SVI·728

ORDENADOR PERSONAL
MANUAL DEL USUARIO

SVI®
SPECTRAVIDEO

SVI•728

ORDENADOR DOMÉSTICO

Manual del usuario

Publicado por
SPECTRAVIDEO INTERNACIONAL LTD.

Primera edición
Primera impresión 1986

Spectravideo Internacional Ltd. no acepta ninguna responsabilidad por los daños producidos directa o indirectamente con ocasión del uso o el funcionamiento de este equipo.

Se ha procurado que la información que se suministra en este manual sea lo más completa y correcta posible. No obstante, dada nuestra política de introducir continuamente mejoras en el diseño y el funcionamiento de nuestros productos, nos reservamos el derecho a cambiar las especificaciones del producto sin previo aviso.

Versión española producida por Vector Ediciones para SPECTRAVIDEO ESPAÑA, S.A.
Traducción: Emilio Benito Santos

Interferencias en radiofrecuencia

Este equipo puede generar y recibir ondas de radiofrecuencia y, si no se lo instala y utiliza adecuadamente, es decir, de acuerdo con las instrucciones del fabricante, puede interferir la recepción de las señales de radio y televisión. A pesar de que ha sido diseñado para reducir estos problemas al mínimo posible, no se puede garantizar que no vayan a presentarse en alguna instalación concreta.

En caso de que se produzcan estas interferencias, lo que se puede comprobar apagado y encendiendo el equipo, sugerimos al usuario que pruebe a hacer lo siguiente:

- Reorientar la antena del receptor.
- Cambiar de sitio el receptor o el ordenador.
- Alejar el ordenador del receptor.
- Conectar el ordenador a una toma de corriente distinta de la del ordenador, de manera que los dos aparatos estén alimentados por circuitos diferentes.

Si los problemas persisten, el usuario debe consultar a su distribuidor o a un técnico experto en radio/televisión.

CONTENIDO

Introducción

¡Bienvenido al mundo de la alta tecnología! Acaba usted de comprar un aparato del que disfrutará durante mucho tiempo. El ordenador doméstico **SVI•728** va a abrirle las puertas del futuro.

Los ordenadores personales **Spectravideo** tienen ciertas características que le permitirán aprovechar plenamente, no sólo la tecnología informática actual, sino también los avances que están por venir.

El **SVI•728** es la herramienta con la que usted va a participar en la revolución informática de esta década, y éste es el manual de funcionamiento de esa herramienta.

Así pues, ¡bienvenido al mañana! Deje que el **SVI•728** le ayude, pero no olvide la regla de oro de la informática: el ordenador no tiene más inteligencia que la que usted quiera darle. Sin su dueño, no es más que un montón de cables, circuitos integrados, metal y plástico. ¡Usted es la pieza clave de su nuevo sistema informático!

Capítulo 1

Descripción general del sistema

La terminología que vamos a emplear en las páginas que siguen puede serle desconocida si es usted un recién llegado al mundo de la informática. Pero no debe preocuparse. Este manual ha sido especialmente diseñado para enseñar a los principiantes a utilizar el potente lenguaje MSX BASIC con que está equipado el **SVI•728.**

Esta primera descripción pretende dar al usuario, tanto al principiante como al experto, una idea de conjunto sobre la potencia y belleza del **SVI•728**, el más asequible y con mayores posibilidades de ampliación de los ordenadores personales.

Su **SVI•728** contiene tres potentes microprocesadores, gracias a los cuales puede visualizar imágenes en color, generar música y controlar un programa, todo ello a un mismo tiempo.

El control de todos los componentes del sistema está confiado a un Z80, que es el microprocesador de 8 bits más popular. Él es el encargado de almacenar, cargar y ejecutar los programas.

Hay un procesador de imágenes (VDP) que genera todas las señales de video, de control y de sincronización necesarias. El VDP puede visualizar 16 colores y controlar 32 sprites, además de la pantalla de texto de hasta 40 columnas de anchura.

Por otra parte, el generador programable de sonido (PSG) puede producir tanto música como una amplia variedad de sonidos.

El lenguaje MSX BASIC está grabado en una ROM (*Read Only Memory*, memoria de sólo lectura) de 32 K. Las instrucciones del usuario (es decir, los programas) que controlan el funcionamiento del ordenador se almacenan en la RAM (*Random Access Memory*, memoria de acceso arbitrario) de 64 K. El resto de la memoria, 16 K, los utiliza el VDP para almacenar la información en la que se basa la construcción de la pantalla.

MSX BASIC

El potente lenguaje MSX BASIC, desarrollado por Microsoft® , es una ampliación del mismo lenguaje BASIC de que están dotados otros ordenadores que cuestan tres veces lo que el **SVI•728**.

Si es usted principiante en programación, encontrará en el MSX BASIC un lenguaje versátil que es muy fácil de aprender y utilizar. Los programadores expertos apreciarán sus avanzados recursos y la flexibilidad que ofrece en la escritura, edición, depuración y ejecución de los programas.

El MSX BASIC tiene incorporadas las siguientes funciones:

a) Editor de pantalla.

b) Control total de gráficos y sonidos.

c) Nuevas órdenes para el control de sprites.

d) Aritmética de doble precisión, utilizable directamente por los programas de aplicación empresariales.

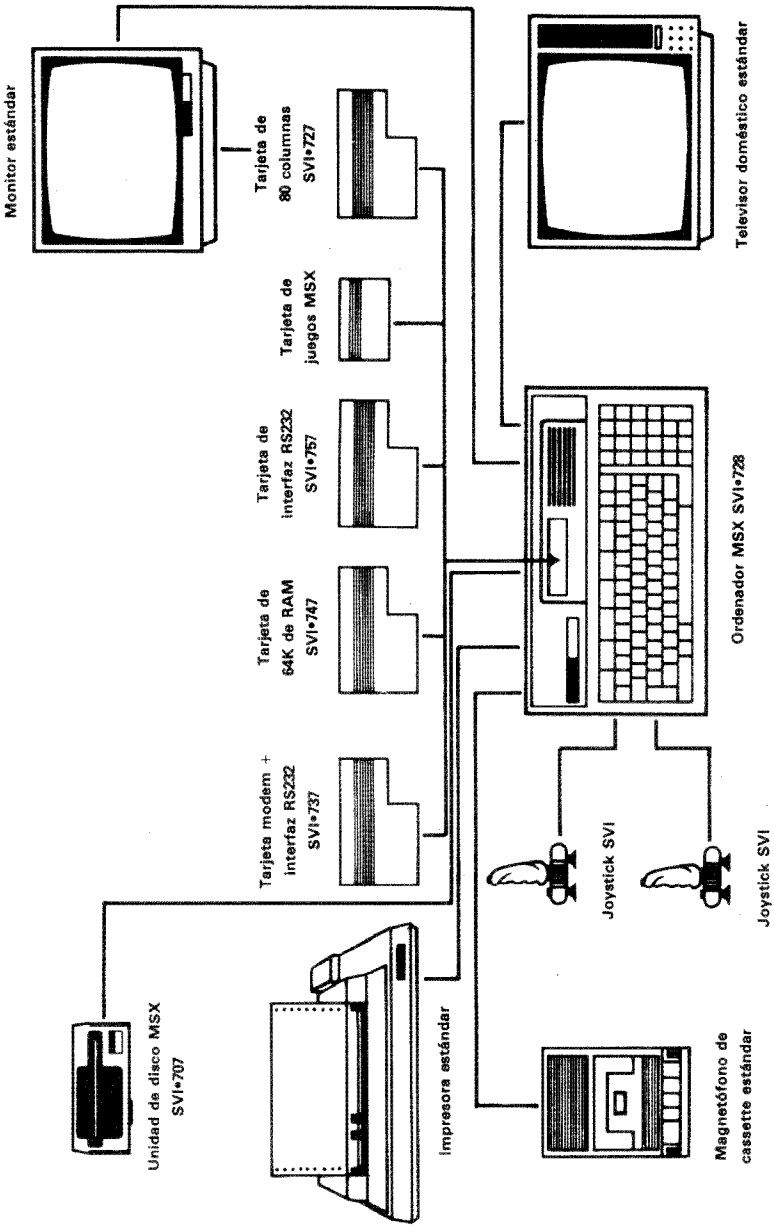
e) Órdenes especiales de gestión de interrupciones, que hacen posible el funcionamiento en tiempo real desde BASIC.

f) Teclas especiales de función que tienen asignadas las órdenes más frecuentemente utilizadas y son reprogramables por el usuario.

g) Un juego de caracteres muy completo, tanto para la pantalla como para la impresora, que incluye letras acentuadas y otros caracteres específicos del idioma castellano.

RECOMENDACIÓN

La forma de llegar a dominar el funcionamiento del **SVI•728** es armarse de paciencia para leer este libro completo y probar todos los ejercicios en el propio ordenador. Ocurre lo mismo que cuando se aprende a conducir: sólo la práctica hace maestros. No se deje desanimar por los errores; corríjalos y vuelva a intentar lo que se hubiera propuesto. Experimente cuanto desee: es imposible estropear el ordenador desde el teclado.



ESQUEMA DE CONEXIÓN DE PERIFÉRICOS PARA EL SISTEMA SVI•728

AMPLIABILIDAD

Para completar el sistema, usted puede añadirle diversos periféricos a través de la puerta de expansión o de la 'ranura de juegos'. Por ejemplo, puede conectar la unidad de disco SVI•707 y la tarjeta de 80 columnas SVI•727, las cuales le permitirán ejecutar los miles de programas disponibles en el sistema operativo CP/M. También puede conectar el sintetizador de voz, el modem de 300 baudios SVI•737 con interfaz serie, así como muchos otros periféricos suministrados por otros fabricantes: sintetizador de música, lápiz óptico, etc.

NOTA

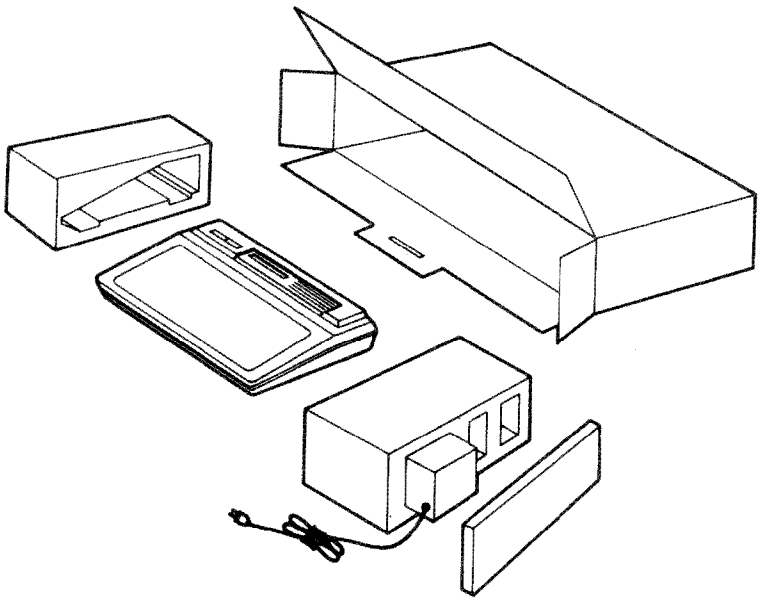
MSX es un marca comercial registrada por Microsoft Corporation.

Microsoft® es marca comercial registrada por Microsoft Corporation.

CP/M es marca comercial registrada por Digital Research.

DESEMBALAJE DEL SVI•728

El ordenador **SVI•728**, los cables de video y audio y la fuente de alimentación vienen en una caja, convenientemente protegidos por molduras de poliuretano.



RECOMENDACIÓN

En previsión de que algún día tenga que volver a transportar el equipo, quizá para enviarlo a reparar, guarde todos los materiales de embalaje.

CONTENIDO

Compruebe que el contenido de la caja es el siguiente:

1. Ordenador **SVI•728**.
2. Cable de audio y video (RF).
3. Fuente de alimentación.
4. Este manual.
5. Tarjeta de garantía.

Si falta alguno de estos objetos, avise inmediatamente a su distribuidor.

INSTALACIÓN DEL SISTEMA

Para conectar el sistema necesitará al menos dos enchufes murales de red: uno para el ordenador y otro para el monitor o el televisor.

Elija una posición que le resulte cómoda. Evite que el equipo quede expuesto a fuentes de calor (luz solar directa, radiadores de calefacción, etc.)

SALIDA DE TV

Envía las señales del **SVI•728** a la entrada de antena del televisor.

SALIDA DE AUDIO Y VIDEO

Envía las señales del **SVI•728** al monitor.

GND

Masa del sistema.

PUERTA DE EXPANSIÓN

Para conectar la unidad de disco SVI•707 y otros periféricos con el **SVI•728**.

PUERTA DE IMPRESORA

Salida hacia una impresora dotada de interfaz tipo Centronics.

PUERTA DE CASSETTE

Para conectar el **SVI•728** con un magnetófono de cassettes.

RANURA DE JUEGOS

Para conectar cartuchos de juegos MSX y diversos periféricos.

ZÓCALO PARA LA FUENTE DE ALIMENTACIÓN

Para conectar la fuente de alimentación. El otro extremo se enchufa en una toma mural.

INTERRUPTOR DE ALIMENTACIÓN

Controla la entrada de corriente en el **SVI•728**.

PUERTA DE JOYSTICK 1

Para conectar un joystick.

PUERTA DE JOYSTICK 2

Para conectar el segundo joystick.

CONEXIÓN DE UN MONITOR

Por el lado del **SVI•728**, el cable debe terminar en dos clavijas de tipo RCA para audio y video. La clavija del otro extremo depende de cómo sea el zócalo del monitor.

1. Conecte las dos clavijas RCA en los zócalos 'audio' y 'video' del **SVI•728**.
2. Conecte el otro extremo del cable en el monitor.

CONEXIÓN DEL TELEVISOR

1. Conecte un extremo del cable de RF en la salida 'TV' del **SVI•728**.
2. Conecte el otro extremo en la entrada de antena del televisor.

CONEXIÓN DE UN MAGNETÓFONO DE CASSETTE

Se necesita un cable que termine, por un extremo, en una clavija DIN de cinco patillas y, por el otro, en las clavijas adecuadas al magnetófono que se vaya a utilizar.

1. Conecte la clavija DIN de 5 patillas en el zócalo 'cassette' del **SVI•728**.
2. Conecte las clavijas del otro extremo en los zócalos del magnetófono.

CONEXIÓN DE UNA IMPRESORA

Se necesita un cable de tipo Centronics SVI•207.

1. Conecte un extremo del cable en la salida de impresora del **SVI•728**.
2. Conecte el otro extremo en el zócalo de la impresora.
3. Enchufe el cable de alimentación de la impresora en una toma mural de red.

CONEXIÓN DE LA FUENTE DE ALIMENTACIÓN

1. Antes de nada, cerciórese de que el interruptor de alimentación está apagado.
2. De los dos cables de la fuente de alimentación, conecte el más grueso en el zócalo del **SVI•728**.
3. Enchufe el otro en una toma mural de red.

ENCENDIDO DEL SISTEMA

Una vez realizadas todas las conexiones, encienda el televisor (o el monitor). Después ponga el interruptor de alimentación del **SVI•728** en la posición 'ENC.'.

El piloto 'POWER ON' debe encenderse.

AUTOCOMPROBACIÓN

El **SVI•728** tiene incorporada una rutina gracias a la cual realiza automáticamente una comprobación del funcionamiento del sistema cada vez que se lo enciende.

En caso de dificultad, consulte el Apéndice G.

RANURA DE JUEGOS

La ranura de juegos de MSX está situada en la parte superior del ordenador.

Antes de introducir o extraer un cartucho MSX, cerciórese de que el ordenador está apagado.

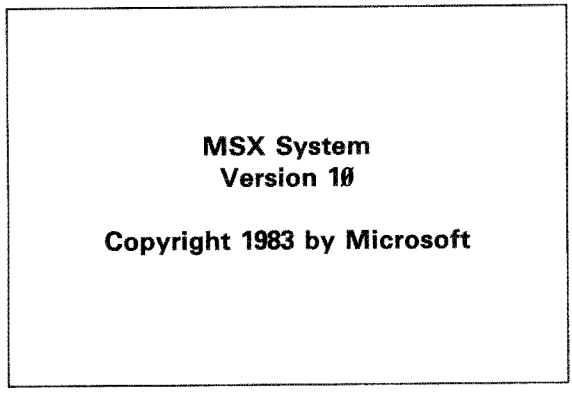
Capítulo 2

El teclado

Al menos en una fase inicial, la programación consiste en dar instrucciones al ordenador a través del teclado. Tanto las instrucciones como las respuestas del ordenador serán visibles en la pantalla del televisor o del monitor.

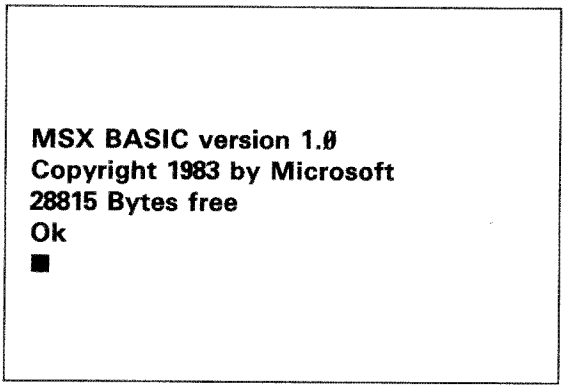
El teclado del **SVI•728** no debería resultarle del todo extraño, ya que se parece bastante al de una máquina de escribir. Sin embargo, en el **SVI•728** hay numerosas teclas adicionales que son necesarias para comunicarse con el ordenador.

Encienda el sistema (con el interruptor de alimentación). Si en la pantalla aparece el siguiente mensaje, hemos empezado con buen pie:



De no ser así, consulte el Apéndice G.

Al cabo de unos segundos, ese mensaje es sustituido por otro:



Ok

EL CURSOR

La palabra 'Ok' es un mensaje que emite el ordenador para indicar que está preparado para aceptar nuestras instrucciones. El cuadrado blanco que hay debajo de 'Ok' es el *cursor*. Indica el lugar de la pantalla en el que aparecerá la próxima letra que escribamos.

Empecemos a escribir para ir familiarizándonos con el **SVI•728** y con las características especiales que nos ayudarán en nuestro trabajo con el ordenador.

Escriba la siguiente palabra:

PRINT

Como puede ver, el cursor se desplaza un lugar a la derecha cada vez que se pulsa una tecla. Observe las teclas [SHIFT] que hay en la parte inferior del teclado, una a cada lado. Pulse una de esas teclas y, antes de soltarla, pulse también la tecla de la coma, [,]. En la pantalla habrá aparecido el signo de *comillas* ".

Ahora escriba lo siguiente:

VIVA EL SVI-728

Complete la instrucción «cerrando» las comillas, es decir, pulse [SHIFT][,]. La pantalla debería estar de la siguiente manera:

```
MSX BASIC version 1.0
Copyright 1983 by Microsoft
28815 Bytes free
Ok
print "viva el SVI-728" ■
```

Si se equivoca al escribir, puede borrar el carácter erróneo pulsando la tecla [←] y luego escribir el correcto.

Pulse ahora la tecla [ENTER] y observe la pantalla:

```
MSX BASIC version 1.0
Copyright 1983 by Microsoft
28815 Bytes free
OK
print "viva el SVI-728"
viva el SVI-728
Ok
■
```

Vamos a explicarle lo que ha ocurrido. El ordenador tiene dos modos principales de funcionamiento: el modo inmediato o directo y el modo de programa. Todavía no hemos escrito ningún programa, luego estamos en modo inmediato. Esto quiere decir que estamos dando instrucciones cuyo efecto queremos ver inmediatamente.

Pues bien, al pulsar [ENTER], le hemos dicho al ordenador que hemos terminado de escribir nuestra instrucción y que queremos que la obedezca. La ins-

trucción consiste en escribir en la pantalla (**PRINT**) un mensaje, y eso es lo que ha hecho el ordenador. Terminado su trabajo, el ordenador emite el mensaje **Ok** para indicar que está esperando nuevas instrucciones.

Para escribir letras mayúsculas, se debe hacer como en las máquinas de escribir: pulsar [SHIFT] al mismo tiempo que la tecla de la letra. Cuando vaya a escribir en mayúsculas un texto relativamente largo, puede «bloquear el teclado en mayúsculas» pulsando [CAPS/LOCK]; de esa forma no hace falta que tenga pulsada la tecla [SHIFT] todo el tiempo.

La tecla [CAPS/LOCK] bascula entre mayúsculas y minúsculas. Así pues, para deshacer el bloqueo de mayúsculas y dejar el teclado en situación normal, pulse [CAPS/LOCK] por segunda vez.

Ahora le toca a usted el turno de pedir al ordenador que escriba otros mensajes. No olvide que debe pulsar [ENTER] al final de cada instrucción. Si el mensaje es suficientemente largo, el ordenador se encargará de pasar automáticamente al principio de la línea siguiente. En esto se distingue de las máquinas de escribir, en las que el usuario debe pulsar la tecla de retorno del carro cuando está llegando al final de la línea; aquí no es necesario (ni se debe) hacer tal cosa.

Cuando se comete un error al escribir algo que esté fuera de las comillas (es decir, algo que debiera ser una palabra reservada de BASIC), el ordenador no obedece la instrucción, sino que avisa de lo que ha ocurrido:

```
PRINT "VIVA EL SVI-728
VIVA EL SVI-728
Ok
PRONT "ES EL MEJOR"
Syntax error
Ok
```

Este mensaje, **Syntax error** (error de sintaxis), indica que hemos escrito algo que no está de acuerdo con las reglas de BASIC. Pero no se preocupe: el ordenador es muy paciente. En el siguiente capítulo le explicaremos cómo corregir errores de este tipo.

