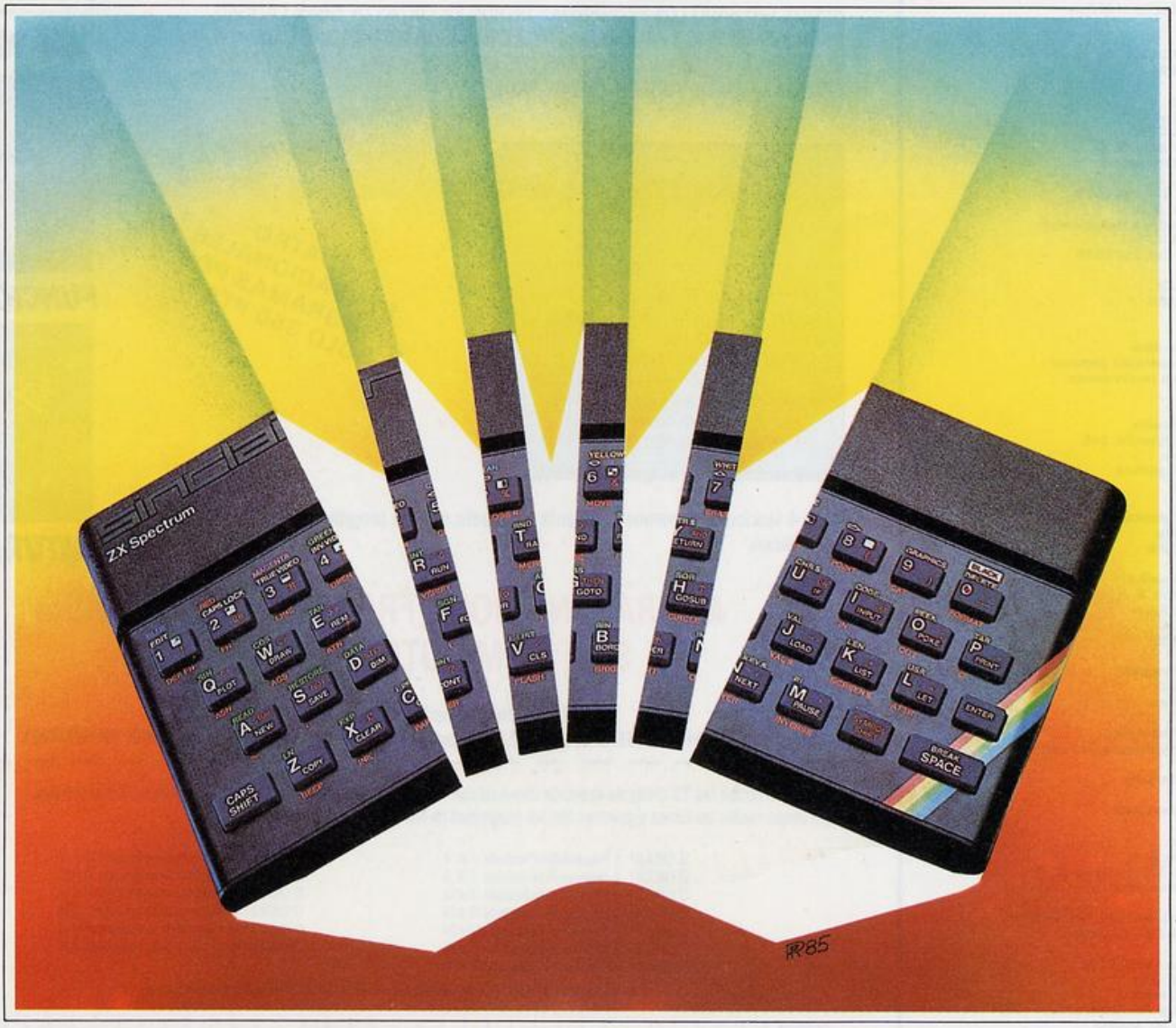
20
150pts.

RUN

Enciclopedia Práctica del Spectrum



Nueva Lente/Ingelek

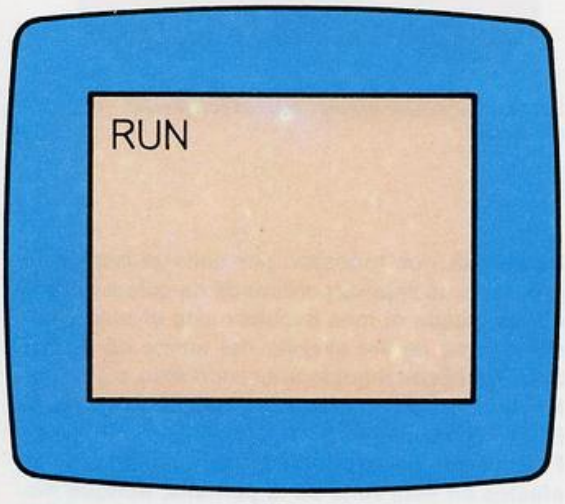
COLOR EN NUESTRO T.V.

El color es uno de los atributos que hacen del Spectrum un ordenador atractivo y versatil, manteniendo una extraordinaria sencillez de manejo. Hasta ahora no sabemos hacer uso de esta importante herramienta, pero en el transcurso de este capítulo y el siguiente, perfilaremos los conocimientos necesarios como para empezar a explotar las posibilidades cromáticas de nuestro aparato, sacándole partido a esta nueva e importante faceta del mismo.

Lo primero que debemos saber sobre este tema es que, a efectos de color, la pantalla se encuentra dividida en dos zonas bien diferenciadas, cuya denominación podría ser: «primer» y «segundo» plano, o zona circundante y zona central de la pantalla.

La zona de la pantalla sobre la cual podemos escribir, es una porción rectangular de 24 filas por 32 columnas. El resto de la misma, es decir, la parte que se encuentra por encima y por debajo, o a la izquierda y derecha de los márgenes citados, no es accesible a través de la sentencia **PRINT**.

La pantalla que se presenta en el televisor puede dividirse en dos zonas: exterior (no accesible por **PRINT***) e interior.*



A partir de este capítulo, podremos empezar a sacar provecho a otra de las cualidades de nuestro Spectrum: la generación de color.

Esta zona no utilizable para la escritura, conforma una especie de «fondo» o segundo plano, bordeando la pantalla, y puede adquirir cualquiera de los ocho colores básicos de que dispone el Spectrum, reseñados en la parte superior de las teclas correspondientes a los números.

Como podemos observar en la tabla adjunta, a cada uno de los colores le corresponde un número determinado, el cual utilizaremos para referirnos a él cuando programemos nuestro ordenador. Así por ejemplo, diremos que vamos a escribir en tinta 0, en vez de en tinta negra, o que vamos a emplear el color 2, en vez del color rojo, etc...

Lo mismo sucede con la parte central de la pantalla, con la ventaja de poder acceder a un segundo juego de colores, basado en el ya descrito. Este segundo juego de ocho colores se compone a partir de los básicos más un atributo denominado BRILLO. De esta forma, podemos decir que los ocho colores básicos son oscuros y los ocho restantes de tono más claro (más brillantes). Con esto hemos hablado de los dos colores de fondo posibles, el del marco de la pantalla, y el de la zona accesible por **PRINT**. Dentro de esta última, podemos distinguir aún dos sub-planos; conformando algo así como un primer plano y un «primerísimo» plano, siendo éste último el color en el cual escribimos realmente.

Llegados a este punto, quizá veamos las cosas

¡!

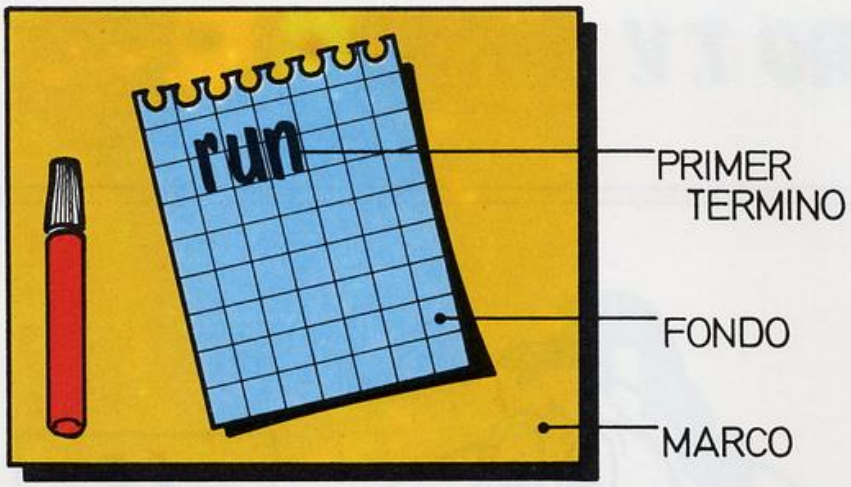
En la pantalla se pueden diferenciar dos zonas: la exterior, denominada marco, y en la cual no se pueden escribir caracteres, y la interior, donde se realizan las impresiones mediante **PRINT**.

*

Cada uno de los colores recibe un código numérico del 0 al 7.

*

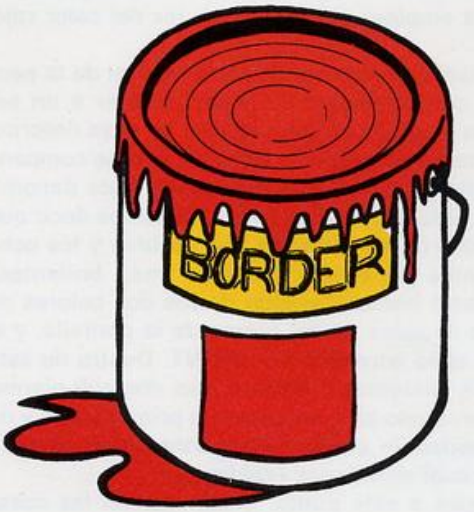
BORDER altera el color del marco de la pantalla de forma inmediata.



