A
8E36  6F                660           LD        L,A
8E37  7D                670 NXTSTP    LD        A,L
8E38  81                680           ADD       A,C
                        690 ;DECIDIR ENTRE UN PASO DIAGONAL O
                        700 ;RECTO ESTA VEZ
8E39  3803              710           JR        C,DIAG
8E3B  B8                720           CP        B
8E3C  3808              730           JR        C,VERHOR
8E3E  90                740 DIAG      SUB       B
```

```
8E3F  6F          750            LD        L,A
8E40  ED5BFF8D    760            LD        DE,(DIASTP)
8E44  1805        770            JR        STEP
8E46  6F          780 VERHOR     LD        L,A
8E47  ED5B018E    790            LD        DE,(VHSTP)
8E4B  E3          800 STEP       EX        (SP),HL
                  810 ;HACER EL PASO A LO LARGO DE X
8E4C  7C          820            LD        A,H
8E4D  82          830            ADD       A,D
8E4E  67          840            LD        H,A
                  850 ;HACER EL PASO A LO LARGO DE Y
8E4F  7D          860            LD        A,L
8E50  83          870            ADD       A,E
8E51  6F          880            LD        L,A
                  890 ;EL PLOT REAL !!!!!
8E52  C5          900            PUSH      BC
8E53  CD048E      910            CALL      PLOT
8E56  C1          920            POP       BC
                  930 ;RECUPERAR CONTADOR
8E57  E3          940            EX        (SP),HL
8E58  25          950            DEC       H
8E59  20DC        960            JR        NZ,NXTSTP
8E5B  E1          970            POP       HL
8E5C  C9          980            RET
```

En el siguiente programa de demostración en lenguaje máquina, he combinado el empleo de CLS (véase cap. 1) con PLOT y DRAW para producir una estructura de 24 líneas que, según me han dicho entendidos en la materia, se parece a un pisapapeles pesado sobre un almohadón blando visto desde arriba.

Si está llamando a la rutina con el comando directo USR del BASIC, sería una buena idea seguir la función USR con una sentencia PAUSE 0 de forma que las dos líneas inferiores no sean destruidas al volver. Los comentarios del listado deben proporcionar una explicación adecuada del funcionamiento del programa.

```
                   10 ;RUTINA DE DEMOSTRACION PARA CLS, PLOT Y DRAW
                   20 ;
8AFA  3E0E         30 MOIRE     LD        A,#E
                   40 ;PAPEL AZUL, TINTA AMARILLA
8AFC  CDFC8A       50           CALL      CLS
                   60 ;OVER1
8AFF  3E02         70           LD        A,2
8B01  32018B       80           LD        (PFLAG),A
                   90 ;BORDER 6
8B04  3E06        100           LD        A,6
8B06  D3FE        110           OUT       (#FE),A
                  120 ;PONER ATRIBUTOS Y MASCARA
8B08  210E00      130           LD        HL,#E
8B0B  220B8B      140           LD        (ATT),HL
                  150 ;TRAZAR BORDE SUPERIOR (BC=0)
8B0E  2100FF      160           LD        HL,#FF00
8B11  CD118B      170           CALL      DRAW
                  180 ;TRAZAR BORDE IZQUIERDO
8B14  2C          190           INC       L
8B15  01BF00      200           LD        BC,#BF
8B18  CD118B      210           CALL      DRAW
8B1B  24          220           INC       H
                  230 ;AHORA CREAR SILUETA EN RESTANTES
                  240 ;(255*191) PIXELS
8B1C  E5          250 NXTDR1    PUSH      HL
                  260 ;TRAZAR DESDE LADO IZQ. AL CENTRO
8B1D  016080      270           LD        BC,#8060
```

```
8B20  CD118B    280           CALL      DRAW
                290 ;AHORA DESDE CENTRO A DER.
8B23  C1        300           POP       BC
8B24  C5        310           PUSH      BC
8B25  06FF      320           LD        B,#FF
8B27  CD118B    330           CALL      DRAW
                340 ;DECREMENTAR CONTADOR A SIGUIENTE FILA
                350 ;HACIA ARRIBA DE LA PANTALLA
8B2A  E1        360           POP       HL
8B2B  2D        370           DEC       L
8B2C  20EE      380           JR        NZ,NXTDR1
8B2E  2C        390           INC       L
8B2F  E5        400 NXTDR2    PUSH      HL
                410 ;TRAZAR DESDE BORDE SUPERIOR AL CENTRO
8B30  016080    420           LD        BC,#8060
8B33  CD118B    430           CALL      DRAW
                440 ;AHORA DESDE CENTRO HASTA EL
                450 ;BORDE INFERIOR
8B36  C1        460           POP       BC
8B37  C5        470           PUSH      BC
8B38  0E8F      480           LD        C,#BF
8B3A  CD118B    490           CALL      DRAW
                500 ;INCREMENTAR CONTADOR A SIGUIENTE
                510 ;COLUMNA DE PIXELS (HACIA LA DER.)
8B3D  E1        520           POP       HL
8B3E  24        530           INC       H
8B3F  20EE      540           JR        NZ,NXTDR2
8B41  C9        550           RET
```

Si lo más importante es la velocidad, sólo le aconsejo utilizar la rutina DRAW si necesita líneas "generales" desde puntos no específicos. Es casi siempre más rápido utilizar una rutina personalizada, quizá empleando una tabla de referencia de las coordenadas del dibujo, en el caso de que se tracen líneas específicas.

Por ejemplo, si necesita trazar con frecuencia una línea recta a lo largo de la fila 0 de la pantalla, entonces es mucho más rápido cargar #FF en los primeros 32 octetos del archivo de pantalla para trazar (DRAW) desde (0,0) hasta (255, 0).

4

Pantallas de carga animadas

No se le habrá pasado por alto que casi todos los programas de juego del Spectrum dignos de mención presentan una bonita imagen en pantalla para entretenerle, mientras el código máquina sigue su lento y laborioso camino a lo largo del cable negro desde el cassette hasta su ordenador. La estética de esta "pantalla de carga" es generalmente proporcional a la cantidad de trabajo duro y papel de gráfico que se emplee en el diseño y codificación de la imagen, siendo ambos considerables a veces, incluso con el empleo de un buen programa de "diseño de gráficos en pantalla".

En este capítulo voy a proporcionarle dos rutinas cortas de utilidad para elaborar un estilo alternativo de pantalla de carga espectacular a la vista pero de fácil realización.

Usted se preguntará: "¿Qué hace el ULA mientras el Z-80 se ocupa de la carga del programa desde la cinta hasta la RAM?" La contestación es la misma de siempre; se hace cargo de toda la entrada, salida y de la generación de la imagen en pantalla. La última función incluye el parpadeo (FLASH) de los colores de tinta (INK) y papel (PAPER) de cualquier celdilla cuyo atributo tenga el bit 7 puesto a uno. Podemos utilizar esta propiedad para conseguir una pantalla que parpadee entre dos imágenes, quizá mostrando una figura en dos posiciones diferentes, de forma que se obtiene "animación" durante la carga u, otra alternativa, mostrando dos palabras distintas del título de juego.

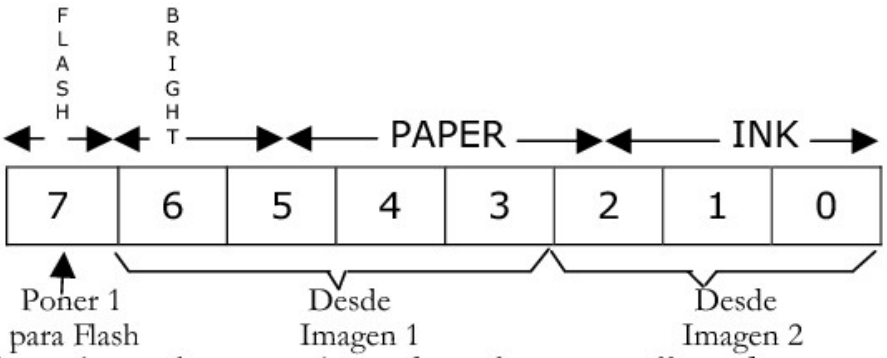
Los conceptos implicados en esta técnica son muy sencillos. Tomaremos una pantalla en blanco y a continuación imprimimos espacios de varios colores

en ella, controlando el color de estos espacios con comandos PAPER. Las celdillas de colores resultantes formarán nuestra primera imagen, y por tanto, tenemos una rejilla de 32 columnas x 24 líneas para diseñar en ella la imagen, teniendo cada celdilla uno de los ocho colores. Cuando la primera imagen está acabada, utilizaremos una rutina corta en lenguaje máquina para copiar los atributos en los 768 octetos de la memoria reservada para su empleo posterior.

Entonces utilizamos otra secuencia BASIC para imprimir una segunda imagen de celdillas coloreadas en blanco en la pantalla, y "emparejamos" los atributos de la segunda imagen con los de la primera. Los atributos de PAPER de la segunda imagen se desplazan 3 bits a la derecha (a la posición de INK) y a continuación se mezclan con los bits de PAPER y de BRIGHT de la primera imagen.

Entonces se pone a 1 el bit de FLASH y se reemplaza el octeto completo del archivo de atributos. A partir de este momento, la celdilla en cuestión empieza a parpadear (FLASH) entre los dos colores proporcionados por las imágenes (éstos pueden, desde luego, ser los mismos).

Puede ser útil un diagrama de composición del nuevo octeto de atributo:



Esta técnica tiene la ventaja sobre las pantallas de carga convencionales de que se necesita únicamente el archivo de atributos de 768 octetos, en comparación con los 6,75K (6912 octetos) de memoria que ocupa la pantalla de carga estándar. De esta forma se puede cargar la pantalla en la novena parte del tiempo que se tardaría normalmente, o unos 5 segundos, pasando rápidamente a cargar el juego propiamente.

Se empleó por primera vez comercialmente una pantalla de carga "animada" en el conocido Manic Miner de Bug Byte, ahora publicado por Software Projects. Las dos imágenes que se utilizaron fueron dos rótulos coloreados de las palabras "Manic" y "Miner".

Bueno, eso es todo sobre la teoría, así que ¿qué le parece algo sobre lenguaje máquina? Primero necesitamos una rutina de desplazamiento de un bloque para copiar el archivo de atributos a memoria "segura", es decir, a la que he reservado como los 768 octetos de la etiqueta IMAGE 1.

```
10 :MOVER IMAGE1 DESDE ARCHIVO DE ATRIBUTOS
20 :AL AREA DE ALMACENAMIENTO
30 :
40 :SE CONSERVA : A
50 :SALIDA : HL=#5B00.BC=0
60 :
```

```
8789  210058      70 ATTSTR    LD        HL,#5800
878C  119587      80           LD        DE,IMAGE1
878F  010003      90           LD        BC,#300
8792  EDB0       100           LDIR
8794  C9         110           RET
8795             120 IMAGE1    DEFS      768
```

La rutina para "mezclar" la imagen almacenada con la que se encuentra en el archivo de atributos es casi igual de sencilla. BLEND coloca la nueva imagen de nuevo en el archivo de atributos y ¡ésa es toda la explicación que necesita este listado!

```
                  10 ;MEZCLAR IMAGE1 DESDE ALMACENAMIENTO CON
                  20 ;ATRIBUTOS ACTUALES
                  30 ;
                  40 ;SALIDA : DE=#5B00,BC=0,A=0
                  50 ;
89FC  110058      60 BLEND     LD        DE,#5800
89FF  21FF89      70           LD        HL,IMAGE1
8A02  010003      80           LD        BC,#300
                  90 ;TOMAR OCTETO DE IMAGE1
                 100 ;
                 110 ;
8A05  7E         120 NXTATT    LD        A,(HL)
                 130 ;
                 140 ;ENMASCARAR SUS VALORES PAPEL Y BRILLO
                 150 ;
8A06  E678       160           AND       #78
                 170 ;
                 180 ;ALMACENAMIENTO DE NUEVO
                 190 ;
8A08  77         200           LD        (HL),A
                 210 ;
                 220 ;TOMAR ATRIBUTOS ACTUALES
                 230 ;
8A09  1A         240           LD        A,(DE)
                 250 ;
                 260 ;CAMBIAR LOS BITS DE PAPEL A BITS DE
                 270 ;TINTA Y ENMASCARARLOS
                 280 ;
8A0A  0F         290           RRCA
8A0B  0F         300           RRCA
8A0C  0F         310           RRCA
8A0D  E607       320           AND       7
                 330 ;
                 340 ;MEZCLAR ESTOS BITS CON LOS DE
                 350 ;PAPEL Y BRILLO DE IMAGE1
                 360 ;
8A0F  B6         370           OR        (HL)
                 380 ;
                 390 ;PONER FLASH1
                 400 ;
8A10  F680       410           OR        #80
                 420 ;
                 430 ;ALMACENAR EL OCTETO ACABADO EN ARCH. ATRI.
                 440 ;
8A12  12         450           LD        (DE),A
                 460 ;
                 470 ;REPETIR PARA LA SIGUIENTE CELDILLA
                 480 ;
8A13  13         490           INC       DE
8A14  23         500           INC       HL
8A15  0B         510           DEC       BC
8A16  78         520           LD        A,B
```

```
8A17  B1        530         OR      C
8A18  20EB      540         JR      NZ,NXTATT
8A1A  C9        550         RET
```

Una vez que la imagen final ha sido creada en el archivo de atributos puede, o salvar (SAVE) los octetos directamente en la cinta utilizando:

```
; SAVE  (NOMBRE)  CODE  22528,768
```

o, si ha estado utilizando las dos líneas inferiores y no quiere que se estropeen con los mensajes de la cinta, entonces emplee ATTSTR de nuevo para desplazar los atributos a memoria "segura" que no sea afectada por la pantalla, y sálvelos (SAVE) desde allí.

5

Exploración del teclado

En este capítulo explicaré cómo es el "mapa" del teclado y cómo se leen las teclas o grupos de ellas en lenguaje máquina.

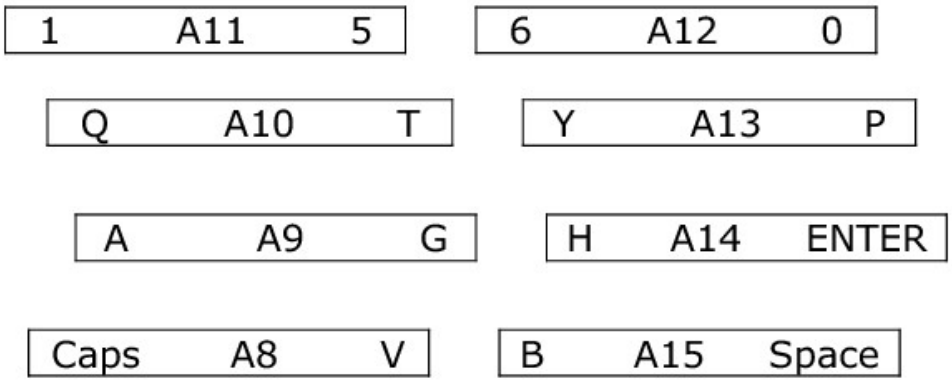
Los que tengan más idea de números se habrán dado cuenta de que el Spectrum tiene 40 teclas. Estas aparecen en cuatro filas de 10, pero el ordenador encuentra más fácil considerarlas como 8 medias filas de 5 teclas, porque hay menos de 10 bits en un octeto.

Si alguna vez se ha atrevido a quitar la tapa de su Spectrum (una cosa que no se recomienda, ya que este detalle anula su garantía), habrá visto que el teclado está conectado con la placa del circuito impreso con dos cables de cinta con aspecto bastante endeble. Un examen más detallado revela que uno de éstos tiene 8 pistas y el otro 5. De hecho, cada pista del mayor de los cables está conectada a uno de los bits 8 a 15 (el octeto de mayor peso) del bus de direcciones, mientras que el cable más pequeño está conectado a los 5 bits inferiores (0 a 4) del bus de datos.

Cuando el Spectrum realiza una exploración completa del teclado (cada 50-avo de segundo), el proceso que efectúa es como sigue. Aplica una "corriente" a cada una de las líneas de dirección por turno. Ahora cada una de las 5 teclas de la media fila correspondiente puede considerarse como un interruptor conectado entre una de las 5 líneas de datos y la línea de direcciones, permitiendo que la corriente circule cuando se encuentra pulsada. El ordenador lee las 5 líneas de datos, y si la corriente llega por una línea, sabe que se encuentra pulsada la tecla correspondiente, y actúa en consecuencia.

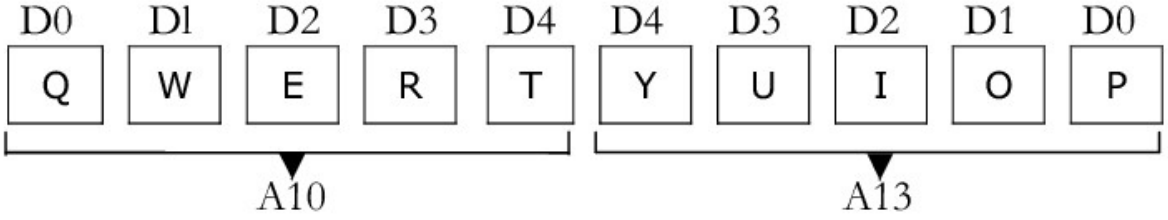
Etiquetamos las líneas de direcciones (por convenio) de A8 a A15 y las

líneas de datos D0 a D4. Las líneas de direcciones están asignadas a las medias filas de la siguiente forma:



Cuando queremos leer una media fila en particular, enviamos su línea de direcciones baja (cero). De la misma forma, cuando se pulsa una tecla en esta media fila, su línea de datos está baja (cero). Si no, está alta (uno).

Las líneas de datos se adjuntan a cualquier media fila con el bit inferior (D0) en la parte de fuera, contando hacia dentro. Por tanto, la forma del mapa para la segunda fila (por ejemplo) es la siguiente:



Bueno, acabada la teoría, pasaremos a la práctica: el teclado en sí es seleccionado (a diferencia de otros periféricos tales como un microdiskette o una impresora) enviando la línea de direcciones A0 baja. Por tanto, el octeto de menor peso de nuestra dirección del puerto de entrada es #FE y, o bien utilizamos la instrucción:

```
IN      A(#0FE)
```

o bien cargamos #0FE en el registro C y utilizamos

```
IN      r,(C)
```

donde r es un registro simple. Sin embargo, primero tenemos que cargar el octeto de mayor peso de la dirección en A o B (dependiendo de la instrucción

que estemos utilizando). Por ejemplo, supongamos que queremos leer la media fila que va desde A hasta C. Esta tiene la línea A9 (bit 1 del octeto de mayor peso), así que cargamos nuestro registro con el número binario 1111 1101 = #FD.

Por consiguiente, un fragmento apropiado para leer la media fila sería:

```
LD    A,#FD
IN    A,(#FE)
```

Para su provecho, he aquí una tabla de octetos de mayor peso para leer cada media fila.

MEDIA FILA	LINEA	BIT	OCTETO MAYOR P.	BITS
CAPS SHIFT-V	A8	0	FE	= 1 1 1 1 1 1 1 0
A-G	A9	1	FD	= 1 1 1 1 1 1 0 1
Q-T	A10	2	FB	= 1 1 1 1 1 0 1 1
1 -5	A11	3	F7	= 1 1 1 1 0 1 1 1
6-0	A12	4	EF	= 1 1 1 0 1 1 1 1
Y-P	A13	5	DF	= 1 1 0 1 1 1 1 1
H-ENTER	A14	6	BF	= 1 0 1 1 1 1 1 1
B-SPACE	A15	7	7F	= 0 1 1 1 1 1 1 1

Ahora podemos elaborar un fragmento de programa para comprobar la tecla BREAK (SPACE). Para probarla por sí sola, en vez de con CAPS SHIFT, utilizamos

```
LD      A,#7F
IN      A,(#FE)
RRA     ;PONE          D0 EN EL CARRY
JP      NC,BREAK       ;BREAK SI D0=0
```

Mientras estamos con el tema de BREAK, le puede interesar saber que, debido a una casualidad del diseño de los circuitos electrónicos del Spectrum, es posible hacer un BREAK (interrupción de la ejecución) al ordenador en BASIC sin pulsar realmente la tecla BREAK. Por alguna razón, al pulsar CAPS SHIFT con cualquiera de las siguientes parejas de teclas hace que D0 se ponga baja en todos los casos que se envíe A15 baja, haciendo que el Spectrum crea que se está pulsando BREAK.

He aquí las cuatro combinaciones mágicas:

CAPS SHIFT con Z y SYMBOL SHIFT
CAPS SHIFT con X y M

CAPS SHIFT con C y N
CAPS SHIFT con V y B

Así que ya sabe lo que tiene que hacer la próxima vez que su tecla BREAK se estropee. Por cierto, verá que si toma cualquier fila entera de teclas, a continuación pulsa dos cualesquiera de ellas que estén ligadas a la misma línea de datos, y luego pulsa cualquier otra tecla de la misma fila, al Spectrum le parecerá que también se ha pulsado la otra tecla de esa fila de la misma línea de datos. Esto puede parecer un poco complicado, por tanto, voy a dar un ejemplo. Pulse T e Y a la vez (ambas de la línea de datos D4 de la segunda fila). Ahora pulse W (en D1). El ordenador pensará que también se está pulsando 0, puesto que también está en la línea D1. Esto tiene poca utilidad en la práctica, pero resulta fascinante.

Es posible leer más de media fila de una vez, simplemente reinicializando más de un bit del octeto de mayor peso de la dirección de entrada. Por ejemplo, para leer la fila inferior completa (líneas A8 y A15), el valor sería el número binario

```
; 0111 1110 = #7E
```

El valor devuelto se determina como se indica a continuación. Si se pulsa cualquiera de las teclas ligadas a una línea de datos en particular, entonces el bit correspondiente es cero. En caso contrario, está a uno. Por tanto, si estamos explorando las dos filas inferiores del teclado, entonces se reinicializará D1 si se pulsa cualquiera de las teclas Z, S, L y SYMBOL SHIFT.

Esto nos lleva a una forma fácil de verificar si está pulsada cualquier tecla del teclado, como podría necesitarse antes del principio de un juego nuevo.

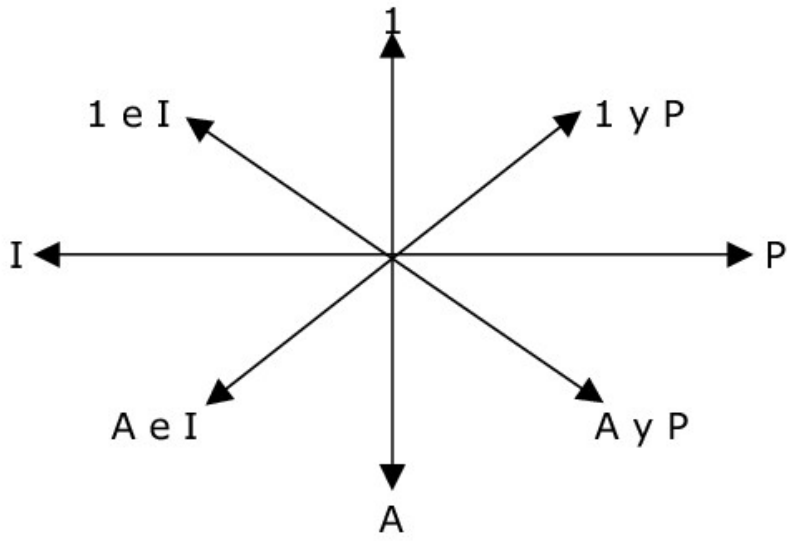
```
        XOR   A    ;A=0,POR LO TANTO BUSCA TODAS MEDIAS FILAS

WAIT    IN    A(#FE) ;LEER TABLA
        CPL   ;DESENMASCARAR LOS BITS NECESARIOS Y
        AND   #1F ;COMPROBAR TODOS LOS UNOS
        JP    NZ,GO ;SALTAR SI TECLA PULSADA
        JR    WAIT
```

Si se está pulsando una tecla, entonces se inicializará la bandera cero y se hará un salto para empezar el juego o cualquier otra cosa que desee. En caso contrario, la rutina volverá a WAIT con el registro A conteniendo cero de nuevo.

Ahora tenemos toda la información necesaria para diseñar una rutina completa de exploración del teclado. Como ejemplo, describiré el desarrollo de una rutina de juegos, que proporciona un control en 8 direcciones y un botón de "disparo".

Utilizaré las teclas 1 para arriba, A para abajo, I para izquierda, P para derecha y cualquier tecla de la fila inferior para "disparo", es decir:

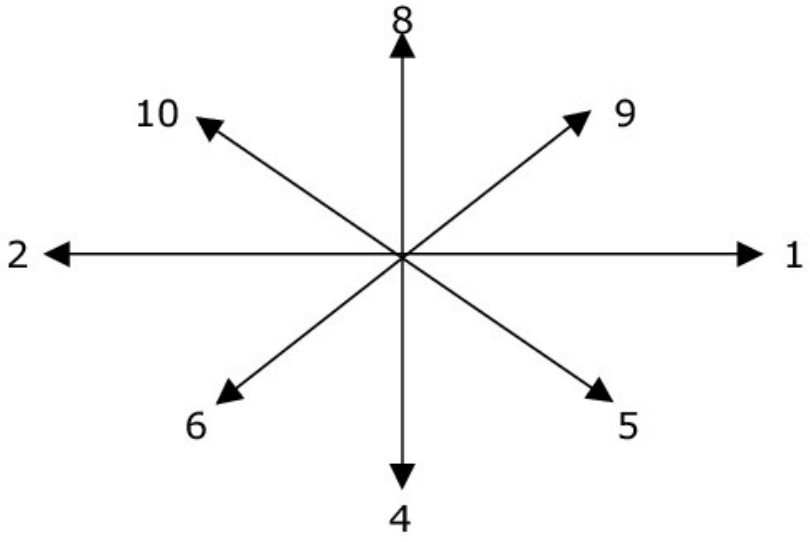


CAPS SHIFT	"DISPARO"	SPACE

La rutina devolverá un "código de control" que dependerá de qué teclas de control estén pulsadas. Asignando un bit del código a cada una de las 4 direcciones principales arriba, abajo, izquierda y derecha, y un bit para nuestra "barra de disparo", podemos denotar todas las demás direcciones haciendo las combinaciones de estos 5 bits. He asignado los bits como sigue:

BIT	CONTROL
0	DERECHA
1	IZQUIERDA
2	ABAJO
3	ARRIBA
4	DISPARO

De esta forma los valores devueltos serán como se indica a continuación:



+16 CUANDO SE PULSA "DISPARO"

Los tres bits de más en el código serán, por supuesto, ceros. Observe que puesto que "Noreste" es una combinación de hacia arriba y a la derecha, el código de control correspondiente es 8 + 1 = 9. De la misma forma para las otras direcciones diagonales.

Ahora bien, está claro que hay algunos valores de control que no tendrían sentido. Por ejemplo, el valor 3 (= 1 + 2) indicaría un deseo de ir a la izquierda y a la derecha a la vez. El usuario ha pulsado demasiadas teclas y, en lugar de una rutina que diera prioridad a una de las direcciones, sería más justo que se ignorasen ambas pulsaciones de las teclas. Esto se puede hacer llamando a la subrutina CHECK dos veces seguidas; una vez para izquierda-derecha y otra para arriba-abajo. Al entrar a CHECK, el registro B contiene una "máscara" para los dos bits que deseamos examinar. Estamos comprobando el código binario "no válido" 11, y para ello utilizamos el fragmento:

```
CHECK   LD      A,C ;C CONTIENE EL CODIGO DE CONTROL
        CPL
        AND     B
        RET     NZ
```

que vuelve si se ve que el control es válido. En caso contrario, hemos encontrado el código no válido 11 en C, y la rutina completa su tarea al volver a inicializar los bits molestos. Para ello:

```
LD      A,B
XOR     C
LD      C,A
RET
```

El resto del listado de la rutina se explica solo y es muy demostrativo. Por tanto, helo aquí

```
                        10 ;ENTRADA NINGUNA
                        20 ;SE CONSERVAN: DE,HL
                        30 ;SALIDA:C=CODIGO DE CONTROL,B=12
                        40 ;
                        50 ;LEER MEDIA FILA Y-P
                        60 ;
8B2E  3EDF              70 SCAN1    LD      A,#DF
8B30  DBFE              80          IN      A,(#FE)
                        90 ;
                       100 ;ENMASCARAR TECLAS I Y P
                       110 ;
8B32  2F               120          CPL
8B33  E605             130          AND     5
                       140 ;
                       150 ;PONER I EN BIT 1, Y P EN EL CARRY
                       160 ;
8B35  1F               170          RRA
                       180 ;
                       190 ;MOVER P DESDE EL CARRY AL BIT 0
                       200 ;
8B36  CE00             210          ADC     A,0
```

```
                        220 ;
                        230 ;ALMACENAR EL CONTROL IZQ. Y DER. EN C
                        240 ;
8B38  4F                250             LD      C,A
                        260 ;
                        270 ;LEER LA TECLA A
                        280 ;
8B39  3EFD              290             LD      A,#FD
8B3B  DBFE              300             IN      A,(#FE)
8B3D  1F                310             RRA
                        320 ;
                        330 ;SALTAR SI A NO ESTA PULSADA
                        340 ;
8B3E  3802              350             JR      C,NDOWN
8B40  CBD1              360             SET     2,C
                        370 ;
                        380 ;LEER TECLA 1
                        390 ;
8B42  3EF7              400 NDOWN       LD      A,#F7
8B44  DBFE              410             IN      A,(#FE)
8B46  1F                420             RRA
                        430 ;
                        440 ;SALTAR SI 1 NO ESTA PULSADO
                        450 ;
8B47  3802              460             JR      C,NUP
8B49  CBD9              470             SET     3,C
                        480 ;
                        490 ;COMPROBAR SI ESTAN PULSADAS
                        500 ;A LA VEZ IZQ. Y DER.
                        510 ;
8B4B  0603              520 NUP         LD      B,3
8B4D  CD608B            530             CALL    CHECK
                        540 ;COMPROBAR SI ESTAN PULSADAS
                        550 ;A LA VEZ ARRIBA Y ABAJO
8B50  060C              560             LD      B,12
8B52  CD608B            570             CALL    CHECK
                        580 ;
                        590 ;LEER LA FILA DE ABAJO
                        600 ;
8B55  3E7E              610             LD      A,#7E
8B57  DBFE              620             IN      A,(#FE)
8B59  2F                630             CPL
8B5A  E61F              640             AND     #1F
                        650 ;
                        660 ;VOLVER EXCEPTO SI "DISPARO"
                        670 ;
8B5C  C8                680             RET     Z
8B5D  CBE1              690             SET     4,C
8B5F  C9                700             RET
                        710 ;
                        720 ;COMPROBAR DIRECCIONES "IMPOSIBLES"
                        730 ;
8B60  79                740 CHECK       LD      A,C
8B61  2F                750             CPL
8B62  A0                760             AND     B
8B63  C0                770             RET     NZ
8B64  78                780             LD      A,B
8B65  A9                790             XOR     C
8B66  4F                800             LD      C,A
8B67  C9                810             RET
```

6

Teclas de control seleccionables por el jugador

En el último capítulo he concluido con un ejemplo de cómo se desarrolla una rutina típica de control del teclado de juegos, utilizando una selección predeterminada de teclas. A menudo puede ser una ventaja, en cuanto a cuestiones de comodidad, si se permite al usuario seleccionar las teclas de control que más le convengan y el número de las mismas. En este capítulo proporciono las rutinas fundamentales para permitirle hacer esto.

Imagínese, si quiere, a nuestro jugador típico, inclinado sobre el teclado y a la espera de cualquier comando, mientras que el juego termina su recorrido largo y tortuoso desde la cinta a la memoria. Se le pide que pulse cualquier tecla (como ocurre siempre). Ahora se le pide que seleccione una tecla para controlar (digamos) el movimiento hacia arriba de su nave espacial. En ese momento, tenemos que esperar que deje de pulsar "cualquier tecla". El siguiente fragmento bastará, y es equivalente a la línea BASIC.

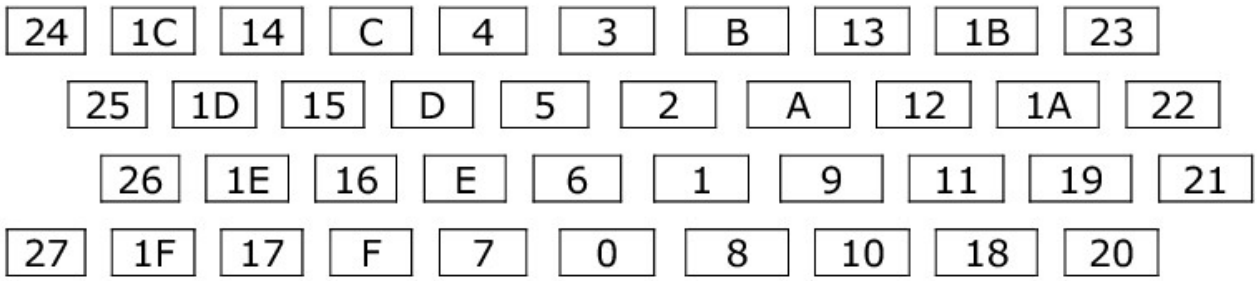
```
; 10 IF INKEY$ <> " " THEN GO TO 10

WAIT    XOR     A
        IN      A,(#FE)     ;LEER EL TECLADO COMPLETO
        CPL     ;SI CUALQUIER TECLA PULSADA
        AND     #1F         ;ENTONCES ESPERA
        JR      NZ,WAIT
```

Ahora estamos preparados para seleccionar el control "hacia arriba". Lo

que se necesita es una rutina que devuelva un valor único para cada tecla que se pulse, y que nos diga cuándo no está siendo pulsada más de una tecla. El valor de la tecla será almacenado para su uso durante el juego, mientras una rutina independiente nos indica si se pulsa la tecla asociada con ese valor.

La siguiente rutina, KFIND1, devuelve el valor de la tecla en el registro D con la bandera cero puesta a uno, si se pulsa solamente una tecla. Si no se pulsa ninguna tecla, entonces la bandera cero será puesta a cero indicando un error. Los valores de teclas, comprendidos en el intervalo desde #0 hasta #27 se encuentran asignados como se indica a continuación (todos los valores están expresados en hexadecimal):



A simple vista, esta presentación puede parecer un poco absurda hasta que se dé cuenta de que hace que las cosas sean más fáciles para la rutina de control del juego posterior. Mirando a los valores hex cuidadosamente, vemos que los 3 bits de menor peso nos dicen en qué media fila se halla la tecla (y por tanto, qué puerto hay que direccionar) mientras que los bits 3, 4 y 5 nos indican qué posición tiene la tecla en esa media fila. He aquí KFIND1:

```
                    10 ;ENTRADA : NINGUNA
                    20 ;SE CONSERVA : L
                    30 ;SALIDA : D=CODIGO TECLA, D=#FF SI NINGUNA TECLA PULSADA
                    40 ;BANDERA CERO DESACTIVADA SI MAS DE UNA TECLA PULSADA
                    50 ;SI NO, BANDERA CERO ACTIVADA   Y A=D
                    60 ;
                    70 ;
8AC5  112FFF        80 KFIND1    LD        DE,#FF2F
8AC8  01FEFE        90           LD        BC,#FEFE
                   100 ;
                   110 ;D COMIENZA EN "NINGUNA TECLA" Y CONTIENE EL VALOR INICIAL D
                     E
                   120 ;TECLA PARA CADA MEDIA-FILA
                   130 ;BC CONTIENE LA DIRECCION DEL PUERTO
                   140 ;
                   150 ;LEER UNA MEDIA-FILA
                   160 ;
8ACB  ED78         170 NXHALF   IN        A,(C )
8ACD  2F           180          CPL
8ACE  E61F         190          AND       #1F
                   200 ;
                   210 ;SALTAR SI NINGUNA TECLA  PULSADA
                   220 ;
8AD0  280C         230          JR        Z,NPRESS
                   240 ;
                   250 ;COMPROBAR SI MAS DE UNA TECLA PULSADA
8AD2  14           260          INC       D
                   270 ;
```

```
                280 ;
                290 ;SI ES ASI VOLVER CON Z A CERO
8AD3  C0        300           RET       NZ
                310 ;
                320 ;
                330 ;CALCULAR VALOR DE TECLA
8AD4  67        340           LD        H,A
8AD5  7B        350           LD        A,E
8AD6  D608      360 KLOOP     SUB       8
8AD8  CB3C      370           SRL       H
8ADA  30FA      380           JR        NC,KLOOP
                390 ;
                400 ;COMPROBAR SI MAS DE UNA TECLA PULSADA
                410 ;
8ADC  C0        420           RET       NZ
                430 ;
                440 ;ALMACENAR VALOR DE TECLA EN D
8ADD  57        450           LD        D,A
                460 ;
                470 ;COMPROBAR LAS OTRAS 7 MEDIAS FILAS
                480 ;
8ADE  1D        490 NPRESS    DEC       E

8ADF  CB00      500           RLC       B
8AE1  38E8      510           JR        C,NXHALF
                520 ;
                530 ;
                540 ;PONER BANDERA CERO
8AE3  BF        550           CP        A
8AE4  C8        560           RET       Z
```

Un fragmento típico para esperar una pulsación válida de tecla en contestación a nuestro mensaje "por favor elija una tecla para el movimiento hacia arriba" sería:

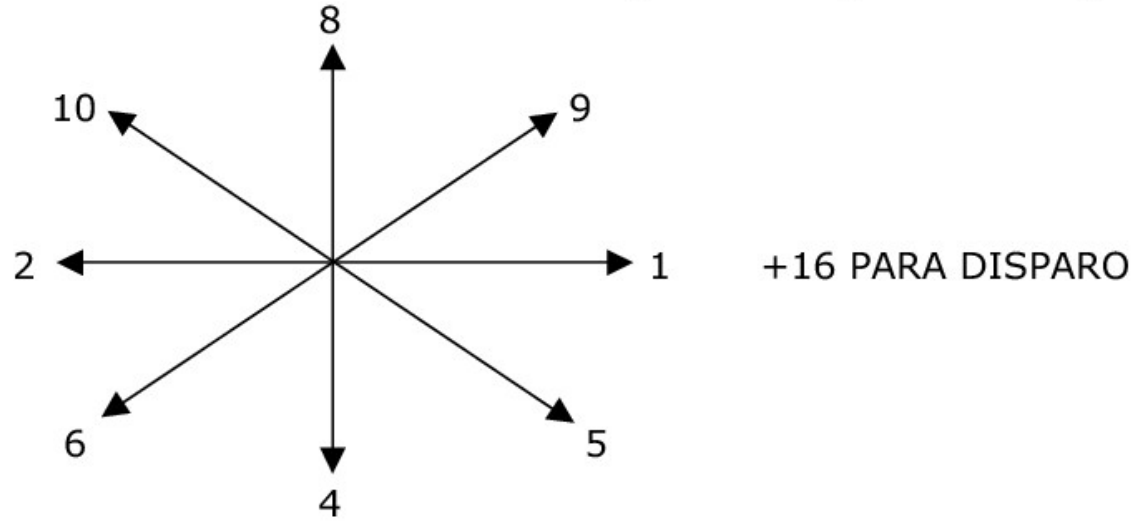
```
REPT    CALL    KFIND1    ;BUSCA TECLADO
        JR      NZ,REPT   ;REPETIR SI ENTRADA ILEGAL
        INC     D         ;REPETIR SI NINGUNA TECLA ESTABA
        JR      Z,REPT    ;PULSADA
        DEC     D
```

He llamado KTEST1 a la rutina complementaria de KFIND1. Cada vez que el Spectrum necesita un control de teclado durante un juego, debemos llamar a KTEST1 una vez para cada tecla de control seleccionada. La rutina leerá dicha tecla y volverá con la bandera carry (acarreo) puesta a cero si está pulsada, o a uno en caso contrario. El único parámetro que KTEST1 necesita es el valor de la tecla que estamos comprobando, introducido en el acumulador. He aquí el listado, seguido por un ejemplo:

```
                 10 ;ENTRADA:A=VALOR DE LA PRUEBA DE LA TECLA
                 20 ;SE CONSERVAN : HL,DE
                 30 ;SALIDA :CARRY A CERO SI TECLA PULSADA
                 40 ;SI NO BC=0
                 50 ;
                 60 ;
893C  4F         70 KTEST1   LD        C,A
                 80 ;
                 90 ;PONER B=16-(NO. LINEA DIRECCION)
```

```
                100 ;
893D  E607      110           AND       7
893F  3C        120           INC       A
8940  47        130           LD        B,A
                140 ;
                150 ;PONER C=(NO. LINEA DE DATOS)+1
                160 ;ES DECIR PONER C=5-INT(C/8)
                170 ;
8941  CB39      180           SRL       C
8943  CB39      190           SRL       C
8945  CB39      200           SRL       C
8947  3E05      210           LD        A,5
8949  91        220           SUB       C
894A  4F        230           LD        C,A
                240 ;CALCULAR EL OCTETO DE MAYOR PESO DE LA DIRECCION DEL PUERTO
                250 ;
894B  3EFE      260           LD        A,#FE
894D  0F        270 HIFIND    RRCA
894E  10FD      280           DJNZ      HIFIND
                290 ;
                300 ;LEER MEDIA-FILA
                310 ;
8950  DBFE      320           IN        A,(#FE)
                330 ;
                340 ;PONER EL BIT DE TECLA NECESITADO EN EL CARRY
                350 ;
8952  1F        360 NXKEY     RRA
8953  0D        370           DEC       C
8954  20FC      380           JR        NZ,NXKEY
8956  C9        390           RET
```

Como ejemplo, supongamos que nuestro juego implica un movimiento en 8 direcciones y un botón de disparo. Para esto, nuestro usuario tendrá que haber escogido 5 teclas de control, para arriba, abajo, izquierda, derecha y disparo. Almacenemos los 5 valores de control elegidos en una tabla apuntada por HL, para poder disparar, subir, bajar, desplazarse a la izquierda, o a la derecha. Utilizaremos los mismos "valores de control" que los del capítulo anterior, es decir:



Una rutina adecuada para elaborar el valor de control en el registro E es la siguiente:

```
        LD    E,8         ;E ES TAMBIEN UN CONTADOR
NXTKEY  LD    A,(HL)      ;TOMAR VALOR DE TECLA
```

```
INC       HL
CALL      KTEST1     ;BUSCAR UNA TECLA PULSADA
CCF       ;1=PULSADA, 0=NO
RL        E          ;PONER BIT DE TECLA EN E
JR        NC,NXTKEY;REPETIR PARA LAS OTRAS
RET       ;4 TECLAS
```

Observe que he hecho que el registro sea "doble" como contador del bucle. El bit inicial 3 es el bit puesto a uno más alto, y se desplaza a la izquierda una vez a cada paso por el bucle hasta que después de 5 pasadas cae en el "carry" (acarreo) provocando que la rutina vuelva con el valor de control completo en el registro E. Las verificaciones usuales de direcciones "imposibles" tales como izquierda y derecha pueden llevarse a cabo entonces.

7

Todo cuanto debe saber sobre interrupciones

Como probablemente sabrá, el microprocesador Z-80 nos ofrece una selección de tres modos de interrupción enmascarables, denominados según las instrucciones que los seleccionan: IM0, IM1 e IM2.

La instrucción IM0 en el Spectrum es bastante redundante. En este modo, el Z-80 espera una instrucción de algún periférico para empezar a realizar su camino por el bus de datos durante el ciclo de interrupción *acknowledge*. Sin embargo, en el caso del Spectrum, el bus de datos generalmente contiene #FF durante una interrupción, y esto es el código de un octeto Hex para RST #38, que el Z-80 ejecuta en su debido tiempo. La razón por la que he dicho que IM0 es redundante es que IM1 realiza exactamente la misma función que RST #38 cuando ocurre una interrupción, sin tener en cuenta el contenido del bus de datos en el momento.

El Spectrum normalmente opera en el modo uno de interrupciones, y en cualquier momento que ocurra una interrupción, la rutina en #38 procede a incrementar el contador del cuadro de la televisión y explora el teclado, actualizando todas las diferentes variables del sistema asociadas con ello. El número de las interrupciones que han sido aceptadas desde que se encendió el ordenador está contenido en la variable del sistema de 3-octetos FRAMES, en #5C78, 23672 decimal. El empleo de este contador está bien documentado tanto en el manual del Spectrum como en otros libros. Por esta razón, no haré más comentarios sobre este tema.

A menos que tenga ganas de utilizar el contador de cuadro o la explora-

ción del teclado mientras ejecuta su programa en lenguaje máquina, debería emplear la instrucción DI para desactivar las interrupciones que de otra forma le harían perder tiempo. Esto tiene especial importancia cuando está generando un sonido o haciendo uso de un trozo de programa que requiere precisión en el tiempo; en caso contrario oirá un zumbido de 50 Hz causado por lagunas en el sonido mientras se procesan las interrupciones.

El modo de interrupción enmascarado IM2 es el más complejo y potente. Cuando ocurre una interrupción, el Z-80 toma el octeto que se encuentra en el bus de datos como la parte menos significativa (de menor peso) de una dirección y el contenido del registro I, o "registro vector de interrupción" como octeto de mayor peso.

Esta dirección apunta a una segunda dirección almacenada (octeto de menor peso primero) en la memoria, que luego se carga en el contador de programa. A continuación comienza la ejecución de la subrutina en esa dirección. Como ejemplo, supongamos que el registro I contenía #FE, el bus de datos contenía #40 y que la dirección almacenada en #FE40 era #0038. Entonces el Z-80 construiría la dirección #FE40 partiendo del registro vector de interrupción y del bus de datos. Tomaría la dirección almacenada en #FE40 y saltaría a #0038, que da la casualidad que es la rutina normal de interrupción.

De hecho, como el bus de datos generalmente contiene #FF durante una interrupción, todas nuestras "direcciones de vector" terminarán en #FF. Un poco de experiencia le mostrará que para evitar una imagen "distorsionada" o "con nieve" en la pantalla del Spectrum, el registro I debe contener el octeto de mayor peso de una dirección de la ROM o de los 32 K superiores de la RAM.

	I = (Hex)	Dirección Vector (Hex)	Contenido (Hex)	Dirección (Decimal)
(1)	2B	2BFF	5C65	23563
(2)	29	29FF	5C76	23670
(3)	2E	2EFF	5CA1	23713
(4)	19	19FF	5D22	23842
	14	14FF	6469	25705
	1E	1EFF	67CD	26573
	0F	0FFF	6D18	27928
	06	06FF	71DD	29149
	28	28FF	7E5C	32348

(1) Los 3 octetos necesarios para una instrucción de salto a almacenar en #5C65 se encuentran contenidos normalmente en las variables del sistema STKEND y BREG y como tales no deben alterarse si pretende utilizar la pila de la calculadora o volver al BASIC después de su programa en lenguaje máquina.

(2) Si su programa es un híbrido BASIC/lenguaje-máquina, entonces observe que los dos primeros octetos de la dirección #5C76 contienen la variable SEED (semilla) para el generador de números seudo-aleatorios del BASIC y se modificará al utilizar RANDOMIZE o con el empleo de la función RND. El tercer octeto después de #5C76 es la parte menos significativa (de menor peso) de FRAMES, por tanto, debe asegurarse de que no se produzca ninguna interrupción IM1 en adelante una vez que se haya definido el interceptor de interrupciones a partir de #5C76. De lo contrario, la dirección de cualquier instrucción de salto insertada en #5C76 aumentaría ¡256 cada 20 ms!

(3) El cuarto registro de 5 octetos del área de la memoria de la calculadora empieza en #5CA1, por tanto, de nuevo, no utilice esta dirección si piensa emplear cualquiera de las rutinas de la calculadora de la ROM en su programa en lenguaje máquina.

(4) Si desea volver a, y utilizar, el BASIC, entonces también queda descartada la dirección #5D22. Sencillamente es demasiado baja: pruebe con el comando CLEAR 23841 ¡y verá lo que quiero decir!

Esto nos limita a los rangos desde #00 a #3F y desde #80 a #FF para I. Ahora bien, si tiene una máquina con 48K, no debería serle difícil encontrar una dirección de vector sin ocupar entre las 127 opciones posibles de los 32K superiores de la RAM (observe que prácticamente no podemos utilizar I = #FF, ya que la dirección almacenada a partir de #FFFF tendría su octeto de mayor peso en la posición 0, que se encuentra en la ROM). Sin embargo, si sólo se tiene una máquina con 16K o no hay sitio en los 32K superiores de la RAM, entonces tenemos que echar mano a las direcciones de vector de la ROM.

De estas 63 direcciones de vector (de nuevo no puede utilizarse realmente #3F, ya que el octeto de mayor peso de la dirección sería el primer octeto de la RAM de la pantalla), solamente 13 apuntan a direcciones de los 16K inferiores de la RAM. 4 de estos 13 están en la memoria de la pantalla, dejando elección entre las 9 direcciones de la página anterior.

El otro tipo de interrupción no implementado en el ZX Spectrum es el NMI, o interrupción no enmascarable. Si un Z-80 recibe una NMI, acaba la instrucción que está efectuando y llama la rutina en #66.

En el Spectrum esta rutina (En ROM) es la fuente de bastantes quebraderos de cabeza entre los fabricantes de periféricos *hardware*. La rutina es corno sigue:

```
#0066   PUSH   AF
        PUSH   HL
        LD     HL,(#5CB0)
        LD     A,H
        OR     L
        *JR    NZ,#0070
        JP     (HL)
#0070   POP    HL
        POP    AF
        RETN
```

La instrucción etiquetada con un * debería haber sido

```
JR      Z,#0070
```

y entonces habría causado un salto a la dirección contenida en #5CB0 (que, por cierto, se habla de ella en el manual del Spectrum como "no utilizada"), a menos que la dirección fuera cero, en cuyo caso se hubiera hecho un retorno. En vez de ello, tal y como está, la única utilidad posible de un NMI en el Spectrum es causar una total puesta a cero *(reset)* del sistema si la dirección contenida en #5CB0 es cero, como ocurre generalmente.

Dentro del Z-80 hay dos bits especiales llamados *flip-flops* de interrupción y se denominan IFF1 e IFF2. Normalmente se manejan juntos bajo el nombre colectivo de IFF, excepto durante un NMI, en cuyo caso IFF2 almacena el valor previo de IFF1, mientras que éste se pone a cero durante la duración de la NMI. La función de IFF es indicar al Z-80 si se permiten en ese momento las interrupciones enmascarables. Si están puestos a 1 se autorizan las interrupciones. Si están puestos a cero (enmascarados), entonces no se detectarán las

interrupciones enmascarables. Por tanto, obviamente EI los pone a 1 mientras que DI los pone a cero. Para ser totalmente preciso, los *flip-flops* están siempre puestos a cero mientras se está procesando DI o EI, y las interrupciones no se activan hasta que se haya ejecutado la instrucción de DESPUES de EI. Merece la pena explicar la razón de todo esto.

En cualquier momento que se acepte una interrupción, se pone a cero IFF automáticamente. Es, sin embargo, responsabilidad del programador reactivar las interrupciones antes de volver de la rutina de interrupciones con RETI. Podría causar un sinnúmero de problemas si tuviera lugar una interrupción entre el momento de activarlas y la vuelta desde la última, de ahí viene la "acción retardada" de EI para permitir que se efectúe un retorno seguro, como ocurre en:

```
EI
RETI
```

el final estándar de una rutina de interrupción.

Una instrucción que a menudo se pasa por alto en los libros sobre el lenguaje de máquina del Spectrum es:

```
LD     A,R
```

Esto a primera vista no parece tener ninguna utilidad, pero observando sus efectos en las banderas se demuestra lo contrario. Cuando se ejecuta la instrucción, la bandera de paridad/rebosamiento (P/V) se pone al contenido de IFF2. Por tanto, podemos utilizar la instrucción para que nos indique si se han activado las interrupciones enmascarables o no. Cuando la bandera P/V está a uno, normalmente indica una paridad par (PE-*parity even*), mientras que cuando se pone a cero indica una paridad impar (PO-*parity odd*).

Supongamos que deseamos conservar los contenidos de IFF mientras que desactivamos las interrupciones, para generar algún sonido "puro", y luego restauramos IFF. Un método adecuado podría ser el siguiente:

```
       LD    A,R         ;PONER P/V A IFF
       PUSH  AF          ;ALMACENAR P/V
       DI    ;INUTILIZAR INTERRUPCIONES
;  (PRODUCE SONIDO)
       POP   AF          ;RECUPERAR P/V
       JP    PO,NOT-ON ;SI PE ENTONCES P/V-1, POR LO TANTO
       EI    ;PONER IFF
NOT-ON
```

De esta forma, si introducimos la rutina con las interrupciones enmascaradas, entonces no estarán activadas al final. La instrucción LD A,I afecta a la bandera P/V de la misma manera que LD A,R.

Dije antes que el bus de datos "usualmente" contiene #FF durante una interrupción. En el caso de un Spectrum aislado, nunca he visto que esto no sea lo que ocurre. Hay, sin embargo ciertos periféricos *hardware* que no decodifican correctamente señales de las líneas IOREQ y READ del Z-80, y como resultado hacen que haya números variables en el *bus* de datos durante el ciclo de *acknow-*

ledge de la interrupción. Por cierto, estos periféricos no incluyen la impresora ZX ni el ZX Interface 1.

Ahora bien, obviamente si el valor en el bus de datos cambia, entonces tendremos que establecer toda una tabla de vectores de interrupción en la memoria para IM2, para que cualquiera de los posibles valores del bus provoque aún un salto a la dirección correcta.

Si sabemos que el valor será par, entonces sencillamente necesitamos una tabla de 128 direcciones de vector terminando en 00, 02, ..., #FE, conteniendo cada entrada la dirección de nuestra rutina de interrupciones. De la misma forma, si el bus de datos tuviera un valor impar, entonces tendríamos una tabla con un octeto más alto de la memoria, de forma que las direcciones de vector terminasen en 01, 03, ..., #FF.

Sin embargo, si el bus de datos contiene cualquiera de los 256 valores posibles, como, por ejemplo, ocurre cuando se conecta un mando de juego *(joystick)* de Kempston Microelectronics al puerto del usuario, entonces tenemos que utilizar una técnica ligeramente diferente. Cada uno de los 257 octetos de la tabla de vectores debe contener el mismo valor para que, sea cual sea la dirección del vector, par o impar, la dirección de interrupción siempre sea la misma. Por tanto, los octetos de mayor y menor peso de la dirección de la rutina de interrupciones deben ser los mismos. En caso de que esto no quede claro, supongamos como contradicción que la dirección de interrupción es #89AB. Si elaboramos una tabla insertando esta dirección 128 veces desde #FE00, entonces un valor par en el bus de datos causaría un salto correcto, pero un valor impar causaría un salto a #AB89: ¡obviamente no es lo que queremos! Observe que la tabla tiene 257 octetos de longitud, y no 256, puesto que debemos tener en cuenta que la dirección de vector termina en #FF, lo que causa que una entrada "se salga" a la siguiente página de la memoria.

Probablemente el valor más adecuado para dar al registro I es #FE, utilizando la página más alta posible de RAM para la tabla de vector. Si llenamos entonces la tabla con #FD, una interrupción hará que haya un salto a #FDFD, que está justo 3 octetos antes del principio de la tabla. Ahora bien, tres octetos son justo lo suficiente para colocar una instrucción de salto en nuestra rutina de interrupción "real". De esta forma hemos confinado la memoria que se necesita para una interrupción IM2 a prueba de errores en un bloque continuo de 250 octetos, sin afectar la versatilidad de la interrupción de ninguna manera (¡con la excepción de añadir los 10T-estados de la instrucción JP al tiempo del proceso !).

He aquí una rutina adecuada para inicializar el sistema IM2 descrito arriba:

```
INT   LD    HL,#FE00   ;CARGA TABLA EN #FE00
      LD    BC,#00FD   ;CON 256 DE #FD
LP1   LD    (HL),C
      INC   HL
      DJNZ  LP1
      LD    (HL),C     ;LA 257-AVA ENTRADA
      LD    A,#FE      ;DEJAR I=#FE
      LD    I,A
      IM    2          ;SELECCIONAR IM2
      RET
```

```
ORG     #FDFD
JP      #0038          ;INSERTA SU PROPIA DIRECCION
```

La técnica anterior está muy bien si tiene 48K de RAM, pero no funcionará con una máquina que disponga de 16K. Como he mencionado antes, al apuntar el registro de vector de interrupción, cualquier página de los 16K inferiores de la RAM se obtendría "nieve" en la pantalla. Todo no se ha perdido, sin embargo, pues, aun en el caso de que haya por lo menos un periférico "granuja", el mando de juegos (*joystick* Kempston), hay una forma, aunque algo molesta y restringida, de usar el modo 2 de interrupción.

El mando de juegos se "lee" normalmente en BASIC con un comando de la forma:

```
; LET A= IN 31
```

pero de hecho todo lo que se necesita para que el interfaz ponga un valor en el *bus* de datos es enviar la línea de dirección A5 bajada (conteniendo cero) de modo que para el comando

```
; LET A= IN (31+64+128+256+512+1024)
```

por ejemplo, se haría igual. Sin embargo, cuando A5 está alta, el mando de juegos no afectará el contenido del *bus* de datos, y debería resultar el valor normal #FF durante el *acknowledge* de interrupción.

Bueno, con esto ya se acabó la teoría. ¿Ahora cómo podemos asegurarnos que A5 esté alta siempre que ocurra una interrupción bajo IM2? Esto puede hacerse asegurándose de que el contador de programa contiene una dirección que tiene el bit 5 puesto a uno antes de una interrupción, y ésta es la razón por la que dije que la técnica era "algo molesta y restringida".

En principio tenemos dos opciones una vez que el contador de programa está en un bloque de 32 octetos que tiene el bit 5 puesto a uno para sus direcciones; podemos, o bien llegar a una instrucción HALT mientras esperemos una interrupción, o podemos aprovechar el tiempo haciendo algo útil como generar un sonido. Si se elige la segunda opción, hay dos puntos principales que hay que recordar.

Primero, no debemos permitir que el bit 5 de PC se ponga a cero; por tanto, la rutina debe o bien encontrarse en un bloque de 32 octetos, o llamar a otras subrutinas que están también en posiciones donde A5 esté alta. Segundo (y suponiendo que no deseamos malgastar el tiempo con esta rutina una vez que ha tenido lugar una interrupción), debemos verificar continuamente algún tipo de bandera que se ponga a uno por la rutina de interrupción, para que sepamos cuándo se ha tomado una interrupción.

Una vez que se ha descartado la interrupción, tenemos hasta unos 20 ms, que son muchísimo tiempo en lenguaje máquina, para hacer tanto proceso "normal" como queramos antes de volver para esperar la siguiente interrupción.

A primera vista, todo el esfuerzo necesario para utilizar las interrupciones

IM2 en el Spectrum puede parecer que no merece la pena, pero en realidad tiene un amplio margen de utilidades. Son el concepto fundamental que hay detrás de muchas de las utilidades disponibles en el mercado, tales como relojes de tiempo real, rutinas TRACE, extensiones del BASIC, teclas de función definibles por el usuario, y así sucesivamente.

Además de todo esto, las interrupciones tienen la propiedad especial de que son generadas precisamente a la misma frecuencia que los cuadros que componen la visualización en TV. Siempre ocurren cuando el haz de electrones está en el punto más alto de su "retroceso al principio" desde la línea inferior hasta la superior de la pantalla, y en consecuencia podemos utilizar interrupciones para producir horizontes de pantalla completa (incluido el borde), animación de *sprites* sin temblores *pixel* a *pixel*, y color en resolución más alta, para mencionar sólo unas pocas de las posibilidades que se obtienen con el proceso sincronizado con TV.

8

Discusión de técnicas de animación de "pixels"

Desde que se lanzó el ZX Spectrum al mercado, la calidad de los juegos *software* que hay para él ha aumentado constantemente y, con ello, la calidad técnica de la animación. El interés principal ha pasado del movimiento de un carácter cada vez al de unos pocos *pixels* cada vez. Al mismo tiempo, se ha exprimido el Spectrum más y más hasta muy cerca de sus límites de diseño, arañando los programadores hasta la última gota de velocidad del microprocesador Z-80, en un esfuerzo para conseguir efectos especiales más espectaculares que los del juego anterior.

En los siguientes capítulos desarrollaré una serie muy potente de rutinas que le permitirán conseguir una animación sin temblores y unos efectos especiales nunca vistos hasta ahora en el Spectrum.

Antes de proseguir, recordemos cómo se genera la pantalla de TV. Aunque cuando vemos la TV vemos una imagen continua, de hecho sólo se trata de uno (en el caso de blanco y negro) o tres (en el caso de la mayoría de los televisores en color) rayos electrónicos que rastrean la pantalla a alta velocidad. Si no fuera por el fenómeno humano de la persistencia de la visión, que "conserva" en la retina del ojo la imagen generada por el haz el tiempo suficiente para que complete un "cuadro" de la TV (20 milésimas de segundo), entonces sólo veríamos un punto brillante iluminado que se movería a gran velocidad, y las pantallas de televisión, tal como las conocemos, no existirían.

En España, los televisores tienen una pantalla de 625 líneas. Esto quiere decir que las imágenes de la televisión son transmitidas como señales para 625 lí-

neas de exploración. Un televisor de tipo medio sólo visualiza unas 540 de estas líneas de exploración; el resto están fuera, en la parte superior e inferior de su pantalla, y parte del tiempo "libre" se consume en un período conocido como "retroceso al principio", en el que los haces de electrones pasan desde la parte inferior de la pantalla a la parte superior, para producir el siguiente cuadro.

Por cierto, algunas de las líneas de exploración sobrantes se utilizan para transmitir datos en los servicios de teletexto. Un decodificador en su TV de teletexto convierte entonces los datos binarios en una imagen de pantalla completa y la visualiza. Es el número de líneas de exploración disponibles para esta característica lo que limita la resolución y la elección de colores en los gráficos de teletexto: no hay bastante espacio para transmitir un teletexto en alta resolución con una velocidad en baudios aceptable.

Bueno, antes de seguir con el tema, volvamos a la discusión de la generación de imagen. El chip responsable de la gestión de la TV en el Spectrum se llama ULA *(Uncommitted Logic Array)* y lo que hace es utilizar dos líneas de exploración por cada fila de la pantalla del Spectrum. Por tanto, el área del texto ocupa 2 x 192 = 384 líneas de exploración; cerca del 70 por 100 de la altura de la pantalla y tarda alrededor del 61 por 100 de cada tiempo de cuadro o 12.288 ms en generarse.

Ahora se preguntará por qué estoy entrando en tanto detalle acerca de la visualización de la TV. Bueno, en el curso de la animación "normal", por medio de una celda cada vez, nada de todo esto sería necesario: los caracteres apenas se mueven, unas 5 veces por segundo, o una vez por cada 10 cuadros o así. Por consiguiente, no se observa ninguna interferencia significativa en la generación de la visualización en la TV.

Sin embargo, cada vez que movemos un carácter, de algún modo debemos "poner en blanco" su imagen vieja e "imprimir" la nueva en el archivo de pantalla. Si por casualidad la televisión está generando las líneas de exploración en las que estamos imprimiendo y borrando, entonces tomará la imagen de la memoria, cualquiera que sea el estado en que se encuentre, y la visualizará en la pantalla. El resultado es que para un cuadro corriente se visualizará una imagen incompleta.

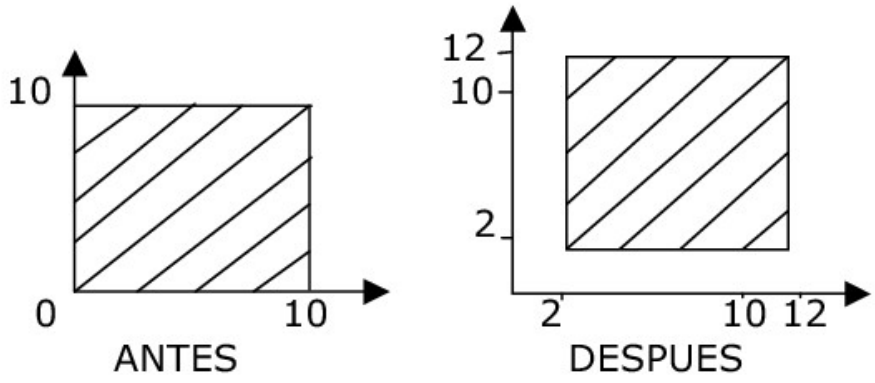
Como ya he dicho, esta interferencia no se observa para un movimiento de baja frecuencia. Sin embargo, la animación de *pixels* necesita hasta 8 veces la frecuencia de movimiento para mover un carácter a la misma velocidad que por celdillas, y éste tiene como resultado unas sombras y temblores inaceptables al utilizar técnicas estándar.

Una solución parcial para el problema es resolver el movimiento en términos de líneas de exploración de TV. Usted toma cada fila que pudiera ser ocupada, bien por la "nueva" o bien por la "vieja" imagen, por orden. Luego deje en blanco cualquier parte de la "vieja" imagen de dicha fila, e imprima cualquier parte de la nueva en la misma. Esto provoca una animación razonablemente suave en pantalla, pues ya no tendremos unas celdillas completamente en blanco donde no debería haberlas. Una técnica similar a ésta ha sido utilizada por Ashby Computers Graphics Limited con sus series de "Ultimate, Play the Game", de tanto éxito, para el Spectrum.

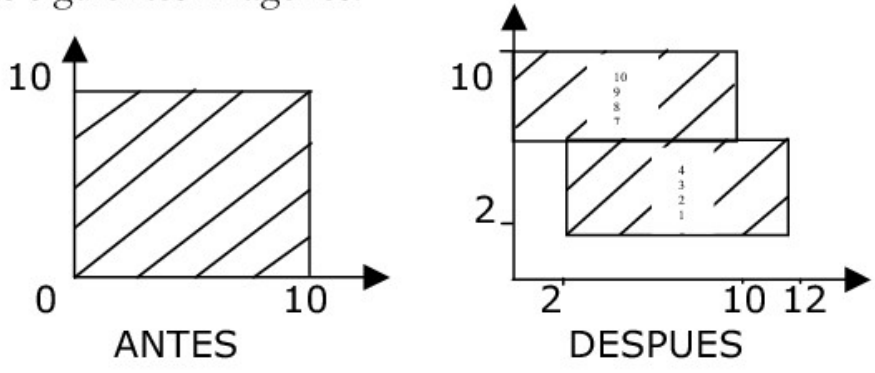
Sin embargo, hay un obstáculo para la técnica descrita arriba. No han sido eliminadas todas las interferencias y nos quedaremos con dos efectos principales. Primero, hay una forma reducida de temblor en la que se visualiza una fila en blanco mientras pasamos de borrar la imagen vieja a imprimir la nueva en esa fila. El resultado es que el carácter se "congela" constantemente en varios sitios según se mueve. Este efecto puede ser minimizado moviendo nuestro carácter desde su línea inferior hacia arriba en lugar de arriba abajo, como se hace tradicionalmente. De esta forma, nos aseguramos de que la rutina de impresión únicamente "colisiona" con la TV una vez durante cada cuadro, cuando los dos procesos se "cruzan", al trabajar en direcciones contrarias. Sin embargo, aún pienso que la interferencia que queda es lo suficientemente frecuente para resultar molesta.

El segundo efecto de la interferencia entre la generación de la imagen y la rutina de impresión es que el carácter se vuelve "estirado" o "encogido" verticalmente, o desunido horizontalmente, en función de la dirección del movimiento.

Imagínese que el carácter está siendo movido (desde la línea inferior hacia arriba) en dirección Noreste, es decir, hacia arriba y a la derecha. Para poner un ejemplo, supondremos que está en un cuadro de *pixels* de 10 x 10 moviendo dos *pixels* a la vez a lo largo de cada eje. En el caso general, cuando no hay interferencia, las imágenes que se observan deberían mostrarse así:



Sin embargo, si la exploración de la TV pasa por encima del área en que estamos imprimiendo mientras que lo estamos haciendo, obtendremos una figura desunida, arrugada en el interior de una zona de 8 *pixels* de longitud. Si la "colisión" ocurre después que se ha movido la cuarta fila de arriba, por ejemplo, obtendremos las siguientes imágenes:



Como puede ver, habremos perdido las filas 5 y 6 de nuestro carácter mutilado durante un cuadro.

La única forma de estar totalmente seguro de obtener una animación totalmente sin temblores es asegurarse de que la exploración de la TV NUNCA pasa por encima del área en la que se está imprimiendo en ese momento. Hay varias maneras diferentes para hacer esto, y todas implican que hay que tener en cuenta las interrupciones que el Spectrum recibe cada 50-avo de segundo. Esta es la misma frecuencia que la de su pantalla de TV y, por consiguiente, el haz de electrones está siempre en el mismo sitio durante el "retroceso al principio" cuando se envía una señal de interrupción.

Mientras podamos limitar nuestra acción de imprimir a los períodos en que la TV no está generando las 384 líneas de exploración del área de texto, podremos estar seguros de que no habrá temblores durante la animación, sin importar el lugar de la pantalla donde se está imprimiendo nuestra forma. Por tanto, los períodos "seguros" son mientras se están generando los bordes inferior y superior y durante los "retrocesos al principio".

Desgraciadamente, a menos que todas nuestras rutinas de juego sean de "tiempo constante", es decir, que siempre tarden lo mismo en ejecutarse, no tendremos forma de saber cuándo se acaba de generar el área del texto, y por tanto, no podemos utilizar el momento del borde "inferior" para imprimir. Esto nos deja con el tiempo entre una interrupción y el tiempo que tarda la exploración de la TV en alcanzar el área del texto, que es alrededor de 14.200 T-estados o 4,06 ms. Por muy increíble que parezca, es, de hecho, tiempo suficiente para imprimir 40 caracteres en la pantalla, utilizando un gestor de interrupciones de "proceso de impresión" especial, que explicaré en el capítulo siguiente.

9

Un procesador de impresión controlado por interrupciones con un generador de horizonte de pantalla completa

Ahora empezaré la explicación de la rutina del "procesador de impresión" controlado por interrupciones que mencioné al final del capítulo anterior. Esto, junto con una amplia serie de rutinas en las páginas siguientes, le permitirá generar la anhelada animación por *pixels* sin temblor de cualquier *sprite* (forma que consiste en un bloque de caracteres) de hasta 5 x 5 ó 7 x 4 celdas de área.

Además de su función principal como procesador de impresión, el gestor de interrupciones también será capaz de generar un horizonte de pantalla completa (incluido el borde). En el momento de escribir este libro, el único juego que tenía un horizonte de pantalla completa era el "Aquaplane" de Quicksilva, programado por John Hollis. A diferencia del horizonte estacionario de "Aquaplane", el horizonte generado por mis rutinas será movible entre uno y ocho *pixels* por cuadro dentro de una región comprendida entre la parte superior del área del texto y la parte más inferior de la pantalla.

Esto es sólo posible gracias a una técnica especial que "engaña" al ordenador para que produzca 3 ó 4 colores en los atributos que cubren el horizonte, de forma que tenemos aún algo que imprimir después que se hayan usado dos colores para el fondo.

En este momento debo advertir que estas rutinas han sido diseñadas para

funcionar en la parte superior de RAM en una máquina con 48K, basándose en la suposición que si es un programador serio del lenguaje máquina, que trabaja con un ensamblador, probablemente tendrá 48K de RAM para disponer de espacio para cualquier cosa que no sea un pequeño archivo de texto una vez que ha cargado el ensamblador.

Aun con el riesgo de repetirme, permítame explicarle que este programa máquina, colocado en los 16K más inferiores de RAM, funcionará alrededor de 20 por 100 más despacio cuando la TV esté generando el área del texto, ya que el ULA y el Z-80 estarán intentando acceder a la vez a los mismos 8 *chips* de la memoria, y el ULA tiene prioridad. Como consecuencia, la parte superior del "procesador de impresión" del gestor de interrupciones no debería necesitar modificación para funcionar en los 16K inferiores (sólo funciona cuando se está generando el borde superior), pero el generador de horizonte necesita una modificación profunda para compensar la pérdida de velocidad y los cambios generales en los tiempos.

En este caso sería probablemente mejor que se conformara con un horizonte estacionario en un límite entre dos de las 24 líneas. De esta forma, no necesita un trabajo especial sobre los atributos y se genera el horizonte del borde sencillamente con un bucle de retardo adecuado, y luego cambiando el color de encima del horizonte por el de debajo utilizando una instrucción OUT.

El sistema del procesador de impresión funciona de la siguiente manera. Cada vez que alguna rutina de animación necesita imprimir algo en la pantalla, en vez de hacerlo directamente, elabora toda la preparación posible y luego deposita los datos resultantes para cada carácter a imprimir en un área de memoria que suele denominarse "memoria tampón o *buffer* de impresión", luego, en cada interrupción, el procesador de impresión "vacía" el *buffer*, una entrada cada vez, y pone el carácter correspondiente y sus atributos en el sitio correcto de los archivos de pantalla y de atributos respectivamente.

Etiquetaremos el principio del *buffer* como BUFFER. Cada entrada suya tiene una longitud de 6 octetos, y el formato de los datos es el siguiente:

1) OCTETO DE ATRIBUTO
2) DIRECCION DE ATRIBUTO (MENOR PESO)
3) DIRECCION DE ATRIBUTO (MAYOR PESO)
4) DIRECCION DEL ARCHIVO DE PANTALLA (MAYOR PESO)
5) DIRECCION DE DATOS DE CARACTER (MENOR PESO)
6) DIRECCION DE DATOS DE CARACTER (MAYOR PESO

Note que no necesitamos almacenar el octeto menos significativo (de menor peso) de la dirección del archivo de pantalla, puesto que es idéntico al de la del archivo de atributos (octeto 2).

Nunca he visto un caso en que haya sido útil o necesario utilizar FLASH 1 cuando se está animando *sprites* por *pixels*, y, por tanto, he decidido sacrificar su bit de atributo, dejándonos espacio para una bandera. El atributo se almacena desplazado un bit a la izquierda en el *buffer*, dejando el bit 0 como bandera.

A menudo resulta útil poder fundir dos *sprites* uno encima del otro, en el caso de que se superpongan, utilizando una operación OR, en vez de emplear la eliminación de la primera imagen por la segunda. He llamado a estos dos tipos de operación de impresión "OR-impresión" y "OVER-impresión" (o impresión y sobreimpresión) respectivamente. Cuando el bit 0 del octeto de atributos está a uno, indica al procesador de impresión que funda este carácter en particular con los contenidos actuales de la celda, con OR-impresión.

La rutina puede modificarse fácilmente si desea utilizarla para sus propios propósitos para imprimir empleando la operación XOR (O exclusivo) simplemente cambiando todas las instrucciones OR precisas por XOR. Poner a uno la bandera significaría que se necesita "XOR-impresión".

Para generar un horizonte estable, es absolutamente necesario que cualquier rutina ejecutada entre la interrupción y la generación del horizonte sea de tiempo constante. Por tanto, he calibrado cuidadosamente la rutina del procesador de impresión de forma que, haya lo que haya en el *buffer*, tarde el mismo tiempo en ejecutarse.

He utilizado varios trucos para hacer que la rutina sea lo más rápida posible. Se vio que había tiempo para imprimir exactamente 40 caracteres, y, por tanto, nuestro *buffer* necesita ser de una longitud de 40 x 6 = 240 bits. Si nos aseguramos de que el octeto de menor peso de BUFFER es 10 Hex, podemos utilizar instrucciones de incremento de un solo registro como:

```
                INC   L
en oposición a  INC   HL
```

para pasar por el *buffer*. Esto ahorra 2 T-estados cada vez que lo utilizamos y tiene además la ventaja de que podemos saber después del proceso de una entrada si se ha alcanzado el final del *buffer* utilizando simplemente:

```
INC   L
```

y luego comprobando el estado de la bandera cero.

Si no está utilizando las primeras líneas superiores del área del texto para la animación, o si no le importa el temblor en esa área, puede pensar en algún momento en que merezca la pena aumentar el número de caracteres con que puede trabajar el procesador de impresión. Esto no plantea problema alguno hasta un límite de 42 caracteres (la respuesta a todo de los autostopistas), ya que el *buffer* estaría todavía contenido dentro de una "página" (256 octetos con la misma parte de mayor peso de la dirección) de la memoria. Sin embargo, más allá de ese límite se necesitarán algunas alteraciones para que la rutina pase correctamente por el límite la página.

Puesto que cada entrada del *buffer* tiene una longitud de 6 octetos, hay 6 instrucciones INC L en cada bucle de la rutina, como verá cuando llegue al listado. La primera es después que se ha recogido el octeto de atributo, la segunda después de que se ha recogido la parte de menor peso de su dirección y así

sucesivamente. Un cálculo matemático rápido nos muestra que un *buffer* que termine en #FCFF y de 43 entradas de longitud debería empezar en #FBFE. Por tanto, para *buffers* de más de 42 entradas hay que cambiar la segunda instrucción INC L a INC HL.

De la misma manera, para un *buffer* de 86 entradas, la dirección de comienzo sería #FAFC, y, por tanto, para *buffers* de más de 85 entradas cambia la cuarta instrucción INC L a INC HL. La longitud máxima del *buffer* por este método es más que suficiente: 128 entradas (768 octetos), en cuyo momento empezaría en un principio de página #FA00. Como guía para el tiempo de proceso extra para un *buffer* más largo, cada entrada tarda aproximadamente igual que la impresión de 1.6 filas ó 3.2 líneas de exploración.

Ahora bien, obviamente habrá ocasiones en que no utilicemos todas las 40 entradas del *buffer*. Sin embargo, debemos asegurar que el procesador de impresión tarda lo mismo en ejecutar una entrada nula y probablemente la forma más fácil para hacer esto es hacer que la rutina THINK imprima un carácter, sin que afecte realmente a la pantalla.

Para indicar una entrada nula en el *buffer*, pondremos el octeto de atributo a cero. Al principio de cada bucle se ejecuta el fragmento siguiente, con HL apuntando el principio de una entrada del *buffer*:

```
NXTCH   LD    A,(HL)
        AND   A
        JR    Z,FAKE
```

En FAKE actualizaremos el puntero de *buffer* de HL y estableceremos los registros necesarios para OR-imprimir (sin efecto neto) en el ángulo inferior derecho de la pantalla. Habrá entonces una pausa corta para poder ecualizar perfectamente el tiempo con el camino de la rutina de impresión normal, seguido de un salto a la sección principal del proceso de la OR-impresión. Esto no afectará a los atributos, y el fragmento de FAKE será el siguiente:

```
FAKE    LD    A,5         ;AJUSTAR PUNTERO DE BUFFER
        ADD   A,L
        LD    L,A
        LD    DE,#50FF    ;DIRECCION DE A.P. DE (23,31)
        LD    A,(DE)      ;ECUALIZADOR DE TIEMPO
        LD    BC,#3D00    ;DIRECCION DE DATOS DEL "ESPACIO"
        EX    DE,HL       ;EN ROM
        NOP   ;ESPERAR    14 T-ESTADOS
        JP    $+3         ;NOTA; $=CONTADOR DE PROGRAMA
        JR    OR          ;SALTA A LA RUTINA PRINCIPAL
```

La parte más importante de las instrucciones de la etiqueta OR es el siguiente fragmento:

```
LD    A,(BC)      ;TOMAR DATO
OR    (HL)        ;'OR' CON FILA DE PANTALLA
LD    (HL),A      ;INSERTAR EN ARCHIVO DE PANTALLA
INC   BC          ;SIGUIENTE BYTE DE DATOS
INC   H           ;SIGUIENTE FILA DE ARCHIVO DE PANTALLA
```

que se repite 6 veces, seguido por

```
LD    A,(BC)     ;IMPRIMIR ULTIMA FILA DE CARACTERES
OR    (HL)
LD    (HL),A
EX    DE,HL
INC   L          ;COMPROBAR FINAL DEL BUFFER
JP    NZ,NXTCH
```

A primera vista, el listado para esto puede parecer torpe, pero debemos recordar que el tiempo es de suma importancia, y que un bucle convencional repetido 7 veces tardaría mucho más en ejecutarse. Por la misma razón, se ha empleado una instrucción JP absoluta (10 T-estados) en vez de un salto relativo (12 T-estados, donde el tiempo extra se utiliza para añadir el desplazamiento al contador de programa).

Mientras tiene todo esto todavía fresco en su memoria y antes de pasar al desarrollo de la parte de generación del horizonte del gestor de interrupciones, haré un listado de la primera parte de la rutina para su observación. Se necesitan unas palabras de explicación para las primeras líneas de la rutina. La primera prioridad en todo gestor de interrupciones debe ser conservar todos los registros que el gestor utilice. Habiendo hecho esto, debemos dar salida al color del borde para el "cielo" de encima del horizonte. Esto también proporciona una oportunidad para enviar un "click" al altavoz, sumando 10 Hex al argumento de la instrucción XOR de la etiqueta TOPBRD. Siempre almacenaremos el último valor al que hemos dado SALIDA por el puerto #FE en la variable BORD, conservando así el estado del altavoz (bit 4).

Las variables CHSTRE y BUFFPT almacenan el número de entradas "reales" y la dirección de la siguiente entrada libre del *buffer*, respectivamente. Estas variables serán de mucha utilidad más adelante. Ahora pasemos a la primera parte del gestor de interrupciones; por favor, lea hasta el final de este capítulo antes de intentar hacer uso de él, ya que, haciéndolo funcionar por sí solo, sería la causa de un tremendo bloqueo del ordenador. También tome nota de que el signo $ de las instrucciones de salto significa "contador de programa", así que:

JR $ + 2 y
JP $ + 3

sencillamente significa "avance a la instrucción siguiente" y se emplean como retardos de tiempo.

```
FF10                10 BUFFER   EQU     #FF10
90E0  10FF          20 BUFFPT   DEFW    BUFFER
90E2  00            30 CHSTRE   DEFB    0
90E3  00            40 BORD     DEFB    0
                    50 ;
```

```
                60 ;
                70 ;GESTOR DE INTERRUPCIONES *****
                80 ;SALVAR REGISTROS
                90 ;
90E4  F5       100 INTERP    PUSH      AF
90E5  C5       110           PUSH      BC
90E6  D5       120           PUSH      DE
90E7  E5       130           PUSH      HL
               140 ;
               150 ;PONER BORDE SUPERIOR
               160 ;
90E8  21E390   170           LD        HL,BORD
90EB  7E       180           LD        A,(HL)
90EC  E610     190           AND       16
90EE  EE05     200 TOPBRD    XOR       5
90F0  D3FE     210           OUT       (#FE),A
90F2  77       220           LD        (HL),A
               230 ;
               240 ;COMENZAR A TRABAJAR MEDIANTE ENTRADAS DEL BUFFER
               250 ;
90F3  2110FF   260           LD        HL,BUFFER
               270 ;
               280 ;UN ATRIBUTO CERO="NINGUNA ENTRADA", POR TANTO IMPRIMIR UN
               290 ;CARACTER FALSO
               300 ;
90F6  7E       310 NXTCH     LD        A,(HL)
90F7  A7       320           AND       A
90F8  286C     330           JR        Z,FAKE
90FA  2C       340           INC       L
               350 ;
               360 ;TOMAR DIRECCIONES DE ATRIBUTOS
               370 ;
90FB  5E       380           LD        E,(HL)
90FC  2C       390           INC       L
90FD  56       400           LD        D,(HL)
90FE  2C       410           INC       L
90FF  1F       420           RRA
               430 ;
               440 ;ALMACENAR NUEVO ATRIBUTO
               450 ;
9100  12       460           LD        (DE),A
               470 ;
               480 ;FORMAR DIRECCIONES A.P.
               490 ;
9101  56       500           LD        D,(HL)
9102  2C       510           INC       L
               520 ;TOMAR DIRECCIONES DE DATOS DE CARACTER
               530 ;
9103  4E       540           LD        C,(HL)
9104  2C       550           INC       L
9105  46       560           LD        B,(HL)
               570 ;
               580 ;DECIDIR SI MEZCLAR VIEJOS CARACTERES
               590 ;
9106  302F     600           JR        NC,NTOR
9108  EB       610           EX        DE,HL
               620 ;
               630 ;IMPRIMIR NUEVO CARACTER UTILIZANDO "OR"
               640 ;
9109  0A       650 OR        LD        A,(BC)
910A  B6       660           OR        (HL)
910B  77       670           LD        (HL),A
910C  03       680           INC       BC
910D  24       690           INC       H
910E  0A       700           LD        A,(BC)
910F  B6       710           OR        (HL)
9110  77       720           LD        (HL),A
```

```
9110  03          730           INC       BC
9112  24          740           INC       H
9113  0A          750           LD        A,(BC)
9114  B6          760           OR        (HL)
9115  77          770           LD        (HL),A
9116  03          780           INC       BC
9117  24          790           INC       H
9118  0A          800           LD        A,(BC)
9119  B6          810           OR        (HL)
911A  77          820           LD        (HL),A
911B  03          830           INC       BC
911C  24          840           INC       H
911D  0A          850           LD        A,(BC)
911E  B6          860           OR        (HL)
911F  77          870           LD        (HL),A
9120  03          880           INC       BC
9121  24          890           INC       H
9122  0A          900           LD        A,(BC)
9123  B6          910           OR        (HL)
9124  77          920           LD        (HL),A
9125  03          930           INC       BC
9126  24          940           INC       H
9127  0A          950           LD        A,(BC)
9128  B6          960           OR        (HL)
9129  77          970           LD        (HL),A
912A  03          980           INC       BC
912B  24          990           INC       H
912C  0A         1000           LD        A,(BC)
912D  B6         1010           OR        (HL)
912E  77         1020           LD        (HL),A
912F  EB         1030           EX        DE,HL
                 1040;
                 1050;BUCLE HACIA ATRAS HASTA FINAL DEL BUFFER
                 1060;
9130  2C         1070           INC       L
9131  C2F690     1080           JP        NZ,NXTCH
                 1090;
                 1100;SALTO HACIA ADELANTE
                 1110;
9134  C33491     1120           JP        ROWS
                 1130;
                 1140;IMPRIMIR ENCIMA DEL CARACTER VIEJO
                 1150;
9137  0A         1160NTOR       LD        A,(BC)
9138  12         1170           LD        (DE),A
9139  12         1180           LD        (DE),A
913A  03         1190           INC       BC
913B  14         1200           INC       D
913C  0A         1210           LD        A,(BC)
913D  12         1220           LD        (DE),A
913E  12         1230           LD        (DE),A
913F  03         1240           INC       BC
9140  14         1250           INC       D
9141  0A         1260           LD        A,(BC)
9142  12         1270           LD        (DE),A
9143  12         1280           LD        (DE),A
9144  03         1290           INC       BC
9145  14         1300           INC       D
9146  0A         1310           LD        A,(BC)
9147  12         1320           LD        (DE),A
9148  12         1330           LD        (DE),A
9149  03         1340           INC       BC
914A  14         1350           INC       D
914B  0A         1360           LD        A,(BC)
914C  12         1370           LD        (DE),A
914D  12         1380           LD        (DE),A
914E  03         1390           INC       BC
```

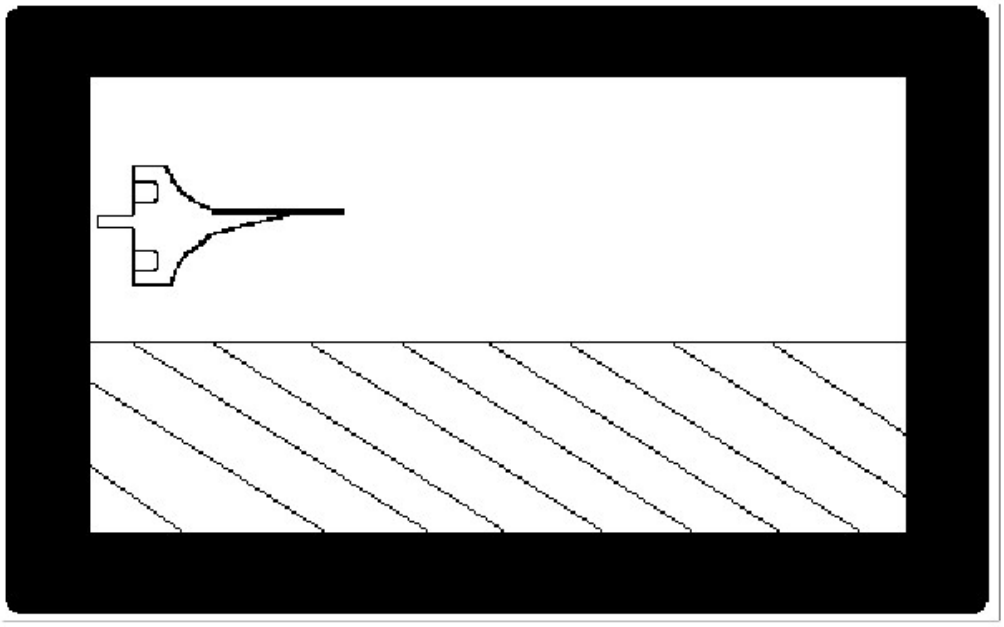
```
914F  14        1400          INC       D
9150  0A        1410          LD        A,(BC)
9151  12        1420          LD        (DE),A
9152  12        1430          LD        (DE),A
9153  03        1440          INC       BC
9154  14        1450          INC       D
9155  0A        1460          LD        A,(BC)
9156  12        1470          LD        (DE),A
9157  12        1480          LD        (DE),A
9158  03        1490          INC       BC
9159  14        1500          INC       D
915A  0A        1510          LD        A,(BC)
915B  12        1520          LD        (DE),A
915C  C35F91    1530          JP        $+3
                1540;
                1550;BUCLE HACIA ATRAS HASTA EL FINAL DEL BUFFER
915F  2C        1560          INC       L
9160  C2F690    1570          JP        NZ,NXTCH
                1580;
                1590;SALTAR ENTONCES PARA GENERAR HORIZONTE
                1600;
9163  C33491    1610          JP        ROWS
                1620;PARA EQUILIBRAR EL TIEMPO IMPRIMIR
                1630;UN ESPACIO CON "OR" EN LA ESQUINA
                1640;INFERIOR DERECHA
                1650;
9166  3E05      1660FAKE      LD        A,5
9168  85        1670          ADD       A,L
9169  6F        1680          LD        L,A
916A  11FF50    1690          LD        DE,#50FF
916D  1A        1700          LD        A,(DE)
916E  01003D    1710          LD        BC,#3D00
9171  EB        1720           EX        DE,HL
9172  00        1730          NOP
9173  C37691    1740          JP        $+3
9176  1891      1750          JR        OR
```

Por cierto, la etiqueta ROWS estará en la primera línea de la siguiente parte de la rutina.

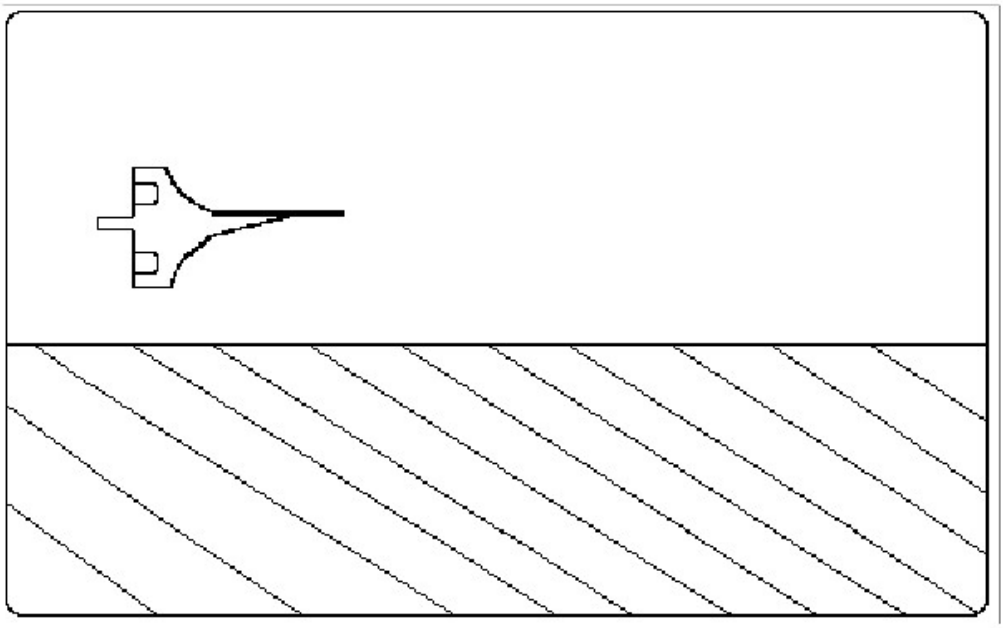
Ahora empezaré una discusión sobre los principios implicados en la generación de un horizonte móvil de pantalla completa. Para simplificar, me referiré al área de la pantalla de encima del horizonte como "cielo" y la de debajo como "mar". Las rutinas producirán un cielo "cian" (azul verdoso) y un mar azul al principio, pero estos colores se cambian muy fácilmente e incluiré más adelante una rutina para ello.

Puede que se esté preguntando: "¿Para qué sirve un horizonte de pantalla completa?" o "¿merece realmente la pena todo el esfuerzo y el cálculo de tiempos necesario?"

La respuesta es que SI merece la pena el esfuerzo, porque, aunque extender el horizonte no permite imprimir en un área mayor de la pantalla, sí aumenta el área de juego efectivo y es mucho más agradable desde el punto de vista estético. Supongamos, por ejemplo, que su juego implicaba el control de un avión que vuela por encima del mar, y que la posición actual del avión estuviera tan a la izquierda en el área del texto como fuera posible, sin salirse fuera de la pantalla. Con un horizonte convencional de texto solamente, veríamos algo como esto:



Como puede ver, nuestro avión parece estar sumamente "fuera de lugar", puesto que está arrinconado lo más cerca posible del borde de la izquierda. Compárelo con el aspecto "espacioso" de un horizonte de pantalla completa, donde el avión parece más real, incluso aunque haya sido impreso en el mismo sitio de la pantalla:



El principio en el que se basan todos los "trucos" de programación que afectan al borde es muy sencillo. El ULA lee continuamente el puerto 254 y envía el color correspondiente al televisor, elaborando así la imagen línea a línea. De este modo, para obtener una barrera nítida entre dos colores de borde, simplemente esperaremos al momento exacto tras cada señal de interrupción antes de enviar el color del "mar" al puerto 254.

Ahora bien, el Z-80 del Spectrum funciona a la velocidad de reloj de

3,5 MHz, es decir, hay 3.500.000 T-estados por segundo. Los cuadros de la televisión se generan a 50 Hz, y con 625 líneas de exploración. Por tanto tenemos:

$$\text{Tiempo necesario para una línea de exploración} = \frac{3.500.000}{625 \times 50} = 112 \text{ T-estados.}$$

Sin olvidar que el Spectrum emplea 2 líneas de exploración para cada fila de la pantalla, tenemos que cada fila necesita 2 x 112 = 224 T-estados para generarse, y esto es cuanto tendremos que esperar para cada fila de la pantalla de encima del horizonte, antes de cambiar el color del borde. Un bucle de retardo adecuado, en el caso que el número de filas se encuentre en el acumulador, sería el siguiente:

```
SCAN1   LD    B,15       ;7 T-ESTADOS
LN      DJNZ  LN         ;14*13+8=190
        AND   #FF        ;7 T-ESTADOS
        INC   HL         ;6 T-ESTADOS
        DEC   A          ;4 T-ESTADOS
        JP    NZ,SCAN1   ;10 T-ESTADOS, BUCLE HACIA ATRAS
                         ;PARA          LA SIGUIENTE FILA
```

Encontrará el fragmento de arriba en la segunda parte del gestor de la interrupción.

Bueno, ya nos hemos ocupado del control del borde. Ahora, ¿qué hay de los atributos? Si el horizonte es una línea divisoria entre dos líneas de la pantalla no tenemos problema. Simplemente utilizamos papel cian para la línea de encima del horizonte y papel azul para la de debajo. Sin embargo, si el número de las filas de texto de encima del horizonte no puede dividirse por 8, necesitaremos producir tanto papel cian como azul en una línea de atributos. No es suficiente el empleo de sólo INK (tinta) cian y PAPER (papel) azul, ya que esto nos dejaría sin colores para imprimir nuestros *sprites* encima del horizonte.

Para producir estos atributos de "dos papeles", necesitamos llenar la línea que contiene el horizonte con papel cian (y cualquier tinta de color que estemos usando en ese momento) y a continuación esperar que la exploración (barrido) de la TV alcance el nivel del horizonte, para luego volver a llenar apresuradamente la línea con atributos de papel azul. El ULA verifica los atributos cada vez que genera una fila en el área de texto; por tanto, el resultado debería ser papel cian por encima del horizonte y papel azul por debajo, junto con nuestra selección de tinta y brillo para cada región.

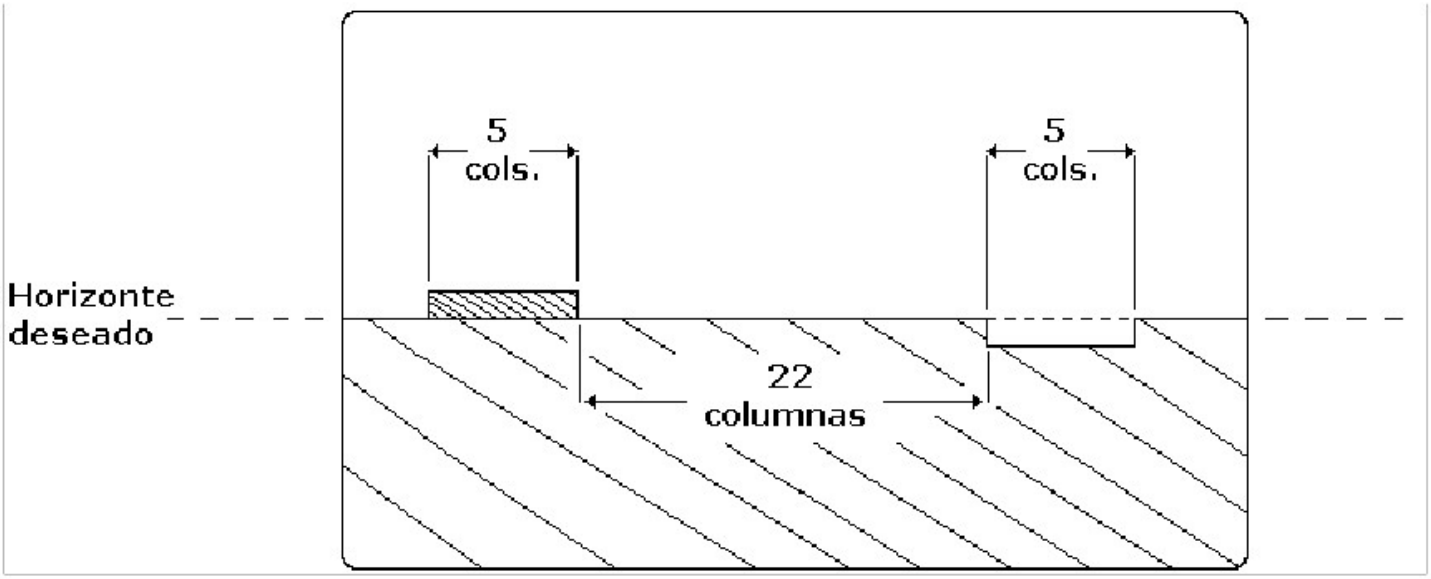
Desgraciadamente, debido a la gran velocidad con la que el haz de electrones barre desde la izquierda hasta la derecha de la pantalla, no tenemos tiempo suficiente para reemplazar toda la fila de atributos antes de que el ULA vuelva a necesitarlos. Un cálculo rápido muestra que el cambio de 32 atributos en 224 T-estados necesitaría un promedio por atributo en 7 T-estados, que es tiempo suficiente para hacer un LD (HL), A básico, sin incrementar ningún puntero.

Hay dos factores más a tener en consideración. Del lado positivo, tenemos,

de hecho, un poco más de tiempo que los 224 T-estados. Supongamos, por ejemplo, que hemos conseguido rellenar la mitad de una fila de TV, empezando por el límite izquierdo del área de texto; el tiempo que hemos tenido para hacerlo sería el que se hubiera tardado para 1,5 filas (336 T-estados), puesto que hubiéramos empezado en el momento en que el haz hubiera dejado el primer atributo, y terminado cuando nos alcanzara a mitad del camino de recorrido de la pantalla.

Del lado negativo, debemos recordar que el cambio de atributos requiere el acceso a los 16K inferiores de la RAM, y las interrupciones que resultan del ULA significarán una ligera disminución general de velocidad.

Después de experimentar, encontré que la mejor solución que podríamos esperar es la de un "horizonte" de nivel continuo de 22 atributos. En la mayoría de los juegos, la acción tiende siempre hacia el centro del área de juego: por tanto, he colocado estos 22 octetos en el centro del área del texto, dejando pasos de una anchura de 5 columnas a cada lado. Tal cual, el horizonte deformado aparecería del siguiente modo:



Ahora bien, esta apariencia es, a no ser que desee una "colina" y un "valle" rectangulares en su pantalla, totalmente intolerable. La mejor solución al problema es llenar las 5 columnas de la izquierda con una fila de tinta inmediatamente encima del horizonte; y las 5 columnas de la derecha con una fila de tinta justo debajo del horizonte. Utilizamos tinta cian a la izquierda y tinta azul a la derecha para crear un horizonte continuo y raso.

Al principio de cada interrupción, todos los atributos en cuestión serán papel cian y tinta blanca. La secuencia con la que cambiaremos los atributos es la siguiente (después de un retardo exacto);

1) Llenar los 5 atributos de más a la derecha con papel cian, tinta azul.
2) Llenar los 5 atributos de más a la izquierda con papel azul, tinta cian.

3) Llenar los 22 atributos del medio con papel azul, tinta blanca.
4) Llenar los 5 atributos de más a la izquierda con papel azul, tinta blanca.
5) Llenar los 5 atributos de más a la derecha con papel azul, tinta blanca.
6) Ahora tenemos que esperar hasta que la TV haya terminado por completo la generación de esta línea de la pantalla. Parte de este tiempo lo podemos emplear en preparar el buffer de impresión para la siguiente interrupción.
7) Llenar los 32 atributos con papel cian, tinta blanca, dispuestos para la siguiente interrupción.

En cierto punto crítico durante la Etapa (3), el haz alcanzará el lado de la derecha de la pantalla habiendo generado la fila cian inmediatamente por encima del horizonte. Cuando el haz está en "retroceso" hacia el lado izquierdo de la pantalla, tenemos que dejar de llenar atributos y dar salida al nuevo valor del borde. Almacenamos el nuevo valor en (BOTBRD + 1) y le sumamos 16 si hay que enviar un click al altavoz. Esto, combinado con el valor en (TOPBRD +1), nos permite una selección entre ningún sonido, o uno de 50 Hz o 100 Hz. En el último caso, verá que; cuando movamos el horizonte hacia arriba y hacia abajo en el siguiente capítulo, la forma de la ola del sonido cambia debido al tiempo que transcurre entre el "click" superior y el "click" inferior.

Volviendo a nuestro procedimiento para cambiar los atributos, el único problema adjunto al uso de la técnica es que hasta 2 filas de cualquier *sprite* impreso en el horizonte en las 5 columnas de la derecha o de la izquierda serán cian y azules respectivamente. Esto apenas se nota y es un precio irrisorio por el efecto total de una visualización en toda la pantalla.

Esta técnica funcionará, como mínimo, 2 filas de texto por encima del horizonte, y el número de estas filas está almacenado en (ROWS + 1). Las filas del texto cero no presentan problema, ya que toda la pantalla será mar y sólo tenemos que saltar derecho hacia la etiqueta NOWAIT cuando se cambia el color del borde. Sin embargo, en el caso muy poco probable de que necesite sólo una fila de texto por encima del horizonte, entonces tendrá que volver a la vieja técnica de llenarlo con tinta y emplear tinta cian y papel azul. Esto es así porque no hay suficiente tiempo para manipular los atributos de la forma requerida por la nueva técnica. En este caso, el gestor de interrupciones salta a la etiqueta WT1LN y espera que la TV alcance la fila uno antes de cambiar el borde.

Repartidas por el listado encontrará etiquetas desde HCOL1 hasta HCOL4; estas etiquetas se utilizarán en los siguientes capítulos para cambiar los colores del cielo y del mar. Se empleará HRZN3 para las rutinas de movimiento del horizonte. Contiene la dirección del 28-avo atributo del horizonte y se le pondrá para apuntar a #001B (en la ROM) cuando no se necesite trabajo alguno de atributos. Este es el caso en la rutina tal y como está, ya que he puesto (ROWS+1) a 96, o a mitad de camino de la pantalla.

Después que se ha generado el horizonte, el gestor de interrupciones tiene dos tareas más por hacer. Primero, tiene que limpiar el buffer de impresión borrando todas las entradas que se acaban de imprimir, e insertando octetos de

atributo cero para indicar 40 entradas nulas. Inicializa BUFFPT con la primera entrada libre, que ahora está en BUFFER y pone el número de entradas, CHSTRE a cero. Finalmente recupera todos los registros almacenados al principio de la interrupción y termina.

Aquí, pues, está la segunda parte del gestor de interrupciones, seguido por una rutina de inicialización:

```
95FF  3E60        10 ROWS      LD        A,96
                  20 ;
                  30 ;A CONTIENE NO. DE FILAS SOBRE HORIZONTE
                  40 ;SI A=0 NO ESPERAR A CAMBIAR BORDE
                  50 ;
9601  D601        60           SUB       1
9603  DAC796      70           JP        C,NOWAIT
                  80 ;
                  90 ;SI A=1 ESPERAR UNA LINEA DE EXPLORACION
                 100 ;
9606  CABD96     110           JP        Z,WT1LN
9609  3D         120           DEC       A
                 130 ;
                 140 ;SI A=2 SALTAR ESTE RETARDO
960A  CA1896     150           JP        Z,GO4IT
                 160 ;
                 170 ;ESTE BUCLE TARDA 224 T-ESTADOS O UNA
                 180 ;LINEA DE EXPLORACION POR PASO
960D  060F       190 SCAN1     LD        B,15
960F  10FE       200 LN        DJNZ      LN
9611  E6FF       210           AND       #FF
9613  23         220           INC       HL
9614  3D         230           DEC       A
9615  C20D96     240           JP        NZ,SCAN1
                 250 ;
                 260 ;EQUILIBRADOR DE TIEMPO
                 270 ;
9618  060A       280 GO4IT     LD        B,10
961A  10FE       290 SELFS     DJNZ      SELFS
961C  E6FF       300           AND       #FF
                 310 ;
961E  211B00     320 HRZN3     LD        HL,#1B
                 330 ;HL=DIRECCION DEL 26-AVO ATRIBUTO DE LA LINEA
                 340 ;
9621  7D         350           LD        A,L
9622  E6E0       360           AND       #E0
9624  47         370           LD        B,A
                 380 ;
                 390 ;BUSCAR COLOR DEL BORDE INFERIOR
                 400 ;
9625  112596     410           LD        DE,BORD
9628  1A         420           LD        A,(DE)
9629  E610       430           AND       16
962B  EE01       440 BOTBRD    XOR       1
962D  12         450           LD        (DE),A
962E  1E29       460 HCOL1     LD        E,41
9630  0E0D       470 HCOL2     LD        C,13
                 480 ;
                 490 ;LLENAR 5 ATRI. A DER. CON PAPEL CIAN, TINTA AZUL
9632  73         500           LD        (HL),E
9633  2C         510           INC       L
9634  73         520           LD        (HL),E
9635  2C         530           INC       L
9636  73         540           LD        (HL),E
9637  2C         550           INC       L
9638  73         560           LD        (HL),E
```

```
9639  2C          570          INC      L
963A  73          580          LD       (HL),E
963B  68          590          LD       L,B
963C  1E0F        600 HCOL3    LD       E,15
                  610 ;
                  620 ;LLENAR 5 ATRI. A IZQ. CON PAPEL AZUL, TINTA CIAN
                  630 ;
963E  71          640          LD       (HL),C
963F  2C          650          INC      L
9640  71          660          LD       (HL),C
9641  2C          670          INC      L
9642  71          680          LD       (HL),C
9643  2C          690          INC      L
9644  71          700          LD       (HL),C
9645  2C          710          INC      L
9646  71          720          LD       (HL),C
9647  2C          730          INC      L
                  740 ;
                  750 ;LLENAR LOS 22 ATRI. DEL MEDIO CON PAPEL AZUL, TINTA BLANCA
                  760 ;
9648  73          770          LD       (HL),E
9649  2C          780          INC      L
964A  73          790          LD       (HL),E
964B  2C          800          INC      L
964C  73          810          LD       (HL),E
964D  2C          820          INC      L
964E  73          830          LD       (HL),E
964F  2C          840          INC      L
9650  73          850          LD       (HL),E
9651  2C          860          INC      L
9652  73          870          LD       (HL),E
9653  2C          880          INC      L
9654  73          890          LD       (HL),E
9655  2C          900          INC      L
9656  73          910          LD       (HL),E
9657  2C          920          INC      L
9658  73          930          LD       (HL),E
9659  2C          940          INC      L
965A  73          950          LD       (HL),E
965B  2C          960          INC      L
965C  73          970          LD       (HL),E
965D  2C          980          INC      L
965E  73          990          LD       (HL),E
965F  2C         1000          INC      L
9660  73         1010          LD       (HL),E
9661  2C         1020          INC      L
                 1030 ;MIENTRAS TANTO LA BUSQUEDA DE TV HA ALCANZADO EL BORDE
                 1040 ;DER. DE ESTE MODO, AHORA CAMBIA EL COLOR DEL BORDE
                 1050 ;
9662  D3FE       1060          OUT      (#FE),A
9664  73         1070          LD       (HL),E
9665  2C         1080          INC      L
9666  73         1090          LD       (HL),E
9667  2C         1100          INC      L
9668  73         1110          LD       (HL),E
9669  2C         1120          INC      L
966A  73         1130          LD       (HL),E
966B  2C         1140          INC      L
966C  73         1150          LD       (HL),E
966D  2C         1160          INC      L
966E  73         1170          LD       (HL),E
966F  2C         1180          INC      L
9670  73         1190          LD       (HL),E
9671  2C         1200          INC      L
9672  73         1210          LD       (HL),E
9673  2C         1220          INC      L
9674  73         1230          LD       (HL),E
```

```
9675  7D          1240          LD        A,L
9676  68          1250          LD        L,B
                  1260  ;
                  1270  ;AHORA LLENA LOS 5 ATRI. DE IZQ. CON PAPEL AZUL, TINTA BLANCA

                  1280  ;
9677  73          1290          LD        (HL),E
9678  2C          1300          INC       L
9679  73          1310          LD        (HL),E
967A  2C          1320          INC       L
967B  73          1330          LD        (HL),E
967C  2C          1340          INC       L
967D  73          1350          LD        (HL),E
967E  2C          1360          INC       L
967F  73          1370          LD        (HL),E
9680  6F          1380          LD        L,A
9681  2C          1390          INC       L
                  1400  ;FINALMENTE LLENAR LOS 5 ATRI. DE DER. CON PAPEL AZUL
                  1410  ;TINTA BLANCA
                  1420  ;
9682  73          1430          LD        (HL),E
9683  2C          1440          INC       L
9684  73          1450          LD        (HL),E
9685  2C          1460          INC       L
9686  73          1470          LD        (HL),E
9687  2C          1480          INC       L
9688  73          1490          LD        (HL),E
9689  2C          1500          INC       L
968A  73          1510          LD        (HL),E
968B  68          1520          LD        L,B
                  1530  ;
                  1540  ;ALMACENAR INICIO LINEA DE ATRI.
968C  E5          1550  INIT3   PUSH      HL
                  1560  ;ASEGURAR QUE EL BUFFER SE LLENA CON CARACTERES FALSOS
                  1570  ;
                  1580  ;
968D  218D96      1590          LD        HL,CHSTRE
9690  7E          1600          LD        A,(HL)
9691  A7          1610          AND       A
9692  280F        1620          JR        Z,END
9694  110600      1630  INIT2   LD        DE,6
                  1640  ;
                  1650  ;NOTAR QUE D=0
9697  72          1660          LD        (HL),D
9698  219896      1670          LD        HL,BUFFER
969B  229B96      1680          LD        (BUFFPT),HL
969E  47          1690          LD        B,A
969F  72          1700  NXTFL    LD        (HL),D
96A0  19          1710          ADD       HL,DE
96A1  10FC        1720          DJNZ      NXTFL
                  1730  ;
                  1740  ;RECUPERAR DIRECCIONES DE ATRI.
                  1750  ;
96A3  E1          1760  END     POP       HL
                  1770  ;
                  1780  ;H=0 SIGNIFICA NINGUN ATRI. A LLENAR
                  1790  ;
96A4  7C          1800          LD        A,H
96A5  A7          1810          AND       A
96A6  CAB696      1820          JP        Z,NOPLG
                  1830  ;
                  1840  ;ESPERAR HASTA QUE LA TV HA TERMINADO CON
                  1850  ;ESTA LINEA DE ATRIBUTOS
                  1860  ;NOTA : SI FLUCTUA ENTONCES
                  1870  ;INCREMENTAR ESTE RETARDO
                  1880  ;
96A9  064D        1890          LD        B,#4D
```

```
96AB  10FE       1900 SELF4   DJNZ     SELF4
                 1910 ;
                 1920 ;LLENAR LA LINEA CON PAPEL CIAN, TINTA BLANCA
                 1930 ;
96AD  0E1F       1940         LD       C,31
96AF  54         1950         LD       D,H
96B0  5D         1960         LD       E,L
96B1  1C         1970         INC      E
96B2  362F       1980 HCOL4   LD       (HL),47
96B4  EDB0       1990         LDIR
                 2000 ;
                 2010 ;RECUPERAR REGISTROS
                 2020 ;
96B6  E1         2030 NOPLG   POP      HL
96B7  D1         2040         POP      DE
96B8  C1         2050         POP      BC
96B9  F1         2060         POP      AF
                 2070 ;
                 2080 ;FINAL DE INTERRUPCION
                 2090 ;
96BA  FB         2100         EI
96BB  ED4D       2110         RETI
                 2120 ;
                 2130 ;
                 2140 ;RETARDO PARA HORIZONTE EN FILA1
                 2150 ;
96BD  060F       2160 WT1LN   LD       B,15
96BF  10FE       2170 SELF11  DJNZ     SELF11
96C1  E6FF       2180         AND      #FF
96C3  23         2190         INC      HL
96C4  3D         2200         DEC      A
96C5  2B         2210         DEC      HL
96C6  3C         2220         INC      A
                 2230 ;
                 2240 ;ENTRA AQUI PARA HORIZONTE FILA 0
                 2250 ;
                 2260 ;BUSCAR NUEVO COLOR DE BORDE
                 2270 ;
96C7  2A2B96     2280 NOWAIT  LD       HL,(BOTBRD)
96CA  112596     2290         LD       DE,BORD
96CD  1A         2300         LD       A,(DE)
96CE  E610       2310         AND      16
96D0  AC         2320         XOR      H
                 2330 ;
                 2340 ;ALMACENARLO Y DARLE SALIDA
                 2350 ;
96D1  12         2360         LD       (DE),A
96D2  D3FE       2370         OUT      (#FE),A
                 2380 ;
                 2390 ;PONER BANDERA PARA NINGUN ATRI. A LLENAR
96D4  2600       2400         LD       H,0
                 2410 ;
                 2420 ;SALTO HACIA ATRAS A LA RUTINA PRINCIPAL
                 2430 ;
96D6  C38C96     2440         JP       INIT3
```

Por supuesto, estaremos empleando el modo 2 de interrupción, y, como verá, he elegido utilizar una tabla de vectores de 257 octetos apuntando a una instrucción de salto al gestor de interrupciones en #FDFD. Esta técnica fue descrita con más detalle en el capítulo 7. Poniendo la tabla en #FE00 y el buffer en #FF10 (recuerde que el octeto de menor peso debe ser #10), he empleado hábilmente el último 1/2K de la RAM, desperdiciando solamente 15 octetos. La siguiente rutina de inicialización establece la tabla de vectores,

selecciona IM2 y luego salta al gestor de interrupciones para asegurar que el buffer sea borrado. He llamado la rutina INT1 y su contraria para volver a seleccionar IM1 (si quisiera volver a BASIC) DISINT.

Las 3 últimas líneas del listado establecen el importantísimo salto en #FDFD.

```
                        10 ;INICIALIZAR PROCESADOR DE INTERRUPCION
                        20 ;CONSERVA REGISTROS COMO ESTABAN CUANDO SALIMOS MEDIANTE
                        30 ;EL GESTOR DE INTERRUPCIONES
                        40 ;
8A2B  F3                50 INT1     DI
8A2C  F5                60          PUSH     AF
8A2D  C5                70          PUSH     BC
8A2E  D5                80          PUSH     DE
8A2F  E5                90          PUSH     HL
8A30  3EFE             100          LD       A,#FE
8A32  ED47             110          LD       I,A
                       120 ;
                       130 ;ESTABLECER TABLA DE VECTORES PARA IM 2 LLENANDO 257 BYTES
                       140 ;DESDE #FE00 CON #FD
                       150 ;
8A34  2100FE           160          LD       HL,#FE00
8A37  45               170          LD       B,L
8A38  3D               180          DEC      A
8A39  77               190 TBLP     LD       (HL),A
8A3A  23               200          INC      HL
8A3B  10FC             210          DJNZ     TBLP
8A3D  77               220          LD       (HL),A
                       230 ;SELECCIONA IM 2 Y PREPARAR PARA ...
8A3E  3E28             240          LD       A,40
8A40  ED5E             250          IM       2
8A42  2600             260          LD       H,0
8A44  E5               270          PUSH     HL
8A45  21458A           280          LD       HL,CHSTRE
                       290 ;
                       300 ;UN SALTO AL GESTOR DE INTERRUPCIONES PARA BORRAR
                       310 ;EL BUFFER DE IMPRESION
8A48  C3488A           320          JP       INIT2
                       330 ;
                       340 ;
                       350 ;UTILICE ESTA RUTINA PARA VOLVER A SELECCIONAR IM 1
8A4B  3E3E             360 DISINT   LD       A,#3E
8A4D  ED56             370          IM       1
8A4F  ED47             380          LD       I,A
8A51  C9               390          RET
                       400 ;
                       410 ;POSICIONAR LAS INSTRUCCIONES DE SALTO
                       420 ;DEL GESTOR DE INTERRUPCIONES
FDFD                   430 LABEL    ORG      #FDFD
FDFD  C3FDFD           440          JP       INTERP
8A52                   450          ORG      LABEL
```

En el próximo capítulo proporcionaré una serie de rutinas para tratar todos los aspectos de la generación del horizonte y movimiento. A continuación encontrará un conjunto de potentes rutinas de animación de *sprites* para hacer uso total del procesador de impresión (todavía sin estrenar).

Si no desea emplear el generador de horizonte de pantalla completa en el gestor de interrupciones, puede desactivarlo con la siguiente secuencia:

```
LD      A,(COLOR DE BORDE)
LD      (TOPBRD+1),A
```

```
LD      (BOTBRD+1),A
XOR     A
LD      (ROWS+1),A
```

Esto establece el horizonte a su nivel máximo con mar y cielo del mismo color, y hace que la rutina evite cualquier trabajo con los atributos. ¡Puede entonces ignorar el siguiente capítulo con tranquilidad!

10

Moviendo el horizonte de pantalla completa por "pixels"

Antes de proporcionarles las rutinas de control de horizonte, se necesitan unas palabras de precaución sobre el uso del procesador de interrupciones.

Aunque el Z-80 siempre recibe una señal de interrupción exactamente en la misma etapa del cuadro de TV, nunca reacciona a esta señal hasta que ha terminado el proceso de la instrucción actual. Esto nos puede llevar a una variación de hasta 23 T-estados (como en el caso de una de las instrucciones más largas, EX(SP), IX) en el tiempo en que se procesa la interrupción.

En términos humanos, esto no parece muy largo, pero es suficiente para que la TV progrese aproximadamente una décima parte del camino a lo largo de una fila, con lo cual afecta nuestro horizonte "artificial", desplazando momentáneamente la parte central, hacia la derecha o la izquierda. De hecho, el generador del horizonte puede tolerar un desplazamiento de tiempo de alrededor ±4 T-estados sin ningún efecto malo; por tanto, siempre que volvemos a una instrucción HALT (que hace que el procesador ejecute continuamente NOPs de 4 T-estados cada uno) antes de que ocurra una interrupción, entonces no habrá problema alguno.

Sin embargo, si desea ejecutar alguna otra rutina mientras espera una interrupción, probablemente encontrará que aparece un temblor. En este caso la solución es ensanchar nuestras "filas de tinta", que, como recordará, tienen 5 columnas de ancho, a ambos lados del horizonte central, hasta que cubran por completo el área del temblor. No necesitará hacerlas más anchas de 7 columnas a cada lado, dejando 18 columnas sin tocar en el centro de la pantalla. Na-

turalmente, necesitará entonces hacer unas ligeras modificaciones en todas las rutinas de horizonte, para que calculen las direcciones correctas de los nuevos atributos y las filas de tinta, y para que trabajen correctamente con ellos. Lo mismo ocurre para el generador de horizonte en el gestor de interrupción.

Ahora empezaré el desarrollo de la primera rutina de horizonte, HRZST1. Su función será borrar las últimas filas de tinta insertas en la pantalla, y calcular las direcciones de las nuevas, después que hayamos movido el horizonte. Almacenaremos la dirección de las 5 filas de tinta de la izquierda en HRZN1, y la de las 5 de la derecha en HRZN2. Cuando no se necesite ninguna fila de tinta (si el horizonte no está en el texto o si está entre dos líneas), pondremos la parte de mayor peso de HRZN1 y HRZN2 a cero, como banderas.

La dirección de la línea de atributos actual siempre estará almacenada en HRZN4 y también inserta en el propio gestor de interrupciones en (HRZN3 + 1). La parte de mayor peso del segundo se pondrá a cero cuando no se necesite ninguna manipulación de atributos, con lo cual el generador de horizonte apuntará al gestor de interrupciones de la ROM. Esta es la forma más fácil para asegurarse de que el generador siga obteniendo bien los tiempos para el cambio de borde.

Por tanto, necesitaremos definir las variables al principio del programa:

```
        ORG     (SU DIRECCION)
HRZN1   DEFW    0
HRZN2   DEFW    0
HRZN4   DEFW    0
```

Y he aquí el listado de HRZST1, seguido de unas notas sobre su empleo.

```
                 10 ;BORRAR FILAS DE TINTA VIEJAS Y ESTABLECER
                 20 ;NUEVOS VALORES PARA LOCALIZACION DE FILAS DE TINTA
                 30 ;Y ATRIBUTOS
                 40 ;ENTRADA : C=NO. DE FILAS DE TEXTO SOBRE EL HORIZONTE
                 50 ;SE CONSERVAN : DE, C
                 60 ;
                 70 ;
8DA2  2AA28D     80 HRZST1    LD        HL,(HRZN1)
                 90 ;
                100 ;BORRAR LAS 5 FILAS DE TINTA A LA IZD.
                110 ;
8DA4  AF        120           XOR       A
8DA6  77        130           LD        (HL),A
8DA7  2C        140           INC       L
8DA8  77        150           LD        (HL),A
8DA9  2C        160           INC       L
8DAA  77        170           LD        (HL),A
8DAB  2C        180           INC       L
8DAC  77        190           LD        (HL),A
8DAD  2C        200           INC       L
8DAE  77        210           LD        (HL),A
                220 ;
                230 ;AHORA LAS CINCO DE LA DERECHA
                240 ;
8DAF  2AAF8D    250           LD        HL,(HRZN2)
8DB2  77        260           LD        (HL),A
8DB3  2C        270           INC       L
```

```
8DB4  77        280           LD        (HL),A
8DB5  2C        290           INC       L
8DB6  77        300           LD        (HL),A
8DB7  2C        310           INC       L
8DB8  77        320           LD        (HL),A
8DB9  2C        330           INC       L
8DBA  77        340           LD        (HL),A
8DBB  79        350 HRZST2    LD        A,C
8DBC  32BD8D    360           LD        (ROWS+1),A
                370 ;ESTA TODAVIA EL HORIZONTE EN EL AREA DE TEXTO?
                380 ;
8DBF  FEC1      390           CP        #C1
                400 ;
                410 ;SI NO, ENTONCES NO SE NECESITA NINGUNA FILA DE TINTA
8DC1  3031      420           JR        NC,NOWRK
                430 ;
                440 ;LOCALIZAR LINEA DE ATRIBUTOS
                450 ;
8DC3  07        460           RLCA
8DC4  07        470           RLCA
8DC5  E603      480           AND       3
8DC7  F658      490           OR        #58
8DC9  67        500           LD        H,A
8DCA  79        510           LD        A,C
8DCB  87        520           ADD       A,A
8DCC  87        530           ADD       A,A
8DCD  E6E0      540           AND       #E0
8DCF  6F        550           LD        L,A
                560 ;ALMACENARLA
8DD0  22D08D    570           LD        (HRZN4),HL
8DD3  79        580           LD        A,C
                590 ;
                600 ;SI EL HORIZONTE ESTA ENTRE 2 LINEAS ENTONCES NO SE
                610 ;NECESITA NINGUNA FILA DE TINTA
                620 ;
8DD4  E607      630           AND       7
8DD6  281C      640           JR        Z,NOWRK
                650 ;
                660 ;LOCALIZAR EL 28-AVO ATRI.
                670 ;
8DD8  7D        680           LD        A,L
8DD9  F61B      690           OR        #1B
8DDB  45        700           LD        B,L
8DDC  6F        710           LD        L,A
                720 ;
                730 ;INSERTARLO EN EL GESTOR DE INTERRUPCIONES
8DDD  22DE8D    740           LD        (HRZN3+1),HL
                750 ;LOCALIZAR EL 28-ESIMO OCTETO DE FILA DEL A.P. DEBAJO HORIZON
                   TE
                760 ;
8DE0  79        770           LD        A,C
8DE1  1F        780           RRA
8DE2  37        790           SCF
8DE3  1F        800           RRA
8DE4  A7        810           AND       A
8DE5  1F        820           RRA
8DE6  A9        830           XOR       C
8DE7  E6F8      840           AND       #F8
8DE9  A9        850           XOR       C
8DEA  67        860           LD        H,A
8DEB  22AF8D    870           LD        (HRZN2),HL
                880 ;AHORA EL PRIMER BYTE DE LA FILA SOBRE EL HORIZONTE
                890 ;
8DEE  25        900           DEC       H
8DEF  68        910           LD        L,B
8DF0  22A28D    920           LD        (HRZN1),HL
8DF3  C9        930           RET
```

```
8DF4  AF          940 NOWRK    XOR       A
                  950 ;CUANDO NO SE NECESITA NINGUNA FILA DE TINTA, SENALAR
                  960 ;LAS VARIABLES; EN LA ROM
8DF5  32A38D      970          LD        (HRZN1+1),A
8DF8  32B08D      980          LD        (HRZN2+1),A
8DFB  32DF8D      990          LD        (HRZN3+2),A
8DFE  C9         1000          RET
```

HRZST1 será llamada por la siguiente rutina HRZMV1, siempre que mueva el horizonte (ST viene de SeT, poner; MV de MoVe, mover). También puede llamarla directamente para mover el horizonte a cualquier nivel en la pantalla de una vez desde cualquier nivel anterior. Basta...

```
LD        C,(NO. DE FILAS SOBRE EL HORIZONTE)
CALL      HRZST1
```

...llene a continuación las filas de tinta y los atributos como desee. Si está poniendo el nivel del horizonte por primera vez debe saltarse la sección que borra las viejas filas de tinta entrando en HRZST2.

Para desarrollar una rutina que mueva el horizonte, HRZMV1, primero debemos definir dos variables. HRZSPD contendrá el número de filas que se mueve el horizonte cada vez que se llama la rutina. Este número variará entre 1 y 8. La dirección del horizonte se almacenará en CNTRL, que puede ponerse (por ejemplo) mediante una rutina de exploración de teclado. El bit 2 de CNTRL se pondrá a uno (valor 4) para un movimiento hacia abajo y el bit 3 (valor 8) para un movimiento hacia arriba. Así que empezamos con las líneas:

```
          ORG       (SU DIRECCION)
HRZSFD    DEFB      0
CNTRL     DEFB      0
```

Debido al problema de un horizonte en la fila uno, descrito en el capítulo anterior, he restringido el movimiento a niveles por debajo de ésta. En mi TV hay unas 236 filas desde la parte superior del área del texto hasta la parte inferior de la pantalla, así es que he puesto el límite más bajo en (ROWS + 1) = 236. Probablemente querrá modificar esto para su propia TV, y en todo caso debe recordar que, cuanto más bajo sea el horizonte, más tiempo tardará en generarlo y, por tanto, menos tiempo tendrá para hacer cualquier otra cosa antes de la siguiente interrupción (como animación de *sprites*, por ejemplo).

La función principal de HRZMV1 es cuidar de los atributos según se mueve el horizonte. Si, por ejemplo, acabamos de mover el horizonte hasta la fila 0 de la línea actual, o dentro de la línea de encima de ella, entonces necesitaremos cambiar una línea de atributos de cielo a mar. De la misma forma, si acabamos de moverlo hacia abajo hasta una nueva línea, debemos llenar sus atributos con cielo, dispuesto para la siguiente interrupción. HRZMV1 será llamado por la rutina de horizonte principal, HRZNMK. Cuando la rutina HRZMV1 concluye su trabajo con los atributos, hace un salto a HRZST1 para esta-

blecer los nuevos valores para HRZN1...HRZN4 y limpia todas las filas de tinta anteriores.

Puede que observe la existencia de dos etiquetas sin usar en el listado, HCOL5 y HCOL6. Estas se utilizarán en una rutina posterior que establecerá nuevos colores para el cielo y el mar. He aquí el listado:

```
                    10 ;RUTINA PARA CAMBIAR NIVEL DE HORIZONTE
                    20 ;POR CANTIDAD (HRZSPD) EN DIRECCION
                    30 ;(CNTRL) NOTA 4=ABAJO, 8=ARRIBA
                    40 ;
                    50 ;
903B  3A3B90        60 HRZMV1   LD      A,(HRZSPD)
903E  47            70          LD      B,A
903F  3A3F90        80          LD      A,(CNTRL)
9042  110000        90          LD      DE,0
                   100 ;
                   110 ;TEST PARA SUBIR
                   120 ;
9045  CB5F         130          BIT     3,A
9047  C27690       140          JP      NZ,UP2
                   150 ;
                   160 ;TEST PARA BAJAR
                   170 ;
904A  CB57         180          BIT     2,A
904C  C8           190          RET     Z
                   200 ;
                   210 ;INCREMENTAR LAS FILAS SOBRE EL HORIZONTE POR HRZSPD
                   220 ;
904D  3A4E90       230          LD      A,(ROW+1)
9050  80           240          ADD     A,B
                   250 ;
                   260 ;COMPROBACION DE SEGURIDAD PARA NIVEL 1 MINIMO DE HORIZONTE
                   270 ;
9051  FEEC         280          CP      236
9053  D0           290          RET     NC
                   300 ;
                   310 ;SI FILAS>192 SALTAR A HRZST1
                   320 ;
9054  FEC1         330          CP      #C1
9056  4F           340          LD      C,A
9057  D25790       350          JP      NC,HRZST1
                   360 ;
                   370 ;ESTAMOS EN LA FILA CERO DE UNA LINEA?
                   380 ;
905A  E607         390          AND     7
905C  2007         400          JR      NZ,NROWZ1
                   410 ;
                   420 ;SI ES ASI, ESTA HRZSPD EN 8 LINEAS POR MOVER?
                   430 ;
905E  CB58         440          BIT     3,B
                   450 ;
                   460 ;SI NO, SALTAR A HRZST1
                   470 ;
9060  CA5790       480          JP      Z,HRZST1
9063  1808         490          JR      ROWZ1
9065  B8           500 NROWZ1   CP      B
                   510 ;
                   520 ;ESTAMOS AHORA MOVIENDONOS DESDE LA FILA CERO DE UNA
                   530 ;LINEA?
9066  2805         540          JR      Z,ROWZ1
                   550 ;
                   560 ;SI NO, SI ESTAMOS EN LA MISMA LINEA, SALTAR
                   570 ;
                   580 ;
```

```
9068  D25790     590          JP       NC,HRZST1
906B  1E20       600          LD       E,#20
                 610  ;
                 620  ;RETARDO PARA ASEGURAR QUE LA TV HA ACABADO LA NUEVA LINEA
                 630  ;
906D  064D       640  ROWZ1    LD       B,77
906F  10FE       650  SELF42   DJNZ     SELF42
                 660  ;
                 670  ;LLENAR LA NUEVA LINEA CON PAPEL CIAN, TINTA BLANCA
                 680  ;
9071  062F       690  HCOL5    LD       B,47
9073  C39A90     700          JP       UPINIT
                 710  ;
9076  3A7790     720  UP2      LD       A,(ROWS+1)
                 730  ;
                 740  ;DECREMENTAR FILAS ENCIMA HORIZONTE POR HRZSPD
                 750  ;
9079  90         760          SUB      B
                 770  ;
                 780  ;VOLVER SI ES NEGATIVO
                 790  ;
907A  D8         800          RET      C
                 810  ;
                 820  ;VOLVER SI ES MENOR QUE 2
                 830  ;
907B  FE02       840          CP       2
907D  D8         850          RET      C
                 860  ;
                 870  ;SI FILAS>184 SALTAR A HRZST1
                 880  ;
907E  FEB9       890          CP       #B9
9080  4F         900          LD       C,A
9081  D25790     910          JP       NC,HRZST1
9084  ED44       920          NEG
9086  E607       930          AND      7
9088  110000     940          LD       DE,0
                 950  ;
                 960  ;SALTAR SI NO ESTAMOS EN FILA 0 DE UNA LINEA
                 970  ;
908B  2007       980          JR       NZ,NROWZ2
                 990  ;
                1000  ;SI NO LLENAR LA LINEA ACTUAL
                1010  ;CON PAPEL AZUL, TINTA BLANCA
                1020  ;
908D  CB58       1030         BIT      3,B
908F  2807       1040         JR       Z,HCOL6
9091  11E0FF     1050         LD       DE,#FFE0
9094  B8         1060 NROWZ2  CP       B
                 1070 ;
                 1080 ;SI AUN ESTAMOS EN LA MISMA LINEA SALTAR
                 1090 ;
9095  D25790     1100         JP       NC,HRZST1
                 1110 ;
                 1120 ;SI NO LLENAR LA ANTERIOR CON PAPEL AZUL
                 1130 ;TINTA BLANCA
9098  060F       1140 HCOL6   LD       B,15
909A  2A9A90     1150 UPINIT  LD       HL,(HRZN4)
909D  19         1160         ADD      HL,DE
                 1170 ;
                 1180 ;RELLENADOR DE PROPOSITO GENERAL
                 1190 ;
909E  54         1200         LD       D,H
909F  5D         1210         LD       E,L
90A0  1C         1220         INC      E
90A1  70         1230         LD       (HL),B
90A2  79         1240         LD       A,C
90A3  011F00     1250         LD       BC,31
```

```
90A6  EDB0      1260          LDIR
90A8  4F        1270          LD        C,A
                1280 ;
                1290 ;FINALMENTE SALTAR A HRZST1
90A9  C35790    1300 ;
                1310          JP        HRZST1
```

Ahora tenemos todo el material necesario para poner y mover el nivel del horizonte. HRZMV1 se ocupa de todo el trabajo de los atributos según se mueve el horizonte, mientras que HRZST1 asegura que sabemos dónde están dichos atributos, borra las filas de tinta anteriores y calcula las direcciones de las nuevas. Solamente queda insertar esas filas de tinta en el archivo de pantalla antes de cada interrupción (puede que hayan sido sobreimpresas por *sprites* desde la última). La rutina maestra HRZNMK (MK de MaKe, hacer) hará esto después de haber llamado a HRZMV1, que hace todos los cambios necesarios en los atributos y las variables. De esta forma, HRZNMK es la única rutina que tenemos que llamar directamente después de cada interrupción, como se verá en la rutina de demostración que sigue a su listado.

```
                 10 ;RUTINA PRINCIPAL DE HORIZONTE
                 20 ;SOLO LLAMARLA JUSTO DESPUES DE CADA INTERRUPCION
                 30 ;HABIENDO PUESTO LAS VARIABLES CNTRL Y HRZSPD
                 40 ;
                 50 ;
895F  CD5F89     60 HRZNMK   CALL      HRZMV1
                 70 ;
                 80 ;VOLVER SI NO SE NECESITA NINGUNA FILA DE TINTA
                 90 ;
8962  2A6289    100           LD        HL,(HRZN1)
8965  24        110           INC       H
8966  25        120           DEC       H
8967  C8        130           RET       Z
                140 ;
                150 ;INSERTAR FILAS DE TINTA PARA EL HORIZONTE
                160 ;
8968  3EFF      170           LD        A,#FF
                180 ;
                190 ;PRIMERO LAS 5 DE LA IZQ.
                200 ;
896A  77        210           LD        (HL),A
896B  2C        220           INC       L
896C  77        230           LD        (HL),A
896D  2C        240           INC       L
896E  77        250           LD        (HL),A
896F  2C        260           INC       L
8970  77        270           LD        (HL),A
8971  2C        280           INC       L
8972  77        290           LD        (HL),A
                300 ;
                310 ;AHORA LAS 5 DE LA DER.
                320 ;
8973  2A7389    330           LD        HL,(HRZN2)
8976  77        340           LD        (HL),A
8977  2C        350           INC       L
8978  77        360           LD        (HL),A
8979  2C        370           INC       L
897A  77        380           LD        (HL),A
897B  2C        390           INC       L
897C  77        400           LD        (HL),A
897D  2C        410           INC       L
897E  77        420           LD        (HL),A
897F  C9        430           RET
```

Para ilustrar su recién encontrado poder sobre el Spectrum, he aquí una rutina de demostración que emplea INT1, HRZST2, HRZNMK, DISINT e indirectamente HRZST1, HRZMV1 y el gestor de interrupciones.

La rutina proporciona control directo sobre el horizonte. Pulsando cualquiera de las teclas de 1 a 5 lo mueve hacia arriba, mientras que las teclas de CAPS SHIFT a V lo mueven hacia abajo. Las teclas 8, 9 y 0 se utilizan para controlar la velocidad del horizonte. Piense en ellas como un número de tres bits que se ponen a uno cuando se pulsa su tecla, luego añada uno para obtener HRZSPD. Por tanto, pulsando las teclas 8 y 0 (101 binario = 5 decimal) da una velocidad de 6 filas por cuadro de TV. Finalmente, aquí está la rutina, DEMO:

```
8AC5  CDC58A      10 DEMO      CALL      INT1
                  20 ;
                  30 ;PONER HORIZONTE INICIAL
                  40 ;
8AC8  0E54        50           LD        C,84
8ACA  CDCA8A      60           CALL      HRZST2
                  70 ;
                  80 ;ESPERAR INTERRUPCION
                  90 ;
8ACD  76         100 DMLP      HALT
                 110 ;
                 120 ;C CONTENDRA LA DIRECCION
                 130 ;
8ACE  0E00       140           LD        C,0
                 150 ;
                 160 ;COMPROBAR MEDIA-FILA INFERIOR-IZQUIERDA
                 170 ;
8AD0  3EFE       180           LD        A,#FE
8AD2  DBFE       190           IN        A,(#FE)
8AD4  2F         200           CPL
8AD5  E61F       210           AND       #1F
8AD7  2802       220           JR        Z,ND1
                 230 ;
                 240 ;SI PULSADA, PONER BIT 2 PARA "ABAJO"
                 250 ;
8AD9  CBD1       260           SET       2,C
                 270 ;
                 280 ;COMPROBAR MEDIA-FILA SUPERIOR-IZQ.
                 290 ;
8ADB  3EF7       300 ND1       LD        A,#F7
8ADD  DBFE       310           IN        A,(#FE)
8ADF  2F         320           CPL
8AE0  E61F       330           AND       #1F
8AE2  2802       340           JR        Z,NU1
                 350 ;
                 360 ;SI PULSADA, PONER BIT 3 PARA "ARRIBA"
                 370 ;
8AE4  CBD9       380           SET       3,C
                 390 ;
                 400 ;ALMACENAR DIRECCION
                 410 ;
8AE6  79         420 NU1       LD        A,C
8AE7  32E78A     430           LD        (CNTRL),A
                 440 ;
                 450 ;COMPROBAR MEDIA-FILA SUPERIOR-DER.
                 460 ;
8AEA  3EEF       470           LD        A,#EF
8AEC  DBFE       480           IN        A,(#FE)
                 490 ;
```

```
                    500 ;UTILIZAR LOS 3 BITS DE TECLA DER. PARA VELOCIDAD HORIZONTE
                    510 ;
8AEE  2F            520         CPL
8AEF  E607          530         AND       7
8AF1  3C            540         INC       A
8AF2  32F28A        550         LD        (HRZSPD),A
                    560 ;
                    570 ;LLAMAR A LA RUTINA MAESTRA DE HORIZONTE
                    580 ;
8AF5  CDF58A        590         CALL      HRZNMK
                    600 ;
                    610 ;COMPROBAR TECLA BREAK
                    620 ;
8AF8  3E7F          630         LD        A,#7F
8AFA  DBFE          640         IN        A,(#FE)
8AFC  1F            650         RRA
8AFD  38CE          660         JR        C,DMLP
                    670 ;
                    680 ;SI PULSADA, VOLVER AL BASIC
                    690 ;
8AFF  CDFF8A        700         CALL      DISINT
8B02  C9            710         RET
```

Como rutina final de esta "serie de horizonte", he preparado HRZCOL, que le permite establecer las demás rutinas para cualquier combinación de colores de cielo y de mar. También establece los colores de tinta para encima y debajo del horizonte (si está moviendo formas sobre el horizonte, probablemente necesitará que ambos sean el mismo). Tiene posibilidad de provocar que el gestor de interrupciones genere un sonido de fondo de "motor", bien a 50 Hz o a 100 Hz, sumando 16 a uno o ambos valores de papel respectivamente. Se debe preparar los registros de la siguiente forma:

H = valor de papel de mar (+ 16 para sonido)
L = valor de papel de cielo (+16 para sonido de 100 Hz)

B = valor de la tinta de mar
C = valor de la tinta de cielo

Por ejemplo, para producir un suelo verde y un cielo blanco, con tinta negra en ambos,

```
LD          HL,#407
LD          BC,#0
CALL        HRZCOL
```

La rutina inserta los atributos correctos en las etiquetas de HCOL1 a HCOL6 que se encuentran en el gestor de interrupciones (HCOL1 a HCOL4) y en HRZMV1 (HCOL5 y HCOL6).