Se puede borrar la pantalla rápidamente pulsando la tecla [CLS/HM] en combinación con [SHIFT]. 'CLS' es abreviatura de *CLear Screen*, 'borrar pantalla'.

Dedique algún tiempo a practicar con órdenes **PRINT**. Cuando empiece a aburrirse, siga leyendo.

DISTRIBUCIÓN DEL TECLADO

Hasta ahora hemos tenido ocasión de conocer muy pocas teclas de las que pudiéramos llamar 'especiales' ([←], [ENTER], [SHIFT], etc.).

El **SVI•728** tiene recursos muy interesantes a los que se accede pulsando ciertas teclas. Dedicaremos el resto de este capítulo a describir esas teclas especiales, muchas de las cuales serán nuevas para usted.

TECLAS DE FUNCIÓN

Fíjese en la fila superior del teclado:

Son las *teclas de función*, que se distinguen por estar marcadas con la letra 'F'. Estas teclas proporcionan un modo de ahorrar trabajo, pues permiten escribir un grupo de caracteres, o incluso una orden completa, con la sola pulsación de una tecla.

Al encender el ordenador, estas teclas tienen asignadas las siguientes funciones:

Tecla	Función
F1	**color**
F2	**auto [10,10]** [ENTER]
F3	**goto**
F4	**list**
F5	**run** [ENTER]
F6	**color 15,4,7**
F7	**cload"**
F8	**cont** [ENTER]
F9	**list.** [ENTER]
F10	[CLS] **run** [ENTER]

Las funciones F1 a F5 se obtienen pulsando la tecla correspondiente. Las funciones F6 a F10 requieren la pulsación simultánea de [SHIFT].

[F1] **COLOR** — La orden **COLOR** se utiliza para controlar los colores del texto, el fondo y el borde de la pantalla.

[F2] **AUTO** — La orden **AUTO** hace que el ordenador genere automáticamente números de línea. Esto es muy interesante cuando se está introduciendo programas por el teclado.

[F3] **GOTO** — **GOTO** es una orden que permite poner en marcha un programa empezando por un lugar (número de línea) especificado.

[F4] **LIST** — Esta orden hace que el **SVI•728** escriba en la pantalla el listado de las instrucciones que forman el programa actualmente almacenado en la memoria.

[F5] **RUN** — **RUN** es una orden que pone en marcha el programa actual; es decir, hace que el ordenador obedezca las instrucciones contenidas en el programa.

[F6] **COLOR 15,4,7** — Esta orden asigna los siguientes colores: blanco para el texto, azul para el fondo y cyan para el borde. Es precisamente la selección de colores que se establece automáticamente cuando se enciende el ordenador.

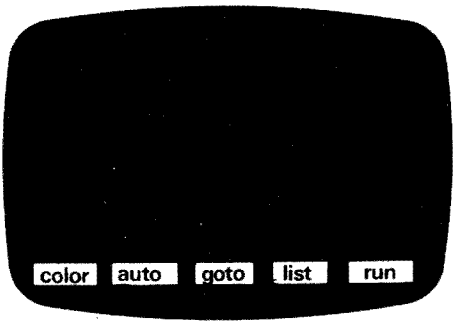
[F7] **CLOAD"** — **CLOAD** es la orden que carga (lee) datos en la cinta y los carga en la memoria del ordenador.

[F8] **CONT** — Esta orden hace que el ordenador reanude la ejecución del programa tras una interrupción.

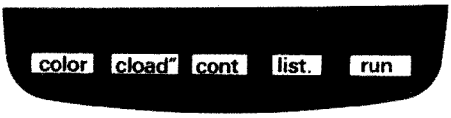
[F9] **LIST.** — Es una orden similar a **LIST**, pero el punto hace que sólo se liste la última línea con la que se ha trabajado (es decir, la 'línea actual').

[F10] **RUN** — Esta orden es similar a la asignada a [F5]. La única diferencia es que [F10] borra la pantalla antes de iniciar la ejecución del programa.

En la última línea de la pantalla se muestra normalmente las funciones que están asignadas a las teclas F1 a F5:



Cuando se pulsa SHIFT aparecen las funciones correspondientes a las teclas F6 a F10:



Estas asignaciones pueden ser modificadas en cualquier momento por el usuario para adapatarlas a sus preferencias o necesidades.

TECLADO NUMÉRICO Y TECLAS DE MOVIMIENTO DEL CURSOR

A la derecha del teclado principal se encuentra un grupo de 24 teclas, especialmente diseñado para facilitar los trabajos de introducción de datos numéricos, proceso de textos y otras aplicaciones profesionales, así como para mover el cursor en todas las situaciones.

Teclado numérico

Las teclas numéricas (0-9) son equivalentes a las de la segunda fila del teclado principal (sin SHIFT). Están repetidas aquí con la disposición habitual de los teclados numéricos. A su lado se encuentran las teclas que producen los operadores aritméticos (+, −, *, /), así como el punto decimal y la coma.

Tecla SELECT

No realiza ninguna función en BASIC, sino sólo en ciertos programas de aplicación (proceso de datos, proceso de textos, etc.).

Tecla [INS/PASTE]

Sirve para poner el ordenador en el 'modo de inserción'. Cuando quiera insertar caracteres en una línea, lleve el cursor a la posición deseada, pulse esta tecla y escriba los caracteres que deba insertar.

Tecla [CLS/HM-COPY]

Lleva el cursor al extremo superior izquierdo de la pantalla. Si se la pulsa en combinación con [SHIFT], hace lo mismo y además borra la pantalla.

Teclas de movimiento del cursor

Estas teclas, [↑], [↓], [←] y [→], permiten mover el cursor por la pantalla. Pulsándolas por parejas, se puede mover el cursor en diagonal; por ejemplo, con [↑] y [→] se desplaza el cursor hacia el rincón superior derecho.

TECLAS DE CONTROL DEL FLUJO DE PROGRAMAS

Las teclas siguientes controlan la ejecución y la interrupción de los programas y de la órdenes directas:

[STOP]

Cuando se pulsa esta tecla una vez, el ordenador interrumpe momentáneamente la tarea que esté llevando a cabo, la cual se reanuda cuando se pulsa por segunda vez la misma tecla.

[CTRL][STOP]

Esta combinación de teclas interrumpe el programa o la orden actual y devuelve el ordenador al modo directo, en el que queda a la espera de nuevas instrucciones del usuario.

[ENTER]

Esta tecla se pulsa cuando se termina de escribir una instrucción. Como ya hemos mencionado, **no** se la debe usar como se haría con la tecla de retorno del carro en una máquina de escribir. En efecto, en el caso de que una instrucción sea tan larga que no quepa en una línea de la pantalla, el propio ordenador se encarga de hacer avanzar el cursor en el momento oportuno hasta el principio de la línea siguiente.

Esta instrucción no cabía en una línea (29 caracteres), y por eso el **SVI•728** ha realizado los avances de línea necesarios. En este ejemplo sólo se ha debido pulsar la tecla [ENTER] después de escribir la palabra 'PROGRAMAR' y cerrar las comillas.

```
ME GUSTARIA TERMINAR CON ESTA
INTRODUCCION Y EMPEZAR A PROG
RAMAR
■
```

OTRAS TECLAS

CAPS/LOCK — Ya hemos visto antes que esta tecla sirve para bloquear el teclado en mayúsculas, situación en la cual las teclas 'literales' producen letras mayúsculas aunque no se pulse SHIFT al mismo tiempo. Este bloqueo de mayúsculas se deshace pulsando CAPS/LOCK por segunda vez.

DEL/CUT — Al pulsar esta tecla se borra el carácter que está bajo el cursor.

ESC — Esta tecla se utiliza en los programas de aplicación, generalmente para interrumpir el programa o para reanudar la ejecución después de una interrupción.

⇒ — No realiza ninguna función en BASIC. Se la suele utilizar en programas de proceso de textos y en aplicaciones similares para producir una sangría de cinco espacios al principio de los párrafos.

⇐ — Al pulsar esta tecla el cursor retrocede a la columna inmediatamente anterior y borra el carácter que estaba en esa posición.

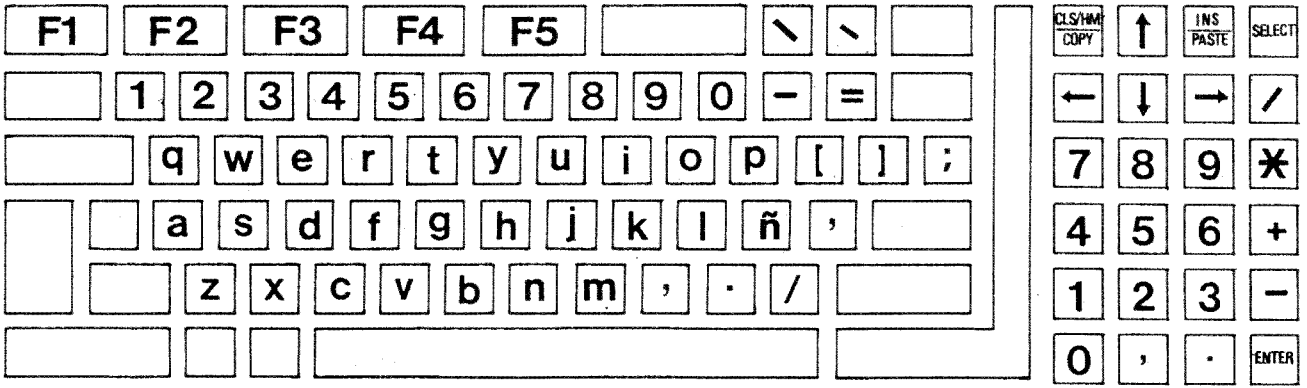
GRAPH — Esta tecla se utiliza en combinación con la teclas 'normales' para generar los diferentes símbolos gráficos que éstas tienen asignados. Para ello se pulsa GRAPH y, sin soltarla, se pulsa también la tecla deseada.

CODE

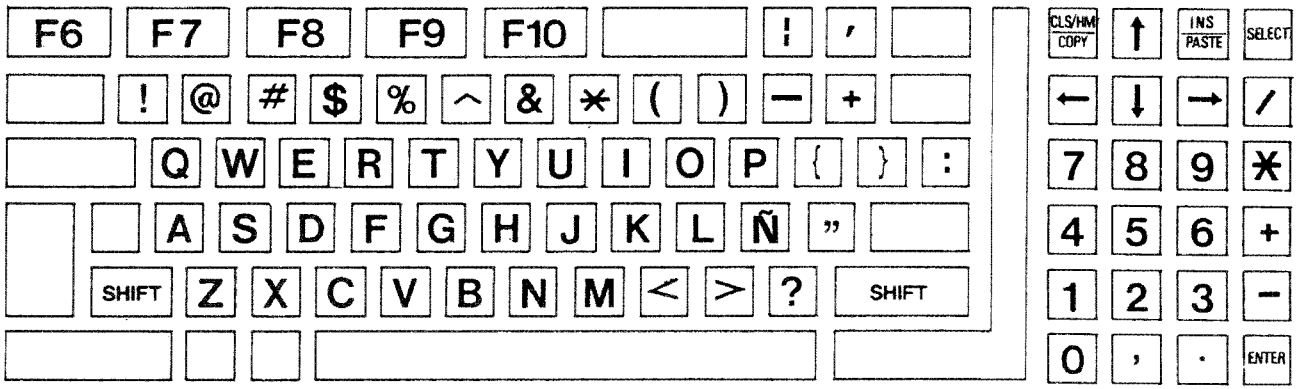
Esta tecla, combinada con las teclas 'normales', con o sin SHIFT, genera letras acentuadas, letras griegas y otros caracteres especiales.

En resumen, en estos dos capítulos usted ha aprendido a conectar el ordenador y ha empezado a familiarizarse con el teclado. Por consiguiente, está en condiciones de abordar la siguiente etapa, que es aprender el funcionamiento del 'editor de pantalla' (capítulo 3).

No obstante, si está impaciente por empezar a hacer algo más interesante con el ordenador, puede ir al Apéndice I, en el que le ofrecemos una breve demostración. Cuando haya terminado con él, prosiga la lectura del manual a partir del capítulo 3.

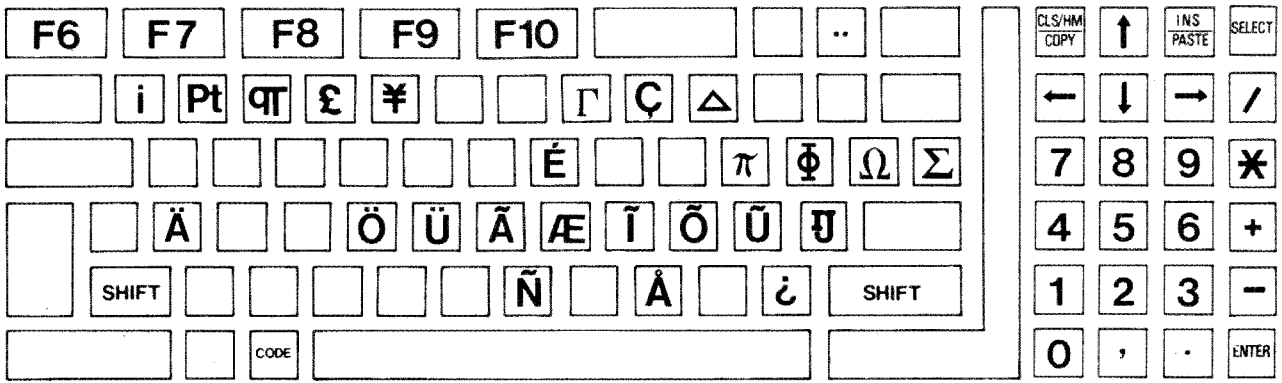


TECLADO NORMAL

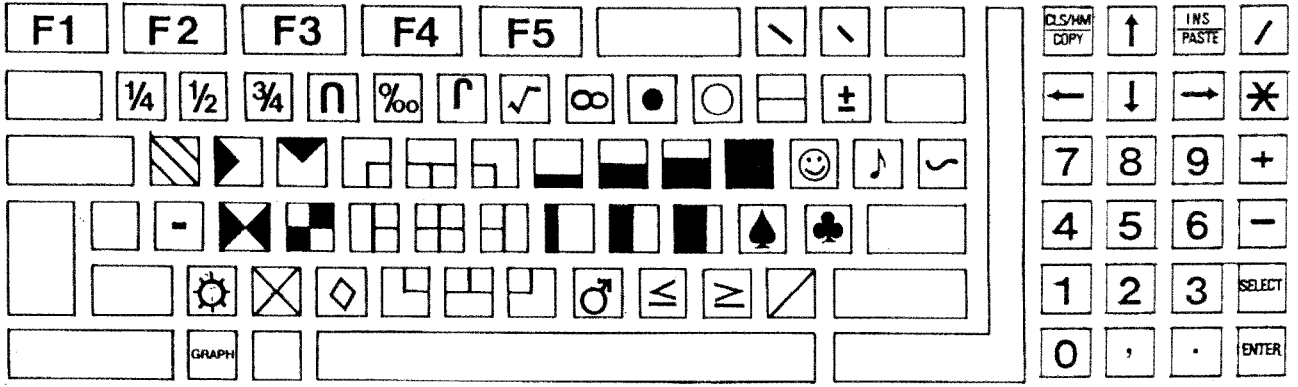


TECLADO CON SHIFT

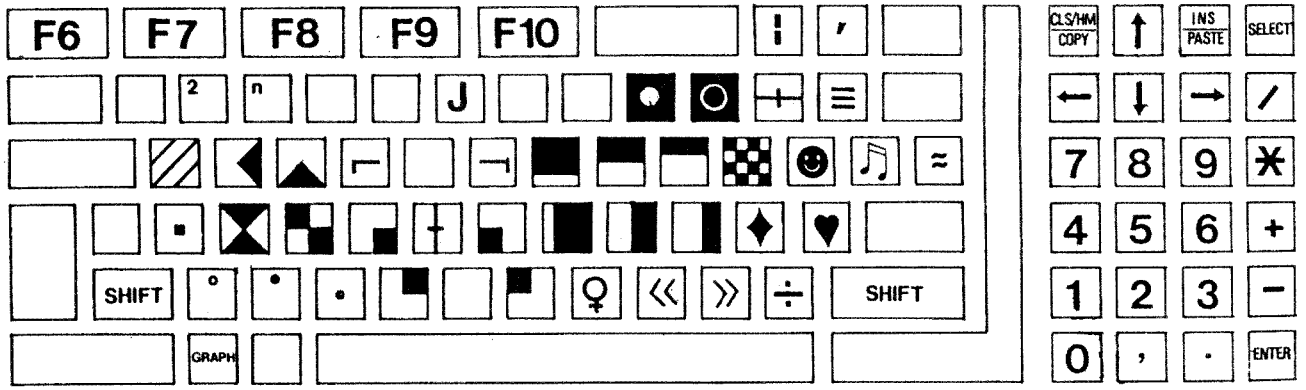
TECLADO CON CODE



TECLADO CON CODE Y SHIFT



TECLADO CON GRAPH



TECLADO CON GRAPH Y SHIFT

Capítulo 3

El editor de pantalla

El 'editor de pantalla' es un recurso de BASIC gracias al cual se puede corregir o modificar cualquier línea que esté visible en la pantalla. Las líneas pueden ser modificadas una a una. El editor de pantalla se activa siempre que aparece el mensaje **Ok**, y permanece activo hasta que se da la orden **RUN** para ejecutar un programa. Mediante las teclas de movimiento del cursor y las teclas de edición se puede ir a cualquier lugar de la pantalla y realizar la corrección deseada.

EL EDITOR DE PANTALLA

Veamos un sencillo ejemplo de utilización del editor de pantalla:

Introduzca el siguiente programa:

```
10 REM DOMOSTRACION DEL
   EDITOR[ENTER]
20 PRINT "DEMOSTRACION DEL"[ENTER]
30 END[ENTER]
```

Nota. Recuerde que toda línea de programa debe empezar con un número de línea. Si se equivoca al transcribir este programa, pulse [ENTER] y vuelva a escribir la línea completa.

Cuando haya terminado, pulse [SHIFT][CLS/HM]. Después pulse [F4] (o escriba **LIST**) y pulse [ENTER]. En la pantalla debería aparecer lo siguiente:

```
LIST

10 REM DOMOSTRACION DEL
   EDITOR
20 PRINT "DEMOSTRACION DEL"
30 END
```

Ahora vamos a corregir la errata de la palabra 'DOMOSTRACIÓN' de la línea 10. En primer lugar pulse [↑] para llevar el cursor al principio de esa línea. Después pulse [→] hasta que el cursor quede sobre la 'O' que debemos corregir:

```
                ↓
10 REM DOMOSTRACION DEL
   EDITOR
20 PRINT "DEMOSTRACION DEL"
30 END

Ok
```

Pulse la letra 'E' para corregir la palabra y pulse [ENTER]. La línea 10 ha quedado almacenada en la memoria en versión corregida:

10 REM DEMOSTRACION DEL EDITOR

Lo que usted acaba de hacer es poner la letra 'E' en lugar de la 'O', no sólo en la pantalla, sino también en la memoria del ordenador. Compruebe que el **SVI•728** ha aceptado el cambio haciendo lo siguiente: pulse [SHIFT][CLS/HM] para borrar la pantalla; pulse [F4] y después [ENTER] para ver el listado de la nueva versión del programa:

```
10 REM DEMOSTRACION DEL
   EDITOR
20 PRINT "DEMOSTRACION DEL"
30 END

Ok
```

La siguiente corrección va a consistir en insertar las palabras 'EDITOR DE PANTALLA' en la línea 20. Empiece por colocar el cursor sobre el segundo signo de comillas de esa línea:

```
10 REM DEMOSTRACION DEL
   EDITOR
20 PRINT "DEMOSTRACION DEL"
30 END

Ok
```

MODO DE INSERCIÓN

Pulse la tecla [INS/PASTE]; la altura del cursor se ha reducido a la mitad, lo que quiere decir que el **SVI•728** ha entrado en el 'modo de inserción'. Escriba las palabras 'EDITOR DE PANTALLA' y pulse [ENTER]. El texto que faltaba ha quedado insertado en su sitio. Compruébelo por el mismo procedimiento que seguimos cuando hicimos la corrección en la línea 10:

```
10 REM DEMOSTRACION DEL
   EDITOR
   20 PRINT "DEMOSTRACION DEL
   EDITOR"
30 END

Ok
```

Otra forma de corregir una línea (sin recurrir al editor de pantalla) consiste en escribir la versión nueva con el mismo número de línea. BASIC almacena entonces la versión nueva en sustitución de la antigua.

Capítulo 4

Introducción a BASIC

ACERCA DE ESTA GUÍA DE BASIC

Para poder controlar el ordenador es necesario que seamos capaces de comunicarle nuestras instrucciones en un lenguaje que él pueda entender. El **SVI•728**, como casi todos los ordenadores personales, entiende un lenguaje llamado BASIC (*Beginner's All Purpose Symbolic Instruction Code*, 'código de instrucciones simbólicas de uso general para principiantes'). Este lenguaje, residente en la ROM del **SVI•728**, es un conjunto de palabras del idioma inglés mediante el cual podemos explicarle al ordenador cómo debe llevar a cabo las tareas que le encomendamos.