La pantalla del Spectrum puede considerarse similar a una mesa (marco), en la cual nos apoyamos para escribir (primer término) sobre un papel (fondo).

más claras con un ejemplo, bastante descriptivo además, de los nombres de las sentencias BASIC que gestionan el color en el Spectrum.
Supongamos que nos disponemos a escribir una carta de forma manual. Para ello, nos proveemos de una mesa donde escribir, papel y bolígrafo. Por comparación de conceptos, podemos decir que el color de la mesa es el del marco de la pantalla, donde no podemos escribir; el del papel es el de la zona imprimible; y por último, el de la tinta del bolígrafo es el color del primerísimo plano anteriormente citado.

La sentencia **BORDER** *nos permite dar color al marco de la pantalla.*



LA SENTENCIA BORDER

Por medio de la sentencia **BORDER**, seguida de un código de color básico, es posible cambiar el tono del marco de la pantalla, el cual, al encendido del ordenador, adopta el blanco.
La gama de colores de este marco de pantalla, está restringida al ámbito de los ocho fundamentales. Veámos algunos ejemplos de los efectos que podemos conseguir, con su empleo, comenzando antes de cada ejercicio por teclear **BORDER 7 (marco blanco)** para restablecer el color inicial: **FOR I=0 TO 7: BORDER I: PAUSE 0; NEXT I**.

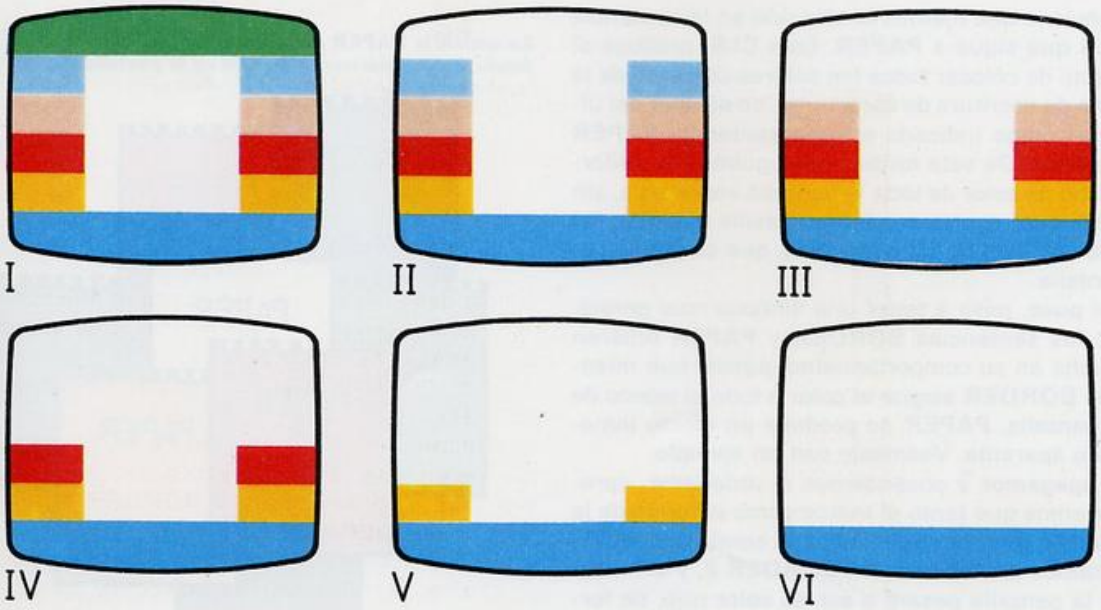
También es posible obtener marcos con franjas multicolor.

Esta línea nos muestra, por cada pulsación de una tecla, la relación ordenada de colores disponibles; desde el más oscuro hasta el más claro. Otra forma de ver el color del marco de la pantalla, es el conseguido alterando éste a gran velocidad: **10 FOR I=0 TO 7:BORDER I:NEXT I:GO TO 10.**
Finalmente, he aquí un último modo de obtener efectos en esta zona de la pantalla, aunque mucho más embarullados: **10 BORDER 2:BORDER 3:BORDER 4:BORDER 5:BORDER 6:GO TO 10.** Si este efecto de bandas de color no nos parece suficientemente sorprendente, podemos

probar a pulsar cualquier tecla, para verlo desplazarse de arriba abajo por la pantalla.
Estos efectos de color no se emplean muy a menudo, puesto que requieren una atención permanente del ordenador en esta tarea, lo cual resulta lógicamente bastante inútil. En todo caso, pueden ser empleados en combinación con otras sentencias BASIC que estudiaremos más adelante, para amenizar algunos tiempos de espera; por ejemplo, aquellos transcurridos desde la edición de una pantalla de instrucciones de un juego, hasta que la hemos leído y pulsamos una tecla para indicar esta circunstancia.
Sin duda, lo más frecuente es utilizar la sentencia **BORDER** del modo más simple estudiado al principio de este epígrafe. De esta forma, una vez que hemos asignado un determinado color al marco de la pantalla, éste permanece así hasta que indicamos lo contrario mediante una nueva sentencia **BORDER**. Explicaremos ahora brevemente el por qué del marco a franjas que hemos visto aparecer, por lo que puede ilustrarnos sobre el funcionamiento interno del ordenador en relación con el color del marco.
Cuando el televisor recibe la señal de vídeo del ordenador, comienza su impresión en la pantalla horizontalmente, de arriba a abajo, de una manera cíclica y constante. A este fenómeno se le conoce bajo el nombre de BARRIDO. El color del marco no es más que una de las señales que el ordenador envía al televisor, y así, si alteramos dicha señal de una forma lo suficientemente rápida, conseguiremos que al comienzo del barrido el marco sea de un color; un lapso de tiempo más tarde, antes de que finalice el barrido, de otro, y así sucesivamente hasta generar las franjas que hemos apreciado.
De hecho, si prestamos atención a lo que sucede en el marco de la pantalla cuando leemos o grabamos un programa en cinta, observaremos que multitud de franjas ocupan esta zona de la pantalla. Ello es debido a que el ordenador envía, al tiempo que lee y escribe, diversas señales al televisor, y puesto que es capaz de hacerlo a una velocidad mayor que la que nosotros conseguimos con **BORDER** desde el BASIC, puede generar un mayor número de franjas.
Aunque en principio el marco de la pantalla sea una zona poco interesante, dado que en ella no podemos escribir, no debemos desechar la posibilidad de su manejo. En el quinto capítulo de la sección de TU SPECTRUM, encontraremos más información sobre este tema.

En el proceso de formación de la imagen en el televisor, se produce un barrido horizontal cíclico de la pantalla.

Si en un corto espacio de tiempo (inferior a un barrido), enviamos varias señales diferentes como color de marco para la pantalla, este adopta en cada momento el correspondiente, generándose un marco a franjas.



¡!

Para que **PAPER** o **INK** afecte simultáneamente a toda la pantalla, hemos de ejecutar a continuación un comando **CLS**.

*

El brillo suplementario que se obtiene mediante la sentencia **BRIGHT** (en inglés, brillo), no afecta al marco de la pantalla, sino sólo a su zona interior.

*

Los atributos **BRIGHT** y **FLASH** (brillo e intermitencia), afectan simultáneamente a la **INK** (tinta) y el **PAPER** (papel) de un caracter.

*

Sólo **BORDER** tiene un efecto inmediato sobre la pantalla; el resto de los atributos de color afectan a los caracteres escritos a continuación, pero no a los ya presentes en la pantalla.

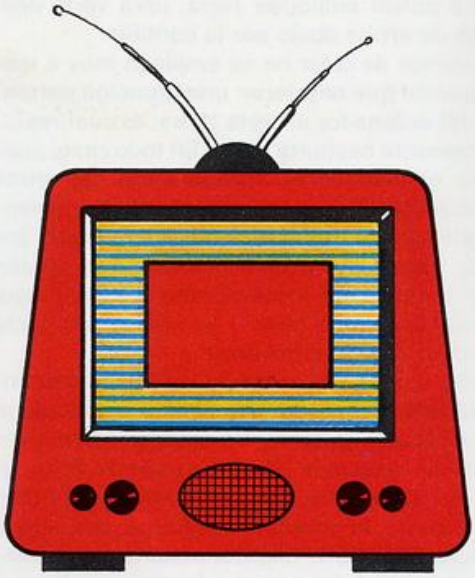
¡!

Al encender el ordenador, el estado original de los atributos de color es el siguiente: **BORDER 7, PAPER 7, INK 0, BRIGHT 0** y **FLASH 0**.

*

Si alteramos la señal de color enviada al marco en un tiempo inferior al de un barrido, podemos obtener marcos con franjas multicolor.

LA SENTENCIA PAPER



La extraordinaria velocidad del código máquina permite generar más franjas que las obtenidas desde el BASIC mediante **BORDER**. *Esto es lo que sucede cuando nuestro Spectrum lee o graba programas.*

Del mismo modo que **BORDER** asigna color al marco de la pantalla, **PAPER** lo hace para el centro de la misma. Sin embargo, la sentencia **PAPER** es mucho más potente: permite asignar un color determinado a cada posición concreta de la pantalla, es decir, a cada caracter en particular, en dos modalidades de brillo, gracias a la combinación con la sentencia BASIC **BRIGHT**, estudiada más adelante.

La estructura de la sentencia **PAPER** es similar en todo a **BORDER**, y va seguida por tanto de un código de color. En el siguiente ejemplo, veremos las posibles combinaciones de colores de marco y fondo, empleando los fundamentales: **FOR I=0 TO 7: FOR J=0 TO 7: BORDER I: PAPER J: CLS: NEXT J: NEXT I.**

Las combinaciones de **PAPER** *y* **BRIGHT** *se pueden utilizar en cada caracter, independientemente del resto de la pantalla.*

ma inmediata. Ahora ejecutemos **PAPER 2**. El mensaje **0 OK** se editará en la última línea de la pantalla, pero sin embargo el interior de la pantalla sigue siendo blanco ¿qué es lo que ha ocurrido? ¿acaso la instrucción no ha sido formulada correctamente?