```
10 ;RUTINA PARA PONER COLORES DE MAR Y CIELO
20 ;ENTRADA : H=PAPEL MAR,L=PAPEL CIELO
30 ;B=TINTA MAR,C=TINTA CIELO
40 ;SUMAR 16 A H O L O AMBOS PARA SONIDO
50 ;
60 ;
```

```
8A88  7C          70 HRZCOL    LD        A,H
8A89  328A8A      80           LD        (BOTBRD+1),A
                  90 ;
                 100 ;DEJAR EL PAPEL MAR EN H
                 110 ;
8A8C  E607       120           AND       7
8A8E  67         130           LD        H,A
8A8F  7D         140           LD        A,L
8A90  32918A     150           LD        (TOPBRD+1),A
                 160 ;
                 170 ;DEJAR EL PAPEL CIELO EN L
                 180 ;
8A93  E607       190           AND       7
8A95  6F         200           LD        L,A
                 210 ;
                 220 ;HCOL1 NECESITA PAPEL PAPEL-CIELO Y TINTA PAPEL-MAR
                 230 ;
8A96  07         240           RLCA
8A97  07         250           RLCA
8A98  07         260           RLCA
8A99  5F         270           LD        E,A
8A9A  B4         280           OR        H
8A9B  329C8A     290           LD        (HCOL1+1),A
                 300 ;
                 310 ;HCOL2 NECESITA PAPEL PAPEL-MAR Y TINTA PAPEL-CIELO
                 320 ;
8A9E  7C         330           LD        A,H
8A9F  07         340           RLCA
8AA0  07         350           RLCA
8AA1  07         360           RLCA
8AA2  57         370           LD        D,A
8AA3  B5         380           OR        L
8AA4  32A58A     390           LD        (HCOL2+1),A
                 400 ;HCOL3 NECESITA PAPEL PAPEL-MAR Y TINTA TINTA-MAR
                 410 ;
8AA7  7A         420           LD        A,D
8AA8  B0         430           OR        B
8AA9  32AA8A     440           LD        (HCOL3+1),A
                 450 ;
                 460 ;...LO MISMO PARA HCOL6
                 470 ;
8AAC  32AD8A     480           LD        (HCOL6+1),A
                 490 ;
                 500 ;HCOL4 NECESITA PAPEL PAPEL-CIELO Y TINTA TINTA-CIELO
                 510 ;
8AAF  7B         520           LD        A,E
8AB0  B1         530           OR        C
8AB1  32B28A     540           LD        (HCOL4+1),A
                 550 ;
                 560 ;...LO MISMO PARA HCOL5
                 570 ;
8AB4  32B58A     580           LD        (HCOL5+1),A
8AB7  C9         590           RET
```

11

Una serie de rutinas para complementar el procesador de impresión

En este capítulo desarrollaré una serie completa de rutinas de impresión para sacar el máximo partido del procesador de impresión controlado por interrupciones que se vio en el capítulo 9. En el capitulo siguiente se verán las rutinas de animación de *sprites*.

Para empezar, sería útil tener una rutina sencilla con la que se pueda enviar cualquier carácter al buffer. En vez de utilizar la dirección de la celdilla actual en el archivo de pantalla, como hace el BASIC del Spectrum con la variable del sistema DF-CC, es más fácil seguir de cerca la posición de impresión utilizando el archivo de atributos. De esta forma definimos una variable ATCC para que contenga la dirección de atributo de la celdilla actual y empezamos con las líneas

```
      ORG    (SU DIRECCION)
ATCC  DEFW   #5800
```

con lo cual inicializamos nuestro marcador al ángulo superior izquierdo del área de texto. La base de la tabla que contiene los datos de carácter será almacenada en CHARS y también podemos empezar por apuntar al juego de caracteres del BASIC del Spectrum, empleando la línea:

```
CHARS DEFW  #3C00
```

Recuerde que CHSTRE contiene el número de entradas utilizadas en el *buffer* de impresión, y BUFFPT apunta a la siguiente entrada libre. Ambas variables se modifican en concordancia. El resto de la rutina es autoexplicativa: por tanto, pasaré directamente a presentarle HIPRINT.

```
                    10 ;ENVIAR UN CARATER AL BUFFER
                    20 ;ENTRADA : A=CODIGO DEL CARACTER
                    30 ;SALIDA : BC=DIRECCION DE LOS DATOS DEL CARACTER
                    40 ;DE=ATCC (VER TEXTO)
                    50 ;HL=SIGUIENTE ENTRADA DE BUFFER
                    60 ;A=OCTETO DE MAYOR PESO DE LA DIRECCION DE A.P.
                    70 ;
                    80 ;MULTIPLICAR CODIGO POR 8
                    90 ;
8CC8  6F          100 HIPRNT    LD        L,A
8CC9  2600        110           LD        H,0
8CCB  29          120           ADD       HL,HL
8CCC  29          130           ADD       HL,HL
8CCD  29          140           ADD       HL,HL
                  150 ;
                  160 ;SUMAR DIRECCION BASE DE LA TABLA
                  170 ;
8CCE  ED5BCE8C    180           LD        DE,(CHARS)
8CD2  19          190           ADD       HL,DE
                  200 ;
                  210 ;HACER BC=DIRECCION DE DATOS
                  220 ;
8CD3  44          230           LD        B,H
8CD4  4D          240           LD        C,L
8CD5  21D58C      250 PLACE     LD        HL,CHSTRE
8CD8  7E          260 PWAIT     LD        A,(HL)
                  270 ;
                  280 ;SI EL BUFFER ESTA LLENO ESPERAR
                  290 ;UNA INTERRUPCION
                  300 ;
8CD9  FE28        310           CP        40
8CDB  DAE18C      320           JP        C,GO
8CDE  FB          330           EI
8CDF  18F7        340           JR        PWAIT
8CE1  F3          350 GO        DI
                  360 ;
                  370 ;ACTUALIZAR CONTADOR DE CARACTERES DEL BUFFER
                  380 ;
8CE2  34          390           INC       (HL)
8CE3  ED5BE38C    400           LD        DE,(ATCC)
                  410 ;
                  420 ;DE=DIRECCION DE ATRIBUTO
                  430 ;
8CE7  2AE78C      440           LD        HL,(ATT)
                  450 ;
                  460 ;H=MASCARA, L=ATRI.
                  470 ;CONSTRUIR NUEVO ATRIBUTO
                  480 ;
8CEA  1A          490           LD        A,(DE)
8CEB  AD          500           XOR       L
8CEC  A4          510           AND       H
8CED  AD          520           XOR       L
8CEE  2AEE8C      530           LD        HL,(BUFFPT)
                  540 ;
                  550 ;HL=PRIMERA LOCALIZACION LIBRE DEL BUFFER
                  560 ;ROTAR Y ALMACENAR ATRI.
                  570 ;
8CF1  07          580           RLCA
```

```
8CF2  77        590          LD       (HL),A
                600 ;
                610 ;ALMACENAR DIRECCION DE ATRIBUTO
                620 ;
8CF3  2C        630          INC      L
8CF4  73        640          LD       (HL),E
8CF5  23        650          INC      HL
8CF6  72        660          LD       (HL),D
8CF7  2C        670          INC      L
                680 ;
                690 ;BUSCAR Y ALMACENAR OCTETO DE MAYOR PESO DE DIRECCION DEL A.
                    P.
                700 ;
8CF8  7A        710          LD       A,D
8CF9  E603      720          AND      3
8CFB  07        730          RLCA
8CFC  07        740          RLCA
8CFD  07        750          RLCA
8CFE  F640      760          OR       64
8D00  77        770          LD       (HL),A
8D01  23        780          INC      HL
                790 ;
                800 ;ALMACENAR DIRECCION DE DATOS DE CARACTERES
                810 ;
8D02  71        820          LD       (HL),C
8D03  2C        830          INC      L
8D04  70        840          LD       (HL),B
8D05  23        850          INC      HL
                860 ;
                870 ;PONER BUFFPT APUNTANDO A LA SIGUIENTE ENTRADA LIBRE DE BUFF
                    ER
                880 ;
8D06  22EE8C    890          LD       (BUFFPT),HL
8D09  C9        900          RET
```

Habrá ocasiones durante algunos programas en que necesitará que el gestor de interrupciones imprima continuamente un carácter en particular o una serie de caracteres en el mismo sitio de la pantalla. Por ejemplo, puede desear superponer "puntos de mira de láser" en el centro de la pantalla, y éstos necesitan una OR-impresión continua en cada interrupción, ya que la nave espacial enemiga (o lo que sea) se mueve detrás de ellos. Como hay tan poco tiempo entre dos interrupciones (especialmente si está empleando el generador de horizonte en un nivel bajo), será muchísimo mejor no tener que cargar todos los datos para los puntos de mira en el buffer de impresión después de cada interrupción.

Para poder hacer esto, he diseñado un sistema de rutinas que emplean una subsección del buffer de impresión, que se llamará "RO-buffer" para indicar "Read only buffer" (Buffer de sólo lectura). A la inversa del caso de las entradas normales en el buffer de impresión, las del RO-buffer no serán borradas por el gestor de interrupciones cuando han sido impresas. Por tanto, sólo tenemos que asegurarnos, después de cada interrupción, de que se hayan insertado los atributos correctos para cada celdilla en cuestión en el RO-buffer. Si la máscara del atributo es el octeto cero, ni siquiera necesitaremos hacer esto, ya que los caracteres siempre se imprimirán con los mismos atributos, sin tener en cuenta los atributos anteriores para esa celdilla.

Las entradas consecutivas normalmente "creced" hacia arriba desde el prin-

cipio del buffer de impresión; por tanto, para mantenerlas separadas, haremos que el RO-buffer crezca hacia abajo desde el final del buffer de impresión. Las dos regiones no deben superponerse nunca. Almacenaremos el número de entradas del RO-buffer en la variable ROLNTH y el principio (dirección más baja) del RO-buffer en la variable ROBFPT. De esta forma los inicializamos (para un RO-buffer de una longitud de cero) con las líneas:

```
        ORG   (SU DIRECCION)
ROLNTH  DEFB  0
ROBFPT  DEFW  0
```

Antes de proseguir, necesitamos una rutina para establecer un RO-buffer. Lo que hará la siguiente rutina es ALTERAR la longitud del RO-buffer con el valor de C, que puede ser negativo o positivo. Habiendo ajustado ROLNTH, la rutina llena todas las entradas entre el final de las entradas "normales" (por ejemplo, BUFFPT) y el principio del RO-buffer, con caracteres "nulos" para evitar que se imprima alguna tontería. Por esta razón la rutina, llamada ALTRBF, siempre se debería llamar con las interrupciones desactivadas. ALTRBF entonces pone ROBFPT a la nueva dirección de principio del RO-buffer y lo devuelve en HL, que se utilizará posteriormente.

```
                 10 ;RUTINA PARA ALTERAR LA LONGITUD DEL RO-BUFFER
                 20 ;ENTRADA : C=ALTERACION DE LONGITUD
                 30 ;SE CONSERVA C
                 40 ;SALIDA : HL=COMIENZO DEL RO-BUFFER,B=0,A=NO. DE ENTRADAS
                 50 ;NO UTILIZADAS EN EL BUFFER DE IMPRESION
                 60 ;
                 70 ;
89F7  21F789     80 ALTRBF    LD        HL,ROLNTH
                 90 ;
                100 ;MODIFICAR ROLNTH POR C
                110 ;
89FA  7E         120          LD        A,(HL)
89FB  81         130          ADD       A,C
89FC  77         140          LD        (HL),A
                150 ;
                160 ;BUSCAR NO. DE ENTRADAS NO UTILIZADAS EN EL BUFFER DE
                170 ;IMPRESION (>=0)
                180 ;
89FD  3AFD89     190          LD        A,(CHSTRE)
8A00  47         200          LD        B,A
8A01  3E28       210          LD        A,40
8A03  90         220          SUB       B
8A04  96         230          SUB       (HL)
8A05  47         240          LD        B,A
                250 ;
                260 ;LLENARLAS CON ENTRADAS "NULAS"
                270 ;
8A06  2A068A     280          LD        HL,(BUFFPT)
                290 ;
                300 ;PERO SALTAR SI NO HAY NINGUNA ENTRADA QUE LLENAR
                310 ;
8A09  2807       320          JR        Z,HOPFL
```

```
8A0B  110600     330           LD        DE,6
                 340 ;
                 350 ;NOTA : D=0
                 360 ;
8A0E  72         370 BLNK      LD        (HL),D
8A0F  19         380           ADD       HL,DE
8A10  10FC       390           DJNZ      BLNK
                 400 ;
                 410 ;PONER ROBFPT APUNTANDO AL COMIENZO DEL RO-BUFFER
                 420 ;
8A12  22128A     430 HOPFL     LD        (ROBFPT),HL
8A15  C9         440           RET
```

Obviamente, cualquier rutina escrita para dar salida a los caracteres al buffer de impresión se adapta fácilmente para utilizar el RO-buffer, que es de hecho una subsección del anterior. Podría modificar HIPRINT o cualquiera de sus propias rutinas de impresión. Proporcionaré una rutina para hacer un volcado de formas pre-definidas, tales como nuestro punto de mira en el RO-buffer, pero primero veremos la rutina sencilla que se necesita para refrescar los octetos de atributo de cada entrada.

La rutina se llama SRVR1 (porque es una rutina de servicio, SeRVice Routine). Toma la dirección de atributos de una entrada, encuentra el atributo en el archivo, a continuación lo enmascara con nuestra variable estándar MASK, que, como siempre, está colocada directamente después de ATT, los atributos para nuestros caracteres. El octeto de atributos completo se rota entonces un bit a la izquierda (para contrarrestar la rotación hacia la derecha del gestor de interrupciones) y se inserta en el RO-buffer. Observe que si deseamos seleccionar OR-impresión ponemos a uno el bit 7 de ATT (el bit 7 de MASK siempre tiene que estar a cero), que luego se rotará para convertirlo en bit 0 antes de su inserción en el buffer. Esto también se aplica a HIPRNT.

Aquí, pues, está el listado de SRVR1. Observe el empleo de la bandera cero (Z) para detectar el final del buffer, cuando el octeto de menor peso de su dirección se convierte en cero.

```
                  10 ;RUTINA DE SERVICIO PARA ACTUALIZAR ATRIBUTOS
                  20 ;EN EL RO-BUFFER
                  30 ;SALIDA : B=MASCARA,C=ATRI. A=0
                  40 ;HL=OCTETO DESPUES DEL BUFFER DE IMPRESION
                  50 ;
89B1  2AB189      60 SRVR1     LD        HL,(ROBFPT)
                  70 ;
                  80 ;HACE B=MASCARA, C=ATRIBUTO
                  90 ;
89B4  ED4BB489   100           LD        BC,(ATT)
                 110 ;
                 120 ;TOMAR DIRECCION DE ATRI.
                 130 ;
89B8  2C         140 NXSRV1    INC       L
89B9  5E         150           LD        E,(HL)
89BA  2C         160           INC       L
89BB  56         170           LD        D,(HL)
89BC  2D         180           DEC       L
89BD  2D         190           DEC       L
                 200
```

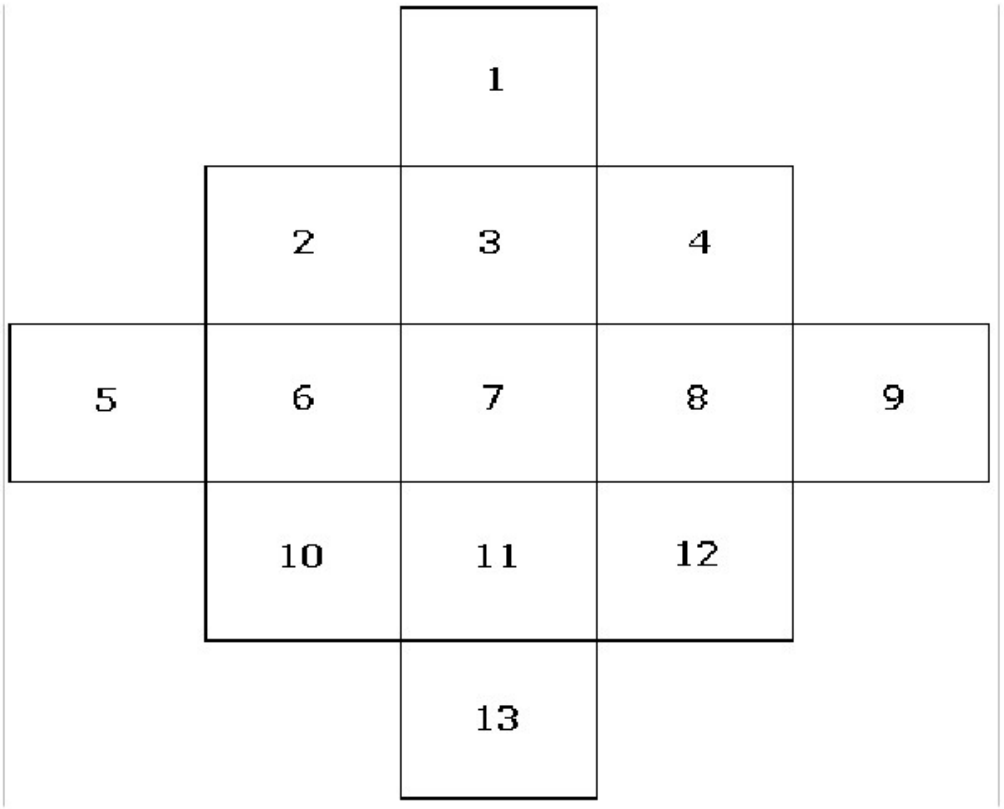
```
                210 ;TOMAR ATRIBUTO ACTUAL
                220 ;
89BE  1A        230         LD      A,(DE)
                240 ;
                250 ;CREAR UNO NUEVO
                260 ;
89BF  A9        270         XOR     C
89C0  A0        280         AND     B
89C1  A9        290         XOR     C
                300 ;
                310 ;ROTAR A LA IZQ. ,CONSERVANDO LA BANDERA OR
                320 ;
89C2  07        330         RLCA
                340 ;
                350 ;ALMACENAR NUEVO ATRI. EN EL BUFFER
                360 ;
89C3  77        370         LD      (HL),A
                380 ;
                390 ;DESPLAZARSE A LA SIGUIENTE ENTRADA
                400 ;
89C4  7D        410         LD      A,L
89C5  C606      420         ADD     A,6
89C7  6F        430         LD      L,A
89C8  20EE      440         JR      NZ,NXSRV1
                450 ;
                460 ;HASTA QUE EL RO-BUFFER HA SIDO COMPROBADO
89CA  C9        470         RET
```

Se debe llamar a SRVR1 en cualquier caso en que sea posible que los atributos de las celdillas empleadas por nuestras entradas del RO-buffer hayan cambiado, por ejemplo, al mover un *sprite* en ellas o al mover un horizonte una línea hacia abajo. La rutina también le da la oportunidad de variar el color de los caracteres "permanentes" de su pantalla, modificando ATT. Puesto que maneja el RO-buffer entero, cada entrada empleará el mismo valor ATT. Si no desea esto, entonces la modificación más fácil es cambiar el fragmento:

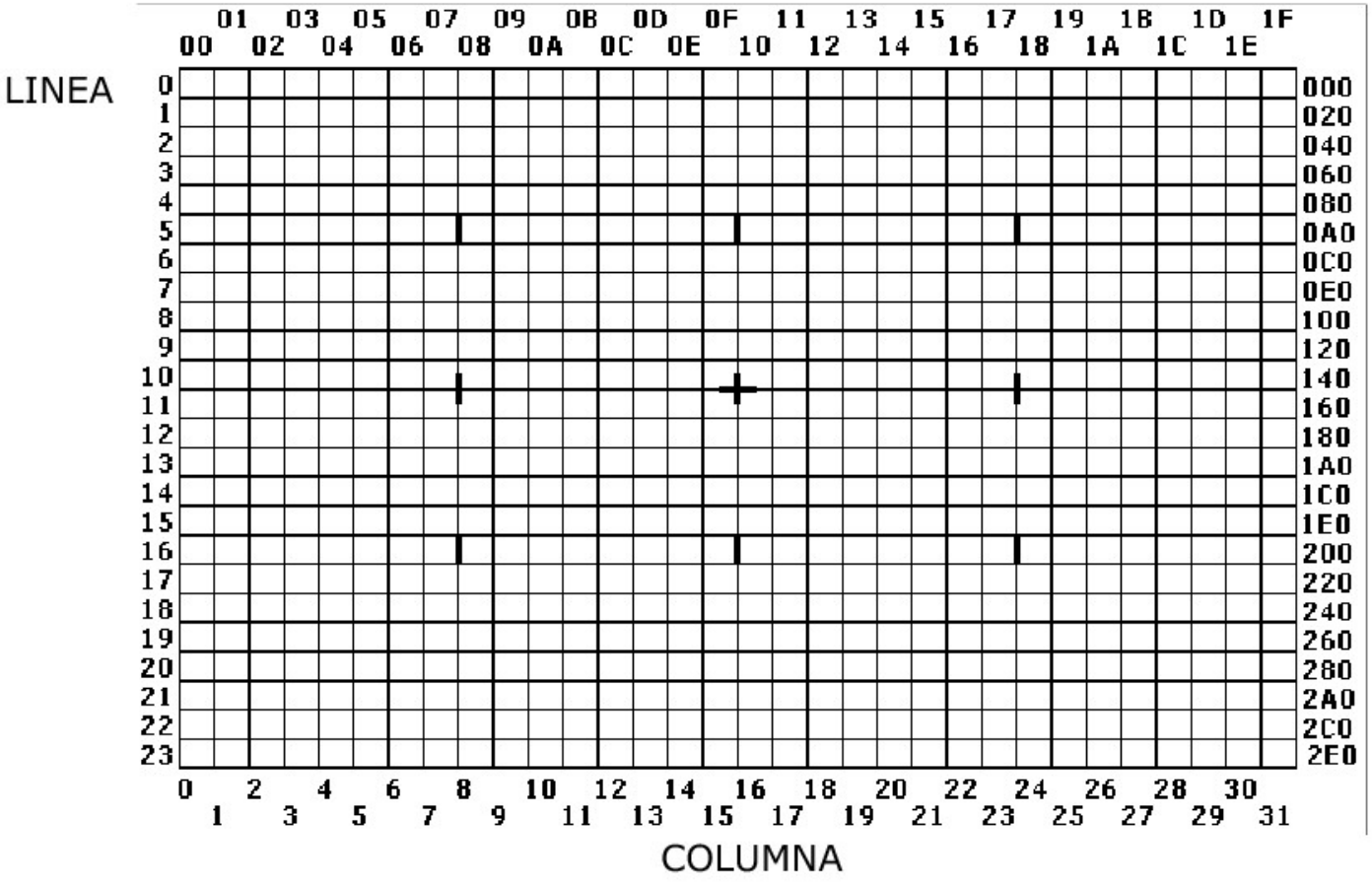
```
        XOR     C
        AND     B
        XOR     C
por:    LD      C,(HL)
        RRC     C
        XOR     C
        AND     B
        XOR     C
```

con lo cual se emplea efectivamente el valor original de ATT, con el cual la entrada fue insertada (por un HIPRNT modificado, por ejemplo).

Continuaré con aquella rutina especializada de la que hablé antes, para enviar "formas" específicas al RO-buffer. Una forma puede constar de varios caracteres separados que juntos forman una imagen que se imprime en la pantalla. Como ejemplo, tomaré el punto de mira láser, que se construirá con 13 caracteres del siguiente modo:



La rutina llamada SRVR2 necesitará dos tablas de datos. Una contendrá la posición deseada de cada carácter en la pantalla, mientras que la otra será una tabla de datos de caracteres, almacenada, como de costumbre, en 8 octetos por carácter. Los datos de posición y los datos de caracteres deben, por supuesto,



estar almacenados en el mismo orden, sin ningún espacio en blanco en las tablas.

Para representar la posición de cada carácter, he decidido numerar las celdillas de 0 a #02FF, en el orden de sus atributos. Esto tiene la ventaja de que el octeto de menor peso será idéntico al del atributo y a las direcciones del archivo de pantalla para la celdilla. Si prefiere emplear un sistema de coordenadas, es fácil escribir una rutina para hacer la conversión de un sistema al otro. O también podría modificar SRVR2.

Como ayuda para facilitar el cálculo de los datos de posición, he aquí un papel cuadriculado etiquetado para la pantalla; sólo tiene que leer el valor de la línea y añadir el valor de la columna (ambos en hexadecimal).

Las tareas de SRVR2 son tomar la posición de un carácter, calcular la dirección del atributo, la del archivo de pantalla y la del de datos del carácter y almacenarlos todos en el orden correcto en el RO-buffer. He aquí un listado de la rutina antes de la demostración de su uso:

```
                    10 ;RUTINA PARA ENVIAR DATOS AL RO-BUFFER
                    20 ;ENTRADA : HL=PRINCIPIO DE LOS DATOS DE POSICION
                    30 ;DE=PRINCIPIO DEL AREA RESERVADA AL RO-BUFFER
                    40 ;BC=DIRECCION DE DATOS DE CARACTER
                    50 ;A= NO. DE CARACTERES
                    60 ;SALIDA : A=0, BC=8
                    70 ;NOTA AF SE DESTRUYE
                    80 ;
8C04  C5            90 SRVR2     PUSH      BC
                   100 ;
                   110 ;UTILIZA BC COMO UNA CONSTANTE
                   120 ;
8C05  010800       130           LD        BC,8
                   140 ;
                   150 ;A SE CONVIERTE EN UN  CONTADOR
                   160 ;
8C08  08           170 NXCHR9    EX        AF,AF'
8C09  1C           180           INC       E
                   190 ;
                   200 ;TRANSFERIR OCTETO DE MENOR PESO DE LA DIRECION DE ATRI.
                   210 ;
8C0A  7E           220           LD        A,(HL)
8C0B  12           230           LD        (DE),A
8C0C  1C           240           INC       E
8C0D  23           250           INC       HL
                   260 ;
                   270 ;FORMAR EL OCTETO DE MAYOR PESO DE DIRECCION DE ATRI.
                   280 ;
8C0E  7E           290           LD        A,(HL)
8C0F  F658         300           OR        #58
8C11  12           310           LD        (DE),A
8C12  1C           320           INC       E
                   330 ;
                   340 ;FORMAR EL OCTETO DE MAYOR PESO DE DIRECCION DEL A.P.
                   350 ;
8C13  E603         360           AND       3
8C15  07           370           RLCA
8C16  07           380           RLCA
8C17  07           390           RLCA
8C18  F640         400           OR        #40
                   410 ;
                   420 ;ALMACENARLO
                   430 ;
```

```
8C1A  12        440          LD       (DE),A
8C1B  1C        450          INC      E
8C1C  23        460          INC      HL
                470 ;
                480 ;RECUPERAR DIRECCION DE DATOS DE CARACTER, SALVAR DIRECCION
                490 ;DE DATOS DE POSICION
                500 ;
8C1D  E3        510          EX       (SP),HL
8C1E  EB        520          EX       DE,HL
                530 ;
                540 ;ALMACENAR DRECCION DE DATOS DE CARACTER EN EL BUFFER
                550 ;
8C1F  73        560          LD       (HL),E
8C20  2C        570          INC      L
8C21  72        580          LD       (HL),D
8C22  2C        590          INC      L
                600 ;
                610 ;SUMAR OCHO A ESTA
                620 ;
8C23  EB        630          EX       DE,HL
8C24  09        640          ADD      HL,BC
                650 ;
                660 ;SALVAR NUEVA DIREC. DATOS DE CAR.,RECUPERAR DIREC. DATOS PO
                    SI.
                670 ;
8C25  E3        680          EX       (SP),HL
8C26  08        690          EX       AF,AF'
                700 ;
                710 ;SIGUIENTE CARACTER
                720 ;
8C27  3D        730          DEC      A
8C28  20DE      740          JR       NZ,NXCHR9
8C2A  E1        750          POP      HL
8C2B  C9        760          RET
```

Haciendo referencia al anterior diagrama de bloques del punto de mira láser, imprimiré el carácter 1 en (10,15). Por tanto, tenemos:

posición = 140 + 0F
= #14F

Con una inspección, vemos que el carácter 2 está una línea hacia abajo (+ #20) y una columna hacia la izquierda (–1); por tanto, su posición es #16E, el carácter 3 está en #16F y así sucesivamente. Etiquetaré el principio de los datos de posición como TRGPOS y el de datos de carácter como TRGDAT (ya ha sido hecho el trabajo pesado para usted, y lo encontrará en el listado de la rutina).

La primera tarea en TARGET es el llamar a INT1 para configurar el gestor de interrupciones. Hay que hacer esto antes de nada, ya que, como recordará, INT1 borra el *buffer* de impresión. Entonces desactivamos las interrupciones, que no son deseables mientras estamos cargando el *buffer* de impresión. Para reservar 13 caracteres en el RO-buffer utilizamos:

```
LD    C,13
CALL  ALTRBF
```

que devuelve la dirección de comienzo del RO-buffer en HL. Necesitamos poner esto en DE, con:

```
EX      DE,HL
```

a continuación preparar los otros registros para SRVR2

```
LD      HL,TRGPOS
LD      BC,TRGDAT
LD      A,13
CALL    SRV2
```

Ahora escoja una máscara completa (#7F, puesto que bit 7 siempre tiene que estar a cero) y OR-imprima (ponga a uno el bit 7 de ATT haciendo ATT = #80). Finalmente, inicialice las entradas de "atributos" del RO-buffer con SRVR1

```
LD      HL,#7F80
LD      (ATT),HL
CALL    SRVR1
```

Como demostración, he puesto la última instrucción en el bucle principal de forma que se llama SRVR1 después de cada interrupción, pero en este caso no es del todo necesario, puesto que el archivo de atributos no se altera dentro del bucle. SERIA necesario, sin embargo, si fuera a incorporar la rutina de demostración de horizonte del capítulo precedente.

Cuando se pulsa la tecla BREAK, TARGET termina por borrar el RO-buffer (poniendo ROLNTH a cero) y vuelve a seleccionar IM1.

```
CALL    DISINT
LD      A,(ROLNTH)    ;SUMA (-ROLNTH) A ROLNTH
NEG
LD      C,A
CALL    ALTRBF
RET
```

Los comentarios explican suficientemente el resto del listado; por tanto, he aquí TARGET:

```
                    10 ;ESTA DEMO. OR-IMPRIME (OR-PRINT) EN ALTA RESOLUCION
                    20 ;UN PUNTO DE MIRA EN EL CENTRO DE LA PANTALLA
                    30 ;
                    40 ;LLAMAR A INT PRIMERO, YA QUE ESTA BORRA EL BUFFER, ASI
                    50 ;COMO INICIALIZA EL GESTOR DE INTERRUPCIONES
                    60 ;
90B7  CDB790        70 TARGET    CALL      INT1
                    80 ;
                    90 ;NO HAY INTERRUPCIONES MIENTRAS SE MODIFICA EL BUFFER
```

```
                100 ;
90BA  F3        110             DI
                120 ;
                130 ;DEFINIR ROBUFF PARA 13 ENTRADAS
                140 ;
90BB  0E0D      150             LD      C,13
90BD  CDBD90    160             CALL    ALTRBF
                170 ;
                180 ;PREPARAR PARA VOLCADO DE DATOS PARA 13 CARACTERES
                190 ;EN EL BUFFER
                200 ;
90C0  EB        210             EX      DE,HL
90C1  21EB90    220             LD      HL,TRGPOS
90C4  010591    230             LD      BC,TRGDAT
90C7  3E0D      240             LD      A,13
                250 ;
                260 ;LLENAR EL RO-BUFFER
                270 ;
90C9  CDC990    280             CALL    SRVR2
                290 ;
                300 ;SELECCIONAR LA MASCARA COMPLETA, Y LA OPERACION OR-PRINT
                310 ;PONIENDO EL BIT 7 DE ATT
                320 ;
90CC  21807F    330             LD      HL,#7F80
90CF  22CF90    340             LD      (ATT),HL
90D2  FB        350             EI
                360 ;
                370 ;CALCULAR ATRIBUTOS
                380 ;
90D3  CDD390    390 TSLP        CALL    SRVR1
                400 ;
                410 ;ESPERAR INTERRUPCION
                420 ;
90D6  76        430             HALT
                440 ;
                450 ;COMPROBAR TECLA BREAK
                460 ;
90D7  3E7F      470             LD      A,#7F
90D9  DBFE      480             IN      A,(#FE)
90DB  1F        490             RRA
90DC  38F5      500             JR      C,TSLP
                510 ;
                520 ;SI PULSADA ENTONCES SELECCIONAR IM 1, BORRAR EL RO-BUFFER
                530 ;Y FIN
90DE  CDDE90    540             CALL    DISINT
90E1  3AE190    550             LD      A,(ROLNTH)
90E4  ED44      560             NEG
90E6  4F        570             LD      C,A
90E7  CDBD90    580             CALL    ALTRBF
90EA  C9        590             RET
                600 ;
                610 ;LOS DATOS DE POSICION
                620 ;
90EB  4F01      630 TRGPOS      DEFW    #14F
90ED  6E01      640             DEFW    #16E
90EF  6F01      650             DEFW    #16F
90F1  7001      660             DEFW    #170
90F3  8D01      670             DEFW    #18D
90F5  8E01      680             DEFW    #18E
90F7  8F01      690             DEFW    #18F
                700
90F9  9001      710             DEFW    #190
90FB  9101      720             DEFW    #191
90FD  AE01      730             DEFW    #1AE
90FF  AF01      740             DEFW    #1AF
9101  B001      750             DEFW    #1B0
9103  CF01      760             DEFW    #1CF
```

```
                    770  ;
                    780  ;LOS DATOS DE CARACTER
                    790  ;
9105  00            800  TRGDAT  DEFB     0
9106  18            810          DEFB     24
9107  10            820          DEFB     16
9108  10            830          DEFB     16
9109  18            840          DEFB     24
910A  10            850          DEFB     16
910B  10            860          DEFB     16
910C  18            870          DEFB     24
                    880  ;
910D  00            890          DEFB     0
910E  00            900          DEFB     0
910F  00            910          DEFB     0
9110  03            920          DEFB     3
9111  0E            930          DEFB     14
9112  18            940          DEFB     24
9113  10            950          DEFB     16
9114  30            960          DEFB     48
                    970  ;
9115  10            980          DEFB     16
9116  10            990          DEFB     16
9117  FE           1000          DEFB     254
9118  93           1010          DEFB     147
9119  10           1020          DEFB     16
911A  18           1030          DEFB     24
911B  10           1040          DEFB     16
911C  7C           1050          DEFB     124
                   1060  ;
911D  00           1070          DEFB     0
911E  00           1080          DEFB     0
911F  00           1090          DEFB     0
9120  80           1100          DEFB     128
9121  E0           1110          DEFB     224
9122  30           1120          DEFB     48
9123  10           1130          DEFB     16
9124  18           1140          DEFB     24
                   1150  ;
9125  00           1160          DEFB     0
9126  00           1170          DEFB     0
9127  00           1180          DEFB     0
9128  92           1190          DEFB     146
9129  FF           1200          DEFB     255
912A  00           1210          DEFB     0
912B  00           1220          DEFB     0
912C  00           1230          DEFB     0
                   1240  ;
912D  21           1250          DEFB     33
912E  61           1260          DEFB     97
912F  43           1270          DEFB     67
9130  4A           1280          DEFB     74
9131  FF           1290          DEFB     255
9132  42           1300          DEFB     66
9133  43           1310          DEFB     67
9134  61           1320          DEFB     97
                   1330  ;
9135  D7           1340          DEFB     215
9136  11           1350          DEFB     17
9137  11           1360          DEFB     17
9138  00           1370          DEFB     0
9139  D7           1380          DEFB     215
913A  00           1390          DEFB     0
913B  11           1400          DEFB     17
913C  11           1410          DEFB     17
                   1420  ;
913D  08           1430          DEFB     8
```

```
913E  0C        1440          DEFB    12
913F  84        1450          DEFB    132
9140  A4        1460          DEFB    164
9141  FF        1470          DEFB    255
9142  84        1480          DEFB    132
9143  84        1490          DEFB    132
9144  0C        1500          DEFB    12
                1510  ;
9145  00        1520          DEFB    0
9146  00        1530          DEFB    0
9147  00        1540          DEFB    0
9148  92        1550          DEFB    146
9149  FE        1560          DEFB    254
914A  00        1570          DEFB    0
914B  00        1580          DEFB    0
914C  00        1590          DEFB    0
                1600  ;
914D  21        1610          DEFB    33
914E  30        1620          DEFB    48
914F  10        1630          DEFB    16
9150  18        1640          DEFB    24
9151  0E        1650          DEFB    14
9152  03        1660          DEFB    3
9153  00        1670          DEFB    0
9154  00        1680          DEFB    0
                1690  ;
9155  D7        1700          DEFB    215
9156  7C        1710          DEFB    124
9157  10        1720          DEFB    16
9158  18        1730          DEFB    24
9159  10        1740          DEFB    16
915A  93        1750          DEFB    147
915B  FE        1760          DEFB    254
915C  10        1770          DEFB    16
                1780  ;
915D  08        1790          DEFB    8
915E  18        1800          DEFB    24
915F  10        1810          DEFB    16
9160  30        1820          DEFB    48
```

Como recordará, hemos previsto, en la parte del procesador de impresión del gestor de interrupciones, una función de "OR-impresión" que funde un nuevo carácter con los contenidos actuales de una celdilla de pantalla en el archivo de pantalla. Ahora proporcionaré las rutinas de apoyo para esta función.

Siempre que vayamos a imprimir un carácter en la pantalla, necesitaremos saber si los contenidos de la celdilla de destino han de ser conservados por la OR-impresión o destruidos por una OVER-impresión. Por ejemplo, cuando dos caracteres en un juego entran en la misma celdilla, probablemente querremos fundirlos, mientras que si movemos un carácter desde una celdilla a la siguiente, arrastrando una celdilla en blanco detrás para borrar la imagen anterior, entonces seguramente no queremos OR-imprimir el espacio con la imagen vieja.

Para este fin necesitamos un mapa en la memoria, que denominaré OR-mapa, para llevar cuenta de qué celdillas están "ocupadas". Sólo se necesita un bit por celdilla; con un 1 indica "celdilla ocupada" y con un 0 indica "celdilla vacía".

Por consiguiente, tenemos cuatro octetos para cada una de las 24 líneas de pantalla, que constituyen un OR-mapa de 96 octetos. No es sorprendente, por

tanto, que haya etiquetado el principio de esto como ORMAP. Para reservar el espacio que se precisa, necesitamos la línea:

```
ORMAP   DEFS   96
```

Habrá que borrar el OR-mapa con regularidad, así es que, antes de continuar, veamos una rutina para llenarla con ceros, denominada CLOR:

```
                 10 ;RUTINA PARA BORRAR EL OR-MAP
                 20 ;
872A  212A87     30 CLOR     LD       HL,ORMAP
872D  015F00     40          LD       BC,95
8730  70         50          LD       (HL),B
8731  54         60          LD       D,H
8732  5D         70          LD       E,L
8733  13         80          INC      DE
8734  EDB0       90          LDIR
8736  C9        100          RET
```

Cada vez que nos preparamos para enviar un carácter al buffer de impresión y deseamos que sea considerado para OR-impresión con los caracteres existentes y futuros, debemos acceder el bit correspondiente a la celdilla de destino en el OR-mapa:

Si este bit está a uno, ya hay algo en esa celdilla, y seleccionamos OR-impresión poniendo a uno el bit 7 del octeto de atributo, ATT. Si el bit es cero, para lo que nos interesa la celdilla está ahora vacía, y como hemos puesto a uno el bit de OR-mapa para indicar que estaba ocupada, ponemos a cero el bit 7 de ATT para seleccionar OVER-impresión. Entonces se envía el carácter al buffer de impresión empleando HIPRINT (o su propia rutina) de la forma usual.

La siguiente rutina, ORCHK, lleva a cabo el proceso descrito arriba, empleando el puntero ATCC, que contiene la situación del octeto de atributo actual, como un medio para localizar el bit correcto del OR-mapa. No se necesita ningún valor de entrada, y los comentarios del listado proporcionan suficiente explicación:

```
                 10 ;RUTINA PARA DECIDIR SI OR-PRINT EN LA CELDILLA DE
                 20 ;CARACTER ACTUAL
                 30 ;
8A91  2A918A     40 ORCHK    LD       HL,(ATCC)
                 50 ;
                 60 ;TOMAR DIRECCION DE ATRIBUTOS Y DIVIDIR SUS 10 BITS
                 70 ;DE MENOR PESO POR 8
                 80 ;
8A94  7D         90          LD       A,L
8A95  CB1C      100          RR       H
8A97  1F        110          RRA
8A98  CB1C      120          RR       H
8A9A  1F        130          RRA
8A9B  CB3F      140          SRL      A
                150 ;
                160 ;PONER EL RESULTADO EN DE
                170 ;
8A9D  5F        180          LD       E,A
8A9E  1600      190          LD       D,0
8AA0  7D        200          LD       A,L
```

```
                    210 ;
                    220 ;SUMAR DIRECCION BASE DE TABLA
                    230 ;
8AA1  21A18A        240           LD        HL,ORMAP
8AA4  19            250           ADD       HL,DE
                    260 ;ROTAR UNA MASCARA HASTA QUE '1' ESTE SOBRE EL BIT REQUERIDO