Este manual se distingue en muchos aspectos de otras obras escritas para explicar BASIC. Una de las principales diferencias estriba en el hecho de que la información aquí presentada ya ha sido utilizada antes en la enseñanza de BASIC en las aulas.

En esta guía empezaremos por describir las órdenes de BASIC que permiten realizar dibujos sencillos y hacer que aparezcan en la pantalla. La razón por la que hemos decidido iniciar el contacto con BASIC es sencilla: aprender BASIC es como aprender un idioma extranjero; si podemos relacionar las palabras de BASIC con conceptos tan intuitivos como los geométricos, el lector podrá aprender muy deprisa y pasárselo bien al mismo tiempo.

En los próximos capítulos explicaremos someramente unas cuantas órdenes que nos permitirán dibujar en la pantalla. Pero la programación de gráficos es bastante más compleja y tiene más posibilidades que las que vamos a ver en esta primera introducción. De su descripción exhaustiva nos ocuparemos en los últimos capítulos del manual.

¿EN QUÉ CONSISTE LA PROGRAMACIÓN?

Programar es comunicar al ordenador las instrucciones y la información que necesita para realizar los trabajos que deseemos encargarle. Los programas difieren entre sí por el hecho de que la informa-

ción que necesita el ordenador y la manipulación a que ha de someterla no son las mismas cuando el objetivo es llevar las cuentas domésticas que cuando se trata de controlar un juego gráfico.

Por otra parte, un programa escrito en un lenguaje determinado tiene una estructura y unas instrucciones concretas que lo hacen incompatible con cualquier otro lenguaje. Ésta es la razón por la que los programadores profesionales se especializan como mucho en un par de lenguajes. Muy pocos son los que siquiera conocen (y no hablemos de dominar) todos los lenguajes de programación que han sido desarrollados en los últimos veinte años.

Hay dos formas principales de dar instrucciones de BASIC al ordenador: el 'modo de programa' y el 'modo inmediato' (o 'modo directo').

Como su nombre indica, el modo de programa es aquél en el que el ordenador funciona controlado por un programa.

Un programa de BASIC es un conjunto de instrucciones escritas en una sucesión de líneas numeradas. Por ejemplo,

```
10 PRINT "ESTOY"
20 PRINT "EMPEZANDO A"
30 PRINT "APRENDER BASIC"
```

Es costumbre espaciar los números de línea de 10 en 10 para que el programa resulte más legible y se pueda insertar líneas cuando sea necesario.

El ordenador empieza a obedecer las instrucciones del programa cuando se le da la orden **RUN**[ENTER].

La otra forma de dar instrucciones al ordenador es, como decíamos, el modo inmediato. Trabajando en este modo, cada vez que terminamos de escribir una instrucción y pulsamos [ENTER], el ordenador la obedece inmediatamente.

Pero ¿cómo sabe el ordenador que lo que hemos escrito es una orden que queremos que obedezca in-

mediatamente, y no una línea de programa? Las instrucciones para el modo inmediato **nunca** empiezan con un número.

UNA REFLEXIÓN ANTES DE EMPEZAR

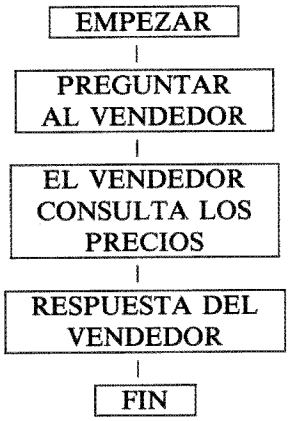
A lo largo de esta guía, casi todo nuestro trabajo lo desarrollaremos en modo de programa. Si usted sigue interesado en la informática, hará lo mismo en el futuro. El modo inmediato se reserva casi exclusivamente para operaciones de 'mantenimiento'; por ejemplo, para grabar programas en cinta o disco, pedir al ordenador la lista de los ficheros grabados y cargar programas y datos leyéndolos en la cinta o el disco.

Sí, nos damos cuenta de que estas palabras y conceptos pueden ser nuevos para usted; pero, como puede suponer, se los iremos explicando en los próximos capítulos.

Vamos a exponerle un símil que le ayudará a comprender qué hace el **SVI•728** con la información y los programas que le suministramos:

Supongamos que usted está pensando en comprar un Rover 2600 con pintura metalizada, techo corredizo, llantas de aleación y tapicería de cuero, y que quiere saber cuánto le va a costar la broma. Lo normal es que vaya a una tienda y le diga al vendedor qué modelo le interesa y con qué opciones. El vendedor consultará su lista, anotará en una hoja de papel los diversos precios, los sumará y le dirá cuál es el precio total.

El proceso de obtener información sobre un coche sería, pues, el siguiente:



Análogamente, cuando trabajamos con el ordenador, éste hace las veces de vendedor. A ambos tenemos que decirle exactamente qué queremos para obtener una respuesta correcta. Pero, antes de poder darla, los dos tienen que realizar unos cálculos.

Los datos que suministramos al vendedor y al ordenador son la información de 'entrada' (INPUT) con la que tienen que trabajar.

Los cálculos que hace el vendedor en una hoja de papel son análogos a los que el ordenador realiza en la memoria RAM.

La respuesta que obtenemos es la 'salida' (OUTPUT).

Leyendo esta guía y practicando este lenguaje, usted entrará en un campo que está en crecimiento continuo. BASIC es solamente el principio.

Capítulo 5

El primer programa

INTRODUCCIÓN

La mejor forma de entender cómo se forman las imágenes en la pantalla del ordenador es observar su analogía con el juego de los barcos.

El objetivo de este juego es adivinar en qué posiciones se encuentran los barcos del contrario. El tablero del juego de los barcos es de la siguiente forma:

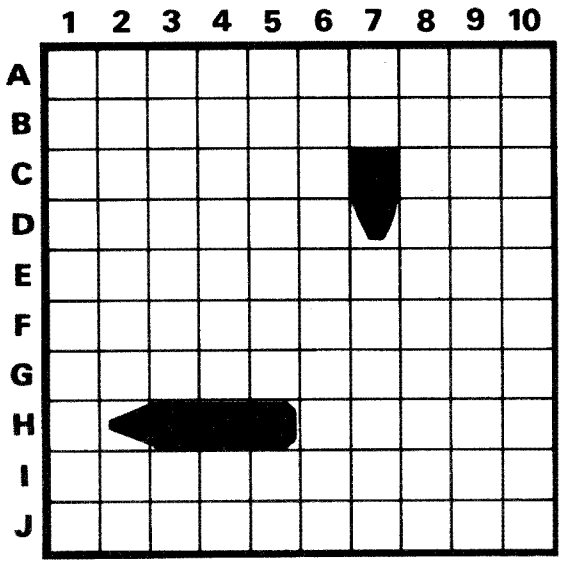


Figura 1. Tablero del juego de barcos

Esta figura muestra la colocación que el jugador contrario ha dado a sus barcos. Uno ocupa las posiciones H2, H3, H4 y H5; el otro está en C7 y D7. Usted debe disparar «apuntando» a una posición de la cuadrícula. Por ejemplo, si apuntamos a H2, el contrario dirá que hemos «tocado» uno de sus barcos; pero, si apuntamos a H1, dirá que hemos dado en el agua.

La relación que hay entre este juego y la pantalla del ordenador está en la forma de dar nombres a las diferentes posiciones del tablero.

RETÍCULA Y COORDENADAS

Para dibujar en el **SVI•728**, también tenemos que dar un nombre distinto a cada una de las celdas que componen la pantalla:

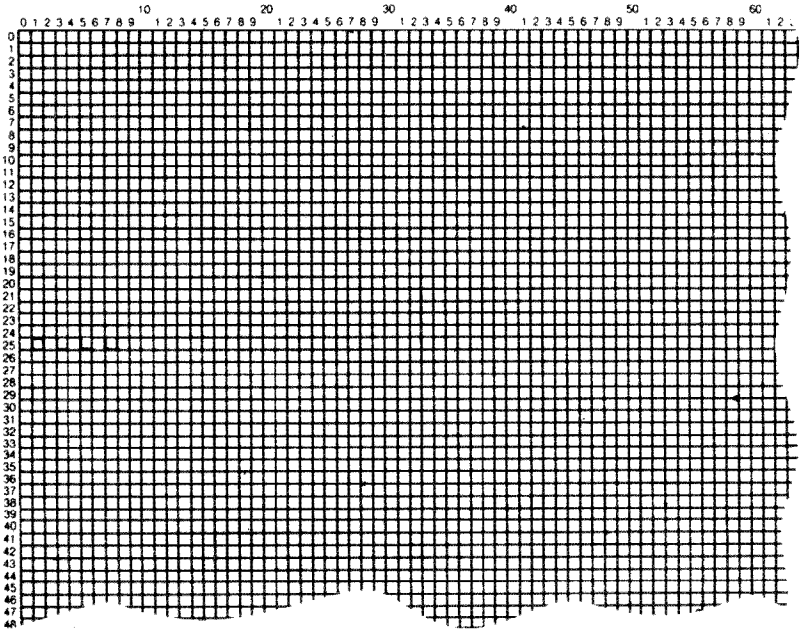


Figura 2

Cada celda tiene un nombre que está formado por dos números: el número que encabeza la columna y el que encabeza la fila. El primero es la *coordenada horizontal*; el segundo es la *coordenada vertical*.

Si sabemos qué nombres tienen las diferentes celdas, empezamos a estar en condiciones de dibujar figuras sencillas. Estudie, pues, el siguiente ejemplo:

En la figura 3 hemos rellenado unas cuantas celdas (lo que para el ordenador equivale a iluminarlas). ¿Sabría usted decir cuáles son sus nombres? Escríbalos en un trozo de papel.

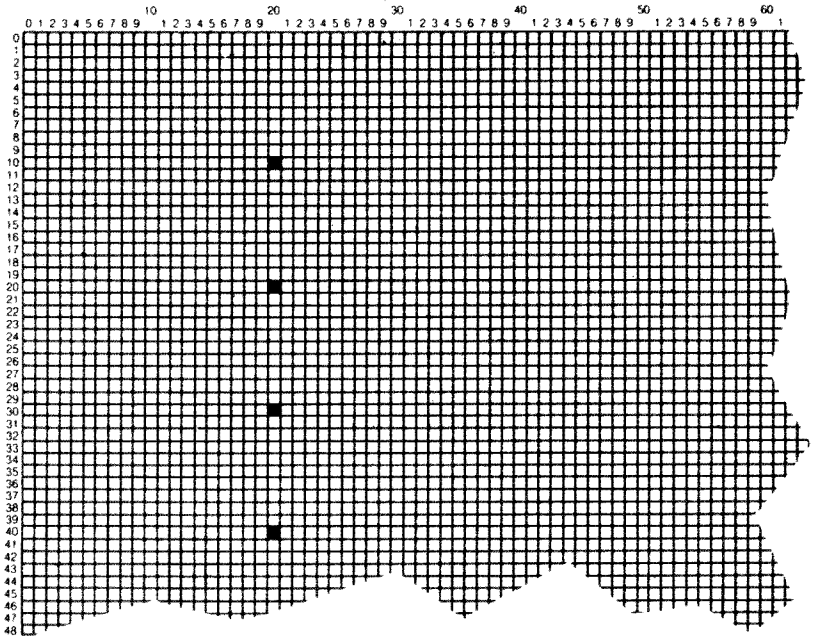


Figura 3

Sus respuestas deberían haber sido las siguientes:

(20,10)
(20,20)
(20,30)
(20,40)

Pues bien, ahora vamos a usar estos nombres en un programa que iluminará las celdas marcadas en la figura 3. Transcriba cuidadosamente las líneas siguientes. No olvide pulsar ENTER al final de cada una. (De ahora en adelante no siempre le recordaremos este detalle; usted debe ir habituándose a pulsar ENTER al final de cada orden directa y de cada línea de programa.)

```
10 SCREEN 3
20 PSET (20,10)
30 PSET (20,20)
40 PSET (20,30)
50 PSET (20,40)
60 END
```

Después de comprobar que ha escrito correctamente todas las líneas, dé al **SVI•728** la orden **RUN**.

¿Qué ha ocurrido? La imagen ha pasado como un relámpago. Bueno, haga lo siguiente:

LIST

1. Escriba:

LIST[ENTER]

Como la propia palabra indica, **LIST** da un listado de las líneas del programa, desde la primera hasta la última. Si hubiéramos escrito las instrucciones del programa en modo inmediato, es decir, sin números de línea, ahora **LIST** no habría producido ningún efecto, pues no habría ningún programa que listar.

2. Añada al programa la siguiente línea:

55 GOTO 55[ENTER]

3. Escriba **LIST**[ENTER].

Observe que el **SVI•728** ha insertado la línea nueva en su sitio, es decir, entre la 50 y la 60. Aquí podemos apreciar por qué decíamos antes que a las líneas no se les debe dar números correlativos; la necesidad de intercalar líneas nuevas es algo habitual cuando se está desarrollando un programa.

4. Escriba **RUN**[ENTER].

El ordenador no sólo dibuja la imagen que queríamos, sino que la deja en pantalla y no parece darse por enterado cuando escribimos algo en el teclado. Compruébelo: los caracteres que usted pulsa no aparecen en la pantalla. ¿Por qué? El ordenador todavía está ejecutando el programa, y seguirá haciéndolo hasta que pulsemos [CTRL][STOP]. Pero examinemos otra vez el programa:

```
10 SCREEN 3
20 PSET (20,10)
30 PSET (20,20)
40 PSET (20,30)
50 PSET (20,40)
55 GOTO 55
60 END
```

Antes de que añadiéramos la línea 55, la imagen aparecía momentáneamente y enseguida se borraba; el programa hacía exactamente lo que le habíamos pedido (muy deprisa) y terminaba. Al intercalar la línea 55, le estamos diciendo al ordenador que cuando llegue a ella se quede en ese punto del programa indefinidamente. Para detener el programa tenemos que pulsar CTRL y STOP simultáneamente.

Esta combinación de teclas interrumpe el *bucle indefinido* en el que el ordenador había entrado. De hecho, interrumpe el programa en la línea que esté siendo ejecutada en ese momento.

Una vez interrumpido el programa, ya podemos ver en la pantalla las letras que escribamos en el teclado. Analicemos el programa:

```
10 SCREEN 3
20 PSET (20,10)
30 PSET (20,20)
40 PSET (20,30)
50 PSET (20,40)
55 GOTO 55
60 END
```

La línea 10 informa al ordenador de que nuestra intención es dibujar una imagen en la 'pantalla 3' (**SCREEN 3**).

SCREEN

En el **SVI•728** disponemos de cuatro pantallas:

SCREEN 0 es la pantalla normal de texto. En ella caben 23 líneas de texto, de 37 caracteres cada una como máximo; si intentamos sobrepasar ese número de líneas, todas las anteriores se desplazan hacia arriba para dejar sitio a una nueva. (Este efecto se denomina también «scrolling».)

SCREEN 1 es la pantalla que el ordenador muestra cuando acabamos de encenderlo. Admite 23 líneas de texto de 29 caracteres cada una.

Estas dos pantallas permiten comunicarnos con el ordenador, es decir, darle instrucciones en modo inmediato e introducir programas.

SCREEN 2 es la pantalla gráfica de alta resolución.

SCREEN 3 es la pantalla gráfica de baja resolución.

(Los términos 'alta resolución' y 'baja resolución' están explicados en el Apéndice F.)

Siempre que se interrumpe un programa pulsando CTRL STOP, el ordenador vuelve a mostrar la pantalla 0 o la 1 (la última utilizada) y queda a la espera de nuevas instrucciones, lo que indica mediante el mensaje **Ok** y el cursor. Entonces se puede, entre otras cosas, listar y modificar el programa.

PSET

La línea 20 contiene una instrucción **PSET**, cuyo efecto es hacer que el ordenador ilumine la celda cuyo nombre es (20,10). Recuerde que el primero de estos números hace referencia a la columna en que se encuentra la celda, mientras que el segundo especifica la fila.

Las líneas 30 a 50 siguen iluminando celdas marcadas en la figura 3. En cuanto a la línea 55, ya hemos dicho para qué sirve, pero la explicaremos detenidamente más adelante.

Escriba **RUN** ENTER y podrá ver la figura. Cuando se canse de observarla, pulse CTRL STOP para detener el programa.

Éste ha sido su primer programa. En general, dar varias instrucciones juntas al ordenador es más eficaz que dárselas una a una en modo inmediato. (Recuerde que en modo inmediato el ordenador obedece inmediatamente cada instrucción en cuanto se pulsa ENTER.) Si no hubiésemos puesto los números de línea para indicar que lo que estábamos escribiendo era un programa, las celdas se habrían encendido una a una: algo equivalente a ir doce veces al mercado para comprar una docena de huevos.

Bien, ha llegado el momento de que usted empiece a poner en práctica todo lo que ha aprendido.

Escriba **NEW** ENTER

La instrucción **NEW** hace que el ordenador «olvide» el programa que tenía en la memoría.

Intente escribir un programa que dibuje la imagen de la figura 4.

Figura 4

UN TOQUE DE COLOR

Cada una de las celdas que iluminemos puede estar en cualquiera de los 16 colores disponibles. La lista de colores y sus números de identificación es la siguiente:

Color n.°	Color
0	Transparente
1	Negro
2	Verde
3	Verde claro
4	Azul oscuro
5	Azul claro
6	Rojo oscuro
7	Cyan
8	Rojo
9	Rojo claro
10	Amarillo oscuro
11	Amarillo claro
12	Verde oscuro
13	Magenta
14	Gris
15	Blanco

Antes de explicar la forma de añadir colores a los dibujos, vamos a hacer desmostración de las posibilidades cromáticas del **SVI•217**.

Pulse [SHIFT][CLS/HM] para borrar la pantalla y luego escriba:

COLOR 4,15[ENTER]

(Observe que la tecla de función [F1] está programada para escribir la palabra 'COLOR'.)

Esta orden ha invertido los colores de texto y fondo.

Los números 4 y 15 son dos de los 16 incluidos en la lista anterior.

Experimentemos un poco con la orden **COLOR**. Escriba:

COLOR 4,4[ENTER]

El texto ha desaprecido; todo lo que podemos ver es una pantalla azul. Si ahora escribe cualquier cosa, por ejemplo su nombre, tampoco podrá verlo en la pantalla. Veamos por qué:

El primer número que hemos puesto en la orden tras la palabra 'COLOR' es el número de color que queremos especificar para el texto; el segundo es el número de color que hemos elegido para el fondo. Como para ambos hemos especificado el color número 4 (azul oscuro), el texto se confunde con el fondo. Para que el texto vuelva a ser visible tendremos que dar otra orden **COLOR** distinta, pero esto no es tan fácil, ya que ni siquiera sabemos dónde está el cursor.

[F6]

Aquí es donde podemos apreciar la utilidad de la tecla de función [F6].

Antes de nada, cerciórese de que el cursor está al principio de una línea, para lo cual basta con pulsar [ENTER].

De momento no se ha producido ningún cambio visible, sólamente se ha oído un pitido. Como lo último que usted había hecho era escribir su nombre, al pulsar [ENTER] ha provocado un error de sintaxis (**Syntax error**) y el consiguiente pitido. Esto no habría ocurrido, por supuesto, si en lugar de su nombre hubiese escrito una orden correcta de BASIC.

Ahora pulse [F6] (o sea, [SHIFT][F1]). La pantalla vuelve a la situación normal, y usted puede ver nuevamente todo el texto anterior, así como los mensajes **Syntax error** y **Ok**.

Fíjese en la última orden que hay en la pantalla:

COLOR 15,4,4

Ésta es la orden generda por la tecla [F6]. (Observe además que después de pulsar [F6] no fue necesario pulsar también [ENTER], ya que el código equivalente

a la pulsación de [ENTER] está incluido en la propia [F6].) En esta orden interviene un tercer número del que todavía no hemos hablado. Ya sabemos que el primero (15) especifica el color del texto y que el segundo (4) especifica el color del fondo. Pues bien, el tercero hace referencia al color del borde de la pantalla. El borde es del mismo color que el fondo en la pantalla de texto número 0; ésta es la razón por la que el color del borde se suele especificar para la pantalla de texto número 1 y para las pantallas gráficas (números 2 y 3).

BORDE, FONDO Y PRIMER PLANO

Podemos considerar que la imagen que vemos en la pantalla está compuesta por tres capas, una encima de otra. Debajo del todo está el *borde*. Por encima de él está el *fondo*. En las pantallas de texto 0 el borde queda totalmente cubierto por el fondo; en cambio, en la pantalla de texto 1 y en las pantallas gráficas (2 y 3) el fondo se «encoge» y permite ver una franja del borde por la parte superior de la pantalla y otra por el extremo inferior.

Superpuesto al borde y al fondo está el *primer plano*; es la imagen propiamente dicha: texto en las pantallas 0 y 1 y dibujos en las pantallas 2 y 3.

Después de esta explicación, comprenderá por qué decimos que el formato de la orden **COLOR** es el siguiente:

COLOR ⟨*n.º color primer plano*⟩,⟨*n.º color fondo*⟩,⟨*n.º color borde*⟩

Le sugerimos que experimente con varios números para familiarizarse con los colores del **SVI•728**.

Volviendo a la orden **PSET**, escriba:

PSET(10,20),3[ENTER]

y cambie el color:

PSET(10,20),8[ENTER]

La orden **PSET** no es más que el prólogo de los inmensos recursos gráficos del **SVI•728**. Pero tenemos que aplazar la descripción de las restantes instrucciones gráficas para cuando hayamos terminado con esta introducción al lenguaje BASIC.