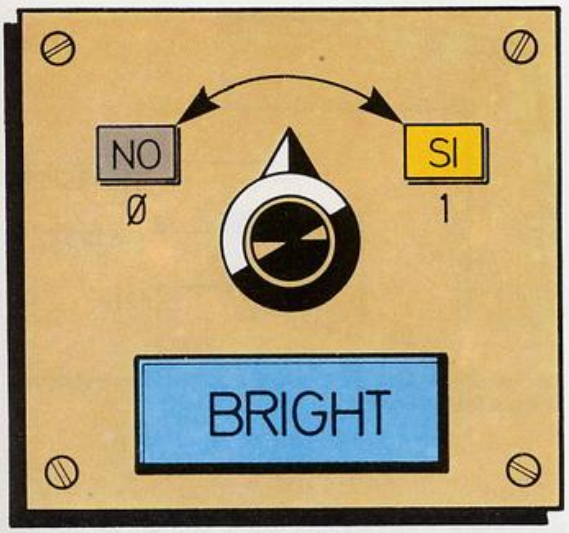
En absoluto. El problema estriba en que el efecto de **PAPER** no es inmediato, como en el caso de **BORDER**, sino que afecta a las instrucciones de

Conviene que fijemos la atención en la sentencia **CLS** que sigue a **PAPER**. Este **CLS**, produce el efecto de colocar todos los colores de fondo de la zona de escritura de caracteres, en el valor del último código indicado en una sentencia **PAPER** genérica. De esta forma, conseguimos la uniformidad de color de toda la pantalla imprimible, sin tener que recurrir a colorear, casilla a casilla, las 768 (24 filas de 32 columnas), que componen la pantalla.

Así pues, pese a tener una sintaxis muy parecida, las sentencias **BORDER** y **PAPER** difieren mucho en su comportamiento, puesto que mientras **BORDER** asigna el color a todo el marco de la pantalla, **PAPER** no produce un efecto inmediato aparente. Veámoslo con un ejemplo.

Si apagamos y encendemos el ordenador, apreciaremos que tanto el marco como el fondo de la pantalla interior son blancos. A continuación ajecutamos una instrucción **BORDER 2**, y el marco de la pantalla pasará a ser de color rojo, de for-

La sentencia **PAPER** *nos permite asignar el color de fondo de los caracteres a escribir en la pantalla.*



Mediante **BRIGHT** *podemos conectar o desconectar el modo de brillo suplementario.*

manejo de pantalla sucesivas, como **PRINT, TAB, CLS,** etc... Probemos a escribir una letra A en la pantalla: **PRINT "A"**. Por fin hace su efecto la sentencia **PAPER**: la letra ha sido escrita sobre un fondo rojo.

En resumen, podemos afirmar que la sentencia **BORDER** indica al Spectrum que debe alterar el color de todo el marco de la pantalla de forma inmediata, mientras que **PAPER** le señala el color que debe emplear como fondo en las próximas operaciones de escritura.

Este hecho nos ha ayudado a descubrir también que el efecto de **CLS** no es sólo el de borrar la pantalla, como habíamos visto hasta ahora, sino que también establece como color de fondo para toda la pantalla el seleccionado con la última sentencia **PAPER** (blanco inicialmente, al encender el ordenador).

Como ya hemos dicho, la sentencia **BRIGHT** es un complemento opcional de **PAPER**, que permite dar brillo al color. Veamos una combinación de sentencias **PAPER** y **BRIGHT**, para conseguir una «carta de ajuste» de color de fondo.

```
 10 REM - AJUSTE DE COLOR
 20 FOR I=0 TO 1
 30 BRIGHT I
 40 FOR J=1 TO 10
 50 FOR K=0 TO 7
 60 PAPER K: PRINT '     ';
 70 NEXT K
 80 NEXT J
 90 NEXT I
100 BRIGHT 0
```

Como podemos apreciar en el listado, la sentencia **BRIGHT** lleva como parámetro un número que adopta los valores 0 ó 1. El primero se utiliza para indicar que el color se encuentra en el modo normal, y el segundo que posee brillo suplementario. Como en el caso de PAPER, las instrucciones **BRIGHT 0** y **BRIGHT 1**, tendrán efecto sobre las escrituras sucesivas, pero no sobre toda la pantalla.

Finalmente, resta advertir que si utilizamos para la instrucción **BRIGHT** un parámetro menor que cero, obtendremos el mensaje de error **B Integer out of range** (entero fuera de rango); por el contrario, empleando parámetros por encima de 1, el mensaje emitido será **K Invalid colour** (código de color no válido). Una excepción a esto último será el código ocho, cuyo peculiar efecto estudiaremos posteriormente.

LA SENTENCIA INK

Hasta ahora hemos visto como dar color a la mesa sobre la que nos apoyamos (**BORDER**), y al papel en el cual escribimos (**PAPER**), por otro, nos falta conocer la manera de depositar la tinta adecuada en nuestro bolígrafo.

A través de la sentencia **INK** se define el color

La sentencia **INK** *nos permite alterar el color de primer término de los caracteres a escribir.*

¡!

Los parámetros para **BRIGHT** (brillo) y **FLASH** deben estar comprendidos entre 0 y 1, significando desconexión y conexión del modo, respectivamente.

*

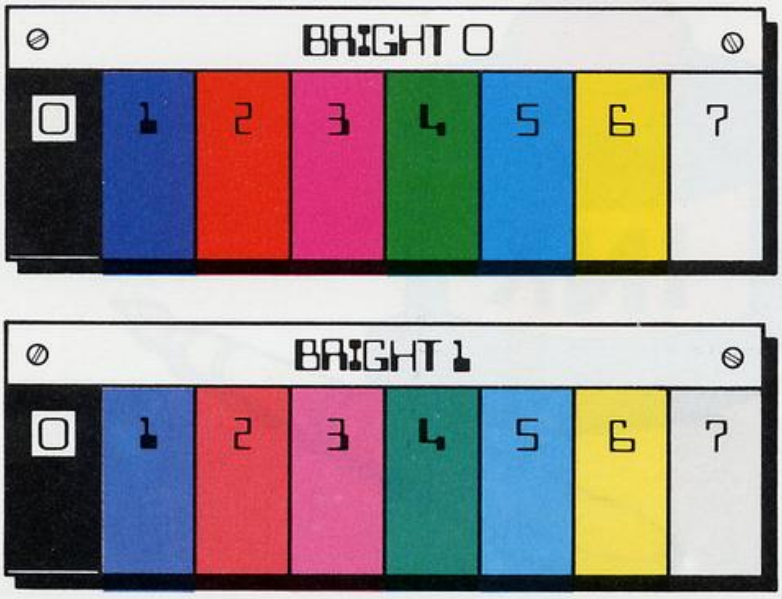
Las diferencias entre los tonos con y sin brillo suplementario de un mismo color, decrecen según se oscurece éste.

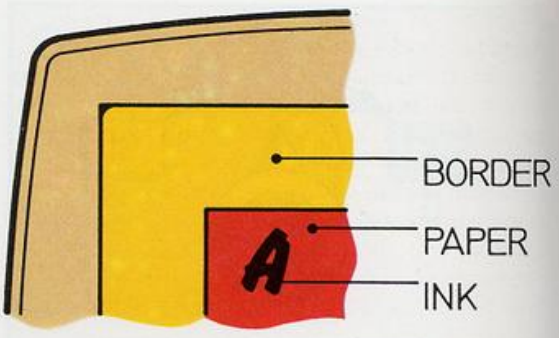
de esta tinta. De forma similar a casos anteriores, la sentencia va seguida de un parámetro que señala el código de color deseado. Como era de esperar, tanto su sintaxis como su funcionamiento es muy similar a **PAPER**.

Así pues, el cambio del color de tinta no afecta de forma inmediata a toda la pantalla, sino sólo a las impresiones que se efectúan a partir de la ejecución de la instrucción correspondiente. Ello nos permite escribir en cada posición de caracter (coordenada fila-columna), con el color de papel (en inglés *paper*) y tinta (en inglés *ink*) que deseemos, independientemente de los que se utilicen para las otras posiciones de la pantalla. Aprovechando parte del programa anterior, podemos escribir ahora uno que nos muestre una «carta de ajuste» completa, de combinaciones de colores de papel y tinta, con y sin brillo.

```
 10 REM - AJUSTE DE COLOR
 20 FOR I=0 TO 1
 30 BRIGHT I
 40 FOR J=1 TO 10
 50 FOR K=0 TO 7
 60 PAPER K: PRINT "    ";
 70 NEXT K
 80 NEXT J
 90 NEXT I
100 FOR I=0 TO 1
110 BRIGHT I: PAPER 7
120 FOR J=0 TO 3
130 INK J: PRINT J;"    ";
140 NEXT J
150 PAPER 0
160 FOR J=4 TO 7
170 INK J: PRINT J;"    ";
180 NEXT J
190 NEXT I
200 PAPER 7: BRIGHT 0: INK 0
```

Tanto para **PAPER** *como para* **INK**, *el rango de su parámetro oscila entre cero y siete, con dos modos de brillo.*





Los colores de **BORDER**, **PAPER** *e* **INK** *son totalmente independientes entre sí.*

Por último, y en relación con el margen en el cual se mueven los parámetros para **INK** y **PAPER**; éstos no deben ser inferiores a cero, puesto que como en el caso de **BRIGHT**, se producirá un error del tipo **B Integer out of range** (entero fuera de rango).

En esta ocasión, los colores admitidos llegan hasta el siete, aunque a ellos hemos de añadir dos códigos más: el ocho y el nueve, cuyo efecto no es exactamente el de un color particular, sino algo más especial que estudiaremos en el próximo capítulo. Cualquier parámetro por encima de nueve, motivará la emisión del mensaje de error **K Invalid colour**, cuyo significado hemos tenido la oportunidad de conocer anteriormente.

CONSIDERACIONES GLOBALES

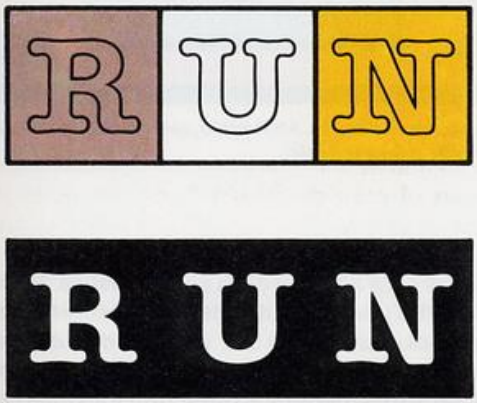
Si somos un poquito observadores, veremos que el orden de los códigos de color se corresponde con una secuencia desde los más oscuros hacia los más claros. De hecho, si anulamos por un momento el color de nuestra T.V., veremos que aparecen diferentes tonalidades de grises, ordenadas de menor a mayor luminosidad.