8AA5  E607          270           AND       7
8AA7  47            280           LD        B,A
8AA8  04            290           INC       B
8AA9  3E01          300           LD        A,1
8AAB  0F            310 NXTROT    RRCA
8AAC  10FD          320           DJNZ      NXTROT
                    330 ;
                    340 ;PONER LA MASCARA EN C
                    350 ;
8AAE  4F            360           LD        C,A
                    370 ;
                    380 ;COMPROBAR SI ESTA CELDILLA YA ESTA OCUPADA
                    390 ;
8AAF  A6            400           AND       (HL)
8AB0  11B08A        410           LD        DE,ATT
8AB3  1A            420           LD        A,(DE)
                    430 ;
                    440 ;SI NO SELECCIONAR OVER-PRINT (SOBREIMPRIMIR)
                    450 ;
8AB4  CBBF          460           RES       7,A
8AB6  2802          470           JR        Z,NOTOR
                    480 ;
                    490 ;SI NO SELECCIONAR OR-PRINT
                    500 ;
8AB8  CBFF          510           SET       7,A
8ABA  12            520 NOTOR     LD        (DE),A
                    530 ;
                    540 ;AHORA PONER EL BIT EN OR-MAP
                    550 ;PARA INDICAR "CELDILLA UTILIZADA"
                    560 ;
8ABB  79            570           LD        A,C
8ABC  B6            580           OR        (HL)
8ABD  77            590           LD        (HL),A
8ABE  C9            600           RET
```

12

Animación perfecta de "sprites"

Por fin estamos preparados para empezar el desarrollo de la generación de *sprites*, de las rutinas de impresión y control para la obtención de gráficos de *pixel* sin temblores junto con el procesador de impresión controlado por interrupciones.

Como recordará, definimos un *sprite* como una imagen contenida en un bloque movible de objeto (de ahí su otro nombre MOB), de caracteres adyacentes en la pantalla. Este bloque siempre será rectangular, y la imagen puede ser de cualquier tamaño, desde un carácter de 1 × 1.

La forma más obvia para mover un *sprite* desde una posición a otra es borrar la vieja imagen imprimiendo espacios encima, y posteriormente imprimir la nueva imagen en la nueva posición. Si las dos imágenes son mutuamente exclusivas (es decir, no se superponen), ésta es una técnica perfectamente aceptable.

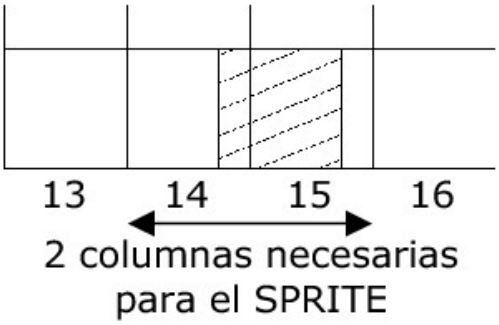
Sin embargo, como suele ocurrir, si sólo hemos movido el *sprite* unos pocos *pixels*, parece que no merece la pena imprimir un espacio en una celdilla común a ambas imágenes, sino reemplazarla con parte de la nueva imagen casi instantáneamente.

Para evitar esta pérdida de tiempo, utilizaremos una aproximación mejor para la animación del *sprite*. Cada forma estará rodeada por una estrecha región de *pixels* en blanco para que, según vayamos moviéndonos de una posición a la siguiente, estos blancos que se arrastran vayan borrando cualquier parte de la imagen vieja que no haya sido borrada por la impresión de la nueva imagen.

De esta forma, se conseguirá la animación con sólo una operación de impresión en lugar de las dos que necesitaba la técnica anterior, y el número de caracteres que necesitamos imprimir será la mitad. Esto es una ventaja considerable teniendo en cuenta que el procesador de impresión estándar sólo puede imprimir 40 caracteres por cuadro de TV.

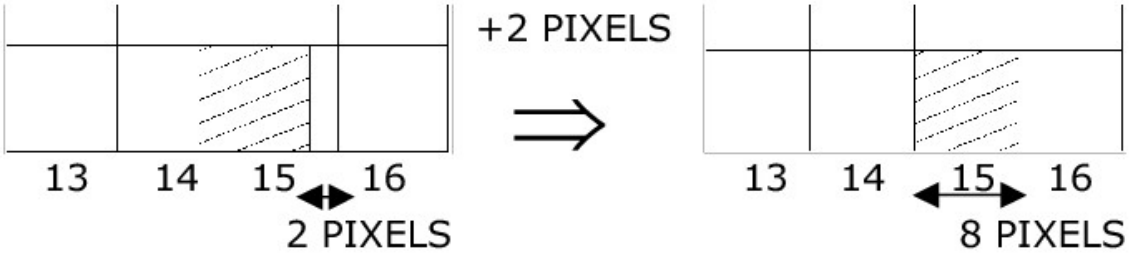
Durante el resto de esta discusión me referiré a una forma de carácter que se mueve con pasos de un *pixel*.

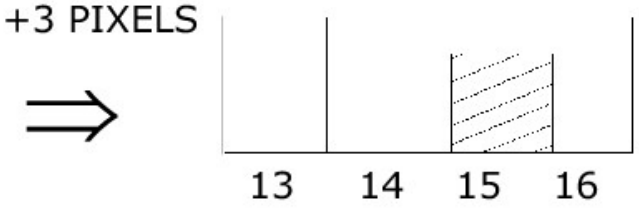
Si nuestra forma puede contenerse dentro de (m) columnas (es decir, es menor o igual que (m × 8) *pixels* de ancho), vemos que puede, a lo sumo, ocupar (m + 1) columnas separadas. De ahí que el *sprite* deba tener por lo menos (m + 1) columnas de ancho. Por ejemplo, nuestra forma de 1 × 1 puede ocupar las columnas 14 y 15 de la forma siguiente:



Si, además, imponemos la restricción de que la forma nunca pueda ocupar más de (m + 1) columnas diferentes al moverse de una posición a la siguiente, vemos que tanto la imagen vieja como la nueva pueden caber en un área de (m + 1) columnas de ancho.

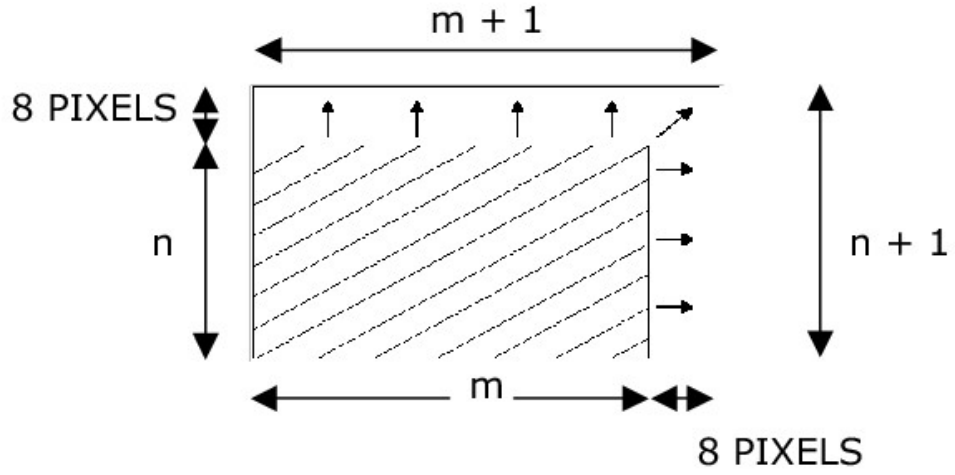
Por ejemplo, debemos restringir el movimiento de nuestra forma 1 × 1 para que, si la vieja imagen ocupa la columna 14, la nueva no ocupe la 16. De ahí, si nuestra forma de las columnas 14 y 15 tiene dos columnas de *pixels* en blanco a su derecha en la columna 15, entonces no debemos permitirle que se mueva más de dos *pixels* a la derecha:



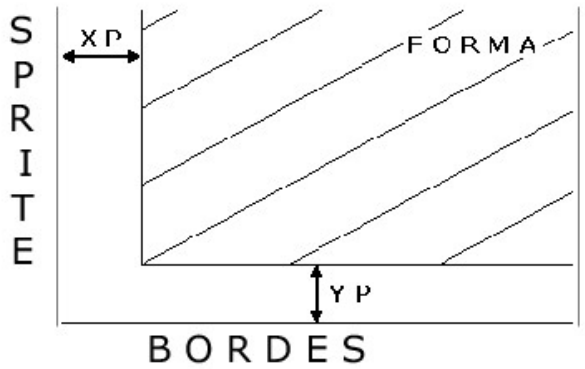


Esta restricción es, en la práctica, de fácil aplicación y tiene como resultado que podemos animar una forma contenida en (m × n) caracteres imprimiendo continuamente una serie de imágenes de *sprite* de dimensiones fijas (m + 1) por (n + 1). Por tanto, para mover nuestra forma de 1 × 1 por la pantalla necesitaremos una celdilla de *sprite* de 2 × 2.

Ahora, si se puede imaginar nuestra forma de m × n flotando por su *sprite* de (m + 1) × (n + 1), verá que, gracias a las 8 diferentes posiciones horizontales y las 8 diferentes posiciones verticales que la forma puede tener, el *sprite* resultante puede ser cualquiera de las 8 × 8 = 64 imágenes posibles.



Si la forma es movible un *pixel* cada vez en cada una de las direcciones de X e Y, cada una de estas 64 estructuras posibles necesitarán ser impresas en algún momento. Antes de seguir con esta discusión, sería prudente definir algunas variables. Llamaremos XP a la distancia horizontal (en *pixels*) de la forma desde el borde izquierdo de su *sprite* e YP a la distancia vertical de la forma desde el borde inferior del *sprite*; en consecuencia:



Ahora tenemos una elección directa sobre el método de generación de las estructuras de *sprite*. Podemos o almacenar sólo una de las "imágenes" del *sprite* en la memoria y manipular la forma dentro de ella para obtener bit por bit la imagen deseada para impresión, o podemos almacenar las imágenes en una tabla algo mayor en RAM, utilizando una técnica de índices para "entresacar" la imagen deseada sin más.

La experiencia ha demostrado que, si bien la primera técnica ocupa tan poco

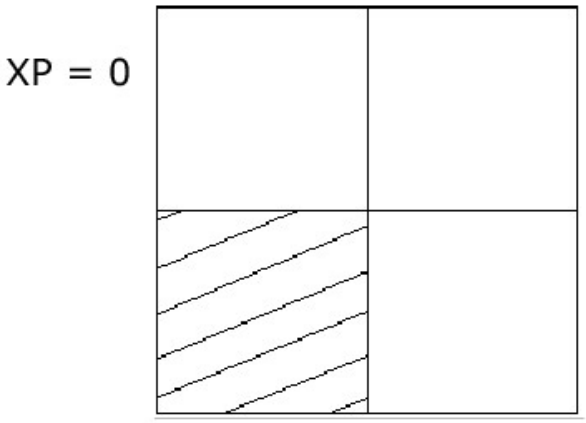
como un octavo de la cantidad de memoria disponible para almacenar la imagen y, por tanto, es útil si el ahorro de RAM es muy importante, tarda mucho en realizar las operaciones de desplazamiento de los bits de XP necesarias para cada uno de los (M + 1) × N × 8 octetos afectados, por lo que es preferible, siempre que sea posible, utilizar la última técnica.

Aunque el *sprite* puede ser una de las 64 estructuras posibles, la situación no es tan seria como parece. No necesitamos almacenar en memoria 64 imágenes diferentes que correspondan a los diferentes valores de YP de una imagen que corresponda a un XP dado. De esta forma sólo necesitamos almacenar (como máximo) 8 imágenes, una para cada valor de XP.

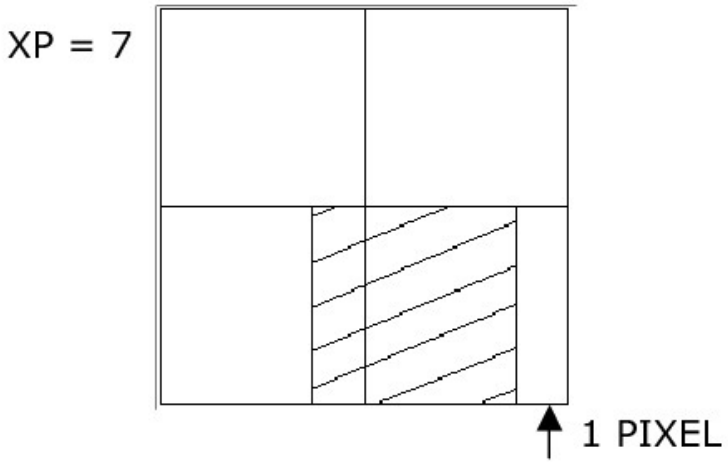
Almacenaremos las imágenes de una en una, cada una de ellas en una columna de cada vez, trabajando desde la parte superior a la inferior y de izquierda a derecha. De esta forma, el orden del almacenamiento de una imagen de nuestra forma de 1 × 1 en su *sprite* de 2 × 2 será:

0	2
1	3

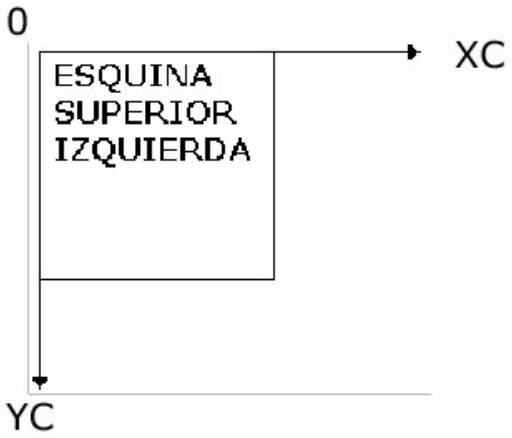
Cada imagen se almacenará suponiendo YP = 0, es decir, la forma se encontrará en el borde inferior del *sprite* y la línea superior de la imagen estará en blanco. Las imágenes se almacenan en orden creciente de XP; así para nuestra forma 1 × 1 la primera imagen será:



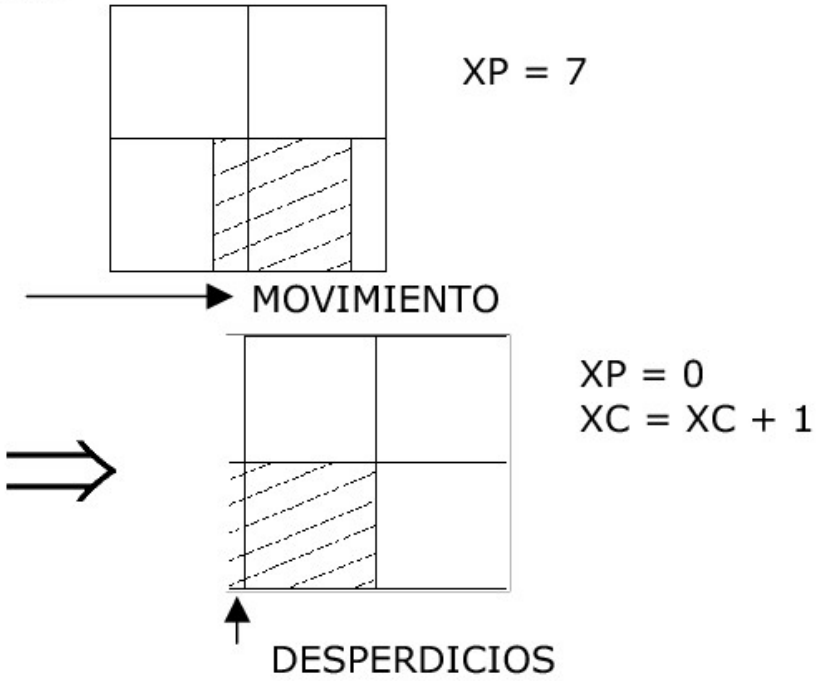
y la última:



Según se desliza nuestra forma por la pantalla, las rutinas de animación van haciendo ciclos a través de la secuencia de imágenes a medida que XP va variando de 0 a 7. Ahora definiré la posición del *sprite* en pantalla referida a la posición de la esquina superior izquierda de la imagen (XC, YC), donde XC se mide hacia la derecha desde cero e YC hacia abajo desde cero y C representa las coordenadas de la celdilla en lugar de la P de *pixels*. Así tenemos:

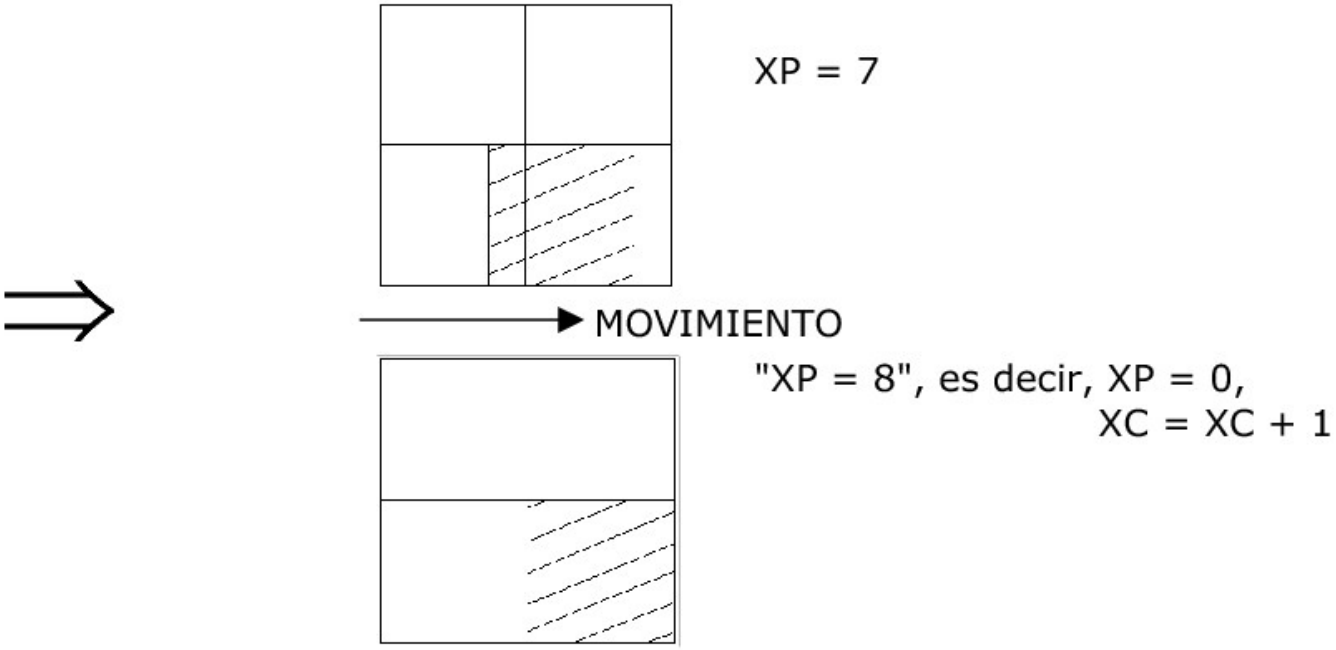


Si la forma está moviéndose hacia la izquierda, estaremos moviéndonos hacia atrás a través de las imágenes. Cuando XP alcanza 0, la columna de más a la derecha estará completamente en blanco y podremos decrementar XC y cambiar XP a 7 sin dejar huella alguna de la vieja imagen en su columna de más a la derecha, que está fuera del área del nuevo *sprite*. Sin embargo, si se está moviendo hacia la derecha, no bastará con cambiar de la imagen 7 a la imagen 0, incrementando XC, ya que esto dejaría una columna de *pixels* de la vieja imagen en la columna que se encuentra inmediatamente a la izquierda de la nueva, de esta forma:

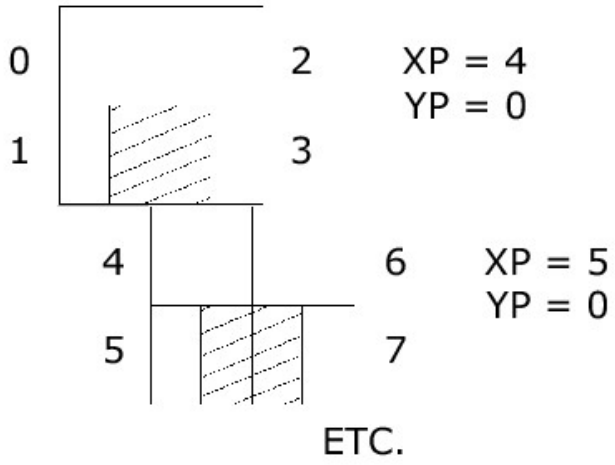


Para solucionar este problema necesitamos simular una "novena" imagen, y esto se hará incluyendo una columna extra de celdillas en blanco inmedia-

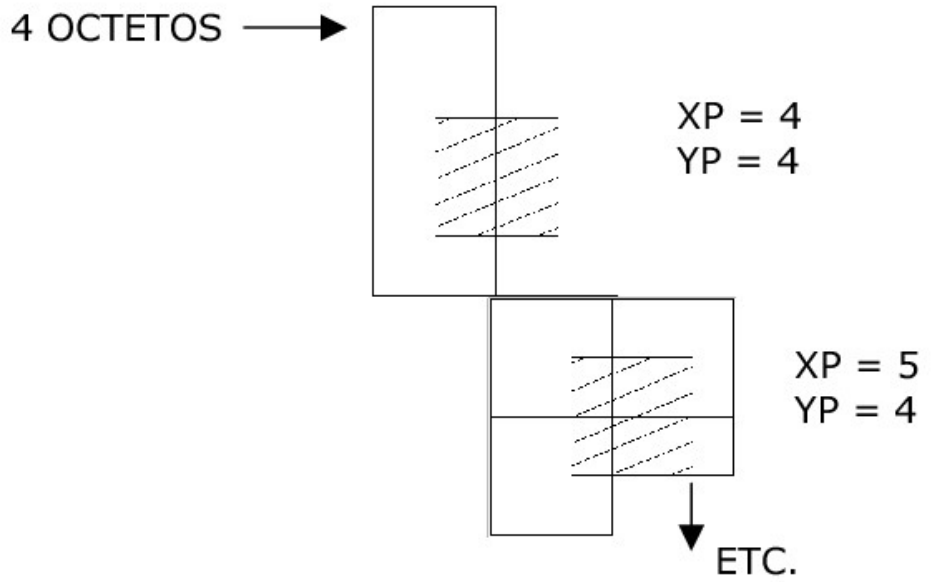
tamente ANTES de la imagen 0. A continuación moveremos desde XP = 7, haciendo apuntar el generador de *sprite* a esta columna en blanco e imaginándonos que XP = 8. Para el resto de las rutinas, XP será 0 y XC será incrementado. Entonces tendremos:



Las diversas imágenes correspondientes a cada valor de YP se producirán haciendo apuntar el generador de *sprite* a la YP-ava fila de la XP-ava imagen, contando hacia abajo desde la fila cero. La razón para almacenar las imágenes columna por columna puede verse ahora. Miremos la disposición en memoria de la imagen 4 de nuestra forma de 1 × 1. Vemos que después de la esquina inferior izquierda de esta imagen viene la superior derecha, a continuación la esquina inferior derecha y seguidamente la superior izquierda de la imagen siguiente. Así:



Al apuntar a la fila 4 de la celdilla 0, desplazamos efectivamente cuatro filas hacia arriba todos los octetos de la imagen y el resultado es una imagen centralizada con XP = 4, YP = 4.

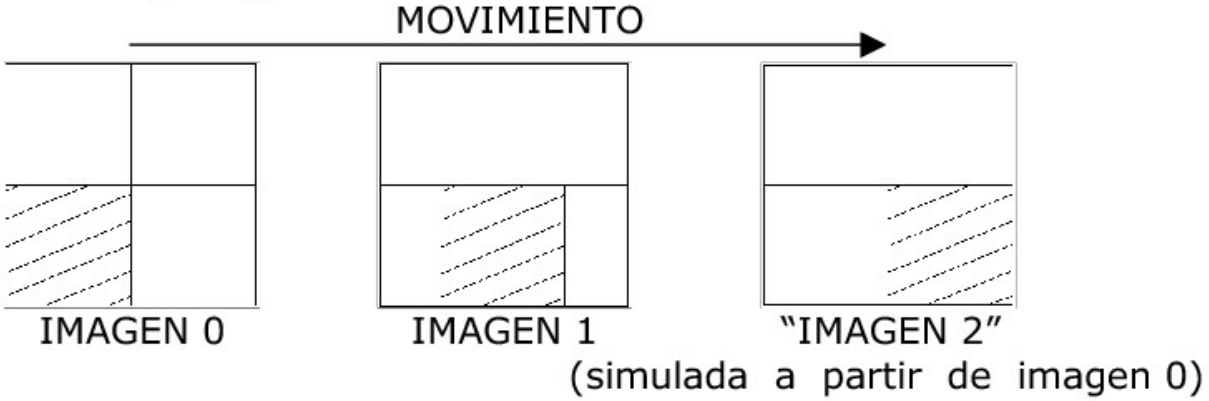


De manera similar a XP = 8, simularemos YP = 8 haciendo apuntar el generador de *sprite* a la fila 0 de la celdilla 1 cuando se esté moviendo hacia arriba desde YP = 7. Observe que YP e YC están aumentando en direcciones opuestas, por tanto, cuando YP alcanza 7 dejamos YP = 0 Y DECREMENTAMOS YC.

Debo decir en este momento que, por supuesto, no es necesario en absoluto almacenar 8 imágenes separadas si no necesitamos un movimiento en la dirección X de un *pixel* por ciclo. Después de todo, hay poca necesidad de almacenar 8 imágenes si efectuamos movimiento sólo en pasos de dos *pixels*, puesto que sólo habría 4 valores alcanzables de XP, y, por tanto, sólo se utilizarían 4 de las imágenes. De ello resulta que tenemos elección entre almacenar 8, 4, 2 y 1 imágenes separadas.

Con 8 imágenes, es posible cualquier velocidad horizontal hasta de 8 *pixels* por movimiento. Con 4 imágenes, tenemos un paso de dos *pixels* entre imágenes y por tanto se permiten unas velocidades de 0, 2, 4, 6 y 8 *pixels*, pero XP siempre tiene que ser par. Con dos imágenes, tenemos un paso de 4 *pixels* entre imágenes, y, por tanto, es posible una velocidad de 0, 4 u 8 *pixels*, siendo XP múltiplo de cuatro.

Por ejemplo, si representamos nuestra forma de 1 × 1 en dos imágenes separadas (teniendo en cuenta que la forma está siempre en la esquina inferior izquierda de la imagen 0), tenemos la secuencia:



Obviamente con una imagen sólo tenemos el caso sencillo del movimiento de un carácter cada vez.

Ahora es posible calcular la cantidad de memoria que se necesita para almacenar las imágenes de cualquier *sprite*. Tome una forma de carácter de (m × n) y enciérrela en un *sprite* de carácter de (m + 1) × (n + 1). Al producir (a) imágenes de este *sprite*, y teniendo en cuenta que cada celdilla necesita 8 octetos y cada serie de imágenes requiere una columna anterior en blanco, tenemos:

$$\text{Memoria necesitada} = 8 \times a \times (m + 1) \times (n + 1) + 8 \times (n + 1) =$$
$$= 8\,(a(m + 1) + 1)(n + 1) \text{ octetos}$$

Por tanto, para una forma de un ancho de tres columnas por dos líneas de profundidad, definida por cuatro imágenes, tenemos m = 3, n = 2, a = 4 y:

$$\text{Memoria necesitada} = 8(4(3 + 1) + 1)(2 + 1)$$
$$= 408 \text{ octetos}$$

Además de esto, y suponiendo que todas las imágenes para los *sprites* actualmente en uso están almacenadas consecutivamente en la memoria, debemos incluir 8 octetos cero después de la última imagen del último *sprite* para permitir la ocupación de la memoria cuando YP = 8 y esté siendo utilizada la última imagen. En este caso, aquellos 8 octetos representarán la esquina inferior derecha del *sprite*.

Como ejemplo, supongamos que tenemos dos *sprites* en uso, ambos de la forma de 3 × 2 como en el cálculo de memoria previo. Si uno es un avión y el otro un tren, entonces una secuencia de reserva de memoria adecuada sería:

```
PLNSPC      DEFS    408
TRNSPC      DEFS    408
            DEFW    0, 0, 0, 0    ;8  OCTETOS
```

(utilizando etiquetas con sufijos SPC de eSPaCio).

Discutamos ahora el método de control y seguimiento de la posición de los *sprites*. Para cada *sprite* utilizaremos una tabla de 17 octetos de información que llamaremos "datos de movimiento del *sprite*". Esta tabla indicará a nuestra rutina de animación a qué velocidad se mueve el *sprite* a lo largo de X e Y, la situación del *sprite* en cualquier momento, la situación de los datos de imagen, las dimensiones del *sprite*, el color a imprimir, y así sucesivamente. Siempre que deseemos mover un *sprite*, haremos apuntar el registro de indice IY al principio de sus datos de movimiento y seguidamente llamaremos a la rutina de animación que hará todo el trabajo que queda, refiriéndose a la tabla de datos de movimiento.

Antes de continuar con un desglose completo de esta tabla, redefiniremos primero XP para que sea el número de la imagen utilizada actualmente por el generador de *sprites*. Por tanto, si hay cuatro imágenes, XP reciclará continua-

mente por los valores (0, 1, 2, 3) según el *sprite* se mueve a través de la pantalla. Es decir, que XP se incrementa continuamente y reducido módulo (número de imágenes diferentes).

Cuando hay ocho imágenes, XP tendrá el mismo valor que antes, o sea el número de *pixels* existentes entre la forma y el borde izquierdo del *sprite*. De no ser así, necesitará multiplicar XP por el paso entre imágenes (2, 4 u 8 *pixels*) para encontrar esta distancia. Vale la pena considerar esta conversión al escribir rutinas de detección de colisiones y cosas por el estilo.

He aquí, pues, un listado de los 18 octetos de datos de movimiento para cada *sprite*, seguido por algunas notas explicativas:

Dirección	Contenidos
IY	XP = Número de imagen actual (<8).
IY + 1	VX = Indice de cambio de XP (positivo o negativo).
IY + 2	N = Número de imágenes = (valor máx. de XP) + 1.
IY + 3	XC = Posición de la columna de *sprite* más a la izquierda.
IY + 4	YP (0-7)
IY + 5	VY = Indice de cambio de YP (positivo o negativo).
IY + 6	YC = Posición de línea de *sprite* más superior.
IY + 7	LO } Dirección de fila 0, celdilla 0 de imagen 0.
IY + 8	HI }
IY + 9	Cuenta de ciclos (véase notas).
IY + 10	Período de ciclos (véase notas).
IY + 11	Anchura del *sprite* expandido.
IY + 12	Altura del *sprite* expandido.
IY + 13	LO } Longitud de una imagen = anchura × altura × 8.
IY + 14	HI }
IY + 15	Octeto de atributo y bandera para OR-impresión.
IY + 16	Máscara de atributo.

La "cuenta de ciclos" y el "período de ciclos" de (IY + 9) e (lY + 10) se utilizarán para aumentar la versatilidad de nuestra rutina de control de *sprite*. En cuanto se llama a la rutina, se decrementará la cuenta de ciclos. Si la cuenta no es cero, se hará un retorno inmediato. En caso contrario, se volverá a cargar la cuenta de ciclos con el "período de ciclo" constante que controla indirectamente la frecuencia de movimiento del *sprite*, y el *sprite* se moverá y se imprimirá. Veremos más detalles más adelante.

En vista de la cantidad de memoria que se necesita para almacenar las imágenes de un *sprite* plenamente operacional, sería aconsejable almacenar las estructuras de bits de las varias formas que necesita un programa de una manera lo más compacta posible, y posteriormente expandirlo a unas imágenes de *sprite* preparadas para ser utilizadas como y cuando se desee. Necesitaremos algunas

rutinas de utilidad para hacer esto, y he preferido proporcionarle un sistema de expansión de *sprite* en dos etapas.

La primera rutina, llamada PADOUT, copiará la forma inactiva desde su área de almacenamiento al "área de imagen de *sprite*", añadiendo una columna de blancos a la derecha, una línea de blancos encima de ella y la columna en blanco reglamentaria precediendo a la imagen 0. La segunda rutina, SPREX trabajará con una copia de la imagen 0, fila por fila, utilizando operaciones de desplazamiento y rotación para generar las otras imágenes.

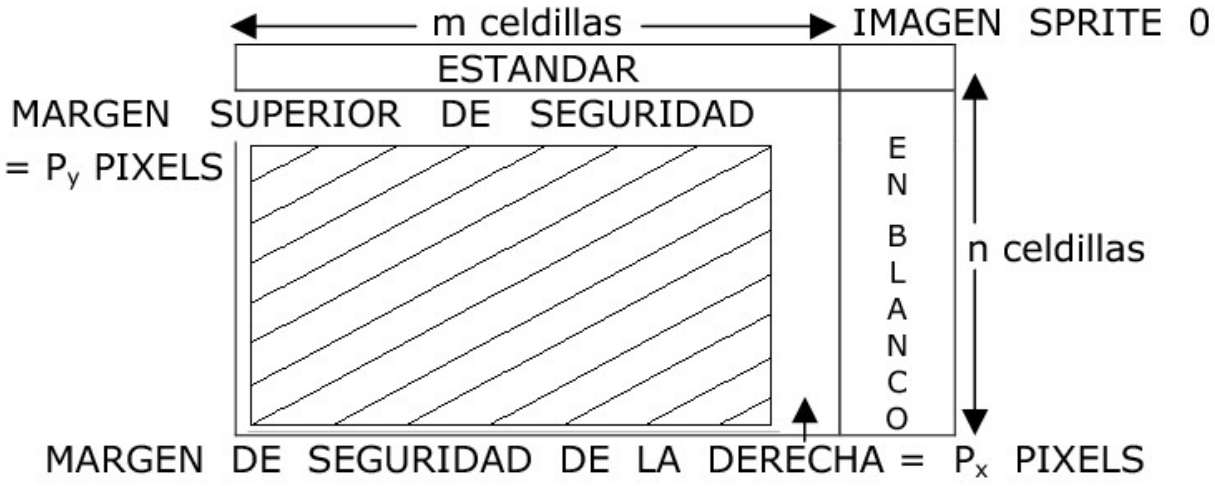
Los datos del *sprite* "pelado" deben almacenarse en una columna cada vez, una fila cada vez, trabajando de izquierda a derecha y desde la parte superior a la inferior, de la misma manera en la que se almacenan las imágenes.

Con referencia a la parte anterior de este capítulo, recordará que el movimiento de la forma de ancho (m) columnas debe restringirse para que no ocupe más de (m + 1) columnas al moverse de una posición a la siguiente. Entonces no estudié tal restricción por ser "de fácil aplicación". Ahora es el momento de explicar cómo se hace.

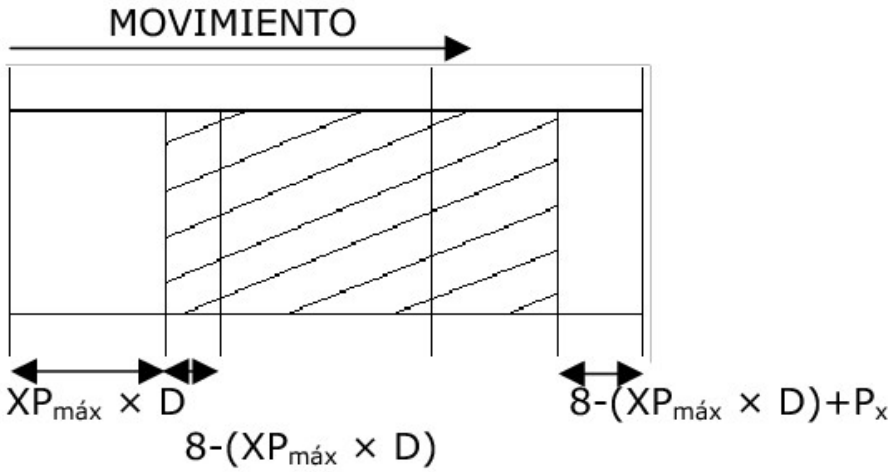
La forma debe incluir una región de blancos en forma de L como bordes superior y derecho de entre 0 y 7 *pixels* de ancho. La anchura de esta región de seguridad está determinada como sigue.

Suponga que la imagen de valor más alto utilizada por el *sprite* es la que corresponde a $XP_{máx}$ y que la velocidad absoluta de cambio de XP es VX por movimiento: sea D el paso en los *pixels* entre las posiciones de la forma en imágenes sucesivas de *sprite*. Entonces el número de *pixels* que se mueven cada vez es D × VX. Vemos que la distancia de la forma desde el borde izquierdo del *sprite* es (XP × D) y, por tanto, como mínimo hay (8 – ($XP_{máx}$ × D)) *pixels* de la forma en la columna de más a la izquierda.

Si el margen de seguridad derecho del *sprite* tiene un ancho de P_x *pixels*, habrá como mínimo (8 – ($XP_{máx}$ × D) + P_x) *pixels* en blanco en la columna de más a la derecha del *sprite*. Algunos diagramas pueden ayudarle:



Arriba vemos cómo una forma tiene que "encogerse" para permitir que quepa un espacio en blanco a su alrededor. Cuando se utiliza la imagen $XP_{máx}$ tenemos:



y vemos que, si se obedece nuestra regla de que la forma de las m columnas no ocupe más de m + 1 durante el movimiento, sólo podemos permitir $(8 - (XP_{máx} \times D) + P_x)$ *pixels* de movimiento a la derecha. Ahora D × VX es la distancia que se mueve a la derecha; por tanto:

$$8 - (XP_{máx} \times D) + P_x = D \times VX \Rightarrow P_x + 8 = D\,(VX + XP_{máx})$$

y finalmente:

$$\boxed{P_x = D\,(VX + XP_{máx}) - 8}$$

Después de tanta teoría, pienso que un ejemplo práctico ayudaría a aclarar la situación.

Supongamos que estamos dando animación a un coche, que en un momento dado estará moviéndose en pasos de un *pixel*, pero que de momento está moviéndose dos *pixels* cada vez. Para ello necesitaré un juego completo de 8 imágenes, lo que significa que "el paso entre imágenes" es de un *pixel*, es decir:

$$D = 1$$

para mover el coche con pasos de dos *pixels*, tenemos

$$VX = 2/D = 2/1 = 2$$

Ahora con esta velocidad constante estaremos reciclando o bien a través de las imágenes "impares", donde:

$XP = [1, 3, 5, 7]$ dando $XP_{máx} = 7$

o bien por las imágenes "pares", donde:

$XP = [0, 2, 4, 6]$ dando $XP_{máx} = 6$

Por tanto, tenemos para el ciclo "impar":

$P_x = 1\,(2 + 7) - 8 = 1$ *pixel*

y para el ciclo par:

$P_x = 1\,(2 + 6) - 8 = 0$

Esto nos da el resultado de que, si podemos restringir XP a los múltiplos de VX, entonces no se necesitará margen derecho, pero, si estamos forzados a utilizar las imágenes para valores "impares" de XP, la columna de *pixels* de más a la derecha de la forma debe estar en blanco. Por tanto, la forma del coche debe incluir un margen de seguridad a la derecha en blanco de un ancho de un *pixel*.

Se puede aplicar un análisis similar para determinar el ancho necesario del margen superior de seguridad. Por tanto, no repetiré todos los detalles engorrosos. Tomando la velocidad vertical absoluta VY (0-8 *pixels* por movimiento) y el valor máximo de YP, $YP_{máx}$ (siempre menor que 8) encontramos que el espesor del margen superior, P_y *pixels*, está dado por

$$P_y = VY + YP_{máx} - 8$$

Supongamos, como ejemplo complementario, que deseamos diseñar, dentro de un *sprite* de 4 × 3, y, por tanto, de una forma de 3 × 2, un avión de combate capaz de moverse hasta cuatro *pixels* por movimiento en la dirección X y hasta 3 *pixels* en la dirección Y. ¿Qué cantidad de la forma 3 × 2 tenemos libre para diseñarlo?

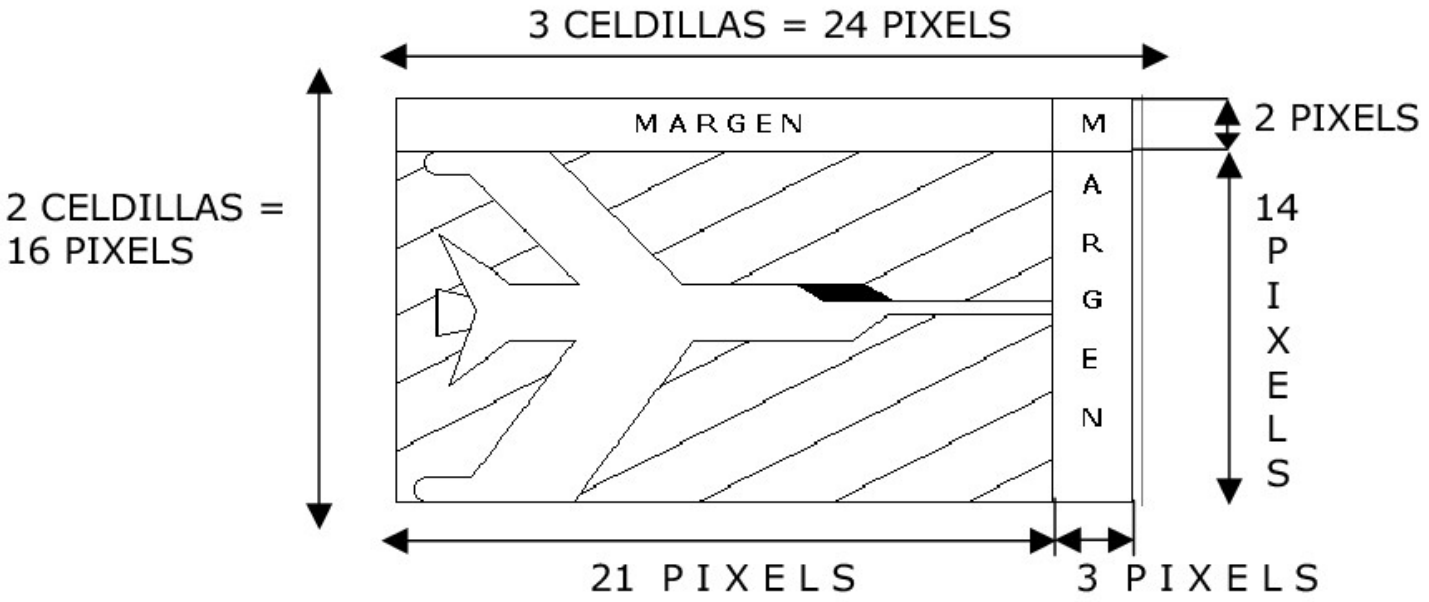
Puesto que las velocidades pueden ser tan bajas como hasta de un *pixel* por cuadro, necesitaremos de nuevo 8 imágenes, y un paso D = 1 *pixel*. Como máximo tenemos VX = 4 y VY = 3. Es bastante posible que en algún momento podamos alcanzar XP = 7 e YP = 7, (si tiene dudas, utilice los valores máximos disponibles, es decir, YP = 7 y XP = [número de imágenes] – 1). Esto nos da:

$P_x = D\,(VX + XP_{máx}) - 8 = 1\,(4 + 7) - 8 = 3$

y

$P_y = VX - YP_{máx} - 8 = 3 + 7 - 8 = 2$

Así es que necesitamos un margen superior de dos *pixels* y un margen derecho de 3. Por tanto, el área que nos queda libre para diseñar es (3 × 8) – 3 = 21 *pixels* de ancho, y (2 × 8) – 2 = 14 *pixels* de alto, así:



Ahora proporcionaré la rutina PADOUT citada previamente que expande el "*sprite* estándar" o los datos "de la forma", almacenados en su forma compacta, hasta un "*sprite* expandido" que se forma como la imagen 0 en nuestra área de imagen de *sprite* reservado anteriormente:

Tomando nota de los requisitos de entrada del listado en ensamblador, vemos que, para la forma de 3 × 2 del ejemplo superior, una rutina rápida adecuada para llamar a PADOUT sería como sigue:

```
LD   HL,TELSPC      ;AREA DE IMAGEN
LD   DE,TELDAT      ;DATOS DE SPRITE
LD   BC,#403        ;B=ANCHURA, C=ALTURA
LD   IY,TELMTN      ;DATOS MOVIMIENTO
CALL PADOUT
LD   D,1            ;PASO ENTRE IMAGENES
CALL SPREX          ;DE UN PIXEL
```

Observe que llamamos a PADOUT con IY apuntando a los datos de movimiento para nuestra forma. Esto es debido a que la rutina inicializa los valores (IY + 7), (IY + 8), (IY + 11), (IY + 12), (IY + 13) e (IY + 14) (véase tabla anterior de los detalles de éstos). ¡Bueno, adelante!

```
 10 ;PARA RELLENAR EL SPRITE VACIO Y DAR ENTRADA A LOS
 20 ;DATOS DE MOVIMIENTO DEL SPRITE
 30 ;
 40 ;ENTRADAS : DE=DIRECCION DE DATOS DE SPRITE
 50 ;HL=AREA DE ALMACENAMIENTO DE IMAGEN
 60 ;B=ANCHURA DE COLUMNA DEL SPRITE ESTANDAR
 70 ;C=PROFUNDIDAD DE LINEA DEL SPRITE ESTANDAR
 80 ;IY=DIRECCION DE DATOS DE MOVIMIENTO DE SPRITE
 90 ;SALIDA : HL= DIRECCION DE IMAGE 0
100 ;B= ANCHURA DE COLUMNA DEL SPRITE EXPANDIDO
110 ;C=PROFUNDIDAD DE FILA DEL SPRITE EXPANDIDO
120 ;NOTA HL Y BC SE UTILIZARAN POR "SPREX"
130 ;AF' SE DESTRUYE
140 ;
150 ;CALCULAR NO. DE FILAS DEL SPRITE EXPANDIDO
```

```
                    160;
                    170;
8F7D  0C            180PADOUT   INC       C
8F7E  FD710C        190         LD        (IY+12),C
8F81  CB21          200         SLA       C
8F83  CB21          210         SLA       C
8F85  CB21          220         SLA       C
                    230;
                    240;ALMACENAR BC PARA SPREX
                    250;
8F87  78            260         LD        A,B
8F88  04            270         INC       B
8F89  FD700B        280         LD        (IY+11),B
8F8C  C5            290         PUSH      BC
                    300;
                    310;A CUENTA LAS COLUMNAS
                    320;
8F8D  08            330         EX        AF,AF'
                    340;
                    350;ALMACENAR DIRECCION DE DATOS
                    360;
8F8E  D5            370         PUSH      DE
                    380;
                    390;COMENZAR CON UNA COLUMNA DE BLANCOS
                    400;
8F8F  0600          410         LD        B,0
8F91  C5            420         PUSH      BC
8F92  70            430         LD        (HL),B
8F93  0D            440         DEC       C
8F94  CDD58F        450         CALL      CL
8F97  C1            460         POP       BC
8F98  79            470         LD        A,C
8F99  D608          480         SUB       8
8F9B  4F            490         LD        C,A
                    500;
                    510;BC=NO. DE FILAS EN LA COLUMNA ESTANDAR
                    520;
8F9C  E1            530         POP       HL
                    540;
                    550;ALMACENAR COMIENZO DE IMAGE 0
                    560;
8F9D  D5            570         PUSH      DE
8F9E  C5            580         PUSH      BC
8F9F  08            590         EX        AF,AF'
                    600;
                    610;INSERTAR UN ESPACIO EN LA LINEA DE ARRIBA
                    620;
8FA0  E5            630NXSCOL   PUSH      HL
8FA1  EB            640         EX        DE,HL
8FA2  3600          650         LD        (HL),0
8FA4  0E07          660         LD        C,7
8FA6  CDD58F        670         CALL      CL
8FA9  E1            680         POP       HL
                    690;
                    700;LLENAR EL RESTO DE LA COLUMNA CON DATOS DE SPRITE
                    710;
8FAA  C1            720         POP       BC
8FAB  C5            730         PUSH      BC
8FAC  EDB0          740         LDIR
                    750;
                    760;HACER LA SIGUIENTE COLUMNA ESTANDAR
                    770;
8FAE  3D            780         DEC       A
8FAF  20EF          790         JR        NZ,NXSCOL
                    800;
                    810;EXPANDIR CON UNA COLUMNA BLANCA MAS A LA DER.
                    820;
```

```
8FB1  C1          830          POP       BC
8FB2  79          840          LD        A,C
8FB3  C607        850          ADD       A,7
8FB5  4F          860          LD        C,A
8FB6  EB          870          EX        DE,HL
8FB7  70          880          LD        (HL),B
8FB8  CDD58F      890          CALL      CL
                  900;
                  910;RECUPERAR DIRECCION DE IMAGE 0
                  920;
8FBB  E1          930          POP       HL
                  940;
                  950;Y VALOR EN DE, PARA SPREX
                  960;
8FBC  D1          970          POP       DE
8FBD  D5          980          PUSH      DE
8FBE  E5          990          PUSH      HL
                 1000;
                 1010;CALCULAR NO. DE BYTES EN UNA IMAGEN Y ALMACENARLA EN
                 1020;DATOS DE MOVIMIENTO DE SPRITE
                 1030;
8FBF  60         1040          LD        H,B
8FC0  68         1050          LD        L,B
8FC1  42         1060          LD        B,D
8FC2  54         1070          LD        D,H
8FC3  19         1080MUL1      ADD       HL,DE
8FC4  10FD       1090          DJNZ      MUL1
8FC6  FD750D     1100          LD        (IY+13),L
8FC9  FD740E     1110          LD        (IY+14),H
8FCC  E1         1120          POP       HL
8FCD  C1         1130          POP       BC
                 1140;PONER LOZALIZACION DE IMAGE 0 EN DATOS DE MOVIM. DE SPRITE
                 1150;
8FCE  FD7507     1160          LD        (IY+7),L
8FD1  FD7408     1170          LD        (IY+8),H
8FD4  C9         1180          RET
                 1190;
                 1200;SUBRUTINA DE BORRADO
                 1210;
8FD5  54         1220CL        LD        D,H
8FD6  5D         1230          LD        E,L
8FD7  13         1240          INC       DE
8FD8  EDB0       1250          LDIR
8FDA  C9         1260          RET
```

Ahora que hemos sacado nuestros datos en bruto del *sprite* sin expandir de la memoria y creado la imagen 0 con ellos, necesitamos generar las otras imágenes. Cada imagen sucesiva se forma desplazando las filas de la anterior uno o más bits hacia la derecha. La rutina SPREX hace esto tomando cada fila de imagen 0 por orden, copiándola dentro del área de "espacio de trabajo" y desplazándola y copiándola repetidamente en la posición apropiada para cada una de las otras imágenes. Etiquetaremos el principio del espacio de trabajo como WKSPC y observe que, puesto que sólo necesitamos colocar una fila de un *sprite* en él cada vez, habrá de sobra con 20 octetos. Por tanto, debe empezar su programa con una línea como:

```
WKSPC   DEFS  20
```

Observe que hay que llamar a SPREX con IY apuntando a los datos de movimiento de su *sprite*, puesto que pone el número de imágenes en (IY + 2). Los

valores de entrada de HL y BC están ya establecidos para usted al llamar a PADOUT, así es que el único parámetro que tiene que establecer después de llamar a PADOUT es el paso (en *pixels*) entre las imágenes, almacenado en D. Por tanto, adjuntamos a nuestro fragmento anterior, para establecer los *sprites* del avión, las líneas:

```
LD     D,1    ;FORMA 8 IMAGENES
CALL   SPREX
```

Tenga cuidado, ya que casi todos los registros alternativos se utilizan en la rutina; por tanto, si tiene intención de volver al BASIC después de utilizar SPREX asegúrese de conservar HL' con:

```
EXX
PUSH   HL
EXX
```

y

```
EXX
POP    HL
EXX
```

al principio y al final de su programa respectivamente.