GOTO

Antes de dar por concluido este capítulo, pruebe el siguiente programa, que le servirá para entender fácilmente el funcionamiento de la orden **GOTO** ('ir a').

Antes de nada, escriba **NEW**[ENTER] y luego

```
10 PRINT "Me encanta mi SVI-728"
20 GOTO 10
```

Dé la orden **RUN**[ENTER] para poner en marcha el programa y pulse [CTRL][STOP] cuando quiera detenerlo. ¿Se ha dado cuenta de por qué el ordenador escribe el mensaje una vez y otra? Por si acaso, vamos a explicarlo:

La línea 10 pide al **SVI•728** que escriba el mensaje que figura entre las comillas. Cuando el ordenador se encuentra con la línea 20, la orden **GOTO 10** le obliga a volver a la línea 10, donde vuelve a escribir el mensaje. Esto se repite incesantemente, hasta que detengamos el programa.

Pruebe con otros mensajes en la línea 10. La forma más rápida de modificarla es la siguiente:

Dé la orden **LIST**[ENTER] para tener el programa a la vista.

Vuelva a escribir

```
10 PRINT "
```

y añada el mensaje de su elección; cierre las comillas y pulse [ENTER] para almacenar en la memoria la versión corregida de la línea.

Escriba **RUN**[ENTER]. El nuevo mensaje aparecerá reiteradamente en la pantalla.

La que hemos visto aquí es la forma más elemental de utilizar la instrucción **PRINT**. Más adelante la explicaremos con todo detalle.

Le sugerimos que consulte el Apéndice F, donde están descritas las características de las cuatro pantallas, antes de empezar a leer el capítulo siguiente.

Capítulo 6

Bucles

Después de ver los ejemplos de **PSET** del capítulo anterior, el lector puede haber pensado que tiene que haber una forma más rápida de dibujar en la pantalla que encender las celdas una a una. Efectivamente, la hay. Pero antes de explicársela, pruebe este programa:

En MSX BASIC hay un recurso que va a resultarle muy útil y a ahorrarle mucho trabajo: en lugar de la palabra clave **PRINT**, basta con escribir el signo de interrogación **?**, que es su abreviatura.

```
10 CLS
20 PRINT "ESCRIBA UN NUMERO:"
30 INPUT Y
40 SCREEN 3
50 PSET(30,Y)
60 PSET(45,Y)
70 PSET(60,Y)
80 PSET(75,Y)
90 GOTO 90
```

Ahora escriba **RUN** ENTER.

El listado del programa debe haber desaparecido inmediatamente; en su lugar podrá ver el mensaje 'ESCRIBA UN NUMERO:'. La pantalla se borró gracias a la orden **CLS** (línea 10), que es abreviatura de *CLear Screen* ('borrar pantalla').

El mensaje lo escribe la línea 20 mediante la orden **PRINT**. Al principio de la línea siguiente ha aparecido un signo de interrogación.

INPUT

El signo de interrogación se debe a la orden **INPUT** de la línea 30. **INPUT** hace que el ordenador, en lugar de pasar a la línea siguiente, espere hasta que escribamos algo en el teclado y pulsemos ENTER. Lo

RECIPIENTES Y VARIABLES

Las variables que vamos a ver en este capítulo son de tipo numérico. Son las que pueden contener solamente valores numéricos. En el capítulo 8 estudiaremos las variables no numéricas (llamadas 'alfanuméricas' o 'literales').

que escribamos quedará almacenado en un «recipiente» en la memoria RAM del ordenador.
Un término más técnico y correcto para esos recipientes es el de *variable*. Para ayudar a los usuarios que no conozcan este concepto vamos a dar una explicación sencilla.

Pero antes de continuar con la descripción de los recipientes vamos a ver uno de ellos en funcionamiento. En respuesta a la interrogación del programa, escriba un número comprendido entre el 0 y el 191 y pulse ENTER. (Si el número es mayor que 191, el programa no producirá ningún efecto visible.)

Si efectivamente el número es igual o menor que 191, el programa dibuja cuatro puntos en la misma horizontal. La altura depende del número introducido.

El número que usted ha escrito ha quedado almacenado en un recipiente etiquetado con el nombre 'Y', según impone la línea 30. Imagine que ese recipiente es una copa y que el número está escrito en un trozo de papel que ponemos en la copa. Cuando el ordenador llega a la línea 50 tiene que iluminar la celda (30,Y); para averiguar qué número debe usar en lugar de la **Y**, mira en la copa etiquetada con ese nombre y encuentra el número introducido por usted. Y esto mismo hace siempre que encuentra la **Y** en el programa. Así pues, si el número era, por ejemplo, el 27, el ordenador ilumina la celda (30,27) en la línea 50, la celda (45,27) en la línea 60, la celda (60,27) en la línea 70 y finalmente la celda (75,27) en la línea 80.

RECIPIENTES EN LA MEMORIA DEL ORDENADOR

Sería muy difícil almacenar y recuperar la información si no dispusiéramos de una forma directa de hacer referencia a ella. Hemos elegido el término 'recipiente' para designar las posiciones de la memoria en las que se almacena la información. La letra 'Y' es sólo uno de los muchos símbolos que podemos usar para «etiquetar» los recipientes.

ETIQUETAS PARA LOS RECIPIENTES

El **SVI•728**, al igual que la mayor parte de los ordenadores personales, puede manejar dos tipos de recipientes: uno que sólo vale para números y otro que vale para números y texto. El ordenador los distingue por el hecho de que los nombres de los recipientes que sirven para números y texto terminan siempre con el signo **$**. Todos los nombres deben empezar con una letra (cualquiera menos la 'ñ' y la 'Ñ') y pueden constar de uno o dos caracteres (aparte del posible '$'). No pueden ser palabras reservadas de BASIC. Por consiguiente, SV, DL, L1, A$, AA$, etc., son nombres correctos.

Transcriba el siguiente programa:

```
10 CLS
20 PRINT "REPETIRE LO QUE USTED
   ESCRIBA EN RESPUESTA AL SIGNO
   DE INTERROGACION"
30 INPUT A$
40 PRINT A$
50 END
```

Escriba **RUN**[ENTER]. Responda al signo de interrogación con un número y observe cómo el ordenador produce el «eco». (No olvide pulsar [ENTER] después de escribir el número.)

Dé otra vez la orden **RUN**[ENTER]. Ahora responda con una palabra, en lugar de con un número.

¿Qué ocurre si mezclamos letras y números? ¿Lo acepta el ordenador? Haga la prueba.

En cada recipiente de este tipo se puede almacenar hasta 255 caracteres (letras u otros signos) como máximo. (La máxima longitud de las líneas de programa de BASIC es también 255.) Compruebe qué ocurre cuando se escribe más de 255 caracteres. Ejecute el programa y responda al signo de interrogación con una cadena de más de 255 caracteres. Observará que el ordenador ignora los caracteres excedentes.

ALMACENAMIENTO Y RECUPERACIÓN DE LA INFORMACIÓN

Hay varias instrucciones que indican al ordenador qué información debe poner en un recipiente determinado. Ya hemos visto una de ellas: **INPUT**. He aquí un ejemplo de otra; transcriba el siguiente programa:

```
10 CLS
20 LET A=10
30 LET B=20
40 LET C=A+B
50 PRINT C
60 END
```

Escriba **RUN**[ENTER].

En la pantalla debe haber aparecido el número 30. Este programa ilustra otra forma de depositar información en los recipientes. La orden **LET** se escribe delante del nombre del recipiente; a su derecha se escribe el signo '='. La información que se quiere depositar en el recipiente se pone a la derecha del signo '='. El programa deposita el número 10 en el recipiente **A** y el número 20 en **B**. Después suma los dos números y pone el resultado en el recipiente **C**. Cuando llega a la línea 50, busca en este último recipiente el dato que debe escribir.

LET

Curiosamente, **no** es necesario escribir la palabra **LET**. Por eso, el siguiente programa funciona exactamente igual que el anterior:

```
10 CLS
20 A=10
30 B=20
40 C=A+B
50 PRINT C
60 END
```

LET sólo sirve para hacer más legible el programa. Esto tiene interés cuando otras personas van a tener

que leerlo, o cuando uno mismo va a leerlo unos meses después de haberlo escrito.

Vamos a explicar una tercera forma de almacenar información en los recipientes, gracias a la cual seremos capaces de dibujar deprisa.

Y no sólo nos permitirá dibujar deprisa, sino que será una herramienta imprescindible en casi todos los programas. Copie y ejecute el siguiente:

```
10 CLS
20 FOR X=0 TO 64
30 PRINT X
40 NEXT X
50 END
```

¿Sorprendido por el resultado? La utilización del contenedor no es nada nuevo. En cambio, el bucle es distinto del que conocimos en el capítulo anterior.

FOR/NEXT

Una vez borrada la pantalla, la línea 20 indica al ordenador que debe poner en el recipiente **X** todos los números del 0 al 64, pero uno a uno. El paciente ordenador empieza por depositar el 0 en **X**. La instrucción **PRINT X** obliga al ordenador a averiguar qué dato tiene almacenado en **X** y a mostrarlo en la pantalla. La línea 40 le dice que debe volver a la instrucción **FOR** y actuar con el número siguiente (*next* es una palabra inglesa que significa 'siguiente').
Por lo tanto, nos encontramos otra vez en la línea 20, pero ahora el número que se va a depositar en **X** es el 1.

Llegados a la línea 30, el ordenador lee el valor que tiene guardado en **X**, encuentra el 1 y lo escribe. La línea 40 vuelve a remitirlo a la 20, para ahora actuar con el número 2. Este *bucle* se repite hasta que finalmente el número depositado en **X** por la línea 20 y escrito por la línea 30 es el 64. Como el 65 ya no pertenece al conjunto especificado en la instrucción **FOR**, el programa sale del bucle y termina al llegar a la orden **END** de la línea 50.

Las palabras clave **FOR** y **NEXT** van siempre empa-

rejadas. No puede haber **FOR** sin **NEXT**, y vice versa.

Los bucles de tipo **FOR/NEXT** son esencialmente diferentes de los formados con **GOTO**. Con el tiempo, usted irá aprendiendo a decidir cuál de los dos es más adecuado a cada tarea.

¿Se atrevería a modificar el programa anterior para escribir uno que dibuje una línea horizontal? Inténtelo antes de leer la solución.

```
10 SCREEN 3
20 FOR X=0 TO 255
30 PSET (X,25)
40 NEXT X
50 END
```

Ejecute el programa y observe con qué facilidad dibuja el ordenador la recta.

Por la instrucción **FOR** de la línea 20, el ordenador sabe que debe poner en el recipiente **X** los números del 0 al 255, sucesivamente. El bucle comienza con el valor 0. Al llegar a la línea 30, el ordenador examina el recipiente y en consecuencia ilumina la celda (0,25).

La instrucción **NEXT X** de la línea 40 reenvía el programa a la línea 20 para dar otra pasada por el bucle, ahora ya con el valor 1 en **X**. La celda que se ilumina es (1,25). El proceso se repite hasta que se sobrepasa el valor 255. Para entonces ya están encendidas todas las celdas de la fila 25.

Como ejercicio, modifique este programa para que dibuje, en lugar de una recta horizontal en la fila 25, una vertical en la columna 30.

```
10 SCREEN 3
20 FOR D=0 TO 191
30 PSET (30,D)
40 NEXT D
50 END
```

En este programa hemos elegido otro nombre, **D**, para el recipiente. El margen de los valores que almacenamos en él es en este caso 0–191. El funcionamiento es análogo al del programa anterior.

Hemos dedicado bastante tiempo a explicarle los recipientes (variables) en forma intuitiva, no técnica. Esto nos parecía importante porque, cuanto mejor conozca usted la relación causa–efecto de todas las instrucciones de BASIC, mejor aprovechará su **SVI•728**.

ALGUNAS SUGERENCIAS

Hemos visto la forma de utilizar los bucles para ahorrar tiempo y trabajo. Hay otros trucos que pueden ayudarle en el mismo sentido. Por ejemplo, no siempre es necesario incluir en los programas la orden **END**. No se pierde nada por ponerla, pero siempre es fácil distinguir cuándo es necesaria y cuándo no lo es. Observe el siguiente programa:

```
10 CLS
20 PRINT "Me encanta mi SVI-728"
30 GOTO 20
```

END

Podríamos haber incluido la línea

```
40 END
```

Pero ¿para qué tomarnos esa molestia? El programa no podría de ninguna manera llegar a ella. Es decir, el programa no termina a causa de ninguna instrucción, sino cuando pulsamos CTRL STOP.

Análogamente, no siempre es necesario borrar la pantalla (**CLS**). Usted sabrá en qué programas le interesa empezar con una pantalla en blanco.

Observe que la orden **SCREEN** automáticamente borra la pantalla a la que afecta.

Capítulo 7

Objetos en movimiento

Vamos a añadir una línea al programa del capítulo anterior que dibujaba una recta horizontal. Con esta modificación conseguiremos desplazar un punto por la pantalla. Transcriba y ejecute este programa.

```
10 SCREEN 3
20 FOR X=0 TO 255
30 PSET(X,25),11
40 PRESET(X,25)
50 NEXT X
```

¿Demasiado rápido? Antes de explicar la forma de retardar el movimiento, estudiemos el programa para ver qué es lo que da precisamente esa ilusión de movimiento.

PRESET

La nueva instrucción **PRESET** de la línea 40 actúa como **PSET**, pero al revés: **PSET** enciende celdas, **PRESET** las apaga. ('PSET' es abreviatura de *point set*, 'activar punto'; 'PRESET' viene de *point reset*, 'desactivar punto'.)

La línea 30 ilumina un punto (con el color especificado, el 11). La línea 40 apaga ese mismo punto; aquí no hay que especificar ningún color, porque «apagar» un punto es, sencillamente, darle el color del fondo.

¿Cómo podemos hacer más lento el movimiento? Se nos ocurre que perdiendo tiempo entre las líneas 30 y 40 pues, cuanto más tardemos en apagar los puntos, más tiempo serán visibles y más fácil será observar la trayectoria. Por lo tanto, introduzcamos un *retardo* entre las líneas 30 y 40:

```
35 FOR T=0 TO 50
37 NEXT T
```

Escriba **LIST** ENTER:

```
10 SCREEN 3
20 FOR X=0 TO 255
30 PSET(X,25),11
35 FOR T=0 TO 50
37 NEXT T
40 PRESET(X,25)
50 NEXT X
```

Y ahora ejecute el nuevo programa.

El movimiento es mucho más lento y cómodo de observar. La estructura de las líneas 35 y 37 ya nos es familiar; se trata de un bucle de tipo **FOR/NEXT**, pero con una particularidad:

En los ejemplos del capítulo anterior siempre intercalábamos alguna orden (**PSET** o **PRINT**) entre el **FOR** y el **NEXT** de los bucles. En este caso, en cambio, el ordenador no hace nada en las sucesivas pasadas por el bucle; eso sí, tarda algún tiempo en recorrer las líneas 35 y 37 y en llevar la cuenta del valor de **T** (para lo que tiene que guardar y leer valores repetidamente en el contenedor **T**). Esta pérdida de tiempo es justamente lo que buscábamos; el bucle se llama *bucle de retardo*.

Si le parece que ahora el movimiento es demasiado lento, puede acelerarlo muy fácilmente: cambie el límite superior del bucle de retardo por un valor más bajo:

```
35 FOR T=0 TO 25
37 NEXT T
```

Para hacer el movimiento aun más lento, ponga en la línea 35 un límite superior más alto:

```
35 FOR T=0 TO 100
37 NEXT T
```

MOVIMIENTO HACIA ATRÁS

Ya sabemos desplazar el balón de izquierda a derecha a cualquier velocidad. ¿Cómo realizar el movimiento contrario, de derecha a izquierda? Añada las siguientes líneas al programa:

```
60 FOR Q=255 TO 0 STEP -1
70 PSET (Q,25),11
80 FOR F=0 TO 10
90 NEXT F
100 PRESET(Q,25)
110 NEXT Q
```

STEP

La novedad está en la línea 60. Se preguntará usted qué es eso de **'STEP −1'**. Vamos allá:

La instrucción

```
FOR X=0 TO 255
```

hace que el ordenador vaya colocando en el recipiente **X** los números del 0 al 255 **de uno en uno**. Este detalle de la instrucción, 'de uno en uno', no es necesario escribirlo explícitamente. En efecto,

```
FOR Q=0 TO 255
```

es equivalente a

```
FOR Q=0 TO 255 STEP 1
```

Cuando se ejecuta el bucle, los números son recorridos de la siguiente forma: 0, 1, 2, . . . , 255. Si no se especifica **STEP**, el ordenador supone que lo que queremos es avanzar de uno en uno.

Para recorrer los números en la forma 0, 2, 4, . . . , 254 tendríamos que haber puesto

```
FOR Q=0 TO 255 STEP 2
```

Fácilmente se deduce que

```
FOR Q=255 TO 0 STEP -1
```

especifica el orden 255, 254, 253, . . . , 2, 1, 0. Ésta es la razón por la que el balón se mueve de derecha a izquierda.

Con sólo borrar una línea obtenemos un resultado drásticamente diferente. Borre la línea 100 escribiendo

100[ENTER]

y dé la orden **LIST**[ENTER] para comprobar que ha quedado borrada.

Ejecute el programa.

Como ya no está la instrucción **PRESET** de la línea 100, los puntos iluminados por la línea 70 no se apagan.

Modifique el programa para que forme un bucle continuo y no termine nunca. Pruébelo antes de seguir leyendo.

Debería haber añadido la siguiente línea:

```
120 GOTO 20
```

UN PROGRAMA PARA PONERLE A PRUEBA

Le proponemos que ponga en práctica todos sus conocimientos elaborando un programa que haga descender un balón por una escalera. El programa no es fácil; puede considerarlo como un reto a su habilidad como programador. Puede empezar por dibujar la escalera y luego concentrarse en el descenso

del balón. El resultado final debería tener el siguiente aspecto:

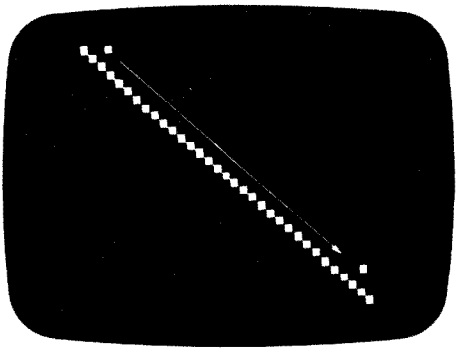


Figura 5

Una solución puede ser ésta:

```
  5 REM Esta sección del programa di-
    buja una escalera
 10 SCREEN 3
 20 FOR X=0 TO 191
 30 PSET(X,Y),6
 40 Y=Y+1
 45 NEXT
 50 REM Esta sección del programa di-
    buja el balón bajando por la escalera
 60 Y=0
 70 FOR X=4 TO 191
 80 PSET(X,Y),15
100 FOR T=0 TO 100
105 NEXT T
110 PRESET(X,Y)
120 Y=Y+1
125 NEXT X
135 CLS
140 Y=0
150 GOTO 20
```

REM

Este programa hace justamente lo que pretendíamos, aunque la solución desarrollada por el lector puede ser distinta. La única instrucción nueva que hemos utilizado es **REM**. Esta palabra es abrevia-

tura de *remark*, 'observación'; el ordenador entiende que todo lo que se escriba a continuación de ella, hasta el final de la línea, debe ser ignorado. Las sentencias encabezadas por **REM** suelen contener comentarios sobre la finalidad de las diversas partes del programa; son totalmente indiferentes para el ordenador.

En las líneas 40 y 120 hay una sentencia que aparece absurda desde el punto de vista del álgebra. En realidad, es fácil entenderla cuando se sabe que su significado es: 'haz que el nuevo valor de **Y** sea igual a su valor antiguo más 1'.

Aunque no haya sido capaz de completar con éxito el programa de la escalera, no se desanime. Como dijimos al principio, esta guía es muy distinta de los demás métodos de enseñanza de BASIC; aparte de explicarle a fondo y con palabras sencillas las órdenes de BASIC, de vez en cuando le plantearemos ejercicios bastante difíciles.

Capítulo 8

Decisiones

En los capítulos anteriores hemos tomado un primer contacto con BASIC a través de los gráficos. A partir de ahora vamos a explorar muchas otras instrucciones de BASIC en contextos diferentes.

IF/THEN

Lo que hace que los ordenadores parezcan inteligentes es su capacidad de tomar decisiones. Esa capacidad se basa en la sentencia **IF/THEN** ('si ... entonces ... '). Veamos cómo funciona; copie el siguiente programa:

```
10 CLS
20 PRINT "TRATE DE ADIVINAR EL
   NUMERO EN EL QUE ESTOY PEN-
   SANDO. COMO PISTA, LE DIRE
   QUE ESTA ENTRE 0 Y 100."
30 A=50
40 INPUT X
50 IF X=A THEN GOTO 100
60 PRINT "LO SIENTO, NO HA ACER-
   TADO. INTENTELO OTRA VEZ."
70 GOTO 40
100 PRINT "¡MUY BIEN! CORRECTO; EL
   NUMERO ERA EL 50."
```

Ejecute el programa. La orden **INPUT** de la línea 40 hace que aparezca un signo de interrogación en la pantalla. Escriba su respuesta. La línea 50 compara la respuesta, que está almacenada en el recipiente **X**, con el número que está almacenado en **A**. Si el número introducido es 50, el programa salta a la línea 100. Si no, el programa continúa en la línea 60, en la que le informa de lo que ha ocurrido; la lí-

nea 70 devuelve el control a la línea 40, en la que el ordenador queda a la espera de una nueva respuesta.