Del mismo modo, se hace patente que la diferencia entre los colores con y sin **BRIGHT** es muy acentuada en los tonos claros, y casi imperceptible en los oscuros. Este efecto debe ser tenido muy en cuenta, puesto que puede ser aprovechado a nuestro favor en alguna ocasión; dentro de este mismo apígrafe, cuando tratemos otra de las características de **BRIGHT**, nos daremos cuenta del por qué de esta afirmación.

Puesto que el marco de la pantalla sólo puede re-

presentar los ocho colores básicos, y no se ve afectado por el factor de brillo, podemos utilizarlo como referencia sobre la luminosidad de los colores empleados en la pantalla interior. Ello nos conducirá a curiosos efectos ópticos, como el que a continuación describimos.

Ya hemos comprobado anteriormente que cuando encendemos el ordenador, el color inicial para marco y fondo de pantalla es el blanco, encontrándose ambos en la modalidad de brillo normal. Pues bien, ejecutemos la siguiente instrucción: **BRIGHT 1: CLS**. Gracias a ella, conectaremos el modo de brillo suplementario, y borraremos la



Para resaltar el brillo suplementario de tinta, sin que aparentemente afecte al papel, debemos emplear fondos oscuros.

pantalla, de forma que el nuevo brillo afecte completamente al interior de la misma.

Casi como por arte de magia apreciamos que el cambio no se ha efectuado en el interior de la pantalla, como nosotros esperábamos, dado que sabíamos que **BRIGHT** no afecta al marco. Aparentemente, el fondo ha permanecido en su color blanco inicial, y el marco de la pantalla ha adquirido un tono gris, que en teoría se encuentra fuera de la gama del Spectrum, y más aún, de las posibilidades de esa zona de la pantalla.

En realidad, todo ha sido un simple efecto óptico. Como nosotros ya suponíamos, **BRIGHT** no ha afectado al marco, sino al fondo; pero el contraste del brillo suplementario entre uno y otro blanco, nos ha hecho creer que el primero de los colores, no es tal blanco, sino un perfecto gris. Este efecto cromático no debe ser desaprovechado, y combinado con inteligencia, puede «arrancarle» al Spectrum algún color más que aquellos de los cuales originalmente dispone, o al menos dar tal impresión.

Como hemos dicho unos párrafos más atrás, esta diferencia entre los tonos brillantes y no brillantes, decrece según se va oscureciendo la gama, y así como en el blanco está acentuadísima, hecho este que hemos podido comprobar tan solo hace unos instantes, en el negro es prácticamente imperceptible.

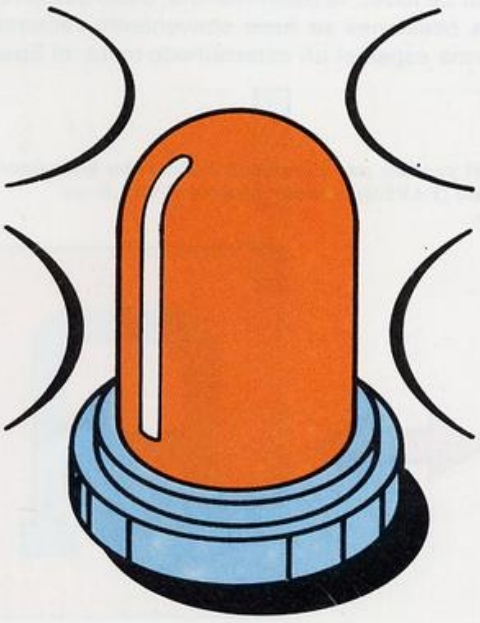
CODIGOS DE COLOR

0	NEGRO
1	AZUL OSCURO
2	ROJO
3	ROSA
4	VERDE
5	AZUL CLARO
6	AMARILLO
7	BLANCO

El siguiente mini-programa nos ayudará a comprobar esta escala decreciente de diferencias, gracias al contraste entre el color del marco (sin brillo), y el del interior (con brillo suplementario); pulsemos una tecla por cada vez que queramos pasar al próximo color: **BRIGHT 1: FOR I=7 TO 0 STEP -1: BORDER I: PAPER I: CLS: PAUSE 0: NEXT I.**

Otra característica a considerar, en lo referente al atributo de color **BRIGHT**, es que afecta tanto al papel como a la tinta que concurren en una determinada posición de la pantalla. Como ya vimos anteriormente, en una misma posición de la pantalla (coordenada fila-columna), podíamos hacer coexistir un color diferente de la tinta y del papel; sin embargo, si decidimos que dicha posi-

La sentencia **FLASH** *nos permite crear caracteres con colores intermitentes.*



¡!

Los colores de fondo y primer término que se otorguen a un caracter, así como los otros atributos (**BRIGHT** y **FLASH**), afectan a cada carácter de la pantalla, independientemente del resto de ellos y del marco de la misma.

*

El mensaje **B Integer out of range** (entero fuera de rango) aparece cuando queremos utilizar valores negativos para los atributos de color.

*

El mensaje **K Invalid colour** (código de color no válido) aparece cuando utilizamos como parámetro para los atributos de color, códigos que superan el nueve.

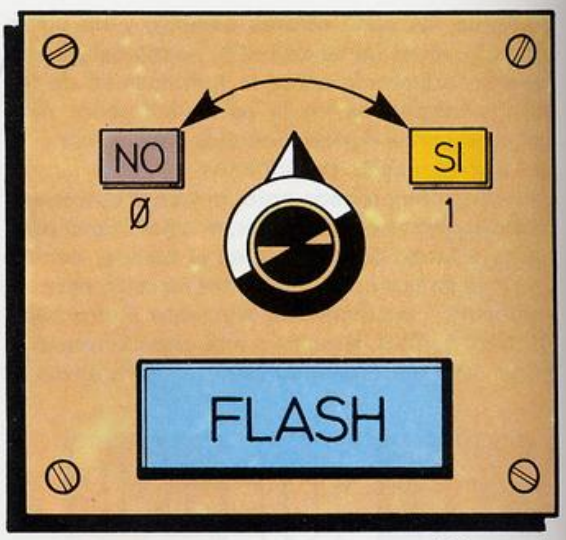
ción se encuentre con brillo suplementario, este hecho afectará tanto al color de la tinta como del papel en esa posición. Una vez que hemos visto esto, estamos en condiciones de explicar por qué dijimos que le podríamos sacar provecho a la diferencia decreciente entre los tonos brillantes y no brillantes de los diversos colores.

Supongamos que queremos escribir la letra inicial de una palabra en tono brillante, para destacarla del resto de la misma, pero sin embargo, nos interesa que permanezca de igual tono el fondo, por ejemplo, porque lo hemos hecho coincidir con el color del marco, para así aumentar las dimensiones aparentes de la pantalla (capítulo cinco, sección de TU SPECTRUM). Esta opción es aparentemente imposible, dado que el brillo afectará tanto a la tinta como al fondo (papel) del caracter.

La solución se encuentra en no utilizar en el programa un color de fondo claro, sino todo lo contrario: el color negro, o en su defecto el azul oscuro. Gracias a ello, aunque la cualidad de brillo suplementario atañe también al fondo, su efecto es prácticamente imperceptible, y no así en la tinta, para la cual habremos empleado un tono muy claro, a ser posible el blanco.

UN ULTIMO TRUCO

Pero además de las posibilidades de color descritas, contamos con una más, quizá la más espectacular de todas: la intermitencia. Dado que en algunas ocasiones se hace conveniente remarcar de forma especial un determinado texto, el Spectrum incorpora además de la característica de brillo (**BRIGHT**), la de intermitencia, obtenida mediante la sentencia **FLASH**, seguida de los parámetros 1 ó 0, para conexión o desconexión del modo, de forma idéntica a como funcionamos con **BRIGHT**.



Los parámetros para **FLASH** *están comprendidos entre 0 y 1: estable o intermitente.*

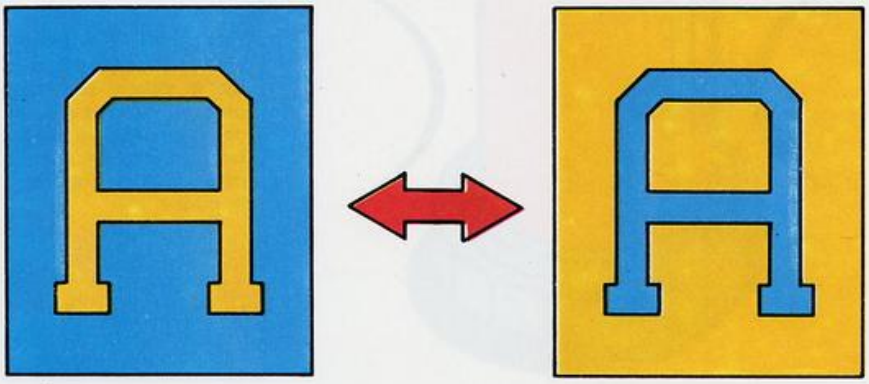
Los errores que se producen al no respetar estos parámetros son los mismos que los emitidos con **BRIGHT**, incluyendo la posibilidad de utilizar el código especial 8, que será objeto de estudio en el próximo capítulo.

El efecto de **FLASH** es producir una alternancia intermitente entre los colores de tinta (**INK**) y fondo (**PAPER**) de un carácter, de forma automática, y con una velocidad constante. Con este nuevo aditamento, son muchas las combinaciones de color posibles dentro del área accesible de la pantalla; podríamos decir que casi infinitas. De hecho, tenemos que considerar lo siguiente:

PAPER.: **Valores de 0 a 7**
INK.: **Valores de 0 a 7**
BRIGHT: **Valores de 0 a 1**
FLASH.: **Valores de 0 a 1**

El número de combinaciones posibles es de 256, lo cual podremos comprobarlo a través de un último programa.