```
                 10 ;RUTINA PARA FORMAR LAS IMAGENES "DESPLAZADAS" DE DATOS DE
                 20 ;SPRITE EXPANDIDO COMO SE PRODUCE POR "PADOUT"
                 30 ;
                 40 ;ENTRADAS : HL=DIRECCION DE IMAGE 0
                 50 ;D=PASO ENTRE IMAGENES
                 60 ;B=ANCHURA DEL SPRITE EXPANDIDO
                 70 ;C=PROFUNDIDAD DEL SPRITE EXPANDIDO EN FILAS
                 80 ;SE CONSERVA : BC
                 90 ;NOTA B'C'D'E'H'L' SON DESTRUIDOS !
                100 ;SALIDA : DE'=VALOR DE ENTRADA DE DE,BC'=0,L'=0
                110 ;
8F29  3E08      120 SPREX    LD       A,8
8F2B  1EFF      130          LD       E,#FF
8F2D  92        140 SUBDIV   SUB      D
8F2E  1C        150          INC      E
8F2F  30FC      160          JR       NC,SUBDIV
8F31  FD7302    170          LD       (IY+2),E
8F34  1D        180          DEC      E
8F35  D5        190          PUSH     DE
8F36  C5        200          PUSH     BC
8F37  0600      210          LD       B,0
                220 ;
                230 ;BC AHORA CONTIENE LA LONGITUD DE 1 COLUMNA EN BYTES
                240 ;
8F39  11398F    250          LD       DE,WKSPC
8F3C  D9        260          EXX
8F3D  E1        270          POP      HL
8F3E  D1        280          POP      DE
                290 ;
                300 ;H'=ANCHURA,L'=NO. DE FILAS
```

```
                        ;D'=PASO DE IMAGEN,E'=NO. DE IMAGENES-1
                        ;GENERAR UNA FILA DE CADA IMAGEN
                        ;
8F3F  D5        NXROW9  PUSH    DE
8F40  44                LD      B,H
8F41  D9                EXX
                        ;
                        ;ALMACENAR DIRECCION DE FILA 0 DE IMAGEN 0
                        ;
8F42  E5                PUSH    HL
8F43  11398F            LD      DE,WKSPC
                        ;
                        ;CONSTRUIR ESTA FILA DEL SPRITE EN EL ESPACIO DE TRABAJO
                        ;
8F46  D9                EXX
8F47  D9        NXBYT3  EXX
8F48  7E                LD      A,(HL)
8F49  09                ADD     HL,BC
8F4A  12                LD      (DE),A
8F4B  13                INC     DE
8F4C  D9                EXX
8F4D  10F8              DJNZ    NXBYT3
                        ;
                        ;ALMACENAR DIREC. DE FILA ACTUAL DE SIGUIENTE IMAGEN EN DE
                        ;
8F4F  D9        NXPOS   EXX
8F50  EB                EX      DE,HL
8F51  D9                EXX
                        ;
                        ;DESPLAZAR FILA POR D' PIXELS
                        ;
8F52  D5                PUSH    DE
8F53  4A                LD      C,D
                        ;
                        ;UN PIXEL CADA VEZ
                        ;
8F54  7C        NXSHF   LD      A,H
8F55  D9                EXX
8F56  21398F            LD      HL,WKSPC
8F59  A7                AND     A
8F5A  CB1E      NXBYT   RR      (HL)
8F5C  2C                INC     L
8F5D  3D                DEC     A
8F5E  20FA              JR      NZ,NXBYT
                        ;
                        ;SIGUIENTE DESPLAZAMIENTO
                        ;
8F60  D9                EXX
8F61  0D                DEC     C
8F62  20F0              JR      NZ,NXSHF
                        ;
                        ;RECUPERAR DIRECCION DE LA SIGUIENTE FILA DE IMAGEN EN HL
                        ;
8F64  D9                EXX
8F65  EB                EX      DE,HL
8F66  11398F            LD      DE,WKSPC
8F69  D9                EXX
                        ;
                        ;TRANSFERIR LA FILA DE H' COLUMNAS AL AREA DE IMAGEN
                        ;
8F6A  44                LD      B,H
8F6B  D9        NXBYT2  EXX
8F6C  1A                LD      A,(DE)
8F6D  77                LD      (HL),A
8F6E  09                ADD     HL,BC
8F6F  13                INC     DE
8F70  D9                EXX
```

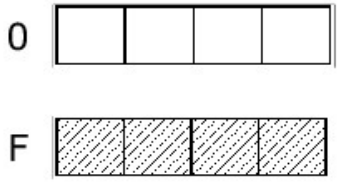
```
8F71  10F8        980          DJNZ     NXBYT2
                  990;
                 1000;BUCLE HACIA ATRAS PARA GENERAR MISMA FILA DE OTRAS IMAGENES

8F73  D1         1010          POP      DE
8F74  1D         1020          DEC      E
8F75  20D8       1030          JR       NZ,NXPOS
8F77  D9         1040          EXX
                 1050;
                 1060;BUSCAR SIGUIENTE FILA DE IMAGEN 0
                 1070;
8F78  E1         1080          POP      HL
8F79  23         1090          INC      HL
                 1100;
                 1110;RECUPERAR DE' Y REPETIR PARA LA SIGUIENTE FILA
                 1120;
8F7A  D9         1130          EXX
8F7B  D1         1140          POP      DE
8F7C  2D         1150          DEC      L
8F7D  20C0       1160          JR       NZ,NXROW9
                 1170;
                 1180;VOLVER CON EL JUEGO DE REGISTROS CORRECTOS
                 1190;
8F7F  D9         1200          EXX
8F80  C9         1210          RET
```

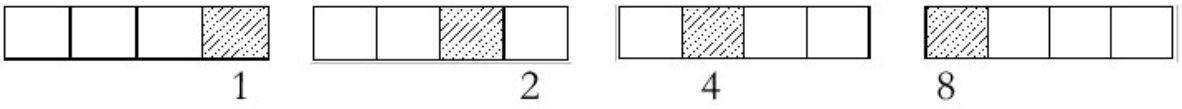
Hasta aquí, muy bien. A estas alturas debe tener un conocimiento razonable de los principios implicados en esta técnica de animación de *sprite*, junto con dos rutinas que hacen casi todo el trabajo de preparación para una animación así. El único trabajo verdadero que ahora necesita realizar siempre que quiera definir un *sprite* es la inevitable tarea aburrida de diseñar la forma y convertirla en los datos del *sprite* original. Muchas personas piensan que los programas de "diseño de caracteres" son útiles y, por supuesto, hay un buen número disponible, incluyendo la versión limitada a un carácter, de las cassettes de introducción de "Horizontes" que se suministran con la máquina.

Podría comprar uno de estos programas o, mejor aún, escribir el suyo propio. Personalmente, prefiero el método más tradicional de utilizar un lápiz, una goma de borrar, un montón de papel de gráfico y un buen suministro de café y de paciencia.

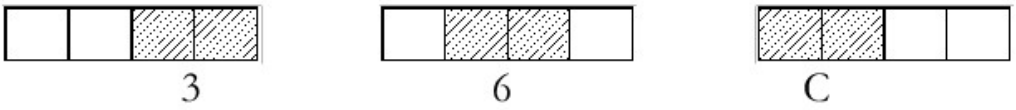
Si toma cualquier fila de un carácter y la divide en dos grupos de cuatro *pixels* cada uno, cada uno de ellos corresponderá a un dígito en la representación hexadecimal de esta octeto-fila. Es entonces fácil ver que los cuatro *pixels* serán una de las sólo 16 estructuras posibles y con un poco de práctica encontrará muy sencillo adjuntar el dígito correcto a cualquier estructura dada. Los más obvios son probablemente:



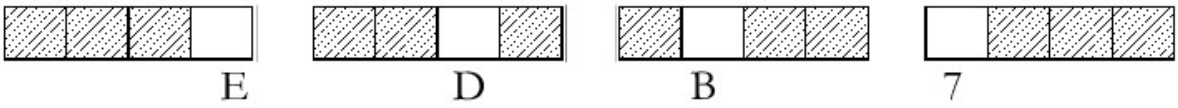
seguido de cerca por las estructuras de un bit:



A continuación tenemos las estructuras de dos bits consecutivos:

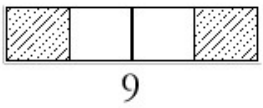


y las que sólo tienen un bit puesto a cero:



Hay dos estructuras posibles con pixels alternados y puestos a uno y a cero. Distíngalas recordando que 5 es impar y, por tanto, tiene puesto a uno el bit-*pixel* de más a la derecha:

y sólo queda por ver la estructura 9, inconfundiblemente simétrica.



Si aún no está muy familiarizado con las estructuras, espero que la división en categorías anterior le proporcionará una ayuda útil.

Ahora proporcionaré la auténtica rutina de "impresión" del *sprite*, que asimila la información correcta para cada uno de ellos y la envía al *buffer* para el procesador de impresión.

Se ha escrito SPRINT tomando velocidad y versatilidad como máximas prioridades. Si movemos *sprites* una vez en cada cuadro de TV, y además, quizá, generamos un sonido y un horizonte de pantalla de nivel bajo, entonces el tiempo es la esencia, y este factor debe tener prioridad por encima del grado de optimización de espacio del programa. Normalmente no se llamará a SPRINT directamente, ya que será subsidiaria de una rutina más general llamada SPRMV, que realizará varias operaciones con los datos de movimiento antes de saltar a SPRINT.

SPRINT nos permite verificar la OR-impresión utilizando el sistema de OR-mapa descrito al final del capítulo anterior. Una versión ligeramente modi-

ficada de ORCHK ha sido construida dentro de SPRINT para efectos de velocidad y la opción de verificación de OR-impresión es seleccionada poniendo a uno el bit 7 del octeto de atributos del *sprite* almacenado en (IY + 15).

Poner a cero este bit tiene como efecto que se ignore el OR-mapa, y en este caso siempre ocurrirá una OVER-impresión. El estado de esta bandera provoca una bifurcación en la rutina a secciones distintas, incluyendo una de ellas ORCHK y la otra no. Se ha visto que este método es mucho más rápido que ejecutar una rutina combinada que implique verificaciones repetidas de las banderas y saltos.

Observe que la rutina llama a ATTLOC, cuyo listado está en el capítulo 1, y se utiliza para proporcionar la dirección de los atributos de la esquina superior izquierda del *sprite.*

Puesto que la rutina destruye los contenidos de todos los registros alternativos, debe una vez más conservar HL' si desea volver al BASIC, SPRINT presupone que realmente hay sitio para su *sprite* en el *buffer* impresión, y como tal no debe llamarse si esto no fuera cierto, en cuyo caso deberá esperar una interrupción para borrar el *buffer.* Observará que la sección implicada en el envío de los datos al buffer utiliza instrucciones de incremento de registro simple para atravesarlo. Si ha extendido el buffer hasta más de 42 entradas de longitud, como se describió en el capítulo 9, necesitará cambiar las instrucciones por incrementos de registro doble, o sea, cambiar INC L por INC HL. Recuerde que la variable de un bit CHSTRE contiene el número de entradas utilizadas en el *buffer* y que BUFFPT apunta a la siguiente entrada libre. Ambos son ajustados convenientemente por la rutina.

No es deseable recibir una interrupción cuando sólo se ha enviado medio *sprite* al *buffer.* Por tanto, a menos que esté utilizando el interceptor de interrupciones para algo diferente del procesador de impresión, debe desactivar las interrupciones antes de llamar a SPRINT, y activarlas a la vuelta.

SPRINT se ocupará admirablemente de los *sprites* que se salgan fuera de la pantalla o incluso aquellos que ni siquiera se encuentren en ella. Por ejemplo, podemos tener sólo la columna de la derecha de un *sprite* 3 × 3 en la pantalla enviando a SPRINT el valor:

XC = − 2, o FE Hex.

Tal como está la rutina, cualquier parte del *sprite* que esté en el área del texto se imprimirá. Sin embargo, podemos, si así lo deseamos, variar la anchura de esta "ventana de *sprite*", cambiando los operandos de las instrucciones etiquetadas LFTLM1, LFTLM2, RGTLM1 y RGTLM2, donde LFTLM significa "Límite izquierdo" (LeFT LiMit) y RGTLM "Límite derecho" (RiGhT LiMit). El límite izquierdo es el valor de la columna de más a la izquierda de la ventana de *sprites*, mientras que el límite derecho es el valor de la columna que se encuentra inmediatamente a la derecha de la ventana (32 en el caso de una ventana máxima).

Por ejemplo, supongamos que queremos que la ventana de *sprites* esté en las 20 columnas centrales de la pantalla (podemos estar empleando las más externas

para el tanteo). Entonces el límite izquierdo será la columna 6, y el derecho la columna 6 + 20 = 26. Así que utilizamos:

```
LD      A,6
LD      (LFTLM1+1),A
LD      (LFTLM2+1),A
LD      A,26
LD      (RGTLM1+1),A
LU      (RGTLM2+1),A
```

Y esto es todo cuanto hay que decir referente a este formidable listado, así es que le dejaré que lo lea y lo introduzca en máquina:

```
                    10 ;ESTA RUTINA ENVIA DATOS DE SPRITE AL BUFFER DE IMPRESION
                    20 ;ENTRADA:B=XP,C=YP,D=YC,E=XC
                    30 ;HL=DIRECCION DE IMAGEN 0
                    40 ;TODO COMO SE DEFINIO POR SPRMV
                    50 ;SALIDA : DE=0
                    60 ; DESTRUIDOS : A'F'B'C'D'E'H'L'
                    70 ;
A06B  D5            80 SPRINT    PUSH      DE
                    90 ;
                   100 ;SI XP=0, DEJAR HL SENALANDO A IMAGEN 0
                   110 ;
A06C  78           120           LD        A,B
A06D  A7           130           AND       A
A06E  2809         140           JR        Z,POSO
                   150 ;
                   160 ;BUSCAR IMAGEN CORRECTA
                   170 ;
A070  FD5E0D       180           LD        E,(IY+13)
A073  FD560E       190           LD        D,(IY+14)
A076  19           200 NXA       ADD       HL,DE
A077  10FD         210           DJNZ      NXA
                   220 ;
                   230 ;BUSCAR POSICION VERTICAL CORRECTA
                   240 ;
A079  09           250 POSO      ADD       HL,BC
                   260 ;
                   270 ;BUSCAR LOCALIZACION DEL ATRI. SUPERIOR IZQ. DEL SPRITE
                   280 ;
A07A  C1           290           POP       BC
A07B  E5           300           PUSH      HL
A07C  CD7CA0       310           CALL      ATTLOC
                   320 ;
                   330 ;E CUENTA LAS COLUMNAS RESTANTES
                   340 ;
A07F  FD5E0B       350           LD        E,(IY+11)
                   360 ;
                   370 ;DECIDIR ENTRE MODOS OR U OVER PRINT EXAMINANDO
                   380 ;EL BIT 7 DE LOS ATRI. DEL SPRITE
                   390 ;
A082  D9           400           EXX
A083  FD6E0F       410           LD        L,(IY+15)
A086  FD6610       420           LD        H,(IY+16)
A089  CB7D         430           BIT       7,L
A08B  CBBD         440           RES       7,L
A08D  D9           450           EXX
A08E  CA45A1       460           JP        Z,SPRTNO
                   470 ;
                   480 ;SE SELECCIONA OR PRINT,BUSCAR DIRECCION APROPIADA
                   490 ;EN OR-MAP
                   500 ;
A091  78           510           LD        A,B
```

```
A092  87          520          ADD       A,A
A093  87          530          ADD       A,A
A094  47          540          LD        B,A
A095  79          550          LD        A,C
A096  CB2F        560          SRA       A
A098  CB2F        570          SRA       A
A09A  CB2F        580          SRA       A
A09C  80          590          ADD       A,B
                  600;
                  610;SUMAR DIRECCION BASE DE OR-MAP
                  620;
A09D  D9          630          EXX
A09E  EB          640          EX        DE,HL
A09F  4F          650          LD        C,A
A0A0  0600        660          LD        B,0
A0A2  21A2A0      670          LD        HL,ORMAP
A0A5  09          680          ADD       HL,BC
A0A6  3AA6A0      690          LD        A,(CHSTRE)
A0A9  47          700          LD        B,A
A0AA  D9          710          EXX
                  720;
                  730;HL' CONTIENE LOCALIZACION EN OR-MAP. ROTAR MASCARA SOBRE
                  740;EL BIT-CELDILLA CORRECTO EN OR-MAP
                  750;
A0AB  79          760          LD        A,C
A0AC  E607        770          AND       7
A0AE  47          780          LD        B,A
A0AF  3E80        790          LD        A,#80
A0B1  2803        800          JR        Z,NROT1
A0B3  0F          810NXTRT     RRCA
A0B4  10FD        820          DJNZ      NXTRT
                  830;
                  840;ALMACENAR MASCARA EN C'
                  850;
A0B6  D9          860NROT1     EXX
A0B7  4F          870          LD        C,A
A0B8  D9          880          EXX
A0B9  79          890          LD        A,C
                  900;
                  910;BC SENALA  LOS DATOS DE IMAGEN
                  920;
A0BA  C1          930          POP       BC
A0BB  E5          940NXTX1     PUSH      HL
                  950;
                  960;ALMACENAR DIRECCION DE OR-MAP
                  970;
A0BC  D9          980          EXX
A0BD  E5          990          PUSH      HL
A0BE  D9         1000          EXX
                 1010;
                 1020;HACER D=PROFUNDIDAD EN LINEAS
                 1030;
A0BF  FD560C     1040          LD        D,(IY+12)
                 1050;
                 1060;SI LA POSICION DE IMPRESION ESTA FUERA DE RANGO-X
                 1070;SALTAR COLUMNA
A0C2  FE20       1080RGTLM1    CP        32
A0C4  3071       1090          JR        NC,HOPCL1
A0C6  FE00       1100LFTLM1    CP        0
A0C8  386D       1110          JR        C,HOPCL1
                 1120;
                 1130;ALMACENAR POSIC. DE COLUMNA EN A'
                 1140;
A0CA  08         1150          EX        AF,AF'
                 1160;
                 1170;SI POSC. DE IMPRESION ESTA DEBAJO DEL AREA DE TEXTO,
                 1180;ENTONCES FINAL
```

```
A0CB  7C        1190NXTY1   LD      A,H
A0CC  FE5B      1200        CP      91
A0CE  3068      1210        JR      NC,OUT81
                1220;
                1230;SI POSIC. DE IMPRESION ESTA ENCIMA DEL AREA DE TEXTO
                1240;IGNORAR ESTA LINEA DE SPRITE
                1250;
A0D0  FE58      1260        CP      88
A0D2  D5        1270        PUSH    DE
A0D3  EB        1280        EX      DE,HL
A0D4  382F      1290        JR      C,NPR1
                1300;
                1310;DECIDIR SI OR-PRINT ES NECESARIO EN ESTA CELDILLA
                1320;
A0D6  D9        1330        EXX
A0D7  79        1340        LD      A,C
A0D8  A6        1350        AND     (HL)
A0D9  CBBB      1360        RES     7,E
A0DB  2802      1370        JR      Z,NOTOR3
                1380;
                1390;SI LA CELDILLA ESTA OCUPADA PONER BANDERA PARA OR-PRINT
                1400;
A0DD  CBFB      1410        SET     7,E
                1420;
                1430;INDICAR CELDILLA OCUPADA
                1440;
A0DF  79        1450NOTOR3  LD      A,C
A0E0  B6        1460        OR      (HL)
A0E1  77        1470        LD      (HL),A
A0E2  D9        1480        EXX
                1490;
                1500;ENVIAR CARACTER AL BUFFER
                1510;
A0E3  1A        1520        LD      A,(DE)
A0E4  D9        1530        EXX
A0E5  AB        1540        XOR     E
A0E6  A2        1550        AND     D
A0E7  AB        1560        XOR     E
A0E8  04        1570        INC     B
A0E9  D9        1580        EXX
A0EA  2AEAA0    1590        LD      HL,(BUFFPT)
A0ED  07        1600        RLCA
A0EE  77        1610        LD      (HL),A
A0EF  2C        1620        INC     L
A0F0  73        1630        LD      (HL),E
A0F1  2C        1640        INC     L
A0F2  72        1650        LD      (HL),D
A0F3  2C        1660        INC     L
A0F4  7A        1670        LD      A,D
A0F5  E603      1680        AND     3
A0F7  07        1690        RLCA
A0F8  07        1700        RLCA
A0F9  07        1710        RLCA
A0FA  F640      1720        OR      64
A0FC  77        1730        LD      (HL),A
A0FD  2C        1740        INC     L
A0FE  71        1750        LD      (HL),C
A0FF  2C        1760        INC     L
A000  70        1770        LD      (HL),B
A001  2C        1780        INC     L
A002  22EAA0    1790        LD      (BUFFPT),HL
                1800;
                1810;INCREMENTAR PUNTERO DE DATOS A SIGUIENTE CELDILLA DE IMAGEN
                1820;
A005  210800    1830NPR1    LD      HL,8
A008  09        1840        ADD     HL,BC
```

```
A109  44        1850          LD        B,H
A10A  4D        1860          LD        C,L
A10B  EB        1870          EX        DE,HL
A10C  D1        1880          POP       DE
                1890;
                1900;SI CONTADOR DE LINEA ES CERO ENTONCES SIGUIENTE COLUMNA
                1910;
A10D  15        1920          DEC       D
A10E  2810      1930          JR        Z,IN16
                1940;
                1950;SI NO, MOVER PUNTERO OR-MAP  A SIGUIENTE LINEA
                1960;
A110  D9        1970          EXX
A111  7D        1980          LD        A,L
A112  C604      1990          ADD       A,4
A114  6F        2000          LD        L,A
A115  D9        2010          EXX
                2020;
                2030;Y MOVER PUNTERO DE ATRI. A SIGUIENTE LINEA
                2040;
A116  7D        2050          LD        A,L
A117  C620      2060          ADD       A,32
A119  6F        2070          LD        L,A
A11A  30AF      2080          JR        NC,NXTY1
A11C  24        2090          INC       H
                2100;BUCLE ATRAS PARA SIGUIENTE LINEA DE SPRITE
                2110;
                2120;
A11D  C3CBA0    2130          JP        NXTY1
                2140;
                2150;INCREMENTAR POSICION DE COLUMNA
                2160;
A120  08        2170IN16      EX        AF,AF'
A121  3C        2180          INC       A
                2190;
                2200;COGER DIRECCION DE OR-MAP Y LLEVAR LA MASCARA A EL
                2210;INCREMENTANDO EL PUNTERO SI ES PRECISO
                2220;
A122  D9        2230          EXX
A123  E1        2240          POP       HL
A124  CB09      2250          RRC       C
A126  3001      2260          JR        NC,NINC2
A128  2C        2270          INC       L
A129  D9        2280NINC2     EXX
                2290;
                2300;HL SENALA AL PRIMER ATRIBUTO DE LA SIGUIENTE COLUMNA
                2310;
A12A  E1        2320          POP       HL
A12B  2C        2330          INC       L
                2340;
                2350;BUCLE HACIA ATRAS PARA SIGUIENTE COLUMNA
                2360;
A12C  1D        2370          DEC       E
A12D  C2BBA0    2380          JP        NZ,NXTX1
                2390;
                2400;PONER NUEVO VALOR DE CHSTRE
                2410;
A130  D9        2420          EXX
A131  78        2430          LD        A,B
A132  32A6A0    2440          LD        (CHSTRE),A
A135  D9        2450          EXX
A136  C9        2460          RET
                2470;
                2480;SALTA HASTA AQUI PARA OMITIR TODA O PARTE DE UNA COLUMNA
                2490;
A137  08        2500HOPCL1    EX        AF,AF'
                2510;
```

```
                        2520;MOVER PUNTERO DE IMAGEN A LA SIGUIENTE COLUMNA DE SPRITE
                        2530;
A138  60        2540OUT81    LD        H,L
A139  69        2550         LD        L,C
A13A  010800    2560         LD        BC,8
A13D  09        2570NXT81    ADD       HL,BC
A13E  15        2580         DEC       D
A13F  20FC      2590         JR        NZ,NXT81
A141  44        2600         LD        B,H
A142  4D        2610         LD        C,L
                        2620;
                        2630;SALTAR HACIA ATRAS A LA RUTINA PRINCIPAL
                        2640;
A143  18DB      2650         JR        IN16
                        2660;
                        2670;OVER-PRINT MUCHO MAS RAPIDA Y CORTA
                        2680;
A145  79        2690SPRTNO   LD        A,C
A146  C1        2700         POP       BC
                        2710;
                        2720;CONTIENE CHSTRE EN E
                        2730;
A147  D9        2740         EXX
A148  ED5BA6A0  2750         LD        DE,(CHSTRE)
A14C  D9        2760         EXX
A14D  E5        2770NXTX2    PUSH      HL
                        2780;
                        2790;HACER D=PROFUNDIDAD EN LINEAS
                        2800;
A14E  FD560C    2810         LD        D,(IY+12)
                        2820;
                        2830;SI POSC. DE IMPRE. FUERA DEL RANGO-X SALTAR COLUMNA
                        2840;
A151  FE20      2850RGTLM2   CP        32
A153  3056      2860         JR        NC,HOPCL2
A155  FE00      2870LFTLM2   CP        0
A157  3852      2880         JR        C,HOPCL2
                        2890;
                        2900;ALMACENAR POSICION DE COLUMNA EN A
                        2910;
A159  08        2920         EX        AF,AF'
                        2930;
                        2940;SI POSC. DE IMPR. ESTA BAJO EL AREA DE TEXTO, FINAL
                        2950;
A15A  7C        2960NXTY2    LD        A,H
A15B  FE5B      2970         CP        91
A15D  304D      2980         JR        NC,OUT82
                        2990;
                        3000;SI POSC. IMPR. SOBRE AREA DE TEXTO ENTONCES IGNORAR LA
                        3010;LINEA DE SPRITE
                        3020;
A15F  FE58      3030         CP        88
A161  D5        3040         PUSH      DE
A162  EB        3050         EX        DE,HL
A163  3822      3060         JR        C,NPR2
                        3070;
                        3080;ENVIAR CARACTER AL BUFFER
                        3090;
A165  1A        3100         LD        A,(DE)
A166  D9        3110         EXX
A167  AD        3120         XOR       L
A168  A4        3130         AND       H
A169  AD        3140         XOR       L
A16A  1C        3150         INC       E
A16B  D9        3160         EXX
A16C  2AEAA0    3170         LD        HL,(BUFFPT)
A16F  07        3180         RLCA
```

```
A170  77        3190          LD        (HL),A
A171  2C        3200          INC       L
A172  73        3210          LD        (HL),E
A173  2C        3220          INC       L
A174  72        3230          LD        (HL),D
A175  2C        3240          INC       L
A176  7A        3250          LD        A,D
A177  E603      3260          AND       3
A179  07        3270          RLCA
A17A  07        3280          RLCA
A17B  07        3290          RLCA
A17C  F640      3300          OR        64
A17E  77        3310          LD        (HL),A
A17F  2C        3320          INC       L
A180  71        3330          LD        (HL),C
A181  2C        3340          INC       L
A182  70        3350          LD        (HL),B
A183  2C        3360          INC       L
A184  22EAA0    3370          LD        (BUFFPT),HL
                3380;
                3390;INCREMENTAR PUNTERO DE DATOS A SIGUIENTE CELDILLA DE IMAGEN
                3400;
A187  210800    3410NPR2      LD        HL,8
A18A  09        3420          ADD       HL,BC
A18B  44        3430          LD        B,H
A18C  4D        3440          LD        C,L
A18D  EB        3450          EX        DE,HL
A18E  D1        3460          POP       DE
                3470;
                3480;SI CONTADOR DE LINEA ES CERO, SIGUIENTE COLUMNA
                3490;
A18F  15        3500          DEC       D
A190  280A      3510          JR        Z,IN17
                3520;
                3530; SI NO MOVER PUNTERO DE ATRI. A SIGUIENTE LINEA
                3540;
A192  7D        3550          LD        A,L
A193  C620      3560          ADD       A,32
A195  6F        3570          LD        L,A
A196  30C2      3580          JR        NC,NXTY2
A198  24        3590          INC       H
                3600;
                3610;BUCLE HACIA ATRAS PARA LA SIGUIENTE LINEA DE SPRITE
                3620;
A199  C35AA1    3630          JP        NXTY2
                3640;
                3650;INCREMENTAR POSICION DE COLUMNA
                3660;
A19C  08        3670IN17      EX        AF,AF'
A19D  3C        3680          INC       A
                3690;
                3700;HL SENALA AL  PRIMER ATRI. DE LA SIGUIENTE COLUMNA
                3710;
A19E  E1        3720          POP       HL
A19F  2C        3730          INC       L
                3740;
                3750;BUCLE HACIA ATRAS PARA LA SIGUIENTE COLUMNA
                3760;
A1A0  1D        3770          DEC       E
A1A1  C24DA1    3780          JP        NZ,NXTX2
A1A4  D9        3790          EXX
A1A5  7B        3800          LD        A,E
A1A6  32A6A0    3810          LD        (CHSTRE),A
A1A9  D9        3820          EXX
A1AA  C9        3830          RET
                3840;SALTAR AQUI PARA OMITIR TODA O PARTE DE LA COLUMNA
                3850;
```

```
A1AB  08        3860HOPCL2   EX       AF,AF'
                3870;
                3880;MOVER PUNTERO DE IMAGEN A LA SIGUIENTE COLUMNA DE SPRITE
                3890;
A1AC  60        3900OUT82    LD       H,B
A1AD  69        3910         LD       L,C
A1AE  010800    3920         LD       BC,8
A1B1  09        3930NXT82    ADD      HL,BC
A1B2  15        3940         DEC      D
A1B3  20FC      3950         JR       NZ,NXT82
A1B5  44        3960         LD       B,H
A1B6  4D        3970         LD       C,L
                3980;
                3990;SALTO HACIA ATRAS A LA RUTINA PRINCIPAL
                4000;
A1B7  18E3      4010         JR       IN17
```

Ahora tenemos las tres rutinas necesarias para preparar los datos y para imprimir realmente nuestros *sprites* en la pantalla. Para completar la serie de rutinas de generación de *sprite* le proporcionaré una rutina maestra de control de *sprites*. La función de SPRMV será actualizar los valores de XP, XC, YP e YC según VX y VY (todos almacenados en los datos de movimiento, indexados por el registro IY), a continuación definir los parámetros correctos en los registros y saltar a hacer una impresión SPRINT del *sprite*. El único parámetro que necesita SPRMV es la dirección de los datos de movimiento, en IY. Y una vez que se ha llamado a SPRMV, no se necesita más trabajo para mover e imprimir su *sprite*.

Antes de explicarle cómo inicializar y manipular los datos de movimiento, le presentaré el listado de SPRMV:

```
                 10 ;
                 20 ;CONTROLADOR DE SPRITE DE PROPOSITO GENERAL
                 30 ;
                 40 ;ENTRADA:IY APUNTA A LOS DATOS DE MOVIMIENTO, PARA MAS DETAL
                  LES
                 50 ;MIRAR TEXTO
                 60 ;NOTA: B'C'D'E'H'L'A'F' DESTRUIDOS
                 70 ;NOTA:IY SE CONSERVA
                 80 ;
                 90 ;
                100 ;DECREMENTAR CONTADOR DE CICLO
                110 ;
8D3D  FD3509    120 SPRMV    DEC      (IY+9)
8D40  C0        130          RET      NZ
                140 ;
                150 ;SI CERO ENTONCES VOLVER A LLENAR EL CONTADOR DE CICLO
                160 ;
8D41  FD7E0A    170          LD       A,(IY+10)
8D44  FD7709    180          LD       (IY+9),A
                190 ;
                200 ;HACER HL=DIRECCION DE IMAGEN 0
                210 ;
8D47  FD6608    220          LD       H,(IY+8)
8D4A  FD6E07    230          LD       L,(IY+7)
                240 ;
                250 ;SUMAR PASO A XP
                260 ;
8D4D  FD7E00    270          LD       A,(IY+0)
8D50  FD8601    280          ADD      A,(IY+1)
8D53  F2628D    290          JP       P,NNEG1
                300 ;
```

```
                    310;SI RESULTADO NEGATIVO HACER XP=XP+XMAX
                    320;Y HACER XC=XC-1
                    330;
8D56  FD8602        340         ADD       A,(IY+2)
8D59  FD3503        350         DEC       (IY+3)
8D5C  FD5E03        360         LD        E,(IY+3)
                    370;
                    380;SALTAR PARA TRABAJAR CON Y
                    390;
8D5F  C37C8D        400         JP        XDN
8D62  FD4602        410NNEG1    LD        B,(IY+2)
                    420;
                    430;SI XP<XMAX IR POR Y
                    440;
8D65  B8            450         CP        B
8D66  FD5E03        460         LD        E,(IY+3)
8D69  3811          470         JR        C,XDN
                    480;
                    490;SI NO, INCREMENTAR XC,HACER XP=XP-XMAX
                    500;
8D6B  FD3403        510         INC       (IY+3)
8D6E  90            520         SUB       B
                    530;
                    540;Y DECREMENTAR HL EN UNA COLUMNA PARA PERMITIR UNA COLUMNA
                    550;BLANCA A LA IZQUIERDA
                    560;
8D6F  08            570         EX        AF,AF'
8D70  FD7E0C        580         LD        A,(IY+12)
8D73  01F8FF        590         LD        BC,#FFF8
8D76  09            600NXUB     ADD       HL,BC
8D77  3D            610         DEC       A
8D78  C2768D        620         JP        NZ,NXUB
8D7B  08            630         EX        AF,AF'
                    640;
                    650;ALMACENAR NUEVO VALOR DE XP
                    660;
8D7C  FD7700        670XDN      LD        (IY+0),A
8D7F  47            680         LD        B,A
                    690;
                    700;SUMAR PASO A YP
                    710;
8D80  FD7E04        720         LD        A,(IY+4)
8D83  FD8605        730         ADD       A,(IY+5)
8D86  F2958D        740         JP        P,NNEG2
                    750;
                    760;SI RESULTADO NEGATIVO HACER YP=YP MOD 8
                    770;
8D89  E607          780         AND       7
8D8B  4F            790         LD        C,A
                    800;
                    810;E INCREMENTAR YC
                    820;
8D8C  FD3406        830         INC       (IY+6)
8D8F  FD5606        840         LD        D,(IY+6)
8D92  C3A28D        850         JP        YDN
                    860;
                    870;SI YP>7 HACER YP=YP-8
                    880;Y DECREMENTAR YC
                    890;
8D95  FE08          900NNEG2    CP        8
8D97  FD5606        910         LD        D,(IY+6)
8D9A  4F            920         LD        C,A
8D9B  3805          930         JR        C,YDN
8D9D  E607          940         AND       7
8D9F  FD3506        950         DEC       (IY+6)
8DA2  FD7704        960YDN      LD        (IY+4),A
                    970;
```

```
                980;SALTO A LA RUTINA DE IMPRESION DEL SPRITE
                990;
8DA5  C3A58D    1000        JP      SPRINT
```

Con referencia a la tabla de contenidos de datos de movimiento encontrada anteriormente en este capitulo, recordaremos que siete octetos de los diecisiete asignados a cada *sprite* serán inicializados por las rutinas PADOUT y SPREX.

Por tanto, sólo necesitamos reservar espacio para ellos con DEFB 0 en el listado en ensamblador. Las variables que tenemos que inicializar nosotros mismos incluyen los valores de posición obvios XP, XC, YP e YC. Recuerde que XP se mide hacia la derecha e YP hacia arriba desde la esquina inferior izquierda del *sprite* mientras que XC se mide hacia la derecha e YC hacia abajo de la esquina superior izquierda de la pantalla. Recuerde también que (XC,YC) son las coordenadas de la esquina superior izquierda del *sprite.*

Las velocidades de VX y VY se miden en las mismas direcciones que XP e YP, y pueden ser mayores, iguales o menores que cero. Si VX > 0, el movimiento es hacia la derecha, mientras que si VX < 0, es hacia la izquierda. De la misma manera, VY > 0 significa un movimiento arriba, mientras que VY < 0 envía el *sprite* hacia abajo. Este orden permite una gran versatilidad en la dirección del movimiento. Por ejemplo, podríamos hacer que un *sprite* hiciera un suave "picado" con una velocidad horizontal de tres *pixels* y una velocidad vertical de un *pixel* por movimiento, haciendo:

VX = # 3 VY = #FF (menos uno)

Como he explicado antes, la "cuenta de ciclos" se proporciona como un medio de regular la frecuencia con la que los *sprites* se mueven, y también si dos o más *sprites* se mueven en fase entre sí.

Todo esto se muestra mejor con un ejemplo. Suponga que tenemos dos *sprites*, con datos de movimiento en las etiquetas MDAT1 y MDAT2 y queremos que un *sprite* se mueva cada cinco cuadros de TV y que el segundo se mueva una vez en cada tres cuadros. Establecemos los respectivos "períodos de ciclo" como los valores cinco y tres y, como siempre, inicializamos las "cuentas de ciclo" a uno, para que ambos *sprites* puedan moverse a la primera llamada de SPRMV. Todo cuanto tenemos que hacer es:

```
LD      IY,MDAT1
CALL    SPRMV
LD      IY,MDAT2
CALL    SPRMV
```

después de cada interrupción.

Si tenemos dos *sprites* moviéndose con la misma frecuencia, y deseamos mantenerlos "desfasados", quizá porque no hay bastante espacio en el buffer de impresión para animar a ambos en el mismo cuadro de TV, utilizamos inicializaciones diferentes de la cuenta de ciclos. Por ejemplo, suponga que se dan períodos de ciclo de dos llamadas a ambos *sprites.*

Entonces hacemos que se muevan en cuadros de TV alternos poniendo la

primera cuenta de ciclo a uno y la segunda al valor dos. Ocurra lo que ocurra, el período de ciclo y la cuenta de ciclo siempre tienen que ser distintos de cero. Si esto no fuera así, se obtendría como resultado una frecuencia de movimiento de una vez cada 256 llamadas a SPRMV.

Utilizando los conceptos de cuenta de ciclos y períodos de ciclos, podemos animar todos los *sprites* implicados en un programa en un bloque. Si colocamos todos los datos de su movimiento consecutivamente en la memoria, un fragmento adecuado para después de cada interrupción (detectado por una instrucción HALT) podría ser el siguiente:

```
MDAT    EQU     (DIRECCION DE DATOS DE MOVIMIENTO)
        LD      IY,MDAT
        LD      B,(NUMERO DE SPRITES)
NXSPRT  PUSH    BC
        CALL    SPRMV
        POP     BC
        LD      DE,17
        ADD     IY,DE
        DJNZ    NXSPRT
```

Refiriéndonos de nuevo a la tabla anterior de los contenidos de datos de movimiento, observará que la dirección del primer octeto de la imagen 0 (inmediatamente después de la columna de blancos precedentes) se almacena en (IY + 7). Este valor también se devuelve en HL después de la llamada PADOUT para establecer los datos del *sprite*.

Podemos utilizar esta entrada en los datos de movimiento como un medio para cambiar o reciclar entre diferentes juegos de imágenes para cualquier *sprite* individual. Por ejemplo, puede desear hacer que su carácter "camine" en vez de deslizarse, o quizá que su nave espacial se desintegre gradualmente durante el vuelo, después de haber sido alcanzada por un disparo de láser enemigo.

Para llevar a cabo esta función, establezca todas las diferentes series de datos de *sprite* que necesite, almacenando los valores devueltos por PADOUT en su propia tabla de consulta. Se debe mantener IY apuntando a una serie de datos de movimiento, que luego será obviamente establecida con la última serie de los datos del *sprite* generado.

Entonces, cuando está ejecutando su programa, utiliza una "cuenta de animación" y un "período de animación" análogos al sistema "cuenta de ciclo" y "período de ciclo" para desplazarse por su juego de imágenes recuperando la dirección apropiada de su tabla de consulta e insertándola en (IY + 7) cada vez que desea cambiar los datos.

Hay varias otras manipulaciones de los datos de movimiento que podría probar: por ejemplo, podría hacer que el *sprite* se mueva en una estructura preprogramada recorriendo una tabla de valores para VX y VY, o podría hacer que el *sprite* hiciera de "camaleón" manipulando el octeto de atributos de (IY + 15) (¡recuerde que hay que conservar el bit 7, la bandera de OR-impresión!). Dejo más variaciones sobre este tema a su imaginación, y empezaré el desarrollo de una rutina de demostración.

Después de pensarlo mucho, he preferido mostrarle cómo mover dos *sprites* en dirección opuesta a lo largo de la línea horizontal central de la pantalla.

Un *sprite* será un naipe especial, el seis de tréboles, conocido tradicionalmente como "carta de Gordon", mientras que el otro, para variar, será un teléfono rojo.

Sería una lástima que no se utilizase la animación más fina posible, así es que moveremos ambos *sprites* por un *pixel* en cada cuadro de TV. Ahora bien, en el buffer de impresión pueden caber 40 caracteres, así es que utilizaremos 20 en cada *sprite*. Sabemos que $5 \times 4 = 20$; por tanto, podemos utilizar una forma de celdilla de 3×4 para el naipe y una forma de celdilla de 4×3 para el teléfono.

Recuerde la fórmula para el área de imagen para imágenes de una forma $(m \times n)$, o sea:

Memoria que se necesita = $8(a(m + 1) + 1)(n + 1)$ octetos
Para el teléfono, $m = 4$, $n = 3$, $a = 8$ y
Memoria que se necesita = $8(8(4 + 1) + 1)(3 + 1) = 1.312$ octetos
Para el naipe, $m = 3$, $n = 4$, $a = 8$ y
Memoria que se necesita = $8(8(3 + 1) + 1)(4 + 1) = 1.320$ octetos

Sin olvidar los 8 octetos con ceros al final de la imagen combinada, reservamos espacio con:

```
TELSPC   DEFS   1312
CARSPC   DEFS   1320
         DEFW   0,0,0,0
```

La velocidad horizontal será $VX = 1$, el valor máximo de XP será $XP_{máx} = 7$, y la distancia entre dos imágenes sucesivas será $D = 1$ *pixel*. Por tanto, la anchura del margen de seguridad a la derecha de nuestro *sprite* será:

$$\begin{aligned} P_x &= D(VX + XP_{máx}) - 8 \\ &= 1(1 + 7) - 8 \\ &= 0 \text{ } pixels \end{aligned}$$

¡es decir, que podemos diseñar nuestras formas en las 3 ó 4 columnas enteras! Como los *sprites* no se moverán verticalmente, no se necesitará ningún margen de seguridad superior. El teléfono y el naipe han sido diseñados en toda el área permitida, y he codificado los datos para usted, cuyos resultados se encuentran tras las etiquetas TELDAT y CARDAT respectivamente. Repasando en su mente el procedimiento para el empleo de PADOUT y SPREX, verá que podemos generar las imágenes de nuestro teléfono con el fragmento:

```
LD     HL,TELSPC     ;AREA DE IMAGEN
LD     DE,TELDAT     ;DATOS DE SPRITE
LD     BC,#403       ;B=ANCHURA, C=ALTURA
LD     IY,TELMTN     ;DATOS MOVIMIENTO
CALL   PADOUT
LD     D,1           ;PASO ENTRE IMAGENES
CALL   SPREX         ;DE UN PIXEL
```

Un fragmento similar generará las imágenes del naipe, en donde TELMTN y CARMTN son las direcciones de comienzo de las tablas de datos de movimiento del teléfono y del naipe, respectivamente.

Inicializaremos la posición del teléfono justo en el límite izquierdo de la pantalla y la base del teléfono en la línea once. Por tanto, la esquina superior izquierda del *sprite* está en (–4, 9) y la forma está en la esquina inferior izquierda del *sprite*, es decir:

XP = 0, XC = #FC, YP = 0, YC = 9

Moveremos los *sprites* una vez en cada cuadro de TV, así que hay que poner la cuenta de ciclo y el período de ciclo a uno. He elegido utilizar OR-impresión en esta demostración, para que los dos *sprites* se fundan al cruzarse. El teléfono será rojo (valor 2) y enmascararemos el papel del atributo actual (es decir, PAPER 8); por tanto, tenemos el octeto de atributo #82 y el de máscara #38. Insertando ceros en los huecos en nuestra tabla, que serán rellenados por PADOUT y SPREX, tenemos los datos de movimiento iniciales:

```
TELMTN    DEFB   0,1,0,#FC,0,0,9,0,0,1,1,
                 0,0,0,0,#82,#38
```

El naipe empezará justo en el límite derecho de la pantalla, con su base en la línea 12. Por tanto, empezamos con

XP = 0, XC = 32, YP = 0, YC = 9

Recuerde que VX = –1 = #FF, puesto que el naipe está moviéndose hacia la izquierda, y utilizando INK cian, PAPER 8 y OR-impresión (bit 7 de atributos puesto a uno) tenemos los datos de movimiento iniciales del naipe:

```
CARDAT          DEFB        0,#FF,0,32,0,0,9,0,0,1,1,
                            0,0,0,0,#85,#38
```

Para utilizar la función OR-impresión, debemos asegurarnos de que el OR-mapa esté borrado (utilizando la rutina CLOR del último capítulo) antes de que empiece cada serie completa de movimientos del *sprite*. De otra forma, terminaríamos OR-imprimiendo la nueva imagen de un *sprite* encima de la vieja, provocando una huella no deseada por la pantalla al moverse el *sprite*. De esta forma, el bucle principal de la demostración incluirá las líneas:

```
CALL   CLOR
LD     IY,TELMTN
CALL   SPRMV
LD     IY,CARMTN
CALL   SPRMV
```

antes de una instrucción HALT, para que los *sprites* se impriman realmente.

El resto del listado de demostración se explica por sí solo. Observe que las rutinas INT1 y DISINT se vieron en el capítulo 9.

He aquí pues la rutina de la "demostración espectacular". Estudie el listado con cuidado y modifique, si así lo desea, la velocidad de los *sprites*, el número de imágenes, etc.