Cuando se acierta, la línea 50 desvía el programa hacia la línea 100, la cual emite el oportuno mensaje.

Analicemos detenidamente la instrucción **IF/THEN**. Al igual que **FOR/NEXT**, interviene en la mayor parte de los programas.

La sentencia **IF/THEN** realiza una comprobación y toma una decisión en función del resultado. En nuestro ejemplo significa: 'si $X=A$ es cierto, debes saltar a la línea 100; en cambio, si no es cierto, continúa normalmente'.

VARIABLES LITERALES

En el capítulo 6 dijimos que el **SVI•728** maneja dos tipos distintos de recipiente. El primero sólo puede contener números y se llama, por consiguiente, *variable numérica*. El otro puede contener tanto letras como números; se le llama *variable literal* o *variable alfanumérica*.

Para distinguir los dos tipos, al final del nombre de las variables literales se escribe el signo **$**. Ejemplos:

```
A$="VARIABLE"
B$="2 variables"
C$="1234"
```

A continuación vamos a proponerle que escriba un programa para el que necesitará las variables literales.

OTRO RETO A SU HABILIDAD

Plantearemos el problema describiéndolo en palabras del lenguaje corriente. Las tres 'etapas' siguientes describen las tres secciones del programa. Las dos primeras pueden ser traducidas a sendas instrucciones de BASIC. La etapa 3 necesita varias líneas para su realización. El programa debe hacer lo siguiente:

Etapa 1. El ordenador pregunta al usuario si quiere que escriba en la pantalla los números del 0 al 100.

Etapa 2. El ordenador espera hasta que el usuario responde, y entonces almacena la respuesta en un recipiente.

Etapa 3. El ordenador analiza la respuesta. Si es la palabra 'SI', escribe los números ofrecidos; si es 'NO', escribe el mensaje '¡ADIOS!'.

Trate de escribir este programa y pruébelo. Tenga en cuenta que muy pocas veces se consigue un programa totalmente correcto al primer intento. Parte de la belleza de la programación radica en que uno aprende de sus propios errores. La descripción que hemos hecho contiene algunas sugerencias. Léala otra vez cuidadosamente antes de empezar a escribir el programa.

Pero no vamos a dejarle desamparado. Por si no ha dado con una solución, he aquí la nuestra:

```
  5 CLS
 10 PRINT "¿QUIERE QUE ESCRIBA LOS
    NÚMEROS DEL 0 AL 100? RESPON-
    DA SI O NO."
 20 INPUT A$
 30 IF A$="NO" THEN GOTO 100
 40 FOR N=0 TO 100
 50 PRINT N
 60 NEXT N
 70 END
100 PRINT "¡ADIOS!"
```

Hemos tenido que incluir el **END** de la línea 70 para evitar que, en caso de que se responda 'SI', el programa también escriba el mensaje de la línea 100. O sea, la orden **END** forma una barrera gracias a la cual el programa sólo puede llegar a la línea 100 desde la 30.

¿Qué ocurre si se responde con 'sí', 'no' o cualquier otra cosa? Compruébelo usted mismo.

Si lo hace, observará que la única respuesta que el ordenador distingue de todas las demás es 'NO', porque, tal como está el programa, se limita a comprobar (en la línea 30) si lo que hay en el recipiente **A$** es exactamente igual a 'NO'. Para mejorar el programa, añada las siguientes líneas:

```
35 IF A$="SI" THEN GOTO 40
37 PRINT "RESPUESTA INCORRECTA.
   SOLO VALE SI O NO. RESPONDA
   OTRA VEZ."
38 GOTO 20
```

El listado completo es el siguiente:

```
 5 CLS
 10 PRINT "¿QUIERE QUE ESCRIBA LOS
NÚMEROS DEL 0 AL 100? RESPONDA SI
O NO."
 20 INPUT A$
 30 IF A$="NO" THEN GOTO 100
 35 IF A$="SI" THEN GOTO 40
 37 PRINT "RESPUESTA INCORRECTA.
    SOLO VIAJE SI O NO. RESPONDA
    OTRA VEZ."
 38 GOTO 20
 40 FOR N=0 TO 100
 50 PRINT N
 60 NEXT N
 70 END
100 PRINT "¡ADIOS!"
```

Pruebe el programa y verá que no admite más respuesta que una de las dos previstas.

MÁS DECISIONES

Si se lo pedimos, el ordenador puede tomar decisiones basadas en comprobaciones más complejas que las que hemos visto hasta ahora. Copie el siguiente programa:

```
10 CLS
20 REM Este programa reúne informa-
   ción acerca de su familia
30 PRINT "¿TIENE ALGUN HERMANO?
   RESPONDA SI O NO."
40 INPUT A$
50 PRINT "¿TIENE ALGUNA HERMA-
   NA? RESPONDA SI O NO."
60 INPUT A$
70 IF A$="SI" AND B$="SI" THEN
   PRINT "SUS PADRES TIENEN AL
   MENOS TRES HIJOS."
80 IF A$="SI" OR B$="SI" THEN
   PRINT "SUS PADRES TIENEN AL
   MENOS DOS HIJOS."
90 IF A$<>"SI" AND B$="SI" THEN
   PRINT "¡QUE SUERTE! NO TIENE
   HERMANOS."
100 IF A$="SI" AND B$<>"SI" THEN
   PRINT "¡QUE SUERTE! NO TIENE
   HERMANAS."
110 IF A$<>"SI" AND B$<>"SI" THEN
   PRINT "ES HIJO UNICO."
120 END
```

Ejecute el programa varias veces y dé respuesta diferentes. Si lo ha escrito en mayúsculas, el programa sólo reconocerá la respuesta 'SI' y tomará cualquier otra como negativa; en cambio, si lo ha introducido en minúsculas, sólo reconocerá 'sí' (con o sin acento, dependiendo de cómo haya escrito usted esta palabra).

El programa es bastante sencillo. La novedad está en **AND** ('y'), **OR** ('o') y <> ('distinto de'), cuyo significado para BASIC es el mismo que en el lenguaje ordinario. Por ejemplo, la línea 90 se lee de la siguiente manera:

> Si A$ es distinto de SI y B$ es igual a SI, entonces escribe ...

ESCRÍBALO A SU GUSTO

Hay varias formas de modificar la orden **PRINT** para controlar la forma en que escribe en la pantalla. La primera posibilidad es escribir una coma o un

signo de punto y coma al final de la instrucción. Copie el siguiente programa:

```
10 FOR N=0 TO 100
20 PRINT "¡HOLA!"
30 NEXT I
```

Ejecútelo. Ésta es la versión de **PRINT** que ya conocemos. Ahora cambie la línea 20:

```
10 FOR N=0 TO 100
20 PRINT "¡HOLA!",
30 NEXT I
```

Ejecute el programa y compare el resultado con el anterior.

Cambie otra vez la línea 20:

```
10 FOR N=0 TO 100
20 PRINT "¡HOLA!";
30 NEXT I
```

Son tres formas muy diferentes de tratar la información en la pantalla. Más adelante utilizaremos selectivamente estos estilos en nuestros programas.

TAB

TAB es otro *modificador* de **PRINT**. Ofrece un medio de decirle al ordenador dónde queremos que escriba.

Introduzca las siguientes órdenes en modo inmediato (o sea, sin números de línea):

```
PRINT TAB(20);"¡HOLA!"
```

y

```
PRINT TAB(30);"¡ADIOS!"
```

Antes de que **PRINT** actúe, **TAB** hace avanzar la posición de escritura hasta la columna especificada entre los paréntesis. La primera columna por la izquierda es la número 0; la última es la 36.ª en **SCREEEN 0** y la 28.ª en **SCREEN 1**.

Capítulo 9

Algunas ideas «aleatorias»

En la mayor parte de los juegos, desde el tute hasta la ruleta, interviene como componente fundamental el azar. El hecho de que no podamos predecir las cartas que recibe cada jugador, el resultado de lanzar un par de dados, el color y número en que se detiene la ruleta, etc., es lo que da interés a los juegos.

El término «aleatorio» hace referencia al azar, a lo impredecible. Un ejemplo de número aleatorio es el que se obtiene cuando se extrae una bola del bombo de la lotería.

Los números aleatorios son importantes en los programas de ordenador, especialmente en los de juegos y en algunas aplicaciones científicas y matemáticas.

Para nosotros es muy fácil generar un número aleatorio lanzando un par de dados. No lo es tanto para el ordenador. Muchos ordenadores ni siquiera son capaces de generar números verdaderamente aleatorios, sino solamente números «pseudoaleatorios», es decir, una sucesión de valores que es predecible por el hecho de que empieza siempre por el mismo número. Es como si las bolas de la lotería estuvieran ensartadas en una cuerda.

Pues bien, el **SVI•728** sí puede generar números auténticamente aleatorios. Basta para ello con poner la línea:

```
5 N=RND(-TIME)
```

al principio de todos los programas en los que se vaya a utilizar números aleatorios.

RND, INT, TIME

En las próximas páginas vamos a explicar tres nuevas palabras clave de BASIC: **RND**, **INT** y **TIME**.

En realidad, los números aleatorios generados por el **SVI•728** forman parte de una sucesión muy larga. Si no tomásemos medidas especiales, la sucesión sería siempre la misma y, por lo tanto, sería predecible. Ahora bien, cada vez que usamos la función **RND( )** con un número negativo entre los paréntesis, lo que estamos haciendo es seleccionar una posición, dentro de esa sucesión, a partir de la cual el ordenador leerá números aleatorios en lo sucesivo. Pero ¿por qué hemos elegido **−TIME** para poner entre los paréntesis de **RND**?

TIME es uno de los conocidos «recipientes», en el que el **SVI•728** deposita un número, constantemente actualizado, que depende del tiempo transcurrido desde que encendimos el ordenador. Por eso, al usar **RND(−TIME)** estamos seleccionando en la sucesión de números aleatorios un punto de partida que es muy difícilmente repetible.

Copie el siguiente programa para observar el funcionamiento de **RND** e **INT**.

```
10 REM Este programa genera un
número aleatorio
20 N=RND(-TIME)
30 X=INT(RND(1)*10)
40 PRINT X
```

En la línea 30 hay una sentencia que aún no conocíamos. La primera instrucción a la que el ordenador responde en esta línea es **RND(1)**. El número que figura entre paréntesis es lo que denominamos un *número auxiliar*. En lugar del **1** podríamos haber puesto cualquier otro, siempre que ese número sea positivo, el ordenador, al encontrarse con **RND( )**, genera un número aleatorio comprendido entre 0 y 1; por ejemplo, el 0,2345.

Lo siguiente que hace el ordenador en la línea 30 es multiplicar el número aleatorio por 10. Como lo que buscamos es un número entero (sin decimales),

podemos quitarle los decimales aplicándole **INT**. De esta manera, si el número aleatorio efectivamente fuese el 0.2345, al multiplicarlo por 10 obtendríamos 2.345, y al aplicarle **INT** resultaría **X=2**.

Vamos a probar un programa que genera números aleatorios para simular el lanzamiento de un par de dados:

```
10 REM Este programa simula el
   lanzamiento de dos dados
20 N=RND(-TIME)
30 X=INT(RND(1)*6+1)
35 IF X=7 THEN GOTO 30
40 Y=INT(RND(1)*6+1)
45 IF Y=7 THEN GOTO 40
50 R=X+Y
60 PRINT R
```

Las líneas 30 y 40 generan los dos números aleatorios. Supongamos que el número generado por **RND(1)** en la línea 30 es 0.9678. Al multiplicarlo por 6 obtenemos 5.8068.

Al sumarle 1 resulta 6.8068. Finalmente, **INT** le quita los decimales, con lo que en el primer dado ha salido un 6. Análogamente funciona la línea 40. La línea 50 suma los dos resultados y la línea 60 los escribe en la pantalla. Las líneas 35 y 45 rechazan los números en los casos extremos en que **RND(1)** genera el 1.

No podemos negar que todo esto de los números aleatorios es algo complicado. Por eso le sugerimos que vuelva a leer esta sección y que practique con los ejemplos.

MÁS SALTOS

El lector ya sabe cómo utilizar **GOTO** para saltar de una línea a otra del programa. Vamos a presentar un segundo tipo de saltos y a explicar las diferencias entre ambos.

Transcriba el siguiente programa:

```
10 REM Este programa convierte años
   en meses
20 CLS
30 PRINT "¿Cuántos años tienes?"
40 INPUT N
50 GOSUB 200
60 PRINT "Cuántos años tiene tu
   padre?"
70 INPUT N
80 GOSUB 200
90 PRINT "Cuántos años tiene tu
   madre?"
100 INPUT N
110 GOSUB 200
120 END
200 PRINT N;"años es igual a";N*12;
   "meses"
210 RETURN
```

Pruebe el programa y responda a las preguntas. ¿Entiende cómo funciona el programa? Vamos a repasarlo.

GOSUB/RETURN

Después de hacer la pregunta en la línea 30 y captar la respuesta en la 40, el ordenador se encuentra la instrucción **GOSUB 200**, que significa 've a la subrutina que empieza en la línea 200'. Una subrutina es una zona del programa que consiste en una o varias líneas y es utilizada frecuentemente por la parte principal del programa. Se le llama *sub*rutina porque está subordinada al resto del programa.

Pues bien, cuando el ordenador encuentra **GOSUB 200** de la línea 50, salta a la línea 200. En ella realiza la conversión de años a meses y escribe el resultado. Después llega a la línea 210, donde la orden le obliga a volver a la sentencia siguiente al **GOSUB 200**; es decir, en este caso vuelve a la línea 60.

Lo mismo ocurre cuando el programa ejecuta las líneas 80 y 110; en estos casos el retorno se produce a las líneas 90 y 120, respectivamente.

¿Cuál es, pues, la diferencia entre **GOTO** y **GOSUB**?

Si en la línea 50 hubiésemos puesto **GOTO 200** en lugar de **GOSUB 200**, en la 210 habríamos necesitado un **GOTO 60** (en vez del **RETURN**). Pero entonces la subrutina ya no podría ser invocada desde las líneas 80 y 110, porque la línea 210 siempre reenviará el programa a la 60.

En el fondo, lo que ocurre es que, cuando el ordenador obedece una instrucción **GOTO**, realiza el salto pedido y **no** recuerda de dónde procede. En cambio, con el par **GOSUB/RETURN**, el ordenador sabe a qué línea tiene que volver cuando termine la subrutina.

El siguiente programa está basado en una variante de **GOSUB/RETURN**.

```
10 REM Este programa es un mini-listín
   telefónico
30 PRINT "Si necesita un número de
   teléfono de urgencia, seleccione una
   de estas opciones:"
40 PRINT "1. Médico"
50 PRINT "2. Policía"
60 PRINT "3. SPECTRAVIDEO"
70 INPUT N
80 ON N GOSUB 100,200,300
90 GOTO 10
100 PRINT "El teléfono del médico es
   1234567"
110 RETURN
200 PRINT "El teléfono de la policía es
   091"
210 RETURN
300 PRINT "El teléfono de Spectravideo
   es 6757822"
310 RETURN
```

Pruebe el programa. La variante de **GOSUB** está en la línea 80. La instrucción **ON N GOSUB 100,200,300** significa: 'según sea el valor de N, ve a la subrutina 100, a la 200 o a la 300'. Así pues, si

el número que está depositado en el recipiente **N** es 1, el programa salta a la subrutina que empieza en la línea 100; si es 2, a la 200; si es 3, a la 300.

EL ORDENADOR COMO CALCULADORA

Se puede hacer que el ordenador funcione como simple calculadora:

Por ejemplo, para sumar 6 más 3 escriba:

```
PRINT 6+3[ENTER]
```

y el ordenador escribirá el resultado:

```
9
```

Para restar,

```
PRINT 6-3[ENTER]
 3
```

Para multiplicar,

```
PRINT 6*3[ENTER]
 18
```

Para dividir,

```
PRINT 6/3[ENTER]
 2
```

Si se omite la palabra **PRINT**, el ordenador no escribe el resultado; ni siquiera lo calcula. Por ejemplo, si escribimos **6+3** sin **PRINT**, el ordenador cree que estamos introduciendo la línea 6 y que su texto es '+3'.

OPERACIONES ARITMÉTICAS

No vamos a insistir en la cuestión de las operaciones aritméticas más de lo necesario. Lo único que nos falta es comentar el orden en que se realizan las operaciones múltiples. En efecto, si escribimos

```
PRINT 3+4*5[ENTER]
```

¿qué resultado nos dará el ordenador? Cabe la duda de si estamos pidiendo al **SVI•728** que sume 3

más 4 y multiplique el resultado por 5, o si queremos que multiplique 4 por 5 y sume 3 al resultado:

(3+4)*5 o 3+(4*5)

Esta duda se resuelve si sabemos cuál es el orden de prioridad de los operadores:

1. ( )
2. **NOT**
3. * /
4. + −
5. > < = >= <= < >
6. **AND**
7. **OR**

De esta lista se deduce, por ejemplo, que la multiplicación (*) **se realiza antes que la suma** (+), de modo que **3+4*5** se calcula de la siguiente forma:

3 + 4 * 5 = 3 + 20 = 23

La lista indica también que las operaciones que están dentro de paréntesis son prioritarias con respecto a todas las demás. Así, podemos forzar un orden distinto del normal mediante la cuidadosa utilización de los paréntesis. Por ejemplo,

(3 + 4) * 5 = 7 * 5 = 35

En caso de que en una operación múltiple intervengan dos operadores de igual prioridad, el cálculo se realiza de izquierda a derecha.

PROCESO DE LA INFORMACIÓN

Al principio de este libro dijimos que un programa de ordenador es un conjunto de instrucciones que procesan información o, lo que es lo mismo, que actúan sobre la información.

La información que un programa necesita para empezar a trabajar es lo que se llama *entrada*. Los resultados del proceso constituyen la *salida*.

En el siguiente ejemplo vamos a ver cómo se le dice al ordenador cuál es la información que queremos procesar. Casi todas las órdenes de BASIC indican

al ordenador cómo debe procesar la información. Hay otras (por ejemplo, **INPUT** y **LET X=20**) que le dicen qué información debe procesar. Vamos a presentar un nuevo par de sentencias, **READ/DATA** ('leer/datos'), que, como su nombre indica, sirven también para suministrar información a los programas.

Transcriba el siguiente programa:

```
 5 CLS
10 REM Este programa lee en una lista los
   nombres de los cinco primeros meses
   del año y los escribe en la pantalla
20 FOR X=1 TO 5
30 READ F$
40 PRINT F$
50 NEXT X
60 DATA Enero,Febrero,Marzo,Abril,Mayo
```

Ejecute el programa.

Si todo ha ido bien, el programa habrá escrito los nombres de los cinco primeros meses del año. Pero veamos cómo funciona este programa.

READ/DATA

La línea 20 abre un bucle que va a ser recorrido cinco veces. La acción del bucle está descrita en las líneas 30 y 40. Consiste en leer (**READ**) una palabra en la lista de datos (**DATA**), depositarla en el recipiente **F$** y escribir en la pantalla el contenido de **F$**. Cuando el ordenador encuentra la orden **READ**, examina el programa en busca de una sentencia **DATA**; en cuanto la encuentra, lee la primera palabra (hasta la coma), 'Enero', y la coloca en el recipiente **F$**. La línea 40 escribe esa palabra en la pantalla y la línea 50 reinicia el bucle.

La segunda vez que se ejecuta **READ**, el ordenador sabe donde terminó de leer la vez anterior; por eso lee 'Febrero', no 'Enero'. Dicho gráficamente, el ordenador mantiene actualizado un puntero que siempre está señalando el dato que está disponible para ser leído por la próxima orden **READ**.

La línea **DATA** puede estar en cualquier lugar del programa, no necesariamente antes de **READ**. Por ejemplo, los siguientes programas son equivalentes al anterior:

```
 5 CLS
10 REM Este programa lee en una lista los
   nombres de los cinco primeros meses
   del año y los escribe en la pantalla
15 DATA Enero,Febrero,Marzo,Abril,Mayo
20 FOR X=1 TO 5
30 READ F$
40 PRINT F$
50 NEXT X
```

```
 2 DATA Enero,Febrero,Marzo,Abril,Mayo
 5 CLS
10 REM Este programa lee en una lista los
   nombres de los cinco primeros meses
   del año y los escribe en la pantalla
20 FOR X=1 TO 5
30 READ F$
40 PRINT F$
50 NEXT X
```

Los diferentes elementos de información se escriben en las líneas **DATA** separados por coma. El ordenador considera que todo lo que hay entre dos comas, incluidos los espacios, es un solo dato.

Copie y ejecute este programa:

```
10 CLS
20 REM Este programa lee en una lista
   los nombres de cinco personas y los
   escribe en la pantalla
30 FOR X=1 TO 5
40 READ F$
50 PRINT F$
60 NEXT X
70 DATA Juan López,Marta Ruiz,Elena
   Fernández,José Martin,Carmen Díaz
```

¿En qué se distingue el programa anterior del siguiente? Trate de averiguar en qué consiste la diferencia sin transcribirlo:

```
10 CLS
20 REM Este programa lee en una lista
   los nombres de cinco personas y los
   escribe en la pantalla
30 FOR X=1 TO 5
40 READ F$:READ G$
50 PRINT F$,G$
60 NEXT X
70 DATA Juan,López,Marta,Ruiz,Elena,
   Fernández,José,Martín,Carmen,Díaz
```

El primer programa lee cada conjunto de nombre, espacio y apellido de una sola vez. El segundo lee el nombre (**F$**) y el apellido (**G$**) por separado.