FLASH *produce una alternancia cíclica entre los colores de fondo (*PAPER*) y primer término (*INK*) de un carácter.*



```
  10 REM - COMBINACIONES DE COLOR
  20 FOR I=0 TO 7: FOR J=0 TO 7: FOR K=0
TO 1: FOR L=0 TO 1
  30 GO SUB 70: PAUSE 10
  40 NEXT L: NEXT K: NEXT J: NEXT I
  50 PAUSE 0: PAPER 7: INK 0: BRIGHT 0:
FLASH 0: CLS
  60 STOP
  70 PAPER I: INK J: BRIGHT K: FLASH L
  80 PRINT "PAP.: ";I;" INK.: ";J;" BRI.
: ";K;" FLA.: ";L;" "
  90 RETURN
```

EL CODIGO MAQUINA

VAMOS ahora a hacer algunas precisiones a cerca de la carga y ejecución de programas en código máquina con el Spectrum; quizás algunos de nosotros ya las sepamos, pero en todo caso no está de más refrescar esos conocimientos.

Una subrutina en código máquina es un conjunto de números (bytes) situados en determinada zona de la memoria, a los cuales se accede mediante la función **USR**. Así pues, para que un programa que utilice determinada subrutina en C/M (código máquina) funcione correctamente, esta se debe encontrar cargada en la memoria.

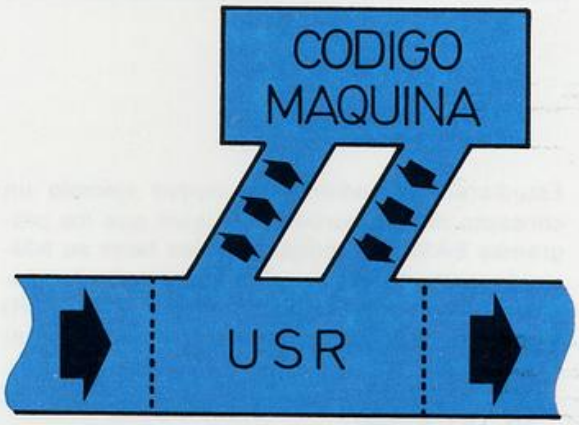
Al desconectar nuestro aparato, o al pulsar el botón de *reset* en el PLUS, la memoria RAM se borra íntegramente. Esto quiere decir, que no es suficiente introducir el código máquina una sola vez, sino siempre que vayamos a cargar el programa que la necesite.

Veamos un ejemplo. Vamos a hacer un pequeño programa en código máquina; realmente, lo único que hace es retornar al BASIC. Así pues, la subrutina va a ser extraordinariamente corta, dado que sólo estará compuesta por el código 201. Este será depositado en la dirección 30000.

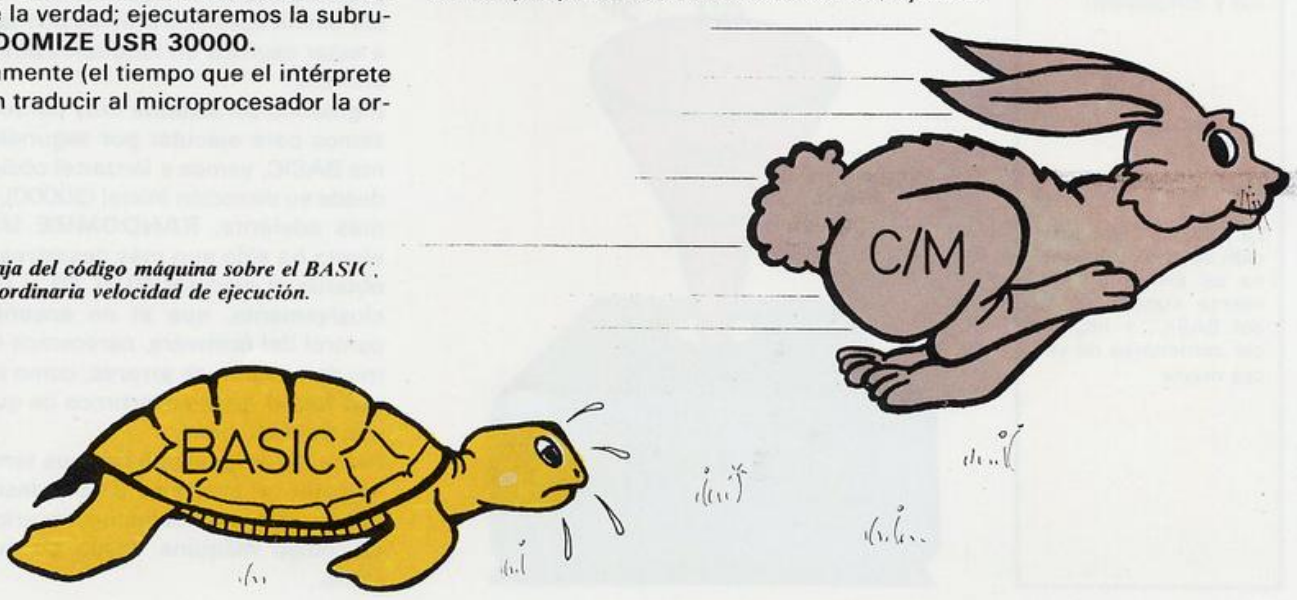
Para introducir en la memoria tan corto programa, sólo es necesario ejecutar la siguiente instrucción: **POKE 30000,201**. Gracias a ella, situamos nuestra mini-subrutina en la dirección 30000. Para comprobar que sigue allí, no tenemos más que preguntar el contenido de esa dirección (**PRINT PEEK 30000**). Bien, ahora llega el momento de la verdad; ejecutaremos la subrutina con **RANDOMIZE USR 30000.**

Casi inmediatamente (el tiempo que el intérprete BASIC tarda en traducir al microprocesador la orden), recibiremos el mensaje **0 OK**. Acabamos de terminar con éxito nuestro primer programa en código máquina, si bien es cierto que no tiene mucha utilidad (simplemente vuelve al BASIC). Ahora bien, él nos va a dar oportunidad de comprobar en la práctica que su efecto desaparece al apagar el ordenador.

Ahora que ya hemos vuelto a encender, ejecute-



La función **USR** *produce una bifurcación temporal del control BASIC hacia una subrutina en código máquina.*



La principal ventaja del código máquina sobre el BASIC, reside en su extraordinaria velocidad de ejecución.

¡!

mos de nuevo la instrucción **RANDOMIZE USR 30000**. Esta vez no ha ocurrido lo mismo, y ello se debe a que en la dirección 30000 ya no se encontraba nuestra subrutina en código máquina

CONCEPTOS IMPORTANTES

Un programa en código máquina sólo puede ser ejecutado desde la dirección de memoria destinada a tal fin.

*

Un programa se denomina reubicable cuando cualquiera que sea la dirección de memoria en que se sitúe, funciona correctamente, siempre y cuando sea ejecutado desde la dirección adecuada.

*

La reubicabilidad no es un hecho común, y prácticamente sólo se da en rutinas relativamente cortas en longitud y complejidad.

*

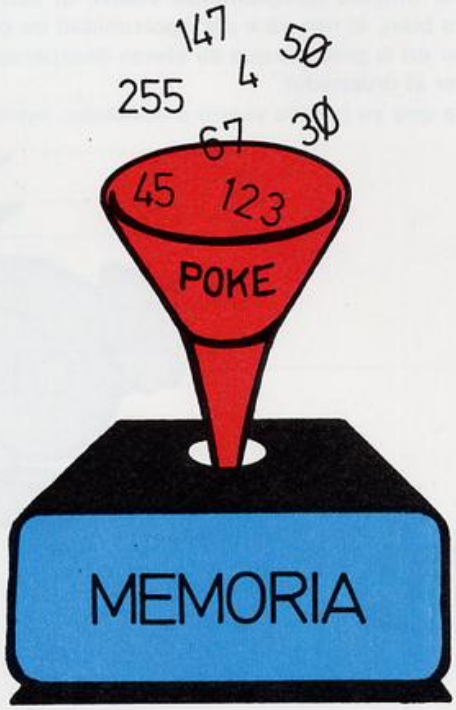
La velocidad de ejecución del código máquina es extraordinariamente superior a la del BASIC, y llega a ser centenares de veces mayor.

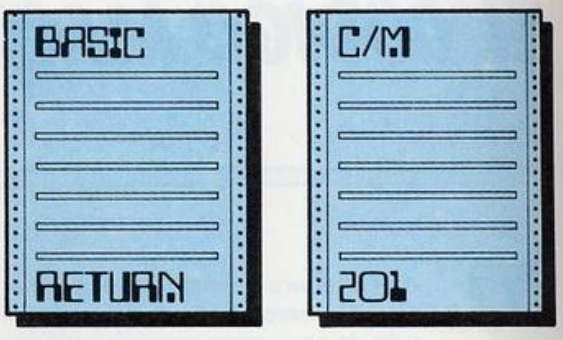
Estudiaremos mediante un nuevo ejemplo un concepto muy importante. Al igual que los programas BASIC, el código máquina tiene su adecuado punto de inicio, y si queremos que funcione correctamente, tenemos que ejecutarlo (**USR**) a partir de éste. Observemos el siguiente programa BASIC:

```
10 LET A=199
20 POKE 22528,A
```

Su misión es encender en la esquina superior izquierda de la pantalla, un cuadrado de color negro y blanco en intermitencia. Si lo ejecutamos mediante **RUN** o **RUN 10**, que en este caso es lo mismo puesto que 10 es la primera línea de instrucción, cumplirá perfectamente su cometido; sin embargo, ejecutado mediante **RUN 20**, producirá un error del tipo **2 Variable not found**.

La sentencia **POKE** *se utiliza para la introducción por el teclado de programas en código máquina.*





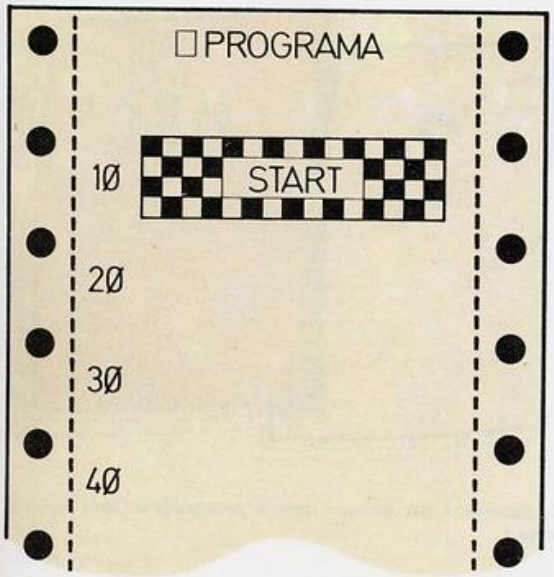
Existen diversas similitudes entre los lenguajes BASIC y código máquina.