```
                    10 ;
                    20 ;
                    30 ;RUTINA DE DEMOSTRACION PARA PADOUT, SPREX, SPRINT Y SPRMV
                    40 ;
                    50 ;CONSERVA HL' PARA LA VUELTA AL BASIC
                    60 ;
99C8  D9            70 TEST     EXX
99C9  E5            80          PUSH     HL
99CA  D9            90          EXX
                   100 ;
                   110 ;PONER HORIZONTE-CERO, CIELO Y MAR NEGRO
                   120 ;
99CB  AF           130          XOR      A
99CC  32CD99       140          LD       (ROWS+1),A
99CF  32D099       150          LD       (TOPBRD+1),A
99D2  32D399       160          LD       (BOTBRD+1),A
                   170 ;
                   180 ;GENERAR DATOS DE SPRITE PARA EL TELEFONO
                   190 ;
99D5  21149B       200          LD       HL,TELSPC
99D8  11549A       210          LD       DE,TELDAT
                   220 ;
                   230 ;TELEFONO ES DE 4 COLUMNAS POR 3 LINEAS
                   240 ;
99DB  010304       250          LD       BC,#403
99DE  FD21329A     260          LD       IY,TELMTN
99E2  CDE299       270          CALL     PADOUT
99E5  1601         280          LD       D,1
99E7  CDE799       290          CALL     SPREX
                   300 ;
                   310 ;GENERAR DATOS DE SPRITE PARA CARTA DE BARAJA
                   320 ;
99EA  2134A0       330          LD       HL,CARSPC
99ED  11B49A       340          LD       DE,CARDAT
                   350 ;
                   360 ;LA CARTA ES DE 3 COLUMNAS POR 4 LINEAS
                   370 ;
99F0  010403       380          LD       BC,#304
99F3  FD21439A     390          LD       IY,CARMTN
99F7  CDE299       400          CALL     PADOUT
99FA  1601         410          LD       D,1
99FC  CDE799       420          CALL     SPREX
                   430 ;
                   440 ;INICIALIZAR PROCESADOR DE IMPRESION DIRIGIDO POR INTERRUPCI
                       ONES
                   450 ;
99FF  CDFF99       460          CALL     INT1
9902  76           470          HALT
                   480 ;
                   490 ;MOVER LOS SPRITES A LO LARGO DE LA PANTALLA 4 VECES A UN
                   500 ;PIXEL POR CUADRO DE TV EN DIRECCIONES OPUESTAS
                   510 ;
9903  0E02         520          LD       C,2
9905  0600         530 NXAM2    LD       B,0
9907  C5           540 NXAM     PUSH     BC
                   550 ;
                   560 ;BORRAR EL OR-MAP ANTES DE CADA GRUPO DE MOVIMIENTOS
                   570 ;
9908  CD089A       580          CALL     CLOR
                   590 ;
                   600 ;MOVER E IMPRIMIR EL TELEFONO
                   610 ;
990B  FD21329A     620          LD       IY,TELMTN
990F  CD0F9A       630          CALL     SPRMV
                   640 ;
                   650 ;MOVER E IMPRIMIR LA CARTA
                   660 ;
```

```
9A12  FD21439A    670          LD       IY,CARMTW
9A16  CD0F9A      680          CALL     SPRMV
9A19  C1          690          POP      BC
9A1A  76          700          HALT
                  710  ;
                  720  ;SIGUIENTE ENCUADRE
                  730  ;
9A1B  10EA        740          DJNZ     NXAM
                  750  ;
                  760  ;DIRECCIONES INVERSAS A LO LARGO DE X
                  770  ;
9A1D  FD7E01      780          LD       A,(IY+1)
9A20  FD77F0      790          LD       (IY-16),A
9A23  ED44        800          NEG
9A25  FD7701      810          LD       (IY+1),A
                  820  ;
                  830  ;SIGUIENTE PASO
                  840  ;
9A28  0D          850          DEC      C
9A29  20DA        860          JR       NZ,NXAM2
                  870  ;
                  880  ;VOLVER A SELECCIONAR IM 1 Y RECUPERAR HL
                  890  ;
9A2B  CD2B9A      900          CALL     DISINT
9A2E  D9          910          EXX
9A2F  E1          920          POP      HL
9A30  D9          930          EXX
9A31  C9          940          RET
                  950  ;
                  960  ;DATOS DE MOVIMIENTO DE SPRITE
                  970  ;
9A32  00          980  TELMTN  DEFB     0
9A33  01          990          DEFB     1
9A34  00         1000          DEFB     0
9A35  FC         1010          DEFB     #FC
9A36  00         1020          DEFB     0
9A37  00         1030          DEFB     0
9A38  09         1040          DEFB     9
9A39  00         1050          DEFB     0
9A3A  00         1060          DEFB     0
9A3B  01         1070          DEFB     1
9A3C  01         1080          DEFB     1
9A3D  00         1090          DEFB     0
9A3E  00         1100          DEFB     0
9A3F  00         1110          DEFB     0
9A40  00         1120          DEFB     0
9A41  82         1130          DEFB     #82
9A42  38         1140          DEFB     #38
9A43  00         1150  CARMTN  DEFB     0
9A44  FF         1160          DEFB     #FF
9A45  00         1170          DEFB     0
9A46  20         1180          DEFB     32
9A47  00         1190          DEFB     0
9A48  00         1200          DEFB     0
9A49  09         1210          DEFB     9
9A4A  00         1220          DEFB     0
9A4B  00         1230          DEFB     0
9A4C  01         1240          DEFB     1
9A4D  01         1250          DEFB     1
9A4E  00         1260          DEFB     0
9A4F  00         1270          DEFB     0
9A50  00         1280          DEFB     0
9A51  00         1290          DEFB     0
9A52  35         1300          DEFB     #85
9A53  38         1310          DEFB     #38
                 1320  ;
                 1330  ;DATOS DE SPRITE SIN EXPANDIR
```

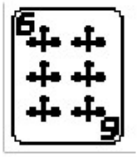
```
                1340  ;
9A54  0F        1350  TELDAT  DEFB    15
9A55  1F        1360          DEFB    31
9A56  3F        1370          DEFB    63
9A57  7F        1380          DEFB    127
9A58  FF        1390          DEFB    255
9A59  FE        1400          DEFB    254
9A5A  FE        1410          DEFB    254
9A5B  FC        1420          DEFB    252
9A5C  78        1430          DEFB    120
9A5D  30        1440          DEFB    48
9A5E  01        1450          DEFB    1
9A5F  03        1460          DEFB    3
9A60  03        1470          DEFB    3
9A61  07        1480          DEFB    7
9A62  07        1490          DEFB    7
                1500  ;
9A63  0F        1510          DEFB    15
9A64  1F        1520          DEFB    31
9A65  3F        1530          DEFB    63
9A66  7F        1540          DEFB    127
9A67  7F        1550          DEFB    127
9A68  7F        1560          DEFB    127
9A69  3F        1570          DEFB    63
9A6A  3F        1580          DEFB    63
9A6B  0F        1590          DEFB    15
9A6C  FF        1600          DEFB    255
9A6D  FF        1610          DEFB    255
9A6E  FF        1620          DEFB    255
9A6F  FF        1630          DEFB    255
9A70  F0        1640          DEFB    240
9A71  60        1650          DEFB    96
                1660  ;
93A2  60        1670          DEFB    96
9A43  60        1680          DEFB    96
9A74  FF        1690          DEFB    255
9A75  FF        1700          DEFB    255
9A76  F8        1710          DEFB    #F8
9A77  F3        1720          DEFB    #F3
9A78  E7        1730          DEFB    #E7
9A79  CF        1740          DEFB    #CF
9A7A  DE        1750          DEFB    #DE
9A7B  9C        1760          DEFB    #9C
                1770  ;
9A7C  9C        1780          DEFB    #9C
9A7D  DE        1790          DEFB    #DE
9A7E  CF        1800          DEFB    #CF
9A7F  E7        1810          DEFB    #E7
9A80  F3        1820          DEFB    243
9A81  F8        1830          DEFB    248
9A82  FF        1840          DEFB    255
9A83  FF        1850          DEFB    255
9A84  FF        1860          DEFB    255
9A85  FF        1870          DEFB    255
9A86  FF        1880          DEFB    255
                1890  ;
9A87  FF        1900          DEFB    255
9A88  0F        1910          DEFB    15
9A89  06        1920          DEFB    6
9A8A  06        1930          DEFB    6
9A8B  06        1940          DEFB    6
9A8C  FF        1950          DEFB    255
9A8D  FF        1960          DEFB    255
9A8E  1F        1970          DEFB    31
9A8F  CF        1980          DEFB    #CF
9A90  E7        1990          DEFB    #E7
9A91  F3        2000          DEFB    243
```

```
9A92  7B          2010            DEFB    #7B
9A93  39          2020            DEFB    #39
9A94  39          2030            DEFB    #39
9A95  7B          2040            DEFB    #7B
                  2050  ;
9A96  F3          2060            DEFB    243
9A97  E1          2070            DEFB    #E1
9A98  CE          2080            DEFB    #CE
9A99  1F          2090            DEFB    31
9A9A  FF          2100            DEFB    255
9A9B  FF          2110            DEFB    255
9A9C  F0          2120            DEFB    240
9A9D  F3          2130            DEFB    248
9A9E  FC          2140            DEFB    252
9A9F  FE          2150            DEFB    254
9AA0  FF          2160            DEFB    255
9AA1  7F          2170            DEFB    127
9AA2  7F          2180            DEFB    127
                  2190  ;
9AA3  3F          2200            DEFB    63
9AA4  1E          2210            DEFB    30
9AA5  0C          2220            DEFB    12
9AA6  80          2230            DEFB    128
9AA7  C0          2240            DEFB    192
9AA8  C0          2250            DEFB    192
9AA9  E0          2260            DEFB    #E0
9AAA  E0          2270            DEFB    #E0
9AAB  F0          2280            DEFB    240
9AAC  F8          2290            DEFB    #F8
9AAD  FC          2300            DEFB    #FC
9AAE  FE          2310            DEFB    #FE
9AAF  FE          2320            DEFB    #FE
9AB0  FE          2330            DEFB    #FE
                  2340  ;
9AB1  FC          2350            DEFB    #FC
9AB2  FC          2360            DEFB    #FC
9AB3  F0          2370            DEFB    #F0
                  2380  ;
9AB4  3F          2390  CARDAT    DEFB    63
9AB5  60          2400            DEFB    96
9AB6  D8          2410            DEFB    #D8
9AB7  A0          2420            DEFB    #A0
9AB8  B9          2430            DEFB    #B9
9AB9  AB          2440            DEFB    #AB
9ABA  B9          2450            DEFB    #B9
9ABB  85          2460            DEFB    #85
9ABC  8F          2470            DEFB    #8F
9ABD  85          2480            DEFB    #85
9ABE  81          2490            DEFB    #81
9ABF  80          2500            DEFB    #80
                  2510  ;
9AC0  81          2520            DEFB    #81
9AC1  83          2530            DEFB    #83
9AC2  81          2540            DEFB    #81
9AC3  85          2550            DEFB    #85
9AC4  8F          2560            DEFB    #8F
9AC5  85          2570            DEFB    #85
9AC6  81          2580            DEFB    #81
9AC7  80          2590            DEFB    #80
9AC8  81          2600            DEFB    #81
9AC9  85          2610            DEFB    #85
9ACA  8F          2620            DEFB    #8F
9ACB  85          2630            DEFB    #85
9ACC  81          2640            DEFB    #81
                  2650  ;
9ACD  83          2660            DEFB    #83
9ACE  81          2670            DEFB    #81
```

```
9ACF  80        2680          DEFB    #80
9AD0  80        2690          DEFB    #80
9AD1  C0        2700          DEFB    #C0
9AD2  60        2710          DEFB    96
9AD3  3F        2720          DEFB    63
9AD4  FF        2730          DEFB    255
9AD5  00        2740          DEFB    0
9AD6  00        2750          DEFB    0
9AD7  00        2760          DEFB    0
9AD8  00        2770          DEFB    0
9AD9  81        2780          DEFB    #81
9ADA  00        2790          DEFB    0
9ADB  42        2800          DEFB    66
9ADC  E7        2810          DEFB    #E7
                2820  ;
9ADD  42        2830          DEFB    66
9ADE  00        2840          DEFB    0
9ADF  00        2850          DEFB    0
9AE0  00        2860          DEFB    0
9AE1  81        2870          DEFB    #81
9AE2  00        2880          DEFB    0
9AE3  42        2890          DEFB    66
9AE4  E7        2900          DEFB    #E7
9AE5  42        2910          DEFB    66
9AE6  00        2920          DEFB    0
9AE7  00        2930          DEFB    0
9AE8  00        2940          DEFB    0
9AE9  42        2950          DEFB    66
9AEA  E7        2960          DEFB    #E7
9AEB  42        2970          DEFB    66
9AEC  00        2980          DEFB    0
9AED  81        2990          DEFB    #81
9AEE  00        3000          DEFB    0
9AEF  00        3010          DEFB    0
9AF0  00        3020          DEFB    0
                3030  ;
9AF1  00        3040          DEFB    0
9AF2  00        3050          DEFB    0
9AF3  FF        3060          DEFB    #FF
9AF4  FC        3070          DEFB    #FC
9AF5  06        3080          DEFB    6
9AF6  03        3090          DEFB    3
9AF7  01        3100          DEFB    1
9AF8  81        3110          DEFB    #81
9AF9  C1        3120          DEFB    #C1
9AFA  81        3130          DEFB    #81
9AFB  A1        3140          DEFB    #A1
9AFC  F1        3150          DEFB    #F1
9AFD  A1        3160          DEFB    #A1
9AFE  81        3170          DEFB    #81
9AFF  01        3180          DEFB    1
9B00  81        3190          DEFB    #81
                3200  ;
9B01  C1        3210          DEFB    #C1
9B02  81        3220          DEFB    #81
9B03  A1        3230          DEFB    #A1
9B04  F1        3240          DEFB    #F1
9B05  A1        3250          DEFB    #A1
9B06  81        3260          DEFB    #81
9B07  01        3270          DEFB    1
9B08  81        3280          DEFB    #81
9B09  A1        3290          DEFB    #A1
9B0A  F1        3300          DEFB    #F1
9B0B  A1        3310          DEFB    #A1
9B0C  81        3320          DEFB    #81
9B0D  DD        3330          DEFB    #DD
                3340  ;
```

```
9B0E  95          3350          DEFB     #95
9B0F  1D          3360          DEFB     29
9B10  05          3370          DEFB     5
9B11  1B          3380          DEFB     27
9B12  06          3390          DEFB     6
9B13  FC          3400          DEFB     #FC
                  3410  ;
                  3420  ;AREA DE IMAGEN PARA DATOS DE SPRITE EXPANDIDO
                  3430  ;LENGTH=4*5*64+(4*8)
                  3440  ;
9B14              3450  TELSPC  DEFS     1312
                  3460  ;
                  3470  ;LENGTH=5*4*64+(5*8)
                  3480  ;
A034              3490  CARSPC  DEFS     1320
A55C  0000        3500          DEFW     0
A55E  0000        3510          DEFW     0
A560  0000        3520          DEFW     0
A562  0000        3530          DEFW     0
```

Si ha seguido debidamente los últimos capítulos de un modo preciso, las imágenes que deben aparecer en su pantalla serán:



13

Color en alta resolución

Enhorabuena a todos aquellos que alguna vez hayan necesitado más de los dos colores que están normalmente disponibles en la celdilla de cada carácter. Albricias, sus deseos están a punto de verse cumplidos. Con las rutinas de este capítulo podrá cubrir un área de la pantalla de un ancho de 8 columnas y de una profundidad de hasta 24 líneas con atributos de color a 8 veces la resolución normal; es decir, un octeto de atributo para cada fila de cada celdilla del área de alta resolución.

La rutina funciona con el empleo de nuestra conocida y bien probada técnica de interrupciones vectorizadas bajo modo 2 de interrupciones (IM2) de nuestro propio gestor de interrupciones, como se describió en el capítulo 7.

Al recibir una interrupción, el Spectrum ejecutará una rutina de retardo adecuada mientras que espera para que el haz electrónico de la TV se aproxime al área de alta resolución. Desde este instante tenemos exactamente 224 T-estados para enviar una fila lo más larga posible de estos atributos de "alta resolución" al archivo de atributos normal. Los experimentos han demostrado que, con la restricción usual de que la rutina esté colocada en los 32 K superiores de la RAM para evitar retrasos debidos a la interferencia del ULA, es posible reemplazar los atributos de sólo 8 celdillas si tenemos que tener tiempo para ajustar nuestros punteros y contadores preparados para la siguiente fila de atributos.

Los nuevos atributos se almacenarán en un archivo de atributos de alta resolución especial, cuyo principio etiquetaremos con la variable de dos octetos HIATT. Para máxima flexibilidad, el área de alta resolución tendrá una longitud

variable y una posición vertical variable. Etiquetando la línea superior de la pantalla como cero y contando hacia abajo, la primera línea del área de alta resolución será STRTLN, y el número de líneas del área serán especificadas por la variable de un octeto DEPTH.

El gestor de interrupciones que he bautizado HIRES incluye dos operaciones de pila *(stack)* dentro de su bucle principal. Para asegurar que éstos no corran el riesgo de interferencia del ULA al acceder a los 16K más bajos de la RAM, la rutina almacena el valor de SP en VALSP y luego utiliza su propia pila de máquina de dos octetos, situada inmediatamente antes de la rutina y por tanto en los 32K superiores de RAM.

En el centro de la rutina hay una secuencia de 8 instrucciones consecutivas LDI para cargar los atributos del archivo de alta resolución al archivo normal. Este es el método más rápido posible para la transferencia de datos, tardando cada operación unos 16 T-estados. Esto puede compararse con los 21 T-estados usuales por repetición de la instrucción LDIR (esto sólo tarda 16 T-estados en su ejecución final, cuando BC = 0).

Los atributos de alta resolución son, por supuesto, puestos en el mapa exactamente de la misma forma que los octetos de atributo estándar. Los bits de 0 a 2 son para INK (tinta), los bits de 3 a 5 son para PAPER (papel), el bit 6 para BRIGHT (brillo) y al poner a uno el bit 7 se indica FLASH 1 (parpadeo).

He aquí pues el listado de HIRES, el gestor de interrupciones, seguido poco después por una rutina de inicialización. Recuerde las restricciones: el gestor de interrupciones, sus variables precedentes y el archivo de atributos de alta resolución deben todos estar en los 32K superiores de la RAM.

```
                    10 ;COLOR EN ALTA RESOLUCION
                    20 ;NOTA:POSICION POR ENCIMA DE LA BARRERA DE LOS 32K
                    30 ;VARIABLES Y ESPACIO PARA UN STACK DE MAQUINA DE DOS-OCTETOS
                     ;
                    40 ;UTILIZADAS POR EL GESTOR DE
                    50 ;INTERRUPCIONES
9029  00            60 STRTLN    DEFB      0
902A  18            70 DEPTH     DEFB      24
902B  0000          80 HIATT     DEFW      0
902D  0000          90 VALSP     DEFW      0
902F  0000         100           DEFW      0
                   110 ;
                   120 ;CONSERVAR REGISTROS
                   130 ;
9031  C5           140 HIRES     PUSH      BC
9032  D5           150           PUSH      DE
9033  E5           160           PUSH      HL
9034  F5           170           PUSH      AF
                   180 ;
                   190 ;ALMACENAR SP Y UTILIZAR LOS DOS OCTETOS QUE PRECEDEN
                   200 ;A ESTA RUTINA COMO UN STACK (PILA)
                   210 ;
9035  ED732D90     220           LD        (VALSP),SP
9039  313190       230           LD        SP,HIRES
                   240 ;
                   250 ;PRODUCIR UN RETARDO EXACTO
                   260 ;
903C  011802       270           LD        BC,#218
903F  0B           280 DELAY     DEC       BC
9040  78           290           LD        A,B
```

```
9041  B1          300           OR        C
9042  20FB        310           JR        NZ,DELAY
                  320 ;
                  330 ;CALCULAR EL NO. DE LINEAS DE TEXTO POR ENCIMA AREA ALT-RES
                  340 ;
9044  3A2990      350           LD        A,(STRTLN)
9047  87          360           ADD       A,A
9048  87          370           ADD       A,A
9049  87          380           ADD       A,A
904A  CA5890      390           JP        Z,GO4IT2
                  400 ;
                  410 ;ESPERAR HASTA QUE EL RAYO ALCANZA EL AREA ALT-RES
                  420 ;CADA BUCLE TARDA 224T-ESTADOS O UNA FILA DE TV
                  430 ;
904D  060F        440 SCANL     LD        B,15
904F  10FE        450 LN2       DJNZ      LN2
9051  00          460           NOP
9052  00          470           NOP
9053  C8          480           RET       Z
9054  3D          490           DEC       A
9055  C24D90      500           JP        NZ,SCANL
                  510 ;
                  520 ;CALCULAR DIRECCION DE ATRIBUTOS PARA (STRTLN,12)
                  530 ;
9058  6F          540 GO4IT2    LD        L,A
9059  3A2990      550           LD        A,(STRTLN)
905C  67          560           LD        H,A
905D  CB3C        570           SRL       H
905F  CB1D        580           RR        L
9061  CB3C        590           SRL       H
9063  CB1D        600           RR        L
9065  CB3C        610           SRL       H
9067  CB1D        620           RR        L
9069  110C58      630           LD        DE,#580C
906C  19          640           ADD       HL,DE
                  650 ;
                  660 ;PONER DIRECCION DE ATRIBUTOS EN DE
                  670 ;
906D  EB          680           EX        DE,HL
                  690 ;
                  700 ;TOMAR PRINCIPIO DE ARCHIVO DE ATRI. ALT-RES
                  710 ;
906E  2A2B90      720           LD        HL,(HIATT)
                  730 ;
                  740 ;A CUENTA EL NO. DE LINEAS QUE QUEDAN
                  750 ;
9071  3A2A90      760           LD        A,(DEPTH)
                  770 ;
                  780 ;BC CUENTA OCTETOS DE COLOR EN ALT-RES PARA ESTA LINEA
                  790 ;
9074  014000      800 NXLINE    LD        BC,64
                  810 ;
                  820 ;SALVAR DIRECCION DEL ATRI. A LA IZQ. DE ESTA LINEA
                  830 ;
9077  D5          840 NXTROW    PUSH      DE
                  850 ;
                  860 ;TRANSFERIR LOS OCHO ATRIBUTOS PARA ESTA FILA
                  870 ;
9078  EDA0        880           LDI
907A  EDA0        890           LDI
907C  EDA0        900           LDI
907E  EDA0        910           LDI
9080  EDA0        920           LDI
9082  EDA0        930           LDI
9084  EDA0        940           LDI
9086  EDA0        950           LDI
                  960 ;
```

```
                    970 ;RECUPERAR DIRECCION DEL ATRI. DE LA IZQ.
                    980 ;
9088  D1            990         POP       DE
                   1000 ;
                   1010 ;SI BC=0 ENTONCES FILA 7 ESTA ACABADA
                   1020 ;
9089  E29390       1030         JP        PO,LSTROW
                   1040 ;
                   1050 ;23T-ESTADOS ECUALIZADOR DE TIEMPO
                   1060 ;
908C  1800         1070         JR        $+2
908E  00           1080         NOP
908F  E6FF         1090         AND       #0FF
9091  18E4         1100         JR        NXTROW
                   1110 ;
                   1120 ;SUMAR 32 A DIRECC. DE ATRI. Y MOVERLA A SIGUIENTE LINEA
                   1130 ;
9093  EB           1140 LSTROW  EX        DE,HL
9094  0E20         1150         LD        C,32
9096  09           1160         ADD       HL,BC
9097  EB           1170         EX        DE,HL
9098  3D           1180         DEC       A
9099  C27490       1190         JP        NZ,NXLINE
                   1200 ;
                   1210 ;RECUPERAR SP, Y SACAR DEL STACK LOS OTROS REGISTROS
                   1220 ;
909C  ED7B2D90     1230         LD        SP,(VALSP)
90A0  F1           1240         POP       AF
90A1  E1           1250         POP       HL
90A2  D1           1260         POP       DE
90A3  C1           1270         POP       BC
                   1280 ;
                   1290 ;VOLVER DESDE INTERRUPCION
                   1300 ;NOTA: AQUI PODRIA INSERTAR UN SALTO A LA
                   1310 ;RUTINA DE INTERRUPCION DE ROM (HASTA #38)
90A4  FB           1320         EI
90A5  ED4D         1330         RETI
```

Las interrupciones pueden interceptarse por medio de una tabla de vectores de 257 octetos, empezando en un borde de página arbitrario que yo he tomado como #FE00. Esta técnica se detalló en el capítulo 7. Necesitamos una rutina para establecer la tabla de vectores y seleccionar el modo de interrupciones 2. Se llamará HIRON para indicar alta resolución activa (HIgh Resolution ON). Tras la rutina hay un fragmento corto para poner la instrucción de salto a HIRES en #FDFD, hacia la cual están vectorizadas todas las interrupciones.

```
                     10 ;INICIALIZAR INTERCEPCION DE INTERRUPCION
                     20 ;CON UNA TABLA DE VECTORES DE 257 OCTETOS EN #FE00
                     30 ;
                     40 ;SALIDA : BC=0,DE=#FF10,HL=#FF01
                     50 ;
8850  3EFE           60 HIRON    LD        A,#FE
8852  ED47           70          LD        I,A
8854  010001         80          LD        BC,#100
8857  67             90          LD        H,A
8858  69            100          LD        L,C
8859  57            110          LD        D,A
885A  58            120          LD        E,B
885B  36FD          130          LD        (HL),#FD
885D  EDB0          140          LDIR
885F  ED5E          150          IM        2
8861  C9            160          RET
```

```
                170 ;
                180 ;PRODUCIR EL SALTO A HIRES(ALT-RESOLUCION)
                190 ;DESPUES DE UNA INTERRUPCION
                200 :
FDFD            210 LABEL     ORG       #FDFD
FDFD  C3FDFD    220           JP        HIRES
8862            230           ORG       LABEL
```

Ahora que tiene la rutina de inicialización y el gestor de interrupciones, tiene todos los medios para obtener color en alta resolución, y ya es casi el momento para unos ejemplos.

El área de alta resolución máxima es de 8 x 192 celdillas, y, por tanto, por lo menos necesita 192 x 8 = 1.563 octetos de atributo o 1,5K de memoria. Tal y como está, la rutina HIRES posiciona esta área en el centro de la pantalla, empezando en la columna 12. Es posible alguna variación modificando la dirección de base del área de atributos contenida en la instrucción

```
LD      DE,#580C
```

que se encuentra poco después de la etiqueta GO4IT2. Puede que sea necesario algún ajuste de tiempos, pero en mi Spectrum he visto que la columna de más a la izquierda del área de alta resolución podía variarse fácilmente entre las columnas 0 y 13. Por tanto, para cubrir el área desde la columna 5 a la columna 12 inclusive, cambiar la instrucción a:

```
LD      DE,#5805
```

Al contrario de lo que ocurre en el generador de horizonte de pantalla completa del gestor de interrupciones del capítulo 9, HIRES no depende de una instrucción HALT antes de cada interrupción para mantener la estabilidad y evitar atributos de parpadeo. Una vez que hemos activado el color en alta resolución, estamos capacitados para realizar cualquier proceso que queramos sin preocuparnos de cuándo tiene lugar una interrupción, hasta que no la desactivemos.

Sustituyendo la instrucción

```
JP      #38
```

por la pareja

```
EI
RETI
```

al final de HIRES, provocaríamos un salto al gestor de interrupción estándar de la ROM, tras haberse generado cada cuadro de alta resolución. Sería seguro volver al BASIC, que funcionaría normalmente, aparte del hecho de que cuanto más baja y amplia sea el área de alta resolución, más lento se hará el BASIC.

Como primer ejemplo, he puesto HIRES apuntando al principio de la ROM y he indicado que visualice los primeros 1,5K del área de alta resolución completa para 256 cuadros de TV (0,12 segundos). La rutina se llama DEMO1 (¿un premio para mi imaginación?).

```
8869  AF          10 DEMO1    XOR       A
886A  326A88      20          LD        (STRTLN),A
                  30 ;
                  40 ;UTILIZAR LAS PRIMERAS 1.5K DE ROM  COMO
                  50 ;ARCHIVO DE COLOR ALT-RES
                  60 ;
886D  6F          70          LD        L,A
886E  67          80          LD        H,A
886F  3E18        90          LD        A,24
8871  327188     100          LD        (DEPTH),A
8874  227488     110          LD        (HIATT),HL
8877  CD7788     120          CALL      HIRON
                 130 ;
                 140 ;NOTA L=0 DESDE HIRON
                 150 ;
887A  45         160          LD        B,L
                 170 ;
                 180 ;PRODUCIR ALT-RES COLOR DURANTE
                 190 ;5.12 SEGUNDOS
887B  76         200 TSLP3    HALT
887C  10FD       210          DJNZ      TSLP3
                 220 ;
                 230 ;VOLVER A SELECCIONAR IM 1 PARA VOLVER AL BASIC
                 240 ;
887E  ED56       250          IM        1
8880  3E3F       260          LD        A,#3F
8882  ED47       270          LD        1,A
8884  C9         280          RET
```

La segunda demostración es ligeramente más exótica e implica el empleo de una subrutina DATPRP para generar un archivo de atributo de 25 líneas. Por supuesto que no todas estas líneas pueden utilizarse en un momento cualquiera, pero reciclando la etiqueta que apunta al "principio" del archivo, HIATT, hacia atrás o hacia adelante en pasos de 8 octetos, podemos hacer que los atributos de alta resolución roten arriba o abajo de la pantalla.

DATPRP, que significa PReParador de DATos, genera un archivo de atributos de 25 líneas, teniendo cada línea misma y cada fila de una línea un sólo color de papel y tinta blanca. Haciendo un esfuerzo para proporcionar algunos colores diferenciables, he utilizado la secuencia magenta, amarillo, azul, verde, blanco, rojo y cian para los colores del papel de filas sucesivas. Sin embargo, si es posible o no que distinga estos colores (o matices para el que esté leyendo en blanco y negro) dependerá de la resolución de su televisor, sobre el que desgraciadamente yo no tengo control alguno.

```
                  10 ;RUTINA DE DEMOSTRACION PARA CONFIGURAR UNA ARCHIVO DE COLOR

                  20 ;ALT-RES DE 25 LINEAS
                  30 ;ESPACIO NECESARIO=25*64=1984
                  40 ;
8901              50 TSTDAT   DEFS      1984
                  60 ;
90C1  210189      70 DATPRP   LD        HL,TSTDAT
```

```
                    80 ;
                    90 ;PARA 25 LINEAS
                   100 ;
90C4  0E19         110           LD        C,25
                   120 ;
                   130 ;UTILIZAR PAPEL NEGRO EN LA FILA 0
                   140 ;
90C6  AF           150           XOR       A
                   160 ;
                   170 ;UTILIZANDO SIEMPRE TINTA BLANCA
                   180 ;
90C7  F607         190 NXCOLR    OR        7
                   200 ;
                   210 ;CREAR UNA FILA DE ATRI. ALT-RES
                   220 ;
90C9  0608         230           LD        B,8
90CB  77           240 FL9       LD        (HL),A
90CC  23           250           INC       HL
90CD  10FC         260           DJNZ      FL9
                   270 ;
                   280 ;SIGUIENTE COLOR DE PAPEL
                   290 ;
90CF  C618         300           ADD       A,24
90D1  E638         310           AND       #38
90D3  20F2         320           JR        NZ,NXCOLR
                   330 ;
                   340 ;SIGUIENTE LINEA
                   350 ;
90D5  0D           360           DEC       C
90D6  20EF         370           JR        NZ,NXCOLR
90D8  C9           380           RET
```

DEMO2 hará que una pila multicolor se encoja hasta el "suelo" con los colores rotando hacia abajo en su trayectoria. La rutina funciona mejor con papel y borde totalmente negros.

```
                    10 ;CONFIGURAR ARCHIVO DE ATRI. DE 25 LINEAS Y ACTIVAR ALT-RES
                    20 ;
8BC8  CDC88B        30 DEMO2     CALL      DATPRP
8BCB  CDCB8B        40           CALL      HIRON
                    50 ;
                    60 ;
                    70 ;CICLO (HIATT) HACIA ATRAS LAS PRIMERAS OCHO FILAS,
                    80 ;HACIENDO EL COLOR FLUYA HACIA LA PARTE INFERIOR DE PANTALLA

                    90 ;OBSERVAR QUE POR ELLO NECESITAMOS 25-LINEAS EN VEZ DE 24
                   100 ;
8BCE  11F8FF       110           LD        DE,#FFF8
                   120 ;
                   130 ;DESPUES DE CADA DECREMENTO DE CICLO (DEPTH) E INCREMENTAR
                   140 ;(STRTLN), HACIENDO ENCOGERSE EL AREA HIRES HACIA ABAJO
                   150 ;
8BD1  0E18         160           LD        C,24
8BD3  79           170 TSLP      LD        A,C
8BD4  32D48B       180           LD        (DEPTH),A
8BD7  3E18         190           LD        A,24
8BD9  91           200           SUB       C
8BDA  32DA8B       210           LD        (STRTLN),A
8BDD  0608         220           LD        B,8
8BDF  211F8C       230           LD        HL,TSTDAT+34
8BE2  22E28B       240 NXRUN     LD        (HIATT),HL
8BE5  76           250           HALT
8BE6  19           260           ADD       HL,DE
8BE7  10F9         270           DJNZ      NXRUN
```

```
8BE9  0D        280          DEC       C
8BEA  20E7      290          JR        NZ,TSLP
                300 ;
                310 ;VOLVER A SELECCIONAR IM 1 PARA VOLVER AL BASIC
                320 ;
8BEC  ED56      330          IM        1
8BEE  3E3F      340          LD        A,#3F
8BF0  ED47      350          LD        I,A
8BF2  C9        360          RET
```

La rutina de demostración final de HIRES es bastante espectacular, y se llama (¡ya lo habrá adivinado!) DEMO3. De nuevo, utiliza la ROM para proporcionar un archivo de atributos bastante aleatorio, pero en esta ocasión tarda cerca de 30,72 segundos en ejecutar HIATT hacia atrás desde #600 hasta cero. El resultado es una estructura muy atrayente. Intente seguir su movimiento de izquierda a derecha, y luego mire de derecha a izquierda a través de él. ¿Observa alguna diferencia en su velocidad aparente?

```
                 10 ;DEFINIR LONGITUD TOTAL DEL AREA ALT-RES
8AE5  AF         20 DEMO3    XOR       A
8AE6  32E68A     30          LD        (STRTLN),A
8AE9  3E18       40          LD        A,24
8AEB  32EB8A     50          LD        (DEPTH),A
                 60 ;
                 70 ;ACTIVAR EL COLOR ALT-RES
                 80 ;
8AEE  CDEE8A     90          CALL      HIRON
                100 ;
                110 ;UTILIZAR LA ROM COMO UN ARCHIVO DE COLOR ALT-RES, PISANDO
                120 ;HIATT HACIA ATRAS DESDE #600 A CERO. OBSERVAR QUE  HAY ALLI

                130 ;#600 ATRI. ALT-RES
8AF1  2606      140          LD        H,6
                150 ;
                160 ;NOTA : L=0 DESDE HIRON
                170 ;
8AF3  22F38A    180 TSLP2    LD        (HIATT),HL
8AF6  76        190          HALT
8AF7  2D        200          DEC       L
8AF8  20F9      210          JR        NZ,TSLP2
8AFA  25        220          DEC       H
8AFB  20F6      230          JR        NZ,TSLP2
                240 ;
                250 ;VOLVER A SELECCIONAR IM 1 PARA VOLVER AL BASIC
                260 ;
8AFD  3E3F      270          LD        A,#3F
8AFF  ED56      280          IM        1
8B01  ED47      290          LD        I,A
8B03  C9        300          RET
```

Para concluir este capítulo debería indicar que el formato de arriba no es la única estructura posible para color en alta resolución. Para empezar, si estuviera dispuesto a tener sólo un octeto de atributo por línea de exploración, no necesitaría hacer un volcado de un archivo de color en alta resolución de uno en uno, y podría cambiar la instrucción de 16 T-estados:

```
LDI
```

por dos instrucciones como:

```
LD      (DE),A
INC     E
```

donde el acumulador contendría el atributo de fila actual y cada pareja tardaría 11 T-estados. De esta forma probablemente aumentaría el ancho del área de alta resolución tres o cuatro columnas.

14

Producción de imágenes en pantalla completa con el borde

Por muy espectacular que fuese, el horizonte de pantalla completa generado en el capítulo 9 "cambiando" el color del borde cien veces por segundo (a 100 Hz) fue un simple rasguño en la superficie de los efectos potenciales de un control del color directo del color del borde. En este capítulo, llevaré a cabo toda la potencia del "cambio" del borde a alta velocidad con una serie de rutinas que le permitirán obtener diez columnas distintas en el borde, teniendo cada fila de ellas cualquiera de los 8 colores. Las velocidades del "cambio" implicadas llevarán al procesador Z-80 hasta sus límites, con un intervalo de 12 T-estados entre los cambios de color y una frecuencia por encima de una fila de TV de 156250 Hz.

Los principios implicados en el "generador de imagen" son muy similares a los de nuestro horizonte de pantalla completa. Utilizamos interrupciones vectorizadas bajo el modo 2 de interrupción para nuestro gestor de interrupción hecho a la medida, que después de ejecutar los retardos apropiados según el haz electrónico de la TV desciende por la pantalla, se precipita a través de una tabla de valores del borde como alma que lleva el diablo, cambiando siempre el color del borde exactamente en las mismas etapas de la generación de cada cuadro de TV.

Para entrar en más detalle, recuerde que el tiempo que tarda la TV en generar una fila de pantalla es exactamente de 224 T-estados. Ahora bien, la forma más rápida de transferir los datos de una tabla al puerto 254 es con el uso de una secuencia de instrucciones OUTI, cada una de las cuales tarda 16

T-estados. Como esta instrucción se utiliza tan pocas veces, me tomaré la molestia de detallarle su funcionamiento.

La pareja de HL contiene la dirección del octeto de datos; el registro C contiene el octeto de menor peso de la dirección del puerto, y el registro B proporciona el octeto de mayor peso de dicha dirección. En cada ejecución, el registro B se decrementa, se forma la dirección de puerto, se envía al puerto el octeto de datos de HL y se incrementa HL. Si B llega a ser cero, se pone a uno la bandera cero; en caso contrario, continúa a cero.

La teoría parece indicar que podemos obtener INT (224/16) = 14 columnas de borde en la pantalla, pero debemos recordar que el haz de la TV tarda un cierto espacio de tiempo en un "retroceso" horizontal desde el extremo derecho de la pantalla hasta el izquierdo. Los experimentos revelan que esta transversal ocupa al haz durante unos 64 T-estados, 2/7 o más o menos el 29 por 100 de su tiempo.

Por consiguiente, tenemos suficiente tiempo para cambiar el color del borde diez veces, mientras que el haz atraviesa la pantalla de izquierda a derecha, y esto tiene como resultado que cada "columna de borde" tenga un ancho de cuatro columnas de texto.

Llamo a este gestor de interrupciones BORPIC por razones obvias (BORder PICture generator). Los datos del borde para BORPIC se almacenarán en cualquier parte que desee de los 32K superiores de la RAM, y deben ser apuntados por la variable PICDAT de dos octetos. Haremos el formato de los datos del borde como sigue:

PRIMER OCTETO: NUMERO DE LINEAS DEL BORDE

luego los datos para cada "línea de borde":

PRIMER OCTETO: NUMERO DE FILAS DE TV EN ESTA LINEA DE BORDE

DIEZ OCTETOS: Los valores del borde para cada una de las 10 columnas de borde.

El concepto de líneas de borde es análogo al de líneas de texto, con la excepción de que las líneas de borde tienen un número variable de filas (hasta 256) y las filas continúan por encima y por debajo del área del texto. Se ve fácilmente que el área de almacenamiento que se necesita para una imagen con n líneas de borde viene dado por:

Memoria que se necesita: (11 * n) + 1

En el listado de BORPIC verá que he reservado espacio para diez líneas de datos de borde y lo he etiquetado BORSTR. Se utilizará esta área más tarde, pero de momento he aquí el listado. ¡Por favor, no lo ejecute hasta que le explique cómo hacerlo!