Mensaje OUT OF DATA

Si todavía no lo ha hecho, copie este último programa. Ejecútelo. Ahora añada la siguiente línea:

```
80 GOTO 30
```

y ejecute otra vez el programa. Ha provocado el mensaje **OUT OF DATA** ('datos agotados'), ¿verdad? Esto se debe a que el programa ha tratado de seguir leyendo datos más allá del 'Díaz' final.

La información se puede leer por segunda vez si previamente se la «restaura» con la orden **RESTORE**:

```
10 CLS
20 CLEAR 500
30 FOR X=1 TO 5
40 READ F$:READ G$
50 PRINT F$,G$
60 NEXT X
70 RESTORE
80 GOTO 30
90 DATA Juan,López,Marta,Ruiz,
   Elena,Fernández,José,Martín,Carmen,
   Díaz
```

CLEAR

Veamos cómo funciona el programa. Hemos añadido una sentencia nueva, **CLEAR**, en la línea 20. Si no se incluye esta orden en el programa, BASIC reserva 200 caracteres para los recipientes en los que se va a almacenar texto. Con **CLEAR** se puede reservar un espacio mayor (en este ejemplo, para 500 caracteres); esto es conveniente en todos los programas que manejen cadenas de texto.

RESTORE

En la línea 70 hemos puesto **RESTORE**. Esta orden restaura los datos para que puedan ser leídos nuevamente por sentencias **READ**. De esta manera, cuando se vuelva a iniciar el bucle como consecuencia de la línea 80, el programa podrá leer los nombres sin que se produzca el mensaje **OUT OF DATA**.

OTRA FORMA DE AHORRAR TRABAJO

Nuestros programas van siendo cada vez más largos y empieza a resultarnos pesado eso de tener que escribir tantos números de línea. El **SVI•728** puede escribirlos AUTOmáticamente si le damos la orden

AUTO[ENTER]

AUTO

El ordenador responde escribiendo el número 10 en la pantalla y dejando el cursor a su derecha, listo para que empecemos a escribir la primera línea del programa. Cuando pulsamos [ENTER] al terminar de escribir la línea, el ordenador nos ofrece el número 20, y así sucesivamente. Así pues, genera números de línea partiendo del 10 e incrementándolos de diez en diez.

Pero también podemos hacer que empiece un número distinto y que el incremento de un número al siguiente sea el que queramos. Por ejemplo, con

AUTO 20,40

el primer número de línea será el 20, el segundo el 60, el tercero el 100, etc.

Capítulo 10

Matrices: una forma de organizar muchos recipientes

Creemos que usted ya es suficientemente experto como para que podamos tomar esta decisión: a partir de ahora, a los «recipientes» les vamos a dar su nombre correcto, que es *variables*.

Por otra parte, es de esperar que usted ya se sienta cómodo en compañía de las variables.

En los capítulos anteriores siempre las hemos utilizado como entes aislados, independientes unos de otros. **A**, **B**, **C$** y **D$** son ejemplos de variables en las que podemos almacenar valores numéricos o literales, según los casos. En este capítulo vamos a estudiar la forma de agrupar variables para facilitar su utilización.

En efecto, hay ocasiones en que es necesario reunir una serie de variables para formar un grupo. Por ejemplo, si queremos guardar las notas que han obtenido 25 estudiantes en 5 asignaturas, es muy deseable que nos las ingeniemos para manejar 25 grupos de variables, en lugar de 125 variables aisladas. El grupo correspondiente a Juan podría tener las cinco notas siguientes:

Juan
(1) 7
(2) 8
(3) 9
(4) 6
(5) 10

Tendríamos un total de 25 listas similares a ésta, una para cada estudiante. A la nota que consiguió Juan en la primera asignatura le podríamos llamar Juan(1); a la de la segunda asignatura, Juan(2); etc.

El programa siguiente muestra cómo podemos decirle al ordenador cuáles han sido las notas de Juan:

```
10 REM Este programa agrupa cinco
   variables y les asigna las notas de
   Juan
20 CLS
30 DIM J(5)
40 FOR X=1 TO 5
50 READ J(X)
60 NEXT X
70 PRINT "¿De qué asignatura quieres
   saber la nota?"
80 INPUT Z
90 PRINT J(Z)
100 DATA 7,8,9,6,10
```

Transcriba y pruebe el programa. Vamos a explicarle cómo funciona.

DIM

La línea 30 dimensiona el grupo de cinco variables, cuyo nombre colectivo es **J**. La misión de **DIM** es hacer que el ordenador reserve espacio para guardar cinco números en cinco variables diferentes. En la línea 50 **READ** coloca los valores en las variables, uno en cada pasada por el bucle.

Cuando el programa llega a la línea 70, las cinco variables ya tienen asignados sus valores respectivos. Esta información está organizada en forma de lista ordenada. Por eso, la línea 90 puede acceder a uno de los cinco datos sabiendo cuál es su posición (**Z**) en la lista.

El nombre correcto para estos grupos de variables es *matriz*.

MATRICES MULTIDIMENSIONALES

La matriz **J( )** del ejemplo anterior era unidimensional. Esto quiere decir que cada variable de la matriz se identifica por un sólo número de orden, o sea, que la matriz tiene una sola dimensión.

Pero en el **SVI•728** también disponemos de matrices de varias dimensiones. Supongamos, por ejemplo, que queremos guardar los nombres de cinco estu-

diantes y sus respectivas notas en química. Esta información se podría tabular de la siguiente forma:

Notas de química

Estudiante n.°	Nombre	Puntos
1	Juan	7
2	Marta	8
3	Elena	7
4	José	9
5	Carmen	6

El siguiente programa lee en una línea **DATA** esta información, la asigna a una matriz y la muestra en la pantalla. Transcriba el programa y trate de entender su funcionamiento antes de que se lo expliquemos.

```
10 REM Este programa forma una matriz
   bidimensional y le asigna valores
20 CLS
30 DIM V$(5,2)
40 FOR N=1 TO 5
50 FOR S=1 TO 2
60 READ V$(N,S)
70 NEXT S:NEXT N
80 REM Esta sección del programa escri-
   be la tabla en la pantalla
90 PRINT "Nombre","Nota":PRINT
100 FOR N=1 TO 5
110 PRINT V$(N,1),V$(N,2)
120 NEXT N
130 DATA Juan,7,Marta,8,Elena,7,José,9,
    Carmen,6
```

¡No se asuste! Éste es el programa más difícil que veremos en esta guía.

La línea 30 dimensiona una matriz de dos dimensiones. En total, las variables agrupadas son 10: cinco para los nombres y cinco para las notas. Cada elemento de la matriz se identifica en este caso por un

número doble. Por ejemplo, el elemento (1,1) hace referencia al nombre del primer alumno, 'Juan'; el elemento (4,2) hace referencia a la nota del cuarto alumno, '9'; etc.

Para asignar valores a las variables de esta matriz necesitamos dos bucles: uno que recorra las cinco filas y otro que recorra las dos columnas. Estos bucles son los que comienzan en las líneas 40 y 50.

La rutina 100–120 escribe en la pantalla los datos almacenados en la matriz. Aquí sólo necesitamos un bucle, ya que las dos columnas están explicitadas en la línea 110.

¿Podríamos haber escrito este programa de otra manera? Desde luego; casi todos los problemas de programación admiten varias soluciones. Las soluciones adoptadas suelen depender del estilo y de las preferencias de cada programador. Con la práctica, usted también aprenderá a discernir qué opción prefiere en cada caso.

Veamos una solución alternativa al programa anterior, en la que hemos utilizado dos matrices unidimensionales en lugar de una bidimensional:

```
 10 REM Este programa forma dos
    matrices unidimensionales y les
    asigna valores
 20 CLS
 30 DIM A$(5),B(5)
 40 FOR M=1 TO 5
 60 READ A$(M),B(M)
 70 NEXT M
 80 REM Esta sección del programa
    escribe la tabla en la pantalla
 90 PRINT "Nombre","Nota":PRINT
100 FOR K=1 TO 5
110 PRINT A$(M),B(M)
120 NEXT M
130 DATA Juan,7,Marta,8,Elena,7,José,9,
    Carmen,6
```

Ejecute este programa y trate de entender su funcionamiento.

Capítulo 11

Cadenas literales

CADENAS

¡Enhorabuena! El hecho de que usted haya llegado hasta aquí es un signo de que ha entrado con buen pie en la informática. En este capítulo vamos a enseñarle a manipular cadenas literales. Una *cadena literal* es sencillamente un grupo de caracteres; en general, una palabra o una frase.

Copie el siguiente programa:

```
10 CLEAR 300
20 INPUT "Escriba una frase no demasiado
   larga.";S$
30 PRINT "Número de caracteres de su
   frase:";
40 PRINT LEN(S$)
```

Ejecute el programa.

LEN es abreviatura de la palabra inglesa *length*, que significa 'longitud'. Así pues, **LEN** calcula la longitud (en número de caracteres) de las cadenas literales. En este ejemplo, la frase que el usuario introduce es almacenada en la variable **S$**.

INPUT

Es de esperar que la estructura de la línea 20 le haya llamado la atención. Hemos combinado en una sola sentencia lo que en capítulos anteriores separábamos en dos: la instrucción **PRINT** con el mensaje «inductor» y la instrucción **INPUT**. Como ha com-

probado al ejecutar el programa, esto es posible; además, es muy conveniente, ya que ayuda a simplificar los programas.

LEFT$
RIGHT$
MID$

Hay tres instrucciones que nos permiten manipular las cadenas literales de varias formas: se trata de **LEFT$**, **MID$** y **RIGHT$** ('izquierda', 'centro', 'derecha'). Vamos a verlas funcionar en los ejemplos siguientes:

```
10 INPUT "Escriba una palabra que tenga
   al menos dos letras.";P$
20 PRINT "La primera letra es
   ";LEFT$(P$,1)
30 PRINT "Las dos últimas letras son
   ";RIGHT$(P$,2)
40 GOTO 10
```

La sentencia **LEFT$(P$,1)** de la línea 20 lee el primer carácter por la izquierda en la cadena **P$**. Si hubiéramos puesto **LEFT$(P$,3)**, habría leído los 3 primeros caracteres.

Análogamente, **RIGHT$(P$,2)** lee los dos últimos caracteres (los dos primeros por la derecha).

La función **MID$** es aún más potente que **LEFT$** y **RIGHT$**. Copie y pruebe el siguiente programa:

```
10 P$="ENCICLOPEDIA"
20 PRINT MID$(P$,3,5)
```

La sentencia **MID$(P$,3,5)** de la línea 20 extrae de la palabra **P$** 5 letras a partir de la 3.ª.

Otra aplicación interesante de **MID$** es buscar una palabra dentro de una frase o, más en general, una subcadena en una cadena. Transcriba y pruebe el siguiente programa:

```
10 CLEAR 500
20 INPUT "Escriba una frase de no más
   de 10 palabras.";F$
30 INPUT "Escriba una palabra que figure
   en la frase.";P$
40 X=LEN(P$)
50 FOR T=1 TO LEN(F$)
60 IF MID$(S$,T,X)=P$ THEN GOTO 100
70 NEXT T
80 PRINT "La palabra buscada no está en
   la frase"
90 END
100 PRINT "La palabra ";P$;" aparece en la
   frase a partir del carácter número";T
```

LEN

En este programa hemos usado **MID$** para inspeccionar la cadena global formando grupos de caracteres, haciendo que la longitud de cada grupo sea igual a la de la subcadena. La línea 40 calcula la longitud (**LEN**) de la subcadena y la asigna a la variable **X**. Ese valor se utiliza en la línea 60 para formar grupos de caracteres tomados de **F$** y compararlos con **P$**.

Capítulo 12

Programación avanzada de gráficos y sonido

En este capítulo vamos a estudiar los recursos gráficos y sonoros del **SVI•728**.

El capítulo está dividido en dos secciones. En la primera repasaremos y ampliaremos las instrucciones de gráficos que usted ya conoce. En la segunda daremos una introducción a la programación de sonidos basada en la orden **PLAY**. En el Apéndice H se explica cómo controlar más directamente el generador de sonido mediante la orden **SOUND**.

Sección 1 Gráficos avanzados

CIRCLE

Para empezar a explorar la capacidad gráfica del **SVI•728**, copie el siguiente programa. No olvide pulsar [ENTER] al final de cada línea.

```
10 SCREEN 2
20 CIRCLE (128,80),60,11
30 PAINT (128,80),11
40 GOTO 40
```

Cuando ejecute el programa verá cómo el **SVI•728** dibuja una circunferencia y luego la rellena de color amarillo. Analicemos las líneas del programa:

```
10 SCREEN 2
```

Esta línea activa la pantalla gráfica.

```
20 CIRCLE (128,80),60,11
```

CIRCLE es una palabra inglesa que significa 'circunferencia'. Los dos números que figuran entre paréntesis especifican las coordenadas del centro de la circunferencia; en este caso, (128,80). El primero es la distancia del centro al borde izquierdo de la pantalla; el segundo es la distancia del centro al borde superior de la pantalla.

El número siguiente especifica el radio de la circunferencia; en este caso, 60. El 11 especifica el color con que se debe dibujar la circunferencia

PAINT

```
30 PAINT (128,80),11
```

La orden **PAINT** hace que el ordenador «pinte» ciertas áreas rellenándolas con un color especificado. Así, la línea 30 pide al ordenador que rellene la circunferencia dibujada en la línea 20. La orden **PAINT** tiene que ir seguida de la especificación de la coordenadas de un punto interior al recinto que se desea rellenar. Si las coordenadas especifican un punto exterior, lo que se rellena no es el recinto, sino todo lo que lo rodea.

Además, hay que especificar también el color con el que se va a rellenar. Ese color tiene que ser el mismo que el del borde del recinto. De hecho, lo que el ordenador hace es empezar a rellenar desde el punto especificado y no se detiene mientras no encuentra en la pantalla algún límite del mismo color con el que se le ha pedido que rellene. Por otra parte, si el recinto no es cerrado, la tinta de rellenado «se escapa» e invade el resto de la pantalla.

```
40 GOTO 40
```

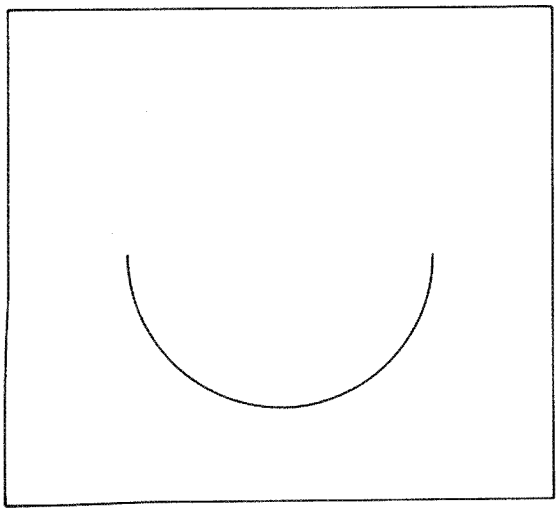
Esta línea mantiene la imagen en la pantalla gráfica. Para interrumpir el programa, pulse CTRL STOP.

Experimente con otros valores para modificar la posición, el radio y el color de la circunferencia.

La orden **CIRCLE** se puede completar con otros parámetros para crear gran variedad de figuras. Veamos un ejemplo:

```
10 SCREEN 2
20 CIRCLE (128,96),80,13,3.14,6.28
30 GOTO 30
```

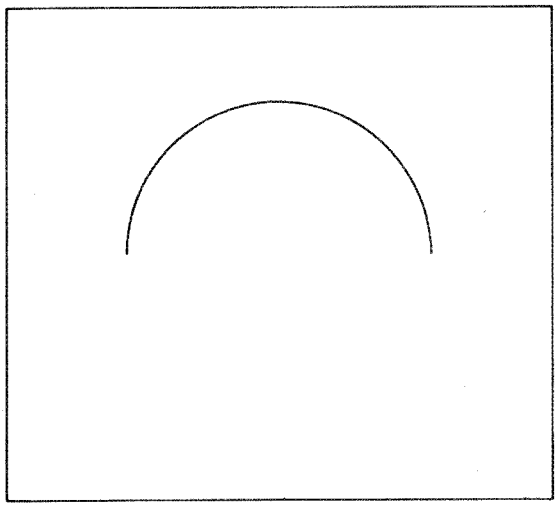
Al ejecutar este programa verá cómo aparece sólo la mitad inferior de la circunferencia:



Si ahora cambiamos la línea 20 por esta otra:

```
20 CIRCLE (128,96),80,13,6.28,3.14
```

el programa dibujará la mitad superior de la circunferencia:



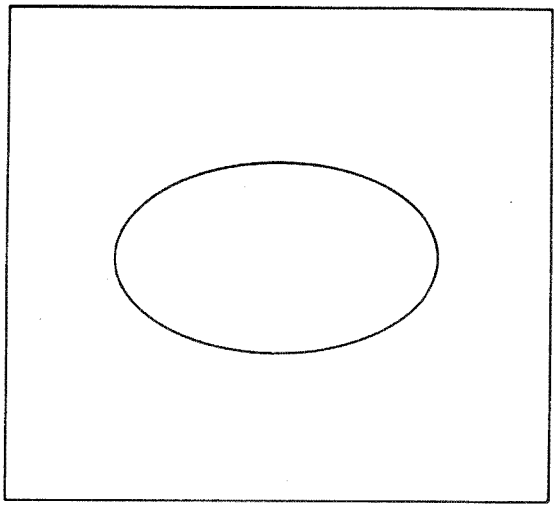
Otra forma de obtener la circunferencia completa sería con esta línea:

```
20 CIRCLE (128,96),80,13,0,6.28
```

¿Qué tienen de mágico los números 3.14 y 6.28? Recuerde de sus estudios de geometría que 3.14 es (aproximadamente) el número π. En el **SVI•728** los ángulos se miden en *radianes*. Media circunferencia es π radianes; la circunferencia completa es 2π radianes. Los números que ponemos a continuación del número de color indican los ángulos inicial y final del arco de circunferencia.

La circunferencia se puede deformar incluyendo un último parámetro en la orden **CIRCLE**: la excentricidad. Cambie otra vez la línea 20:

```
20 CIRCLE (128,96),80,13,,,1/2
```

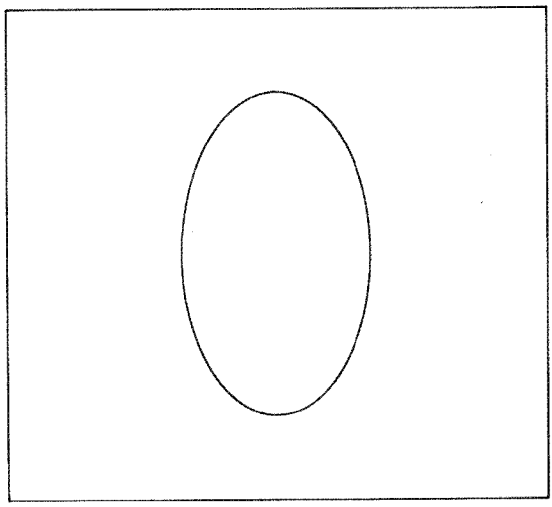


¿De dónde ha salido la elipse? Las tres comas seguidas indican que no queremos especificar los ángulos inicial y final, sino dibujar una circunferencia completa. El último número indica que en realidad no queremos una verdadera circunferencia, sino una elipse, y que la excentricidad (relación entre el diámetro vertical y el horizontal) va a ser 1/2.

La circunferencia es un caso particular de la elipse: una elipse en la que los dos diámetros son iguales (excentricidad igual a uno).

Si la excentricidad es menor que 1, como en el ejemplo anterior, la elipse es más ancha que alta. Si es mayor que 1, la elipse es más alta que ancha:

```
20 CIRCLE (128,96),80,13,,,2
```

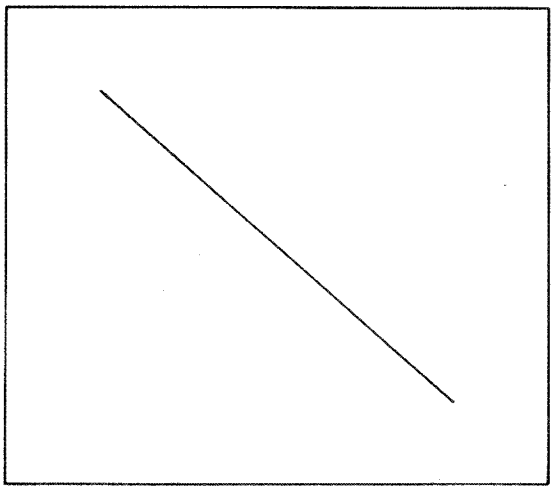


RECTAS (LINE)

Puesto que ya sabemos dibujar circunferencias y elipses y rellenarlas, vamos a ocuparnos de las rectas y los rectángulos. El **SVI•728** dispone de una orden muy potente para dibujar rectas y rectángulos: **LINE** ('recta'). Dé la orden **NEW** para borrar la memoria del ordenador y copie el siguiente programa:

```
10 SCREEN 2
20 LINE (50,40)-(200,150),14
30 GOTO 30
```

Cuando ejecute el programa, verá que el ordenador dibuja una recta que desciende por la pantalla diagonalmente de izquierda a derecha:



La clave está en la línea 20:

```
20 LINE (50,40)-(200,150),14
```

Las dos parejas de números, (50,40) y (200,150), son las coordenadas de los dos puntos extremos de la recta. El último número, 14, especifica el color.