La traducción de este programa al lenguaje máquina será algo más complicada que un mero código 201, pero no mucho más. Supongamos que como en el caso anterior, deseamos ubicar la subrutina a partir de la dirección 30000; de ser así, la introducción del programa se conseguirá mediante la siguiente línea de instrucción: **POKE 30000,62: POKE 30001,199: POKE 30002,50: POKE 30003,0: POKE 30004,88: POKE 30005,201.**

Ejecutando este mini-programa con **RANDOMIZE USR 30000**, podremos ver que se obtiene un efecto idéntico al observado con la ejecución del último listado BASIC; es decir, la aparición de un cuadrado en intermitencia (negro y blanco), en la posición superior izquierda de la pantalla. Pero las similitudes entre estos dos programas, pese a estar escritos en lenguajes diferentes, no acaban ahí.

Siguiendo un sistema muy parecido al que utilizamos para ejecutar por segunda vez el programa BASIC, vamos a lanzar el código máquina, no desde su dirección inicial (30000), sino desde una más adelante: **RANDOMIZE USR 30001**. El efecto ha sido aun más desastroso que el que se obtenía al ejecutar **RUN 20**, y ello se debe exclusivamente, que al no encontrarnos bajo el control del *firmware,* carecemos de un mecanismo de emisión de errores, como el de **2 Variable not found,** para advertirnos de que algo no marcha del todo bien.

Así pues, al igual que hay que tener cuidado para ejecutar un programa BASIC desde el lugar adecuado, aún más debemos tenerlo, para ejecutar el código máquina desde su punto exacto de inicio.



Los programas, tanto en BASIC como en código máquina, deben ejecutarse desde la línea para la que están preparados.

REUBICABILIDAD

Ahora vamos a utilizar un nuevo concepto: la REUBICABILIDAD; aplicable tanto al BASIC como al C/M (código máquina), pero que como en el último ejemplo estudiado, tiene aún más importancia al tratarse en lenguaje máquina, puesto que es más susceptible de producir «desastres».
Un programa se dice REUBICABLE cuando cualquiera que sea su situación en la memoria, o su numeración de referencia, funciona igualmente. Con un ejemplo quedará más clara esta idea. En el caso del BASIC, la numeración de referencia está constituida por los números de línea de instrucción; por tanto, un programa BASIC que funcione correctamente cualquiera que sean los números de línea que se utilicen, siempre y cuando se respete el orden de ejecución, es reubicable. El programa del cuadrado intermitente ejemplo es reubicable:

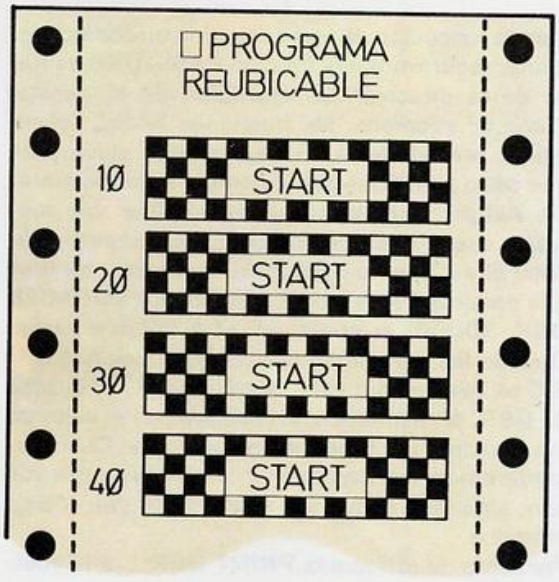
```
10 LET A=199
20 POKE 22528,A
```

El programa funciona correctamente cualquiera que sean los números de línea que se utilicen, respetando lógicamente dos normas evidentes: las líneas han de encontrarse en el mismo orden, y el programa se debe ejecutar desde el punto adecuado (en nuestro caso la primera instrucción). Como prueba de ello, tenemos el siguiente programa, que habrá de ejecutarse mediante **RUN 50** (o simplemente **RUN**).

```
50 LET A=199
98 POKE 22528,A
```

Al igual que para el BASIC los puntos de referencia son los números de instrucción, para el lenguaje máquina lo son las direcciones de memoria donde se encuentra. Vamos a cambiar de lugar nuestro programa en M/C (*machine code* o código máquina) del cuadrado intermitente: **POKE 31000,62: POKE 31001,199: POKE 31002,50: POKE 31003,0: POKE 31004,88: POKE 31005,201.**
Como recordaremos, este programa se ejecuta desde la primera dirección, que en este caso ya no es 30000, sino 31000. Efectivamente, con **RANDOMIZE USR 31000** seguirá funcionando a la perfección. Este programa es por tanto REUBICABLE también en código máquina. Sin embargo, esta propiedad no depende de la misión que cumpla el programa, sino de la manera en que es llevado a cabo. Así pues, podemos encontrar programas que cumplan un mismo objetivo en BASIC y en código máquina, y que sin embargo en un caso sean reubicables y en otro no. De todas formas, la reubicabilidad es un factor de comodidad muy a tener en cuenta, pero lamentablemente no es posible en todas las ocasiones.

Los programas que funcionan cualquiera que sea la referencia que se les otorgue, se denominan reubicables.



BITS

Por corto que sea un programa en lenguaje máquina, por ejemplo un mero código de retorno (201), siempre tarda en ejecutarse un tiempo mínimo, debido al trabajo del intérprete BASIC en ejecutar el **USR** BASIC, y manejar los parámetros de retorno según ha indicado la palabra clave que precede a la función.

*

Para hacernos una idea de la velocidad del lenguaje máquina, basta con saber que la instrucción más básica del Z-80 tarda en ejecutarse 4 estados (T). En el caso del Spectrum, **4T=4/3500000 seg.= 0.0000011428-5741 seg.**: y la más complicada, **23T = 23/3500000 seg. = 0.00000657142857**. Es decir, su velocidad de ejecución por código oscila entre menos de 2 microsegundos (millonésimas de segundo) y menos de 7 microsegundos.

En el siguiente ejemplo, veremos un programa BASIC no reubicable:

```
10 LET A=199
20 POKE 22528,A
30 GO TO 50
40 NEW
50 STOP
```

Este programa se ejecutará perfectamente desde la primera instrucción, pero sin embargo, sólo con hacer un pequeño cambio en los números de línea, puede que tenga resultados totalmente diferentes, aún siendo ejecutado desde el punto adecuado (la primera de sus líneas):

```
10 LET A=199
20 POKE 22528,A
30 GO TO 50
50 NEW
55 STOP
```

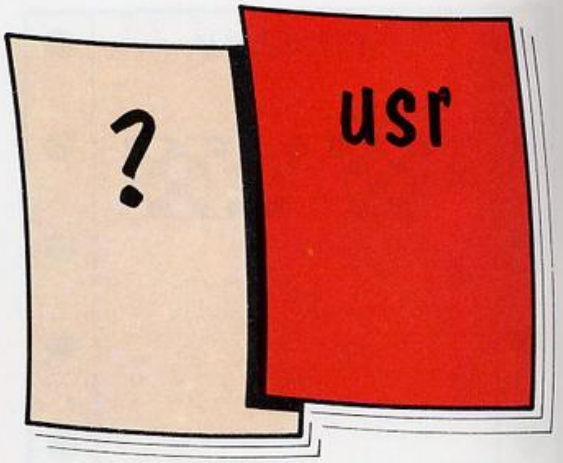
Esto mismo sucede con el código máquina. Aquellos programas que para funcionar correctamente, sin realizar en ellos ningún cambio, deben estar situados en determinado punto de la memoria se denominan NO REUBICABLES.

LA FUNCION USR

Para la ejecución de una subrutina en código máquina, recurrimos a la función BASIC **USR** seguida de la dirección de ejecución de la misma. Como ya sabemos, las funciones BASIC tienen que ir precededidas en una palabra clave para que sean admitidas por el comprobador de sintaxis. Así por ejemplo, si quieres ejecutar una subrutina desde la dirección 30000, no puedes introducir directamente **USR 30000**, sino que tendrás que emplear **PRINT USR 30000, RANDOMIZE USR 30000**, o cualquier otra palabra clave. Otra de las características de las funciones BASIC es devolver un valor determinado. En el caso de **USR**, el resultado corresponde con el valor de los registros B y C del código máquina. Como recordaremos del capítulo anterior, los registros son algo así como las «variables» del código máquina.

Por tanto, si utilizamos **PRINT USR**..., a la vuelta de la subrutina encontraremos un número escrito en la pantalla, puesto que la palabra clave utilizada como «trampolín» de **USR** ha sido **PRINT**. Para solucionar este problema, normalmente se utiliza la palabra clave **RANDOMIZE**, dado que no produce ningún efecto en la pantalla.

Pese a todo, este último sistema puede ocasionar ciertos problemas en programas que utilicen la generación de números aleatorios (**RND**). El método más eficaz para que la llamada al código máquina no provoque problemas, es el almacenamiento en una variable del resultado devuelto: **LET <variable> = USR ...** Por ejemplo, **LET A=USR 30000**.



La función **USR** *siempre debe ir precedida de una palabra clave.*

El sistema ideal para utilizar **USR** *es almacenar su resultado de retorno en una variable.*

¡!

Durante la ejecución de un programa en código máquina, puede ser que se produzca una detención con un extraño mensaje del Spectrum; esto será debido a que por un error en nuestra subrutina, la ejecución ha pasado a alguna zona de la ROM *(firmware)*.

*

La unidad de medida de tiempo de ejecución para el código máquina es el ESTADO (T). Este equivale a uno partido por la frecuencia de reloj del microprocesador. En el caso concreto del Z-80 que incorpora nuestro Spectrum, la frecuencia de reloj es de 3.500.000 Hz; es decir, T=1/3500000 seg.