```
8D15  0000        10 PICDAT   DEFW     0
                  20 ;
                  30 ;PICDAT CONTIENE LA DIRECCION DE DATOS DEL BORDE
                  40 ;ESPACIO NECESARIO=1+11*(NO. DE LINEAS DEL BORDE)
                  50 ;
8D17              60 BORSTR   DEFS     111
                  70 ;
                  80 ;EL GENERADOR DEL CUADRO DEL BORDE CONSERVA LOS REGISTROS
                  90 ;
8D86  C5         100 BORPIC   PUSH     BC
8D87  D5         110          PUSH     DE
8D88  E5         120          PUSH     HL
8D89  F5         130          PUSH     AF
8D8A  08         140          EX       AF,AF'
8D8B  F5         150          PUSH     AF
                 160 ;
                 170 ;ESPERAR 38T-ESTADOS
                 180 ;
8D8C  E3         190          EX       (SP),HL
8D8D  E3         200          EX       (SP),HL
                 210 ;
                 220 ;ESPERAR DURANTE (FLYBAK+1) FILAS DE TV MIENTRAS EL RAYO
                 230 ;ALCANZA LA PARTE SUPERIOR DE LA PANTALLA
                 240 ;
8D8E  3E1F       250 FLYBAK   LD       A,31
8D90  060F       260 SCANM    LD       B,15
8D92  10FE       270 LN4      DJNZ     LN4
8D94  00         280          NOP
8D95  A7         290          AND      A
8D96  C8         300          RET      Z
8D97  3D         310          DEC      A
8D98  C2908D     320          JP       NZ,SCANM
                 330 ;
                 340 ;AJUSTE DE TIEMPO DE 5T-ESTADOS
                 350 ;
8D9B  C0         360          RET      NZ
                 370 ;
                 380 ;HL SENALA LOS DATOS DE LA IMAGEN
                 390 ;
8D9C  2A158D     400          LD       HL,(PICDAT)
                 410 ;
                 420 ;C CONSERVA EL VALOR DEL PUERTO
                 430 ;
8D9F  0EFE       440          LD       C,#FE
                 450 ;
                 460 ;A CUENTA LAS LINEAS DE DATOS DEL BORDE
                 470 ;
8DA1  7E         480          LD       A,(HL)
8DA2  23         490          INC      HL
                 500 ;
                 510 ;ALMACENAR EL COMIENZO DE ESTA FILA DE DATOS EN DE
                 520 ;
8DA3  54         530          LD       D,H
8DA4  5D         540          LD       E,L
                 550 ;
                 560 ;EL NUCLEO DE 10 CAMBIOS SUCESIVOS DE BORDE
                 570 ;
8DA5  EDA3       580 NXTRW    OUTI
8DA7  EDA3       590          OUTI
8DA9  EDA3       600          OUTI
8DAB  EDA3       610          OUTI
8DAD  EDA3       620          OUTI
8DAF  EDA3       630          OUTI
8DB1  EDA3       640          OUTI
8DB3  EDA3       650          OUTI
8DB5  EDA3       660          OUTI
8DB7  EDA3       670          OUTI
```

```
                    680  ;
                    690  ;GENERAR LA SIGUIENTE FILA DE LA PANTALLA
                    700  ;
8DB9  3D            710          DEC       A
8DBA  CAC88D        720          JP        Z,NXTLN
8DBD  62            730          LD        H,D
8DBE  6B            740          LD        L,E
                    750  ;
                    760  ;PRIMERO ESPERAR 30T-ESTADOS
                    770  ;
8DBF  0600          780          LD        B,0
8DC1  0600          790          LD        B,0
8DC3  1800          800          JR        $+2
8DC5  00            810          NOP
8DC6  18DD          820          JR        NXTRW
                    830  ;SIGUIENTE LINEA DE DATOS DEL BORDE
                    840  ;
8DC8  08            850  NXTLN   EX        AF,AF'
8DC9  3D            860          DEC       A
                    870  ;
                    880  ;ECUALIZADOR DE 7T-ESTADOS
                    890  ;
8DCA  E6FF          900          AND       #FF
8DCC  C2CC8D        910          JP        NZ,NXTLN2
                    920  ;
                    930  ;RECUPERAR REGISTROS Y VOLVER DESDE LA INTERRUPCION
                    940  ;
8DCF  F1            950          POP       AF
8DD0  08            960          EX        AF,AF'
8DD1  F1            970          POP       AF
8DD2  E1            980          POP       HL
8DD3  D1            990          POP       DE
8DD4  C1           1000          POP       BC
8DD5  FB           1010          EI
8DD6  ED4D         1020          RETI
```

Como dije, BORPIC debe funcionar como un gestor de interrupciones que utiliza IM2. Emplearemos la tabla usual de vectores de 257 octetos para una instrucción de salto a BORPIC, una técnica descrita con detalle en el capítulo 7. Como siempre, es cosa suya dónde pone la instrucción de salto y en qué zona de la página coloca la tabla. Si está indeciso, ¿por qué no pone la tabla de vectores en #FE00 y la instrucción de salto en #FDFD? Esto se puede conseguir añadiendo las líneas:

```
       LABEL  ORG     #0FDFD
C30000        JP      BORPIC
              ORG     LABEL
```

Entonces se puede utilizar la rutina HIRON del capítulo 13 para establecer la tabla de vectores y seleccionar el modo dos de interrupciones (se empleó previamente para establecer la misma tabla para la rutina de color en alta resolución HIRES).

Ahora pues, tenemos las rutinas necesarias para hacer que el cuadro del borde sea una realidad. El generador es extremadamente sensible a las variaciones de tiempos, así es que, como en el caso del generador de horizonte de pantalla completa del capítulo 9, siempre tenemos que volver a una instrucción HALT antes de una interrupción. De este modo tenemos una variación máxima de 4 T-estados, el tiempo que tarda el procesador en ejecutar un NOP, que es lo que hace repetidamente cuando se alcanza la instrucción HALT.

Refiriéndome al listado de BORPIC, verá que hay una etiqueta misteriosa en la línea:

```
FLYBAK LD    A,31
```

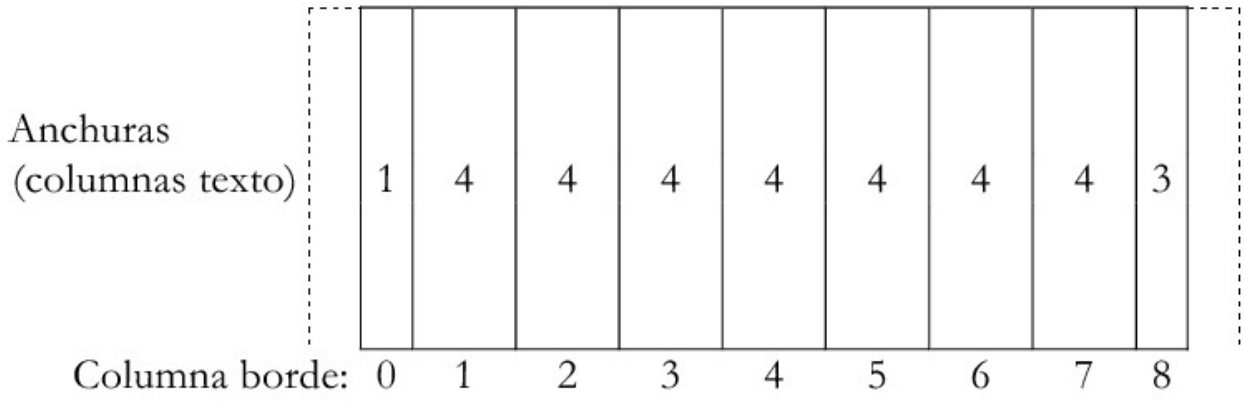
(El *flyback* es el retroceso del haz de electrones del televisor al principio de una nueva línea en el barrido de la pantalla).

Utilizaremos esto para hacer ajustes en la altura a la que empieza el cuadro del borde en la pantalla. El valor que se carga en A es el número de filas de TV que tiene que esperar la rutina antes de comenzar el proceso de los datos del borde. Si desea que un cuadro empiece justo en la parte superior de la pantalla, ajuste (FLYBAK + 1) hasta que lo haga. El valor resultante dependerá de su aparato particular de TV así como de su Spectrum.

En el caso de mi portátil de color, descubrí que al cargar (FLYBAK + 1) con 31 llevaba el haz hasta la parte superior de la pantalla. Entonces había 32 filas del borde superior que quedaban antes del área del texto, y, desde luego, es lo más general que la altura de este "margen superior" más el valor en (FLYBAK + 1) sea 63. Por supuesto, hay 192 filas en el área del texto. Debajo de esto está el "margen inferior", cuya altura visible varía con los distintos televisores y Spectrums, pero en mi sistema es de unas 44 filas de profundidad, dando un total de 32 + 192 + 44 = 268 filas en la pantalla.

Aunque ahora podemos producir una imagen estable en el borde de la pantalla, será algo incompleta, a menos que podamos mostrar las partes de las "columnas de borde" y de las "líneas de borde" que están, por decirlo de algún modo, "detrás" del área de texto. Lo que necesitamos es una rutina que examine los datos para la parte "invisible" del borde y establezca los atributos del papel en todas las celdillas del texto apropiadamente, para que después de activar el generador del cuadro parezca que tenemos una imagen de pantalla completa y la barrera entre el área del texto y el borde no se pueda detectar. Ahora desarrollaré una rutina semejante y la llamaré ATTSET.

Nueve de las columnas del borde se superponen al área del texto de la siguiente forma:



Para los propósitos de esta rutina supondremos que el área del texto ha sido dividida en exactamente seis líneas de borde, cada una de las cuales tiene 32 filas de altura. Si prefiere tener líneas de borde más estrechas, o posiblemente

líneas de borde de altura variable, ATTSET se ajusta con facilidad. No se le habrá escapado el detalle de que 32 filas = 4 líneas de texto en altura; por tanto, con este formato acabamos produciendo "celdillas de borde" que son un cuadrado de 4 celdillas de texto.

Los principios implicados en ATTSET son realmente muy sencillos; entramos en la rutina con HL apuntando a los datos del borde para la primera columna de la primera línea de borde del área del texto, ATTSET toma este octeto, lo multiplica por 8 para obtener un valor de PAPER, efectúa una operación OR con este valor y el de INK de la primera celdilla y a continuación pone el resultado en el primer octeto del archivo de atributos. Se toma el siguiente valor del borde y se utiliza para las siguientes cuatro columnas de texto de la línea 0, y este procedimiento se repite para las siguientes 6 columnas de borde. El valor de la columna 8 del borde se utiliza para las tres columnas de texto finales, y esta línea de datos de borde se vuelve a procesar tres veces para las líneas de texto que quedan de esta línea de borde.

Todo el procedimiento arriba mencionado se repite para cada una de las cinco líneas de borde que quedan en el área del texto. He aquí el listado de ATTSET, seguido por una demostración:

```
                10 ;RUTINA PARA PONER ATRI. DE PAPEL DADOS UNOS DATOS DEL BORDE

                20 ;ENTRADA : HL=DIRECCION DEL PRIMER OCTETO DE DATOS DEL BORDE

                30 ;EN EL AREA DE TEXTO COMO SE PRODUJO POR "EXPAND"
                40 ;SALIDA:BC=0,HL=#5B00
                50 ;
                60 ;HL SENALA AL COMIENZO DE LOS ATRIBUTOS
8F37  110058    70 ATTSET    LD        DE,#5800
8F3A  EB        80            EX        DE,HL
                90 ;
               100 ;B CUENTA LAS LINEAS DE BORDE
               110 ;
8F3B  0606      120           LD        B,6
               130 ;
               140 ;C CUENTA LAS LINEAS DE ATRIBUTOS (4 POR LINEA DE BORDE)
               150 ;
8F3D  0E04      160 NXTLN3    LD        C,4
               170 ;
               180 ;ALMACENAR DIRECCION DE DATOS DE BORDE
               190 ;
8F3F  D5        200 NXT14     PUSH      DE
8F40  C5        210           PUSH      BC
               220 ;C CONTIENE LA MASCARA PARA EL PAPEL
               230 ;
8F41  0E38      240           LD        C,#38
               250 ;
               260 ;TOMAR OCTETO DE BORDE MULT. POR 8 PARA OBTENER BITS DE PAPE
                   L
               270 ;
8F43  1A        280           LD        A,(DE)
8F44  07        290           RLCA
8F45  07        300           RLCA
8F46  07        310           RLCA
               320 ;
               330 ;UTILIZAR TINTA DE LA CELDA  CON NUESTRO PAPEL PARA FORMAR
               340 ;UN NUEVO OCTETO DE ATRIBUTO
               350 ;
8F47  AE        360           XOR       (HL)
```

```
8F48  A1          370          AND       C
8F49  AE          380          XOR       (HL)
8F4A  77          390          LD        (HL),A
                  400  ;
                  410  ;HACER LO MISMO PARA LAS 7 SIGUIENTES COLUMNAS DE BORDE
                  420  ;QUE TIENEN CUATRO COLUMNAS DE ANCHURA
                  430  ;
8F4B  13          440          INC       DE
8F4C  2C          450          INC       L
8F4D  0607        460          LD        B,7
8F4F  1A          470  NXT12   LD        A,(DE)
8F50  07          480          RLCA
8F51  07          490          RLCA
8F52  07          500          RLCA
8F53  AE          510          XOR       (HL)
8F54  A1          520          AND       C
8F55  AE          530          XOR       (HL)
8F56  77          540          LD        (HL),A
8F57  2C          550          INC       L
8F58  AE          560          XOR       (HL)
8F59  A1          570          AND       C
8F5A  AE          580          XOR       (HL)
8F5B  77          590          LD        (HL),A
8F5C  2C          600          INC       L
8F5D  AE          610          XOR       (HL)
8F5E  A1          620          AND       C
8F5F  AE          630          XOR       (HL)
8F60  77          640          LD        (HL),A
8F61  2C          650          INC       L
8F62  AE          660          XOR       (HL)
8F63  A1          670          AND       C
8F64  AE          680          XOR       (HL)
8F65  77          690          LD        (HL),A
8F66  2C          700          INC       L
8F67  13          710          INC       DE
                  720  ;
                  730  ;SIGUIENTE COLUMNA DEL BORDE
                  740  ;
8F68  10E5        750          DJNZ      NXT12
                  760  ;
                  770  ;AHORA HACER LAS TRES COLUMNAS DE ATRI. DE MAS A LA DERECHA
                  780  ;
8F6A  1A          790          LD        A,(DE)
8F6B  07          800          RLCA
8F6C  07          810          RLCA
8F6D  07          820          RLCA
8F6E  AE          830          XOR       (HL)
8F6F  A1          840          AND       C
8F70  AE          850          XOR       (HL)
8F71  77          860          LD        (HL),A
8F72  2C          870          INC       L
8F73  AE          880          XOR       (HL)
8F74  A1          890          AND       C
8F75  AE          900          XOR       (HL)
8F76  77          910          LD        (HL),A
8F77  2C          920          INC       L
8F78  AE          930          XOR       (HL)
8F79  A1          940          AND       C
8F7A  AE          950          XOR       (HL)
8F7B  77          960          LD        (HL),A
8F7C  23          970          INC       HL
                  980  ;
                  990  ;HL SENALA AHORA LA SIGUIENTE LINEA DE ATRIBUTOS
                 1000  ;
8F7D  C1         1010          POP       BC
                 1020  ;
                 1030  ;REPETIR PARA LAS TRES LINEAS DE ATRI. SIGUIENTES
```

```
                    1040  ;
8F7E  0D            1050          DEC       C
8F7F  2803          1060          JR        Z,OUT1
8F81  D1            1070          POP       DE
8F82  18BB          1080          JR        NXT14
                    1090  ;
                    1100  ;ELIMINAR LA ULTIMA ENTRADA DEL STACK (PILA)
                    1110  ;
8F84  F1            1120  OUT1    POP       AF
                    1130  ;INCREMENTAR EL PUNTERO A SIGUIENTE LINEA DE DATOS DE BORDE
                    1140  ;
8F85  13            1150          INC       DE
8F86  13            1160          INC       DE
8F87  13            1170          INC       DE
                    1180  ;
                    1190  ;REPETIR PARA 5 LINEAS DE BORDE
                    1200  ;
8F88  10B3          1210          DJNZ      NXTLN3
8F8A  C9            1220          RET
```

Como demostración de BORPIC y ATTSET, obtendremos una estructura multicolor de ocho líneas de borde por ocho columnas, teniendo cada línea una altura de 32 filas, como requiere ATTSET. Las primeras 32 filas por encima del área del texto son necesarias para la primera línea, así es que tenemos que poner (FLYBAK + 1) a 63 – 32 = 31 para empezar a generar la imagen en el sitio correcto. En BORSTR se preparan los datos del borde, y el espacio que se necesita será de 1 + (8 × 11) = 89 octetos, que están dentro de los 111 octetos que reservamos en BORPIC.

Puesto que tenemos un octeto para el número de líneas, once para la primera línea de borde y uno para la altura del segundo, el primer valor de borde de la segunda línea de borde estará en (BORSTR + 1 + 11 + 1) = (BORSTR + 13). Por lo cual establecemos los atributos de PAPER con:

```
LD      HL,BORSTR+13
CALL    ATTSET
```

Los comentarios del listado en ensamblador proporcionan suficiente explicación del resto de la rutina, llamada BPDEMO.

```
                     10 ;DEMOSTRACION PARA BORPIC Y ATTSET
                     20 ;
8B7F  3E1F           30 BPDEMO   LD        A,31
8B81  32828B         40          LD        (FLYBAK+1),A
                     50 ;
                     60 ;CONSTRUIR DATOS DE BORDE EN BORSTR
                     70 ;
8B84  21848B         80          LD        HL,BORSTR
8B87  22878B         90          LD        (PICDAT),HL
                    100 ;
                    110 ;COMENZAR CON BORDE NEGRO
                    120 ;
8B8A  AF            130          XOR       A
                    140 ;
                    150 ;DENOTAR "8 LINEAS DE BORDE"
                    160 ;
8B8B  3608          170          LD        (HL),8
8B8D  23            180          INC       HL
                    190 ;
                    200 ;BUCLE PARA GENERAR DATOS PARA CADA LINEA DE BORDE
```

```
                    210 ;INDICAR "32 FILAS EN ESTA LINEA"
8B8E  3620          220 NXBLIN    LD        (HL),32
8B90  23            230           INC       HL
                    240 ;HACER NEGRA LA PRIMERA COLUMNA DEL BORDE
                    250 ;
8B91  3600          260           LD        (HL),0
8B93  23            270           INC       HL
                    280 ;
                    290 ;PASAR POR OCHO COLORES PARA LAS OCHO COLUMNAS CENTRALES
8B94  0608          300           LD        B,8
8B96  77            310 NXBCLM    LD        (HL),A
8B97  C603          320           ADD       A,3
8B99  E607          330           AND       7
8B9B  23            340           INC       HL
8B9C  10F8          350           DJNZ      NXBCLM
                    360 ;
                    370 ;HACER NEGRA LA ULTIMA COLUMNA
                    380 ;
8B9E  70            390           LD        (HL),B
8B9F  23            400           INC       HL
                    410 ;
                    420 ;CAMBIAR COLOR DE SEGUNDA COLUMNA A LA SIGUIENTE EN LA SERIE

8BA0  C603          430           ADD       A,3
8BA2  E607          440           AND       7
8BA4  20E8          450           JR        NZ,NXBLIN
                    460 ;
                    470 ;PONER ATRI. DE PAPEL PARA HACER COINCIDIR CON DATOS DEL
                    480 ;BORDE
                    490 ;
8BA6  21918B        500           LD        HL,BORSTR+13
8BA9  CDA98B        510           CALL      ATTSET
                    520 ;
                    530 ;ACTIVAR LA IMAGEN DEL BORDE
                    540 ;
8BAC  CDAC8B        550           CALL      HIRON
                    560 ;
                    570 ;GENERARLA DURANTE 5.12 SEGUNDOS
                    580 ;NOTAR QUE B=0 DESDE ATTSET
8BAF  76            590 TSLP9     HALT
8BB0  10FD          600           DJNZ      TSLP9
                    610 ;
                    620 ;VOLVER A SELECCIONAR IM1 PARA EL BASIC
                    630 ;
8BB2  ED56          640           IM        1
8BB4  3E3F          650           LD        A,#3F
8BB6  ED47          660           LD        I,A
8BB8  C9            670           RET
```

Como última rutina de utilidad para BORPIC, pensé que sería práctico tener una que genere los datos de borde, dando una serie de estructuras de bits y valores de color que llamaré colectivamente "datos de borde compactos".

La rutina EXPAND nos permitirá especificar cualquier número de líneas de borde, teniendo cada una de ellas cualquier altura (hasta 256 en cada caso) y utilizar dos colores para cada línea de borde, que entonces será definida por los diez bits de más a la izquierda de dos "octetos de datos compactos". Cada uno de estos diez bits corresponde a una columna de borde de una línea de borde. Utilizando un sistema análogo a los valores de INK y PAPER del BASIC del Spectrum, dejaremos que los dos colores disponibles de cada línea de borde sean BINK y BAPER, indicando BINK una celdilla de borde con un 1, y una celdilla BAPER con un 0.

Para reducir la cantidad de datos que se necesitan para una imagen, sólo

especificaremos los valores de BINK y BAPER al principio de los datos y siempre que queramos cambiar sus valores según vamos trabajando hacia abajo de la pantalla. Necesitamos algún modo de decir a EXPAND que empiece a utilizar nuevos colores, y probablemente la forma más fácil para ello es el empleo de los seis bits que sobran en la parte derecha de los datos para una línea. Los pondremos a #3F para un cambio de colores, a continuación seguimos este octeto con otros dos que contienen los valores BINK y BAPER respectivamente. Poniendo los mismos bits a #3E se indicará "fin de datos". Este procedimiento se aclarará con los ejemplos que siguen al listado de EXPAND.

```
                    10 ;RUTINA PARA EXPANDIR LOS DATOS DEL BORDE PARA BORPIC
                    20 ;
                    30 ;ENTRADA:HL=COMIENZO DE DATOS DEL BORDE COMPACTO
                    40 ;SALIDA:HL=DIRECCION DEL PRIMER VALOR DEL BORDE DE PRIMERA
                    50 ;LINEA DEL BORDE EN AREA DE TEXTO
                    60 ;A=0,B=BAPER COLOR,C=BINK COLOR
                    70 ;DE=SIGUIENTE OCTETO TRAS  DATOS COMPACTOS
                    80 ;CONSTRUIR LOS DATOS EN EL ESPACIO EN BORSTR
                    90 ;
8F5A  115A8F       100 EXPAND    LD        DE,BORSTR
                   110 ;
                   120 ;TRANSFERIR "NO. DE LINEAS"
                   130 ;
8F5D  EDA0         140           LDI
                   150 ;
                   160 ;UN REG. CONTENDRA LAS FILAS DE TV VISUALIZADAS HASTA EL
                   170 ;MOMENTO
                   180 ;
8F5F  AF           190           XOR       A
8F60  08           200           EX        AF,AF'
                   210 ;
                   220 ;C=BINK COLOR
                   230 ;
8F61  4E           240 NEWCOL    LD        C,(HL)
8F62  23           250           INC       HL
                   260 ;
                   270 ;B=BAPER COLOR
                   280 ;
8F63  46           290           LD        B,(HL)
8F64  23           300           INC       HL
                   310 ;
                   320 ;SI ESTAMOS EN EL AREA DE TEXTO ENTONCES ALMACENAR
                   330 ;DIRECCION DE DATOS EXPANDIDOS
                   340 ;
8F65  08           350 NXTWD     EX        AF,AF'
                   360 ;
                   370 ;EL SIGUIENTE VALOR PUEDE ALTERARSE PARA CAMBIAR EL VALOR
                   380 ;DE PROFUNDIDAD DEL BORDE SUPERIOR
8F66  FE21         390 TPMRGN    CP        33
8F68  C26C8F       400           JP        NZ,NYET
8F6B  D5           410           PUSH      DE
                   420 ;
                   430 ;INCREMENTAR CONTADOR DE FILA
                   440 ;
8F6C  86           450 NYET      ADD       A,(HL)
8F6D  08           460           EX        AF,AF'
                   470 ;
                   480 ;TRANSFERIR PROFUNDIDAD DE ESTA LINEA (EN FILAS)
                   490 ;
8F6E  EDA0         500           LDI
8F70  03           510           INC       BC
```

```
                520 ;
                530 ;TOMAR PRIMER OCTETO DE DATOS COMPACTOS
                540 ;
8F71  7E        550         LD      A,(HL)
8F72  EB        560         EX      DE,HL
8F73  D5        570         PUSH    DE
                580 ;
                590 ;PARA CADA UNO DE LOS OCHO BITS ..
                600 ;
8F74  1E08      610         LD      E,8
                620 ;
                630 ;PONER OCTETO DE BAPER EN DATOS DE BORDE SI BIT = 1
                640 ;
8F76  17        650 ABC     RLA
8F77  70        660         LD      (HL),B
8F78  D27C8F    670         JP      NC,PAPER
                680 ;
                690 ;SI NO INSERTAR UN OCTETO DE BINK
                700 ;
8F7B  71        710         LD      (HL),C
                720 ;
                730 ;DESPLAZARSE AL SIGUIENTE BIT
                740 ;
8F7C  23        750 PAPER   INC     HL
8F7D  1D        760         DEC     E
8F7E  C2768F    770         JP      NZ,ABC
                780 ;
                790 ;TOMAR EL SEGUNDO OCTETO COMPACTO
                800 ;
8F81  D1        810         POP     DE
8F82  13        820         INC     DE
8F83  1A        830         LD      A,(DE)
                840 ;
                850 ;ELEGIR BAPER O BINK PARA CADA UNO DE DOS BITS MAS A LA IZQ
8F84  17        860         RLA
8F85  70        870         LD      (HL),B
8F86  D28A8F    880         JP      NC,PAPER2
8F89  71        890         LD      (HL),C
8F8A  17        900 PAPER2  RLA
8F8B  23        910         INC     HL
8F8C  70        920         LD      (HL),B
8F8D  D2918F    930         JP      NC,PAPER3
8F90  71        940         LD      (HL),C
8F91  23        950 PAPER3  INC     HL
                960 ;
                970 ;COMPROBAR BITS 0-5 DEL SEGUNDO OCTETO COMPACTO DE DATOS
                980 ;
8F92  1A        990         LD      A,(DE)
8F93  13       1000         INC     DE
8F94  EB       1010         EX      DE,HL
               1020 ;
               1030 ;#3F INDICA NECESIDAD DE NUEVOS COLORES
               1040 ;
8F95  F6C0     1050         OR      #C0
8F97  3C       1060         INC     A
8F98  28C7     1070         JR      Z,NEWCOL
               1080 ;
               1090 ;#3E INDICA FINAL DE DATOS
               1100 ;
8F9A  3C       1110         INC     A
8F9B  C2658F   1120         JP      NZ,NXTWD
               1130 ;
               1140 ;EN CUYO CASO RECUPERAR DIRECCION POR ATTSET
               1150 :
8F9E  E1       1160         POP     HL
8F9F  23       1170         INC     HL
8FA0  C9       1180         RET
```

Fíjese en la línea:

```
TPMRGN    CP 32
```

El valor de esta instrucción es el número de filas del cuadro del borde que está encima del área del texto, y siempre debe ser igual a 63 – (FLYBAK + l), es decir, la suma de los dos valores de las etiquetas FLYBAK en BORPIC y TPMRGN en EXPAND debería ser 63:

```
(FLYBAK + 1) + (TPMRGN + 1) = 63
```

TPMRGN, en caso de que no lo haya adivinado, representa margen superior (ToP MaRGiN), EXPAND emplea este valor para encontrar la dirección correcta de los datos de borde que hay que utilizar como valor de entrada a ATTSET, donde se puede aplicar este uso. Luego almacena este valor y lo devuelve en HL dispuesto para un uso inmediato, si así se desea, con ATTSET.

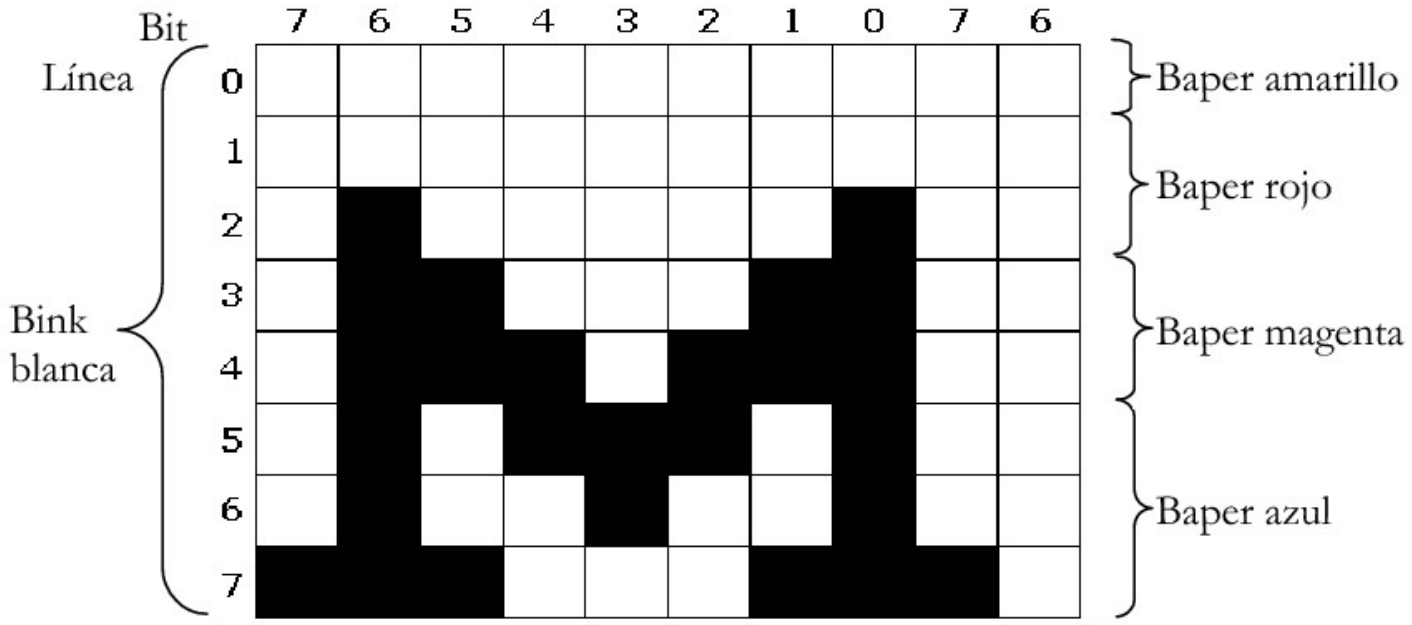
Como demostración sencilla de EXPAND, he escrito una rutina para mostrar una burda pero gran manifestación de un logotipo. Utilizaremos una reja de 10 × 8 celdillas de borde cuadradas, así es que necesitamos que el margen superior sea de una altura de 32 filas. Esto se establece con:

```
LD    A,32
LD    (TPMRGN + 1), A
LD    A, 31
LD    (FLYBAK + 1), A
```

EXPAND construye los datos en el espacio previamente reservado en BORSTR, y así debemos hacer que PICDAT apunte a él:

```
LD    HL,BORSTR
LD    (PICDAT),HL
```

La imagen deseada es la siguiente:



A primera vista, la estructura de bits para esto viene dada por los valores hexadecimales;

```
;00    00
;00    00
;41    00
;63    00
;77    00
;5D    00
;49    00
;E3    80
```

Ahora debemos incorporar la otra información. El primer octeto debe ser el número de las líneas de borde (8) seguido por los primeros valores de BINK y BAPER (7 y 6 respectivamente). Necesitamos un nuevo BAPER después de la línea 0, así que indicamos esto cambiando los datos

de 00 00 a 00 3F

y luego incluyendo el nuevo valor de BAPER después del valor de RINK, el cual queda igual. Los siete primeros octetos son ahora:

```
08     - LINEAS BORDE
07 06 - BINK, BAPER
00 3F- DATOS PARA LINEA 0
07 02 - BINK, BAPER
```

El resto de los datos se tratan de la misma manera, sumando #3E al último valor para indicar "fin de datos", por lo cual los últimos 2 octetos cambian

de E3 80 a E3 BE

El listado final de los datos de borde compacto se encuentra en la etiqueta MELDAT del listado en ensamblador; por tanto, establecemos la imagen con la sencilla y rápida secuencia:

```
LD     HL,MELDAT
CALL   EXPAND
CALL   ATTSET
CALL   HIRON
```

He aquí el listado completo, llamado EXDEMO

```
                 10 ;RUTINA DE DEMOSTRACION PARA  EXPANDIR ATTSET Y BORPIC
                 20 ;GENERA EL LOGOTIPO DE MELBOURNE
                 30 ;
                 40 ;32 LINEAS DEL CUADRO ESTARAN POR ENCIMA DEL TEXTO
                 50 ;
8C64  3E20       60 EXDEMO   LD        A,32
8C66  32678C     70          LD        (TPMRGN+1),A
                 80 ;
                 90 ;OBSERVAR 63-32=31 PARA FLYBAK
                100 ;
```

```
8C69  3E1F      110           LD        A,31
8C6B  326C8C    120           LD        (FLYBAK+1),A
                130 ;
                140 ;CONSTRUIR DATOS BORDE EN BORSTR ...
                150 ;
8C6E  216E8C    160           LD        HL,BORSTR
8C71  22718C    170           LD        (PICDAT),HL
                180 ;
                190 ;UTILIZANDO "EXPAND" EN DATOS BORDE COMPACTO
                200 ;
8C74  218A8C    210           LD        HL,MELDAT
8C77  CD778C    220           CALL      EXPAND
                230 ;
                240 ;AHORA PONER ATRIBUTOS DE PAPEL APROPIADAMENTE
                250 ;
8C7A  CD7A8C    260           CALL      ATTSET
                270 ;
                280 ;ACTIVAR EL GENERADOR DE CUADRO DE BORDE
                290 ;
8C7D  CD7D8C    300           CALL      HIRON
                310 ;
                320 ;GENERAR CUADRO DURANTE 5.12 SEGUNDOS
                330 ;OBSERVAR QUE B=0 DESDE HIRON
                340 ;
8C80  76        350 XLP       HALT
8C81  10FD      360           DJNZ      XLP
                370 ;
                380 ;VOLVER A SELECCIONAR IM 1 PARA BASIC
                390 ;
8C83  ED56      400           IM        1
8C85  3E3F      410           LD        A,#3F
8C87  ED47      420           LD        I,A
8C89  C9        430           RET
                440 ;
                450 ;DATOS DE BORDE COMPACTO
                460 ;8 LINEAS DE BORDE CON BINK BLANCO
8C8A  08        470 MELDAT    DEFB      8
                480 ;
                490 ;UNA CON BAPER AMARILLO
                500 ;
8C8B  07        510           DEFB      7
8C8C  06        520           DEFB      6
8C8D  20        530           DEFB      32
8C8E  00        540           DEFB      0
8C8F  3F        550           DEFB      #3F
                560 ;
                570 ;DOS CON BAPER ROJO
                580 ;
8C90  07        590           DEFB      7
8C91  02        600           DEFB      2
8C92  20        610           DEFB      32
8C93  00        620           DEFB      0
8C94  00        630           DEFB      0
8C95  20        640           DEFB      32
8C96  41        650           DEFB      #41
8C97  3F        660           DEFB      #3F
                670 ;
                680 ;DOS CON BAPER VIOLETA
                690 ;
8C98  07        700           DEFB      7
8C99  03        710           DEFB      3
8C9A  20        720           DEFB      32
8C9B  63        730           DEFB      #63
8C9C  00        740           DEFB      0
8C9D  20        750           DEFB      32
8C9E  77        760           DEFB      #77
8C9F  3F        770           DEFB      #3F
```

```
                        780 ;
                        790 ;Y TRES CON BAPER AZUL
                        800 ;
8CA0  07                810             DEFB    7
8CA1  01                820             DEFB    1
8CA2  20                830             DEFB    32
8CA3  5D                840             DEFB    #5D
8CA4  00                850             DEFB    0
8CA5  20                860             DEFB    32
8CA6  49                870             DEFB    #49
8CA7  00                880             DEFB    0
8CA8  20                890             DEFB    32
8CA9  E3                900             DEFB    #E3
8CAA  BE                910             DEFB    #BE
```

Las rutinas proporcionadas en este capítulo tienen muchos usos posibles. Podría utilizar BORPIC para proporcionar unos gráficos excelentes en el margen superior y/o inferior (recuerde que puede utilizar unas líneas de borde tan reducidas como de una fila de altura) utilizando una línea de borde de 192 filas de altura de un color en medio. Alternativamente y recordando que las rutinas no afectan al archivo de pantalla o a los atributos de INK, podría utilizar BORPIC, ATTSET y EXPAND para proporcionar un fondo espectacular cuando, por ejemplo, se detiene un juego o muere el jugador.

Concluiré este capítulo con sólo un ejemplo, produciendo una secuencia de cuatro imágenes de los números 3, 2, 1 y 0, en ese orden. Esta cuenta atrás debe utilizarse, por ejemplo, como fondo para una imagen de texto de un submarino a punto de lanzar uno de sus misiles Polaris, en un espectacular efecto de tres dimensiones, por supuesto.

```
                         10 ;3-2-1-0 CUENTA ATRAS
                         20 ;
                         30 ;DATO DE BORDE COMPACTO
                         40 ;
                         50 ;IMAGEN "3"
                         60 ;
90D0  08                 70 DAT3    DEFB    8
90D1  05                 80         DEFB    5
90D2  02                 90         DEFB    2
90D3  20                100         DEFB    32
90D4  FF                110         DEFB    #FF
90D5  80                120         DEFB    #80
90D6  20                130         DEFB    32
90D7  01                140         DEFB    1
90D8  80                150         DEFB    #80
90D9  20                160         DEFB    32
90DA  01                170         DEFB    1
90DB  80                180         DEFB    #80
90DC  20                190         DEFB    32
90DD  FF                200         DEFB    #FF
90DE  80                210         DEFB    #80
                        220
90DF  20                230         DEFB    32
90E0  FF                240         DEFB    #FF
90E1  80                250         DEFB    #80
90E2  20                260         DEFB    32
90E3  01                270         DEFB    1
90E4  80                280         DEFB    #80
90E5  20                290         DEFB    32
90E6  01                300         DEFB    1
90E7  80                310         DEFB    #80
```

```
90E8  20          320           DEFB      32
90E9  FF          330           DEFB      #FF
90EA  BE          340           DEFB      #BE
                  350 ;
                  360 ;IMAGEN "2"
                  370 ;
90EB  08          380 DAT2      DEFB      8
90EC  02          390           DEFB      2
90ED  04          400           DEFB      4
90EE  20          410           DEFB      32
90EF  FF          420           DEFB      #FF
90F0  80          430           DEFB      #80
90F1  20          440           DEFB      32
90F2  01          450           DEFB      1
90F3  80          460           DEFB      #80
90F4  20          470           DEFB      32
90F5  01          480           DEFB      1
90F6  80          490           DEFB      #80
90F7  20          500           DEFB      32
90F8  FF          510           DEFB      #FF
90F9  80          520           DEFB      #80
90FA  20          530           DEFB      32
90FB  FF          540           DEFB      #FF
90FC  80          550           DEFB      #80
90FD  20          560           DEFB      32
90FE  C0          570           DEFB      #C0
90FF  00          580           DEFB      0
9100  20          590           DEFB      32
9101  C0          600           DEFB      #C0
9102  00          610           DEFB      0
9103  20          620           DEFB      32
9104  FF          630           DEFB      #FF
9105  BE          640           DEFB      #BE
                  650 ;IMAGEN "1"
9106  08          660 DAT1      DEFB      8
9107  07          670           DEFB      7
9108  02          680           DEFB      2
9109  20          690           DEFB      32
910A  1C          700           DEFB      #1C
910B  00          710           DEFB      0
910C  20          720           DEFB      32
910D  7C          730           DEFB      #7C
910E  00          740           DEFB      0
910F  20          750           DEFB      32
9110  0C          760           DEFB      12
9111  00          770           DEFB      0
9112  20          780           DEFB      32
9113  0C          790           DEFB      12
9114  00          800           DEFB      0
9115  20          810           DEFB      32
9116  0C          820           DEFB      12
                  830 ;
9117  00          840           DEFB      0
9118  20          850           DEFB      32
9119  0C          860           DEFB      12
911A  00          870           DEFB      0
911B  20          880           DEFB      32
911C  7F          890           DEFB      #7F
911D  80          900           DEFB      #80
911E  20          910           DEFB      32
911F  7F          920           DEFB      #7F
9120  BE          930           DEFB      #BE
                  940 ;
                  950 ;IMAGEN "0"
                  960 ;
9121  08          970 DAT0      DEFB      8
9122  01          980           DEFB      1
```

```
9123  05          990          DEFB     5
9124  20         1000          DEFB     32
9125  7F         1010          DEFB     #7F
9126  00         1020          DEFB     0
9127  20         1030          DEFB     32
9128  FF         1040          DEFB     #FF
9129  80         1050          DEFB     #80
912A  20         1060          DEFB     32
912B  C7         1070          DEFB     #C7
912C  80         1080          DEFB     #80
912D  20         1090          DEFB     32
912E  CD         1100          DEFB     #CD
                 1110  ;
912F  80         1120          DEFB     #80
9130  20         1130          DEFB     32
9131  D9         1140          DEFB     #D9
9132  80         1150          DEFB     #80
9133  20         1160          DEFB     32
9134  F1         1170          DEFB     #F1
9135  80         1180          DEFB     #80
9136  20         1190          DEFB     32
9137  FF         1200          DEFB     #FF
9138  80         1210          DEFB     #80
9139  20         1220          DEFB     32
913A  7F         1230          DEFB     #7F
913B  3E         1240          DEFB     #3E
                 1250  ;
                 1260  ;TABLA DE DIRECCIONES DE DATOS DE BORDE COMPACTO
                 1270  ;
913C  D090       1280  BORTAB   DEFW     DAT3
913E  EB90       1290          DEFW     DAT2
9140  0691       1300          DEFW     DAT1
9142  2191       1310          DEFW     DAT0
                 1320  ;
                 1330  ;RUTINA DE CUENTA ATRAS ****
                 1340  ;UTILIZAR 32 FILAS DE TV SOBRE EL AREA DE TEXTO
                 1350  ;
9144  3E20       1360  CNTDWN   LD       A,32
9146  324791     1370          LD       (TPMRGN+1),A
9149  3E1F       1380          LD       A,31
914B  324C91     1390          LD       (FLYBAK+1),A
914E  214E91     1400          LD       HL,BORSTR
9151  225191     1410          LD       (PICDAT),HL
                 1420  ;
                 1430  ;CONFIGURAR LA TABLA DE VECTORES, PERO NO ACTIVAR EL
                 1440  ;GENERADOR DE CUADRO HASTA QUE LOS DATOS BORDE ESTEN DEFINID
                     OS
                 1450  ;
9154  F3         1460          DI
9155  CD5591     1470          CALL     HIRON
9158  213C91     1480          LD       HL,BORTAB
915B  0604       1490          LD       B,4
                 1500  ;
                 1510  ;TOMAR DIRECCION DE DATOS DE BORDE COMPACTO
                 1520  ;
915D  5E         1530  NXTNUM   LD       E,(HL)
915E  23         1540          INC      HL
965F  56         1550          LD       D,(HL)
9160  23         1560          INC      HL
9161  C5         1570          PUSH     BC
9162  E5         1580          PUSH     HL
                 1590  ;
                 1600  ;DEFINIR DATOS DE MARGEN Y ATRI. DE PAPEL
                 1610  ;
9163  EB         1620          EX       DE,HL
9164  CD6491     1630          CALL     EXPAND
9167  CD6791     1640          CALL     ATTSET
```

```
                1650  ;
                1660  ;PINTAR LA IMAGEN DURANTE UN SEGUNDO
                1670  ;
916A  FB        1680          EI
916B  0650      1690          LD        B,#50
916D  76        1700  PSE     HALT
916E  10FD      1710          DJNZ      PSE
9170  F3        1720          DI
                1730  ;
                1740  ;SIGUIENTE IMAGEN
                1750  ;
9171  E1        1760          POP       HL
9172  C1        1770          POP       BC
9173  10E8      1780          DJNZ      NXTNUM
                1790  ;
                1800  ;VOLVER A SELECCIONAR IM 1 PARA BASIC
                1810  ;
9175  3E3F      1820          LD        A,#3F
9177  ED47      1830          LD        I,A
9179  ED56      1840          IM        1
917B  FB        1850          EI
917C  C9        1860          RET
```

Apéndice A

Lista de las principales rutinas

Apéndice B

Ensambladores y monitores-desensambladores recomendados

"DEVPAC 3" de Hisoft

... está formado por el ensamblador "GENS 3" y el monitor/desensamblador "MONS 3". Las rutinas de este libro se desarrollaron sobre "GENS 3". Hay una versión compatible para microdrive. "GENS 3" tiene una longitud de 7K y, por tanto, es muy práctico para utilizar en su Spectrum de 48K.

Hisoft
13 Gooseacre
Cheddington
Leighton Buzzard
Beds.
LU7 0SR

Importado por VENTAMATIC.

Oxford Computer Publishing (OCP)

... proveen con dos programas separados para 16 y 48K. Se trata de "Full Screen Editor/Assembler" y "Machine Code Test Tool": un tutor y monitor-*debugger*.

Oxford Computer Publishing Ltd.
4 High Street
Chalfont St. Peter
Bucks
SL9 9QB Importado por ABC-SOFT.

Picturesque

... hacen una pareja potente de utilidades llamadas "Editor Assembler" y "Spectrum Monitor", siendo la última también un desensamblador, y ambas para los Spectrums de 16 ó 48 K.

Picturesque
6 Corkscrew Hill
West Wickman
Kent
BR4 9BB

Sinclair Research

... una compañía que debería serle familiar, publica el "ZEUS Assembler" y "Monitor/Disassembler", ambos para el Spectrum de 48K y escritos originalmente por Crystal Computing Limited.

Sinclair Research
Stanhope Road
Camberley
Surrey
GU15 3BR

Lecturas recomendadas

A lo largo de este libro habrá observado que se hace referencia a los "T-estados" y a los tiempos empleados por diversas instrucciones en su ejecución. Si desea encontrar un desglose total de los tiempos de cada instrucción del Z-80, su código de operación y su efecto sobre las banderas, puede recurrir a lo que muchas personas consideran una guía completa del Z-80, *Programming The Z-80*, de Rodnay Zaks, publicado por Sybex.

Este libro merece la pena sin duda alguna como guía de referencia, aunque es algo caro.

Otro libro que no debe faltar a ningún buen programador en lenguaje máquina de Spectrum es *The Complete Spectrum ROM Disassembly*, del Dr. Ian Logan y el Dr. Frank O'Hara, publicado por Melbourne House.

Un listado en ensamblador, totalmente explicado, de la ROM, ocupa por completo 236 páginas del libro. Este listado deberá estar a su alcance cuando necesite estudiar cómo ha sido programada una rutina completa de la ROM, o qué valores se necesita para su utilización.

¿Te has preguntado qué hay tras el lenguaje máquina de alto nivel de muchos de los juegos con más éxito para el Spectrum?

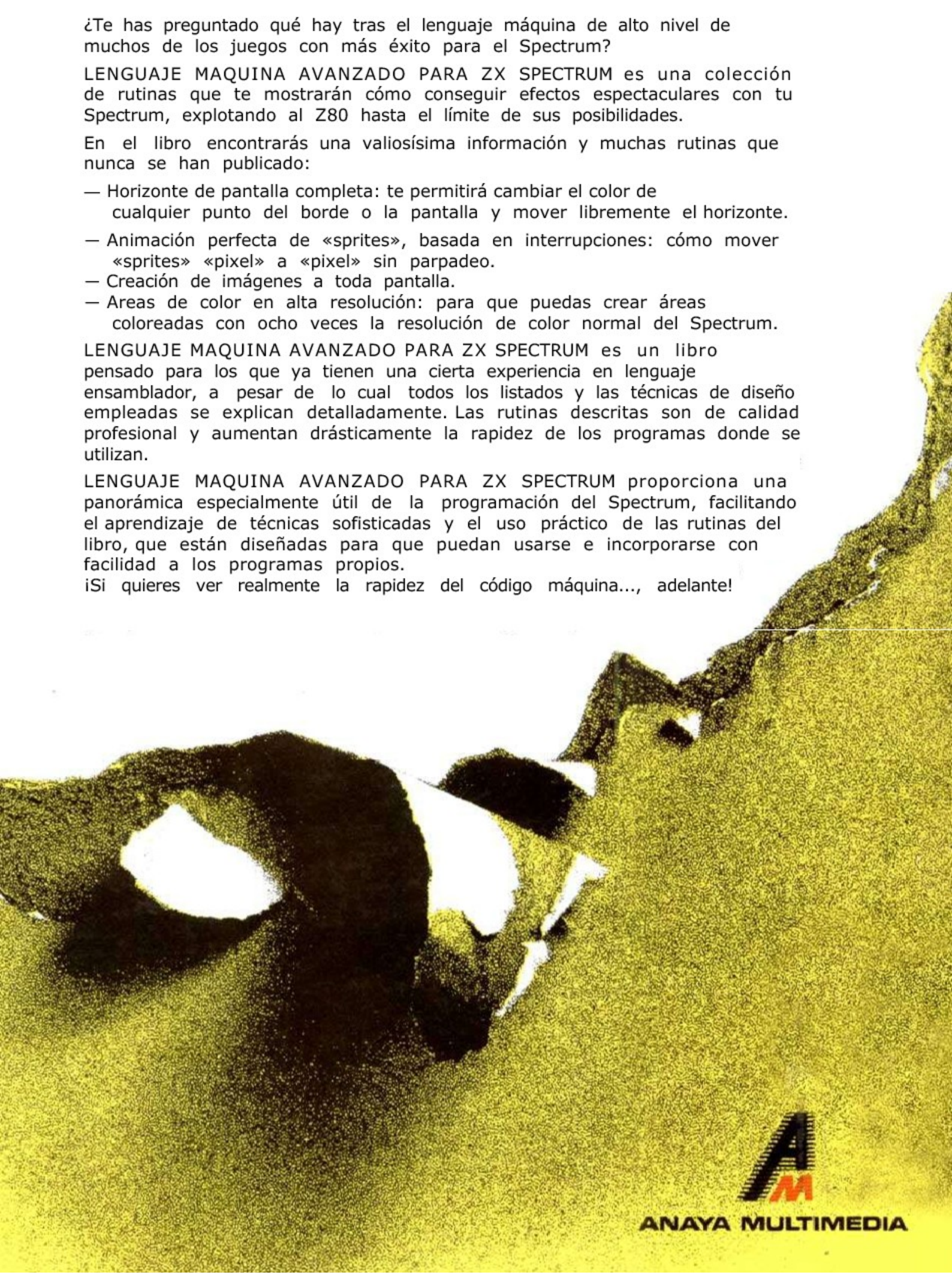
LENGUAJE MAQUINA AVANZADO PARA ZX SPECTRUM es una colección de rutinas que te mostrarán cómo conseguir efectos espectaculares con tu Spectrum, explotando al Z80 hasta el límite de sus posibilidades.

En el libro encontrarás una valiosísima información y muchas rutinas que nunca se han publicado:

— Horizonte de pantalla completa: te permitirá cambiar el color de cualquier punto del borde o la pantalla y mover libremente el horizonte.
— Animación perfecta de «sprites», basada en interrupciones: cómo mover «sprites» «pixel» a «pixel» sin parpadeo.
— Creación de imágenes a toda pantalla.
— Areas de color en alta resolución: para que puedas crear áreas coloreadas con ocho veces la resolución de color normal del Spectrum.

LENGUAJE MAQUINA AVANZADO PARA ZX SPECTRUM es un libro pensado para los que ya tienen una cierta experiencia en lenguaje ensamblador, a pesar de lo cual todos los listados y las técnicas de diseño empleadas se explican detalladamente. Las rutinas descritas son de calidad profesional y aumentan drásticamente la rapidez de los programas donde se utilizan.

LENGUAJE MAQUINA AVANZADO PARA ZX SPECTRUM proporciona una panorámica especialmente útil de la programación del Spectrum, facilitando el aprendizaje de técnicas sofisticadas y el uso práctico de las rutinas del libro, que están diseñadas para que puedan usarse e incorporarse con facilidad a los programas propios.
¡Si quieres ver realmente la rapidez del código máquina..., adelante!



ANAYA MULTIMEDIA