RECTÁNGULOS (B)

Sin más que añadir una **B** a la **LINE** podemos convertir la recta en un rectángulo. (La 'B' es la inicial de la palabra inglesa *box*, que significa 'recuadro'.) Modifique la línea 20:

```
20 LINE (50,40)-(200,150),14,B
```

Ejecute la nueva versión del programa.

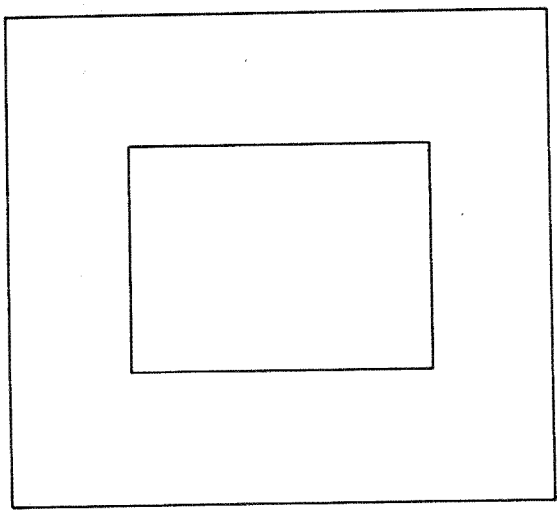
La letra **B** hace que el **SVI•728** dibuje el rectángulo de lados verticales y horizontales que encaja en los dos puntos especificados.

RECTÁNGULOS RELLENADOS (BF)

Si además queremos rellenar el rectángulo, debemos poner la letra **F** (de *fill*, 'rellenar') a continuación de la **B**, sin separarlas con comas. Modifique la línea 20:

```
20 LINE (50,40)-(200,150),14,BF
```

Ejecute el programa y verá cómo el ordenador dibuja el rectángulo y lo rellena (con el mismo color del borde).



El siguiente programa da un repaso a las órdenes de creación de gráficos que hemos estudiado hasta ahora.

```
10 COLOR 1,9
20 SCREEN 2
30 CIRCLE (126,110),60,9,,,1.3
40 CIRCLE (110,96),10,2
50 PAINT (110,96),2
60 CIRCLE (142,96),10,2
70 PAINT (142,96),2
80 LINE (100,125)-(105,135),14
```

```
 90 LINE (152,125)-(147,135),14
100 LINE (105,135)-(147,135),14
110 CIRCLE (74,110),25,11,,,5
120 CIRCLE (178,110),25,11,,,5
130 Y=85:R=50:C=1
140 FOR Q=1 TO 10
150 Y=Y-5:R=R-4:C=C+1
160 CIRCLE (126,Y),R,C,,,,.2
170 NEXT Q
180 FOR T=1 TO 500:NEXT T
190 N=RND(-TIME)
200 FOR T=1 TO 30
210 X=X+10:Y=100
220 C=INT(RND(1)*15)+1
230 LINE (X,Y)-(X+35,Y +35),C,BF
240 LINE (X,Y)-(X+35,Y+35),1,BF
250 NEXT T
260 GOTO 260
```

DRAW

La orden **DRAW** es la puerta de acceso a un mini-lenguaje incluido en BASIC y denominado **GML** ('Graphic Macro Language'). Borre la memoria del ordenador con **NEW** y copie el siguiente programa:

```
10 SCREEN 2
20 PSET (50,60),1
30 DRAW "D50 R50 U50 L50"
40 GOTO 40
```

La línea 20 coloca el cursor gráfico en el punto de coordenadas (50,60) y selecciona el color número 1.

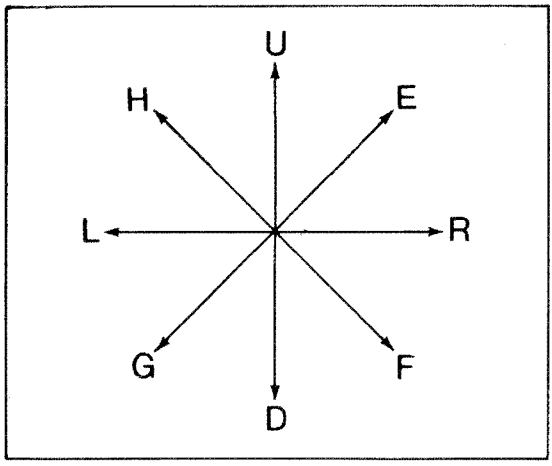
La línea 30 dibuja a partir de la posición del cursor previamente establecida por **PSET**. Las letras **D**, **R**, **U** y **L** son las iniciales de *down* ('abajo'), *right* ('derecha'), *up* ('arriba') y *left* ('izquierda'). Por consiguiente, **D50** significa 'bajar 50 unidades', etc.

DRAW actúa sobre una cadena literal, que puede ser una constante, como en el ejemplo anterior, o una variable, como en el siguiente:

```
10 SCREEN 2
20 PSET (50,60),1
30 T$ = "U50R50D50L50"
40 DRAW T$
50 GOTO 50
```

La línea 30 define la figura que queremos realizar, **T$**; la línea 40 la dibuja.

También se puede dibujar en diagonal. Las letras necesarias son las indicadas en la figura siguiente:



ESCALA (S)
COLOR (C)

Copie y ejecute el siguiente programa:

```
10 COLOR 15,1
20 SCREEN 2:X=14:Y=120:CO=2:Z=1
30 PSET (X,Y)
40 A$="S=Z;C=CO;F15R50E15
   U45H15L30H15U20E10R20F10
   R10U12H15L30G20D35F25R25
   F10D18G10L35H10L10D14S0"
50 DRAW A$
```

```
60 X=X+16:Y=Y+8:
   CO=CO+1:Z=Z+1
70 IF Z>10 THEN 20
90 GOTO 30
```

La cadena **A$** contiene las sentencias **S=Z** y **C=CO**. **S** es abreviatura de *scale* ('escala'); por lo tanto, **S=Z** hace la escala del dibujo igual a **Z**. **C** es abreviatura de *color*; **C=CO** selecciona el color con el que se va a dibujar. Los valores de **Z** y **CO** han sido definidos en la línea 20. Cada vez que se realiza un dibujo, la línea 60 amplía la escala (**Z=Z+1**) y cambia de color (**CO=CO+1**).

MOVIMIENTO (M) SIN DIBUJAR (B)

La orden **M** lleva el cursor hasta el punto de coordenadas especificadas. Si la **M** va precedida de una **B**, el cursor se desplaza sin dibujar. Borre la memoria con **NEW** y pruebe el siguiente programa:

```
 10 REM Dibujos
 20 SCREEN 2
 30 A$="BM30,156C9F15R50E15U
    45H15L30H15U20E10R20F10R
    10U12H15L30G20D35F25R25F
    10D18G10L35H10L10D14"
 40 DRAW A$
 50 PAINT (42,154),9
 60 LINE (150,171)-(210,171),2
 70 LINE (210,171)-(230, 34),2
 80 LINE (230, 34)-(210, 34),2
 90 LINE (210, 34)-(190,151),2
100 LINE (190,151)-(170,151),2
110 LINE (170,151)-(150, 34),2
120 LINE (150, 34)-(130, 34),2
130 LINE (130, 34)-(150,171),2
140 PAINT (165,165),2
150 GOTO 150
```

En este ejemplo, la definición de la cadena **A$** (línea 30) contiene la sentencia **BM30,156**, gracias a la cual el cursor se desplaza hasta el punto de coordenadas sin dibujar. Si no dispusiéramos de **BM**, todos los dibujos empezarían en el extremo superior izquierdo de la pantalla gráfica.

SPRITE$

Con las órdenes que hemos manejado hasta ahora podemos realizar dibujos sencillos y estáticos. Para mover figuras por la pantalla disponemos en el **SVI•728** de los *sprites*, que son una especie de duendes que podemos crear y controlar con facilidad.

El proceso de visualización de un sprite en la pantalla consta de las siguientes fases:

Etapa 1. Seleccionar un sprite.

Etapa 2. Decirle qué ropa queremos que se ponga (o sea, qué forma debe tener).

Etapa 3. Decirle de qué color debe ser la ropa.

Etapa 4. Pedirle que se nos aparezca (en alguno de los 32 planos disponibles).

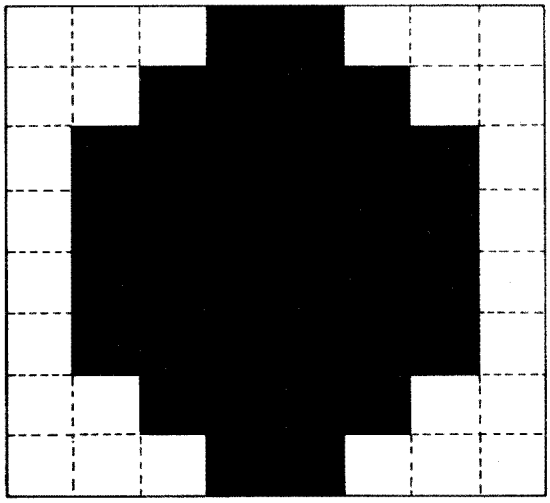
Es necesario comprender bien estas fases y no olvidar ninguna. Siempre que no se obtenga los resultados esperados al manejar sprites, es casi seguro que se ha olvidado sumistrar al ordenador parte de esta información.

Veamos un ejemplo. Transcriba y ejecute este programa:

```
 10 SCREEN 2
 20 FOR T=1 TO 8
 30 READ A$
 40 S$=S$+CHR$(VAL("&B"+A$))
 50 NEXT T
 60 SPRITE$(1)=S$
 70 PUT SPRITE 0,(128,96),8,1:
    GOTO 70
100 DATA 00011000
110 DATA 00111100
120 DATA 01111110
130 DATA 01111110
140 DATA 01111110
150 DATA 01111110
160 DATA 00111100
170 DATA 00011000
```

En el centro de la pantalla debe haber aparecido una bola roja. Se trata del sprite creado por el programa.

Las líneas 100 a 170 describen la forma (la ropa) del sprite, diseñada en una retícula de ocho filas y ocho columnas:



Cada línea **DATA** contiene una serie de ceros y unos que especifican una fila. Los ceros representan puntos transparentes; los unos, puntos iluminados. Los puntos de la retícula se denominan también *pixels* (de *picture element*, 'elemento de imagen'.)

Las líneas 20 a 50 forman un bucle que lee las líneas **DATA** y construye la cadena **S$**. A la salida del bucle, **S$** contiene la descripción de la forma del sprite.

La línea 60 asigna la descripción **S$** al sprite número 1.

La línea 70 coloca el sprite en el punto de coordenadas (128,96), selecciona para él el plano número 0 y el color 8. El 1 especifica el número del sprite.

(La sintaxis de la orden es: **PUT SPRITE** *<número del plano>*,**(X,Y)**,*<número del color>*,*<número del sprite>*. Los planos posibles son 0, 1, 2, . . ., 31. Gracias a los números de plano podemos tener varios sprites en la pantalla simultáneamente.)

Esta forma de crear y controlar imágenes es muy cómoda. Si usted conoce otras versiones de BASIC, en las que el movimiento de las imágenes se basa en el control directo de la memoria con PEEK y POKE, apreciará las ventajas del sistema de sprites del **SVI•728**.

Los sprites no están limitados al tamaño de la retícula de 8×8 pixels. En efecto, también podemos ampliarlos a tamaño doble y definirlos en una retícula de 16×16 pixels.

El tamaño de los sprites se especifica en la orden **SCREEN**, a continuación del número de la pantalla. Los sprites de 8×8 se seleccionan con el número 0. El número 1 selecciona sprites de 8×8 pixels ampliados al doble. El 2 selecciona sprites de 16×16. El 3 selecciona sprites de 16×16 ampliados al doble.

El método de definición de sprites en la retícula de 16×16 puntos está ilustrado por este programa:

```
10 SCREEN 2,2
20 FOR X=1 TO 32
30 READ A$
40 RESTORE
50 S$=S$+CHR$(VAL("&B"+A$))
60 SPRITE$(0)=S$
70 PUT SPRITE 0,(128,96),1,0
80 NEXT X
90 GOTO 90
100 DATA 11111111
```

La definición se realiza en el orden siguiente: primero un bloque de 8 columnas y 16 filas, y luego un bloque igual a su derecha. Por lo tanto, para definir un sprite de 16×16 se requiere un total de 32 datos. (En el ejemplo anterior lo que hacemos es leer 32 veces el mismo dato.)

El siguiente programa demuestra la utilización del joystick para mover un sprite por la pantalla y para «disparar» un segundo sprite. Si usted no dispone de joystick, ponga un 1 en lugar del 0 en las sentencias **STICK** y **STRIG** de las líneas 170 y 180; utilice las teclas de movimiento del cursor para desplazar el sprite y la barra espaciadora para disparar.

```
 10 COLOR 15,1,1
 20 SCREEN 2
 30 REM Esta sección forma los sprites
 40 FOR T=1 TO 8
 50 READ A$
 60 S$=S$+CHR$(VAL("&B"+A$))
 70 NEXT T
 80 SPRITE$(1)=S$
 90 FOR T=1 TO 8
100 READ B$
110 U$=U$+CHR$(VAL("&B"+B$))
120 NEXT T
130 SPRITE$(2)=U$
140 REM Esta sección define la posición
    inicial del sprite
150 X=128:Y=96
160 Put SPRITE 1,(X,Y),9,1
170 D=STICK(1)
180 F=STICK(1)
185 PRINTF,D
190 REM Esta sección mueve el sprite en
    función del estado del joystick
200 IF F<>0 THEN GOSUB 460
210 IF D=1 THEN X=X   :Y=Y-1
220 IF D=2 THEN X=X+1:Y=Y-1
230 IF D=3 THEN X=X+1:Y=Y
240 IF D=4 THEN X=X+1:Y=Y+1
250 IF D=5 THEN X=X   :Y=Y+1
260 IF D=6 THEN X=X-1:Y=Y+1
270 IF D=7 THEN X=X-1:Y=Y
280 IF D=8 THEN X=X-1:Y=Y-1
290 GOTO 160
300 DATA 00111100
310 DATA 01000010
320 DATA 10000001
330 DATA 11111111
```

```
340 DATA 01000010
350 DATA 10000001
360 DATA 10000001
370 DATA 10000001
380 DATA 00010000
390 DATA 00101000
400 DATA 00101000
410 DATA 00111000
420 DATA 00000000
430 DATA 00000000
440 DATA 00000000
450 DATA 00000000
460 FOR I=Y-3 TO-20 STEP-2
470 PUT SPRITE0,(X,I),9,2
480 NEXT I
490 RETURN
```

LOCATE

La orden LOCATE sirve para colocar el cursor en la pantalla:

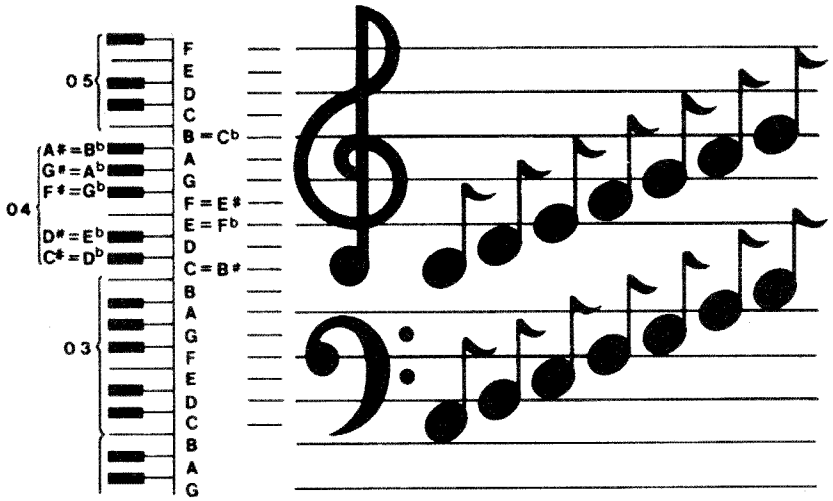
```
10 CLS
20 LOCATE 10,18: PRINT"ejemplo de
   LOCATE"
30 LOCATE 10,17:PRINT"Esto es un"
```

Con esto termina nuestra introducción a los recursos gráficos del BASIC del **SVI•728**. En la sección siguiente vamos a explorar los recursos sonoros.

Sección 2 Sonido

Dentro del **SVI•728** hay un potente sintetizador de música que podemos programar con sencillas instrucciones de BASIC. Además, el funcionamiento del sintetizador es independiente del microprocesador principal. Esto implica que el sintonizador puede estar generando sonidos al mismo tiempo que la pantalla, la impresora o cualquier otro periférico está haciendo su trabajo.

La figura siguiente muestra la escala musical que puede generar el **SVI•728.**



Nombres españoles de las notas y equivalencia:

DO	C
RE	D
MI	E
FA	F
SOL	G
LA	A
SI	B

PLAY

La orden fundamental del sistema de sonido es **PLAY**. Para empezar, dé la siguiente orden en modo inmediato:

```
PLAY "CDE"
```

(y pulse ENTER). El **SVI•728** acaba de interpretar las notas DO, RE y MI. Si no las ha oído, suba el volumen del monitor o televisor.

Copie las siguientes líneas:

```
10 PLAY "CDE"
20 GOTO 10
```

En la páginas siguientes iremos modificando este programa para presentarle las numerosas opciones de la orden **PLAY**.

O (OCTAVA)

Para empezar, modifique la línea 10:

```
10 PLAY "O1CDE"
```

Cuando ejecute el programa observará que las notas son ahora mucho más graves que antes. Esto se debe a que hemos especificado la octava 1 (**O1**), que es la más baja de las 8 disponibles en el **SVI•728**. Si no se especifica ninguna octava, el ordenador toma por defecto la número 4. Añada la siguiente línea:

```
11 PLAY "O4CDE"
```

El programa genera ahora tres notas graves seguidas de tres más agudas. La octava más alta disponible se obtiene con **O8**.

T (TEMPO)

Cambie la línea 10:

```
10 PLAY "T32O1CDE"
```

El programa interpreta ahora las mismas notas, pero a velocidad mucho más lenta. La cláusula **T32** ha especificado ese tempo. Los valores posibles van desde 32 (lento) hasta 255 (rápido).

Observe que también las notas generadas por la línea 11 son más lentas. Una vez establecido un tempo, queda en vigor mientras no se especifique otro distinto. Cambie la línea 11:

```
11 PLAY "T255O4CDE"
```

Las notas de la línea 11 son ahora mucho más rápidas que las de la línea 10.

L (DURACIÓN)

La cláusula **L** (de *length*, 'longitud') controla la duración de las notas. Compruébelo cambiando la línea 10:

```
10 PLAY "T255O1CDL1E"
```

Hemos hecho la nota E mucho más larga que las otras dos. Este cambio ha afectado también a las notas de la línea 11. Los valores especificables tras **L** van desde 1 (máxima duración) hasta 64 (mínima).

Hay otras dos cláusulas, **S** y **M**, que determinan las cualidades tonales de las notas, es decir, controlan la *envolvente* del sonido. Una misma nota suena de forma diferente en los diversos instrumentos musicales; lo que caracteriza los instrumentos es el *timbre*. **S** y **M** nos permiten generar notas con diferentes timbres.

S (FORMA DE LA ENVOLVENTE)

Mediante la cláusula **S** podemos seleccionar la forma de la variación del volumen de la nota con respecto al tiempo, es decir, la envolvente de volumen.

En el **SVI•728** disponemos de 8 envolventes distintas; sus formas y números de referencia son los que muestra la figura siguiente.

Como ejemplo, cambie la línea 10:

```
10 PLAY "S1O4CDE"
```

y suprima la línea 11. Ejecute el programa y observe el sonido.

M (MODULACIÓN)

M especifica el periodo de la envolvente, es decir, cuánto tiempo va a estar sonando la envolvente elegida. Pruebe la siguiente versión de la línea 10:

```
10 PLAY "S10M5000O4CDE"
```

Está claro que esta cláusula ha cambiado drásticamente el sonido. **M** puede ir seguida de cualquier valor comprendido entre 1 y 65535, aunque los valores más útiles están entre 100 y 2000.

R (SILENCIO)

Mediante la cláusula **R** podemos intercalar silencios entre las notas. Cambie la línea 10:

```
10 PLAY "O4CR1DR10E"
```

Hemos intercalado un silencio entre la primera nota y la segunda, y un silencio de mayor duración entre las dos últimas. La duración de los silencios se especifica de la misma manera que la de las notas (**L**).

V (VOLUMEN)

La última cláusula que vamos a estudiar aquí es la que controla la intensidad de las notas: **V**. Cambie la línea 10:

```
10 PLAY "O4V5CV10DV15E"
```

En este ejemplo cada nota va precedida de una cláusula de volumen. El parámetro de **V** puede estar entre 0 (silencio) y 15 (intensidad máxima).