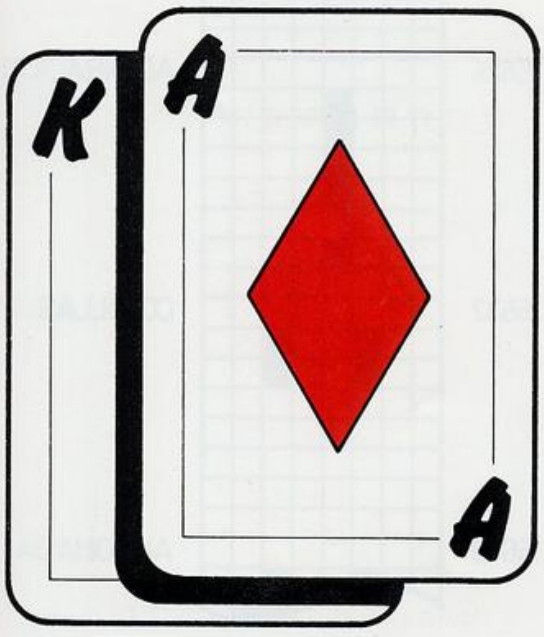
BLACK JACK

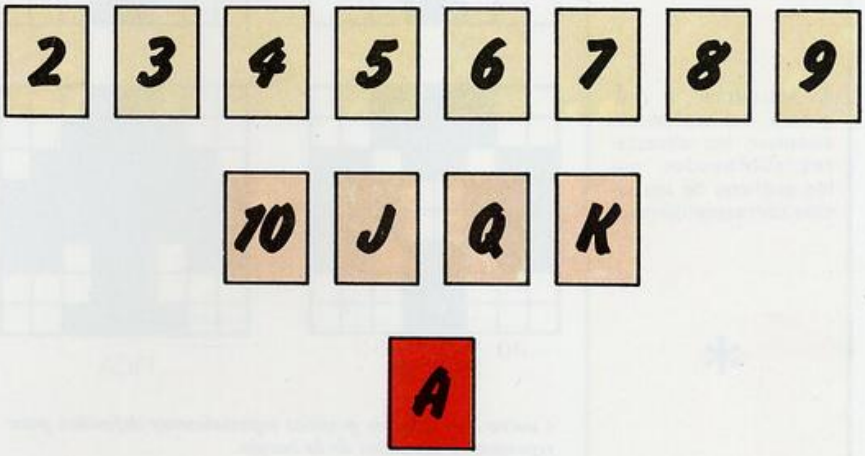
HEMOS creado para nuestros incondicionales lectores el RUN'S CASINO. Como acabamos de inaugurarlo, andamos escasos de croupieres que atiendan la totalidad de nuestras mesas; pero nuestro fiel amigo Spectrum se ha presentado voluntario para atender la mesa de Black Jack. Muchos de nosotros hemos oído hablar del Black Jack, pero desconocemos sus reglas de juego. Por ello, a continuación aparece una pequeña sinopsis de dichas normas, para que así lleguemos a ser unos empedernidos y expertos jugadores de Black Jack.

Antes de nada hemos de anticipar que las reglas de nuestro Black Jack no son exactamente las tradicionales del casino. Su mayor mejora consiste en que no se juega con pesetas, sino con una nueva moneda: el RUN DOLAR; un considerable ahorro para aquellos que no tienen buena fortuna con las cartas.

Por otra parte, aunque algunos desconozcamos el juego, su mecanismo no nos será del todo ajeno, puesto que es muy similar al popularísimo «siete y media».

La mejor jugada es el Black Jack, la cual se premia con el doble de lo apostado.





Los valores de los naipes corresponden con los números de las cartas, a excepción de las figuras, que tienen valor diez, y el as, el cual puede adoptar indistintamente los valores once o uno.

REGLAS DEL JUEGO

El Black Jack se juega con una baraja francesa (diamantes, corazones, tréboles y picas). El valor numérico de las cartas, es el impreso en las mismas, excepto en el as (A), el jack (J), la dama (Q) y el rey (K), cuyos valores respectivos son los siguientes: el as puede aumir los valores 1 u 11 según nos convenga, aunque en nuestro caso particular va a ser el Spectrum quien dé el valor que se le antoje dentro de los dos citados. El valor de las tres cartas restantes (J,Q,K) es 10.

Al comenzar el juego, el ordenador nos reparte una carta, y se despacha a sí mismo otra, ambas boca arriba. Aunque nuestros naipes aparezcan al descubierto en la pantalla, no debemos preocuparnos: el Spectrumn es honrado a carta cabal, y no nos las mirará ni lo más mínimo. El objetivo del juego es quedar los más próximos

BITS

La grabación del programa debe llevarse a cabo mediante el comando **SAVE "BLACKJACK"**. Si utilizamos el sistema de autoejecución, ésta ha de comenzar a partir de la línea 1 (**SAVE "BLACKJACK" LINE 1**).

¡!

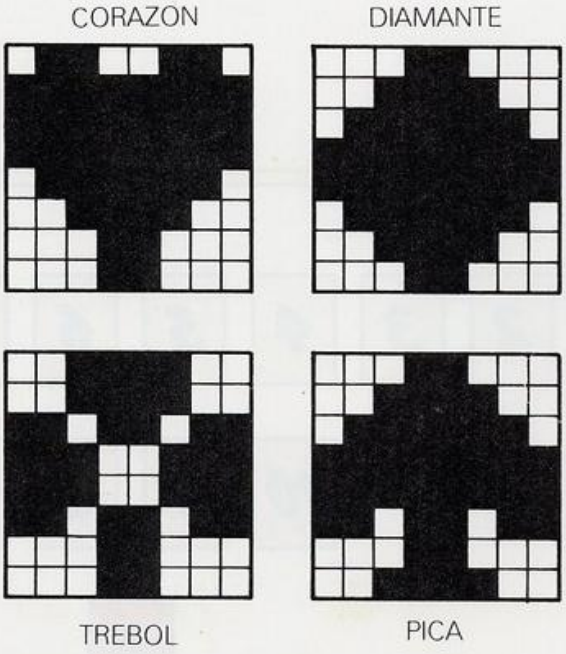
Al introducir el programa, no olvidemos sustituir los caracteres subrayados, por los gráficos de las teclas correspondientes.

*

En este programa, la duplicación de los caracteres se obtienen mediante una subrutina en BASIC, la cual accede a la dirección de la ROM donde se encuentrán almacenados los códigos que definen los caracteres del Spectrum.

*

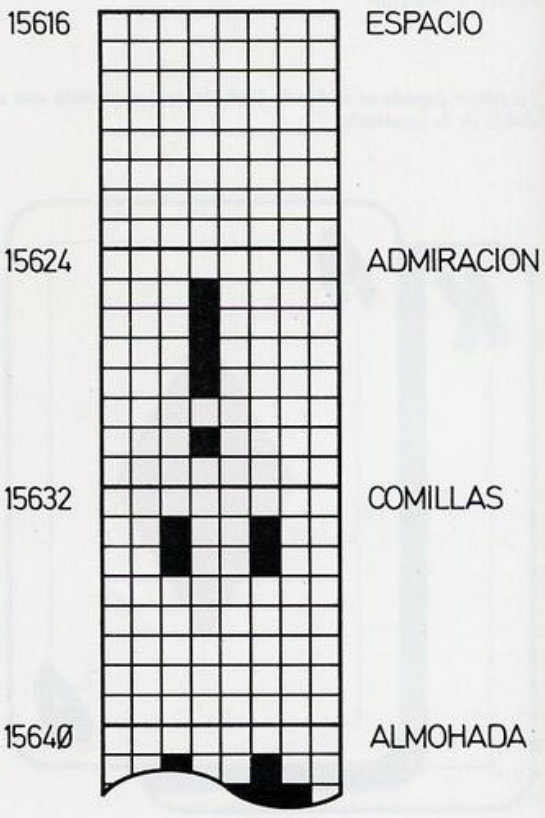
El premio por obtener la jugada de Black Jack, es en nuestro juego, el doble de lo apostado.



Cuatro han sido los gráficos especialmente definidos para representar los palos de la baraja.

posible a veintiún puntos, pero sin sobrepasarlos, puesto que en ese momento quedamos descalificados. Para conseguirlos, a la vista de nuestras cartas, y de la que le vemos al ordenador, podremos pedir un naipe más, hasta que nos plantemos en determinada puntuación. El número máximo de cartas que podemos solicitar es cinco. Una vez llegados a este punto, será al Spectrum al que le toque jugar, aunque en su caso el abanico de posibilidades es mucho más restringido: tendrá que ir sumando puntos hasta un mínimo de 17, estando obligado a plantarse en el momento en que alcance o supere dicha cantidad.
A continuación se procede a la comprobación de cual de los dos oponentes ha obtenido la mejor jugada. En este aspecto, el Spectrum se guarda dos bazas bastante importantes: la primera es que en caso de empate a puntos, siempre se declara vencedor el croupier, y la segunda, que si hemos sobrepasado la puntuación de veintiuno, habremos perdido la «mano», sin tener para nada en cuenta si a él tampoco le acompañó la suerte.
Por último, existe una jugada que supera a todos; incluso a la de obtener veintiún puntos. Esta maravillosa combinación de cartas se denomina Black Jack, y consiste en obtener de entrada, es decir, con las dos primeras cartas, una puntuación de veintiuno (un as y una carta de valor diez).
Así pues, podemos resumir las reglas del juego en los cinco puntos siguientes:
1. Se consigue Black Jack cuando con nuestras dos primeras cartas obtenemos veintiún puntos.
2. Si una vez pedidas dos cartas no hemos conseguido Black Jack, seguiremos sumando puntos, intentando acercarnos lo más posible al valor 21.
3. Podemos plantarnos en cualquier momento del juego, siempre que no hayamos sobrepasado los veintiún puntos, en cuyo caso ya no tendría sentido seguir la mano.
4. Si el Spectrum iguala nuestra puntuación habremos perdido esa baza, y en consecuencia lo que hubiésemos apostado.
5. Si la puntuación del Spectrum es 16 o menor, nuestro crupier personal tiene la obligación de pedir una nueva carta. Por el contrario, si la suma de los puntos obtenidos por el Spectrum es 17 o superior, se plantará, aunque con ello sepa que pierde la jugada.
La arquitectura del programa es completamente estructurada y sencilla de comprender, a excepción quizá de la subrutina que duplica el tamaño de los caracteres impresos en los naipes. Esta parte del programa se encuentra en la línea 8000, y a breves rasgos, funciona de la manera que a continuación explicamos.
La subrutina busca a partir de la dirección de memoria 15616 (ROM), el carácter elegido por el programa para su aumento. A partir de dicha dirección, se hayan almacenados todos los caracteres, tanto numéricos como alfabéticos, de nues-

A partir de la dirección de memoria 15616 se encuentra el área de generación de caracteres.



tro Spectrum. Una vez encontrado el carácter oportuno, el programa duplica su tamaño y lo ubica en el área destinada a los gráficos de usuario para su posterior uso.

Este método es algo lento. Podíamos haber utilizado la subrutina de caracteres gigantes de Psion Computers, y evitarnos así el aumento de caracteres, pero... como bien es sabido: «En la variación está el gusto». Nuestro objetivo ha sido además, ilustrar otra técnica con la que poder conseguir un efecto similar.

Para la introducción del programa, tengamos en cuenta que los caracteres subrayados corresponden a los gráficos de las teclas afectadas; así por ejemplo, A equivale al carácter gráfico de la A, es decir, el que se obtiene pulsando la tecla A cuando nos encontramos en el modo GRAPHICS (cursor G). Por otra parte, los caracteres doblemente subrayados (este hecho sólo se da con números), corresponden a los gráficos cambiados de los números afectados, los cuales se obtienen de forma similar a los anteriores, pero pulsando además **CAPS SHIFT**.

Para finalizar, resta advertir que es una buena

idea destacar los **REM** del programa escribiéndolos en vídeo inverso, de forma que sean más fácilmente localizables al efectuar un listado por pantalla. Los literales subrayados **INV.** entre corchetes, y **TRUE,** también entre corchetes, deben de interpretarse por tanto como **INVERSE VIDEO** y **TRUE VIDEO,** respectivamente; o lo que es lo mismo, **CAPS SHIFT + 4** y **CAPS SHIFT + 3.**

```
 10 REM ************************
 20 REM * J.M.MAYORAL SERRANO *
 30 REM ************************
 40 REM *   BLACKJACK  1985   *
 50 REM ************************
 60 LET DIREC=15360
 70 LET F$="": BRIGHT 0
 80 GO TO 140
 90 PRINT  INK 2; PAPER 7;AT P1-3,COL+1;"3333"
100 FOR W=P1-2 TO P1+2
110 PRINT  PAPER 7;AT W,COL+1;"    "
120 NEXT W
130 RETURN
140 BORDER 4
150 PAPER 4
160 INK 9
170 CLS
180 REM [INV.] INIC. VARIABLES [TRUE]
190 GO SUB 940: LET B$=F$
200 LET VOC=RIG
210 LET DIN=1000
220 GO SUB 810
230 LET IND=ABS SQR (VOC+4*RIG)+2
240 LET IND=IND+2
250 GO SUB 1530
260 REM [INV.] FIN DEL JUEGO [TRUE]
270 IF DIN<=0 THEN  PRINT  INK 9; PAPER 2; FLASH 1;AT 10,3;"TE HAS QUEDADO SIN DINERO!";AT 19,2; PAPER 6; "PULSA ~C~ PARA OTRA PARTIDA!": GO TO 1510
280 REM [INV.] CONT. JUEGO [TRUE]
290 PRINT  INK 9; PAPER 2;AT 10,2;"POSEES UN FONDO DE ";DIN;" RUN$": PRINT AT 19,1; PAPER 6; INK 9;"CUANTO DINERO QUIERES APOSTAR?"
300 INPUT "RUN$= ";PASTA
310 CLS
320 IF PASTA>DIN THEN  GO TO 290
330 IF PASTA<=0 THEN  GO TO 290
340 IF PASTA-INT PASTA<>0 THEN  GO TO 290
350 LET DIN=DIN-PASTA
360 REM [INV.]  MENSAJES  [TRUE]
370 PRINT  INK 9; PAPER 6;AT 0,0;"          TUS  CARTAS          "; PAPER 1;AT 11,0;"       CARTAS    SPECTRUM          "
380 PRINT  INK 9;AT 9,1;"Total=";AT 20,1;"Total="
390 REM [INV.]  CALCULOS  [TRUE]
400 REM [INV.]  ALEATORIEDAD CARTAS [TRUE]
410 FOR I=2 TO 1 STEP -1
420 GO SUB 650
430 NEXT I
440 FOR I=1 TO 2
450 LET VOC=U(I)=11
460 IF I=1 THEN  PRINT  FLASH 1; PAPER 2; INK 7;AT 21,2;" QUIERES UNA CARTA MAS ? ": LET A$=INKEY$: GO TO 480
470 GO TO 520
480 BEEP .05,40: LET A$=A$+" "
490 IF A$(1)="N" OR A$(1)="n" THEN  GO SUB 1470: GO TO 570
500 IF A$(1)="S" OR A$(1)="s" THEN  RANDOMIZE : GO SUB 1470: GO TO 520
510 GO TO 460
520 GO SUB 650
530 IF U(I)>10+(RIG+1)*11 THEN  LET I=2: GO TO 570
540 IF U(I+RIG+2)=-20+(17*RIG)+25 THEN  GO TO 570
550 IF I=VOC*RIG+2 AND U(I)>(RIG+4)*4 THEN  GO TO 570
560 GO TO 460
570 NEXT I
580 IF U(RIG+1)<=21 AND U(RIG+3)=5 AND (U(RIG+2)<>21 OR U(RIG+4)<>((RIG+1)*2)) THEN  GO TO 620
590 REM [INV.]  BLACKJACK  [TRUE]
600 IF U(RIG+1)=21 AND U(RIG+3)=(RIG+1)*2 AND (U((RIG+1)*2<>21 OR U(RIG+4)<>2)) THEN  LET PASTA=PASTA*1.5 : PRINT  FLASH 1;AT 5,18;" BLACKJACK ": GO TO 620
610 IF (U((RIG+1)*2)<=21 AND U(2)>=U(RIG+1)) OR U(RIG+1)>21 THEN  LET PASTA=0
620 LET DIN1=DIN
630 LET DIN=DIN+(RIG+2)*PASTA
640 GO TO 220
650 LET QUO=I+2
660 LET U(QUO)=U(QUO)+1
670 LET C=INT (RND*13)+2
680 IF C=VOC*RIG+11 THEN  LET VOC=VOC+1
690 LET U(QUO-2)=U(QUO-2)+C*(C<12)+10*(C>11)
700 IF RIG=2 THEN  GO SUB 1180
710 IF RIG*VOC>RIG THEN  GO TO 650
720 IF U(I)<20+RIG+2 OR VOC=RIG THEN  GO TO 750
730 LET VOC=VOC-1
740 LET U(I)=U(I)-10
750 GO SUB 1010
760 IF NOT RIG THEN  GO SUB 1180
770 LET FILA=RIG-2+(I*11): LET COLUM=9
780 PRINT  INK 0;AT FILA,COLUM-1;U(I)
790 RETURN
800 REM [INV.] GRAFICOS USUARIO [TRUE]
810 DATA 0,78,209,81,81,81,78,0
820 DATA (RIG+51)*2,255,255,255,126,60,24,24
830 DATA 24,60,126,255,255,126,60,24
840 DATA 24,60,126,255,255,219,24,60
850 DATA RIG+6*10,RIG+6*10,219,231,231,219,24,24
860 DATA 78,50,51,52,53,54,55
870 DATA 56,57,84,65,74,81,75
880 RESTORE 810
890 FOR K=USR "E" TO USR "E"+39
900 READ X
910 POKE K,X
920 NEXT K
930 RETURN
940 RESTORE 860: FOR B=1 TO 14
950 READ A: LET F$=F$+CHR$ A
960 NEXT B
970 LET RIG=0
980 LET POS=RIG+4
990 RETURN
1000 REM [INV.] DIBUJO CARTAS [TRUE]
1010 LET P=11*I+(POS-11)
1020 LET COL=6*U(I+POS-2)-5
1030 LET P1=P
1040 LET Q1=COL
1050 INK 2: PAPER 4
1060 PRINT AT P-3,COL;"533335"
1070 FOR K=1 TO 5
1080 PRINT AT P-3+K,COL;"5    5"
1090 NEXT K
1100 PRINT AT P+3,COL;"133332"
1110 INK 0
1120 GO SUB 90
1130 LET COLA=INT (RND*POS)+1
1140 INK 2*(COLA<3)
1150 PRINT  PAPER 7;AT P-2,COL+3;CHR$ (148+COLA)
1160 RETURN
1170 REM [INV.]  SBR DUPLIC. CARAC. [TRUE]
1180 LET X=USR "A"
1190 FOR K=0 TO 31
1200 POKE X+K,0
1210 NEXT K
1220 BRIGHT 0
1230 PAPER 7
1240 LET S=POS*2*CODE B$(C)+DIREC
1250 IF B$(C)=CHR$ 84 THEN  LET S=USR "E"
1260 FOR J=0 TO 7
1270 LET V=PEEK (S+J)
1280 LET C=0
1290 LET D=64*POS
1300 FOR K=8 TO 1 STEP -1
1310 LET D=D/2
1320 IF V>=D THEN  LET C=C+3*D*D: LET V=V-D
1330 NEXT K
1340 LET H=INT (C/256)
1350 LET L=C-256*H
1360 POKE X,H
1370 POKE X+1,H
1380 POKE X+8,L
1390 POKE X+9,L
1400 LET X=X+2+8*(J=3)
1410 NEXT J
1420 PRINT  BRIGHT 1;AT P,COL+2;"AB"
1430 PRINT  BRIGHT 1;AT P+1,COL+2;"CD"
1440 PAPER 4
1450 BRIGHT 0:
1460 RETURN
1470 REM [INV.]  BORRADO MENSAJE [TRUE]
1480 PRINT  PAPER 4;AT 21,0;"                "
1490 RETURN
1500 REM [INV.] OTRA JUGADA ? [TRUE]
1510 IF INKEY$="C" OR INKEY$="c" THEN  RUN
1520 GO TO 1510
1530 DIM U(IND): RETURN
```