ARMONÍA

En todos los ejemplos que hemos visto hasta ahora no ha sonado más que una nota al mismo tiempo. Sin embargo, el **SVI•728** dispone de tres canales de sonido independientes, que podemos combinar para producir acordes (varias notas simultáneas). Pruebe el programa con esta versión de la línea 10:

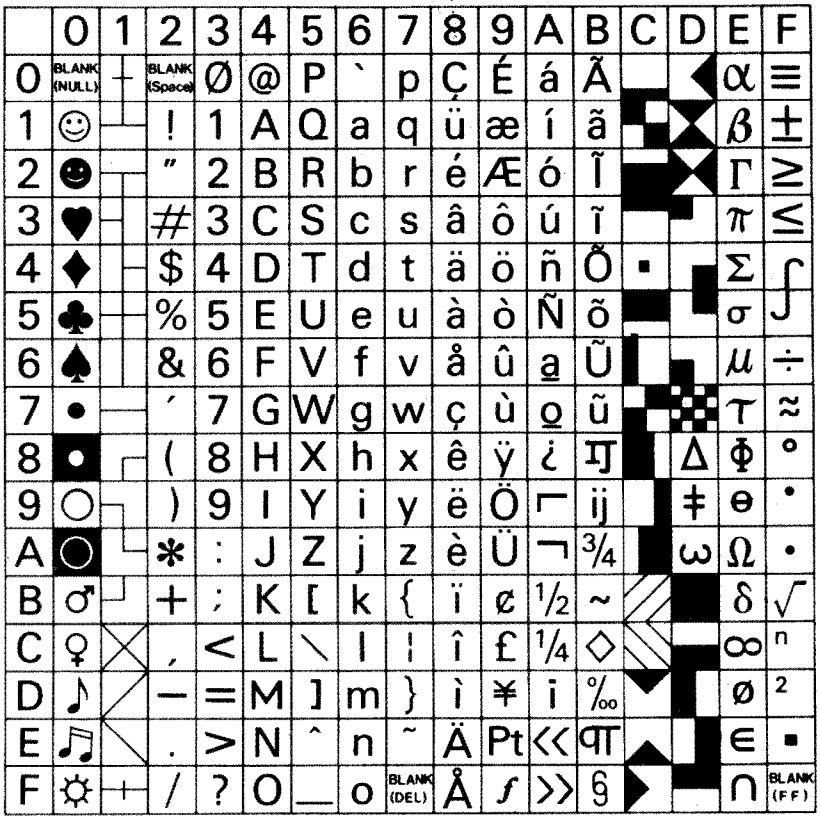
```
10 PLAY "O1CDE","O3EFC","O5GAB"
```

Las tres notas suenan al mismo tiempo, una en cada canal, produciendo un acorde. También es posible, por ejemplo, programar una melodía en un canal y un acompañamiento en los otros.

Hay otras formas de controlar el generador de sonido del **SVI•728**. Están descritas en el Apéndice H.

Apéndice A

Juego de caracteres y códigos ASCII



Apéndice B

Funciones matemáticas

FUNCIONES DERIVADAS

Las funciones matemáticas no disponibles directamente en MSX BASIC pueden ser obtenidas mediante las siguientes fórmulas:

Función	Expresión en MSX BASIC
Secante	= 1/COS(X)
Cosecante	= 1/SIN(X)
Cotangente	= 1/TAN(X)
Arco seno	= ATN (X/SQR(−X*X + 1))
Arco coseno	= −ATN (X/SQR(−X*X+1)) + 1.5708
Arco secante	= ATN (1/SQR (X*X − 1) + (SGN(X) − 1)* 1.5708
Arco cosecante	= ATN (X/SQR (X*X − 1) + (SGN(X) − 1)* 1.5708
Arco cotangente	= −ATN(X) + 1.5708
Seno hiperbólico	= (EXP(X) − EXP (−X))/2
Coseno hiperbólico	= (EXP(X) + EXP(−X))/2
Tangente hiperbólica	= (EXP(X) − EXP(−X))/(EXP(X) + EXP(−X))
Secante hiperbólica	= 2/(EXP(X) + EXP (−X))
Cosecante hiperbólica	= 2/(EXP(X) − EXP(−X))
Cotangente hiperbólica	= (EXP(X) + EXP(−X))/(EXP(X) − EXP(−X))
Arco seno hiperbólico	= LOG(X + SQR(X*X + 1))
Arco coseno hiperbólico	= LOG(X + SQR(X*X − 1))
Arco tangente hiperbólica	= LOG ((1 + X)/(1 − X))/2
Arco secante hiperbólica	= LOG ((SQR(−X*X + 1) + 1)/X)
Arco cosecante hiperbólica	= LOG ((SGN (X)*SQR (X*X + 1) + 1)/X)
Arco cotangente hiperbólica	= LOG ((X + 1)/(X − 1))/2

Apéndice C

Códigos y mensajes de error

Código	Mensaje de error
1	**NEXT without FOR** Se ha encontrado un **NEXT** cuya variable no corresponde con la de ningún **FOR**.
2	**Syntax error** BASIC no puede entender una línea porque algo en ella no se ajusta a las reglas de sintaxis del lenguaje.
3	**RETURN without GOSUB** Se ha encontrado un **RETURN** sin que antes se haya ejecutado el **GOSUB** correspondiente.
4	**Out of DATA** Una orden **READ** ha intentado leer más datos que los incluidos en líneas **DATA**.
5	**Illegal function call** Se ha entregado a una función numérica o literal un argumento que está fuera de los márgenes permitidos, o bien: 1. Un subíndice es negativo o demasiado grande. 2. El argumento de **LOG** es negativo o nulo. 3. El argumento de **SQR** es negativo. 4. Se ha puesto un argumento incorrecto en alguna de las siguientes sentencias: **MID$, LEFT$, RIGHT$, INP, OUT, PEEK, POKE, TAB, SPC, STRING$, SPACE$, INSTR$, ON . . . GOTO**.
6	**Overflow** El resultado de un cálculo es tan grande que no puede ser representado en los formatos numéricos de BASIC.

Código	Mensaje de error
7	**Out of memory** El programa actual es demasiado grande, o sus variables ocupan demasiado espacio en la memoria, o tiene demasiados ficheros abiertos, o las expresiones son demasiado complejas, o el anidamiento de las estructuras de control (**FOR, GOSUB**) es demasiado profundo.
8	**Undefined line number** Se ha mencionado un número de línea que no existe en el programa.
9	**Subscript out of range** Se ha tratado de hacer referencia a un elemento de una matriz con un índice o grupo de índices incompatible con la estructura de la matriz.
10	**Redimensioned array** Se ha tratado de dimensionar con **DIM** una matriz que ya ha sido dimensionada antes, bien explícitamente (con un **DIM** anterior), bien implícitamente (las matrices se dimensionan implícitamente por el mero hecho de usarlas; el margen de los índices va entonces de 0 a 10).
11	**Division by cero** División por cero, que puede ocurrir en la división de números reales, en la división entera, en la operación **MOD** y al tratar de elevar 0 a una potencia negativa.
12	**Illegal direct command** Se ha intentado dar como orden directa una orden que sólo es válida si va precedida de número de línea.
13	**Type mismatch** Incongruencia de tipos. Un dato es literal cuando tenía que ser numérico, o vice versa.
14	**Out of string space** Se han creado demasiadas cadenas literales y ya no queda espacio para más.

Código	Mensaje de error
15	**String too long** Se ha intentado crear una cadena de más de 255 caracteres, quizá por concatenación de otras.
16	**String formula too complex** Expresión literal demasiado compleja. Se la puede descomponer en varias expresiones más sencillas.
17	**Can't continue** No se puede reanudar la ejecución del programa con **CONT**. Esto ocurre cuando: 1. El programa se ha detenido como consecuencia de un error. 2. El programa ha sido modificado durante la interrupción. 3. No existe ningún programa.
18	**Undefined user function** Se ha invocado una **FN** que no ha sido definida con **DEF FN**.
19	**Device I/O error** Error de entrada/salida en el funcionamiento del cassette, unidad de disco, impresora o consola. Es un error fatal en el sentido de que es irrecuperable para BASIC.
20	**Verify error** El programa residente en la memoria es distinto del grabado en la cinta.
21	**No RESUME** Se ha llegado al final del programa mientras se estaba ejecutando una rutina de gestión de errores.
22	**Resume without error** **RESUME** sólo es válida dentro de una rutina de gestión de errores.
23	**Unprintable error** Se ha producido un error para el que BASIC no dispone de mensaje aclaratorio.

Código	Mensaje de error
24	**Missing operand** BASIC ha encontrado una operación en la que falta un operando.
25	**Line buffer overflow** Línea de programa demasiado larga.
26-49	Errores sin mensaje aclaratorio. Estos números están reservados para futuras ampliaciones de BASIC.

Errores relacionados con el sistema de disco

Codigo	Mensaje de error
50	**FIELD overflow** Una sentencia **FIELD** ha tratado de asignar más bytes que los especificados en la orden **OPEN** como longitud de registro de un fichero de acceso arbitario. O bien se ha alcanzado el final del tampón de registros al utilizar en modo secuencial (con **PRINT#** o **INPUT#**) un fichero de acceso arbitrario.
51	**Internal error** Error interno que no debería ocurrir nunca. En el caso improbable de que usted llegue a observarlo, le rogamos que informe a Microsoft de las condiciones en que se ha producido.
52	**Bad file number** Se ha especificado un número de fichero que está fuera del margen permitido (establecido con **MAXFILES**), o se ha intentado leer o escribir en un fichero que no está abierto.
53	**File not found** En una sentencia **OPEN, LOAD** o **KILL** se ha hecho referencia a un fichero que no existe.

Código	Mensaje de error
54	**File already open** Se ha intentado abrir en modo de salida secuencial un fichero que ya está abierto.
55	**Input past end** Al leer un fichero se ha intentado sobrepasar su final, o bien se ha intentado leer un fichero vacío. Para evitar este error se puede detectar el final del fichero con la función **EOF**.
56	**Bad file name** En una sentencia **LOAD**, **SAVE**, **KILL**, **NAME**, etc. se ha utilizado un nombre de fichero mal construido.
57	**Direct statement in file** Se ha encontrado una línea no numerada al cargar un programa que había sido grabado en modo ASCII. Como consecuencia del error se abandona el proceso de carga.
58	**Sequential I/O only** Se ha tratado de acceder en modo arbitrario a un fichero que ha sido abierto para acceso secuencial.
59	**File not OPEN** El fichero especificado en **PRINT#**, **INPUT#**, etc. no ha sido abierto previamente.
60-255	Errores sin mensaje aclaratorio. El usuario puede crear sus propios mensajes y asignarlos a los números altos de este margen.

Apéndice D

Palabras clave de MSX BASIC

Los nombres de las sentencias y funciones de BASIC son palabras reservadas; es decir, no pueden ser utilizadas como nombres de variables. A continuación damos la lista de palabras reservadas para MSX BASIC. Si se intenta usar alguna de estas palabras como nombre de variable, el ordenador responde con un mensaje de error.

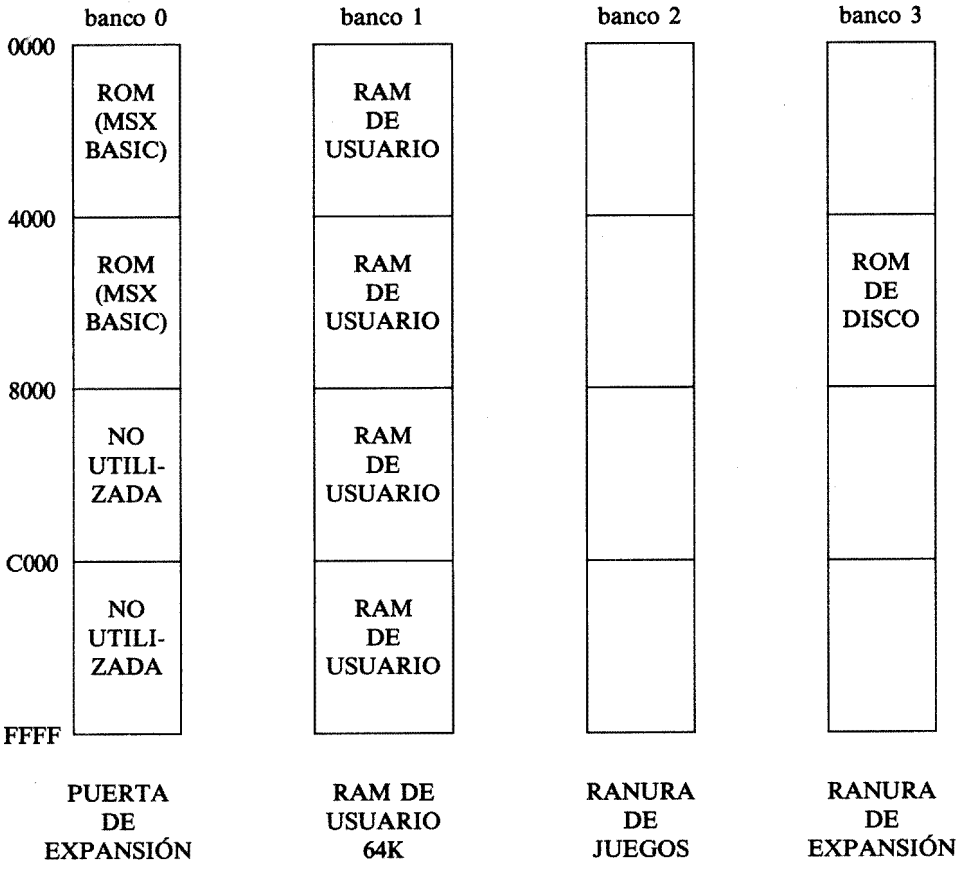
ABS	END	LINE	POKE
ASC	EOF	LINE INPUT	POS
ATN	ERASE	LINE INPUT#	PRESET
AUTO	ERL	LIST	PRINT
BASE	ERR	LLIST	PRINT USING
BEEP	ERROR	LOAD	PRINT#
BIN$	EXP	LOCATE	PRINT# USING
BLOAD	FIX	LOG	PSET
BSAVE	FOR	LPOS	PUT SPRITE
CALL	GOSUB	LPRINT	READ
CDBL	GOSUB	LPRINT USING	REM
CHR$	GOTO	MAXFILES	RENUM
CINT	HEX$	MERGE	RESTORE
CIRCLE	IF GOTO	MID$	RESUME NEXT
CLEAR	IF THEN	MOTOR ON	RESUME O
CLOAD	INKEY$	MOTOR OFF	RETURN
CLOAD?	INP	NEW	RIGHT$
CLOSE	INPUT	NEXT	RND
CLS	INPUT$	OCT$	RUN
COLOR	INPUT#	ON ERROR GOTO	SAVE
CONT	INSTR	ON GOSUB	SCREEN
COS	INT	ON GOTO	SGN
CSAVE	INTERVAL ON	ON INTERVAL GOSUB	SIN
CSNG	INTERVAL OFF	ON KEY GOSUB	SOUND
CSRLIN	INTERVAL STOP	ON SPRITE GOSUB	SPACE$
DATA	KEY	ON STOP GOSUB	SPC
DEF FN	KEY LIST	ON STRIG GOSUB	SPRITE ON
DEFINT	KEY (n) OFF	OPEN	SPRITE OFF
DEFDBL	KEY (n) OFF	OUT	SPRITE STOP
DEFSNG	KEY (n) STOP	PAD	SPRITE$
DEFSTR	KEY ON	PAINT	SQR
DEFUSR	KEY OFF	PDL	STR$
DELETE	LEFT$	PEEK	STICK
DIM	LEN	PLAY	STOP
DRAW	LET	POINT	STOP ON

STOP OFF	STRIG STOP	TIME	VARPTR
STOP STOP	STRING$	TROFF	VDP
STRIG	SWAP	TRON	VPEEK
STRIG ON	TAB	USR	VPOKE
STRIG OFF	TAN	VAL	WIDTH
			WAIT

Apéndice E

Mapa de memoria y descripción de los conectores

MAPA DE MEMORIA



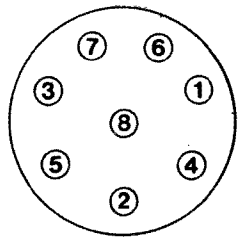
Patillas del conector de expansión

Patilla	Nombre	E/S*	Descripción
1	$\overline{CS1}$	S	Selección de ROM 4000-7FFF
2	$\overline{CS2}$	S	Selección de ROM 8000-BFFF
3	$\overline{CS12}$	S	Selección de ROM 4000-BFFF
4	$\overline{SLTSL}$	S	Señal de selección de bancos. Señal fija para cada canal.
5			Reservada para su uso futuro
6	$\overline{RFSH}$	S	Señal de regeneración de memoria
7	$\overline{WAIT}$	E	Señal de espera hacia la CPU
8	$\overline{INT}$	E	Señal de petición de interrupción
9	$\overline{M1}$	S	Señal de ciclo de lectura de la CPU
10	$\overline{BUSDIR}$	E	Esta señal controla el sentido del bus de datos cuando está seleccionada la tarjeta. Se pone a nivel bajo cuando los datos son emitidos por la tarjeta.
11	$\overline{IORQ}$	S	Señal de petición de entrada/salida
12	$\overline{MERQ}$	S	Señal de petición de memoria
13	$\overline{WR}$	S	Señal de escritura
14	$\overline{RD}$	S	Señal de lectura
15	$\overline{RESET}$	S	Señal de reinicialización del sistema
16			Reservada para su uso futuro
17	A9	S	
18	A15	S	
19	A11	S	
20	A10	S	
21	A7	S	
22	A6	S	
23	A12	S	
24	A8	S	A0–A15: bus de direcciones
25	A14	S	
26	A13	S	
27	A1	S	
28	A0	S	
29	A3	S	
30	A2	S	
31	A5	S	
32	A4	S	
33	D1	E/S	
34	D0	E/S	
35	D3	E/S	
36	D2	E/S	D0–D7: bus de datos
37	D5	E/S	
38	D4	E/S	
39	D7	E/S	
40	D6	E/S	
41	GND		Masa
42	CLOCK		Reloj de la CPU, 3.579 MHz

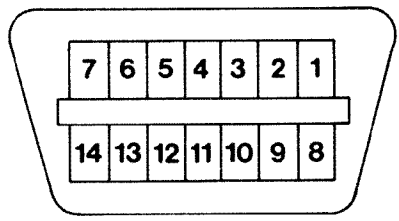
Patilla	Nombre	E/S*	Descripción
43	GND		Masa
44	SW1		Protección
45	+5V		Alimentación a +5 V
46	SW2		Protección
47	+5V		Alimentación a +5 V
48	+12V		Alimentación a +12 V
49	SUNDIN	E	Entrada de sonido (−5 dbm)
50	−12V		Alimentación a −12 V

* Entrada/salida referida al **SVI•728.**

Conector de cinta

Patilla	Nombre	E/S	Distribución de las patillas
1	GND	—	
2	GND	—	
3	GND	—	
4	CMTOUT	S	
5	CMTIN	E	
6	REM +	S	
7	REM −	S	
8	GND	—	

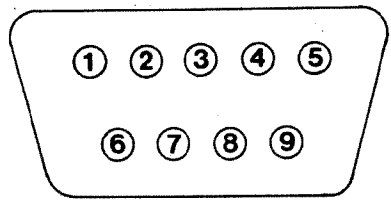
Conector de impresora

Patilla	Nombre	E/S	Distribución de las patillas
1	PSTB	S	
2	PDBϕ	S	
3	PDB1	S	
4	PDB2	S	
5	PDB3	S	
6	PDB4	S	
7	PDB5	S	
8	PDB6	S	
9	PDB7	S	
10	NC	—	
11	BUSY	E	
12	NC	—	
13	NC	—	
14	GND	—	

Teclado

Patilla	Nombre	E/S	Patilla	Nombre	E/S
1	INPUT 7	E	2	INPUT 6	E
3	INPUT 5	E	4	INPUT 4	E
5	INPUT 3	E	6	INPUT 2	E
7	INPUT 1	E	8	INPUT 0	E
9	OUTPUT 9	S	10	OUTPUT 8	S
11	OUTPUT 7	S	12	OUTPUT 6	S
13	OUTPUT 5	S	14	OUTPUT 4	S
15	OUTPUT 3	S	16	OUTPUT 2	S
17	OUTPUT 1	S	18	OUTPUT 0	S
19	CAPS	S	20	+5V	—
21	GND	—			

Conector de joystick

Patilla	Nombre	E/S	Distribución de las patillas
1	FORWARD/SCK	E	
2	BACKWARD/SI	E	
3	LEFT/CS	E	
4	RIGHT/SENCTR	E	
5	+5V	—	
6	TRIGGER 1	E/S	
7	TRIGGER 2	E/S	
8	OUTPUT	S	
9	GND	—	

Apéndice F

Notas acerca de las pantallas

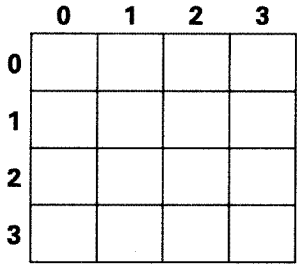
PANTALLAS DE TEXTO

La orden **SCREEN 0** activa la pantalla de texto de 37 caracteres por línea. **SCREEN 1** activa la de 29 caracteres por línea. En ambos casos el número de líneas es 24. Cuando se enciende el ordenador, éste selecciona automáticamente la pantalla número 0. Con la orden **WIDTH x** se puede modificar la anchura de ambas pantallas. En la pantalla 0 los valores posibles de **x** van de 1 a 40; en la pantalla 1, de 1 a 32.

ALTA Y BAJA RESOLUCIÓN

La orden **SCREEN 3** activa la pantalla multicolor de baja resolución. Esta pantalla se puede considerar dividida en 64 columnas y 48 filas, según se muestra en las figuras 2 y 3 del capítulo 5. La orden **SCREEN 2** activa la pantalla de alta resolución: 256 columnas y 192 filas. Por razones de compatibilidad entre las dos pantallas, el sistema de coordenadas es el mismo en ambas. Por consiguiente, los números de columna y fila de las figuras mencionadas deben ser multiplicados por 4 para convertirlos en coordenadas.

La distribución es tal que un pixel de la pantalla 3 se divide en 16 pixels en la pantalla 2:



En la pantalla de alta resolución (2) cada pixel puede ser iluminado o apagado individualmente. En cambio, en la pantalla de baja resolución (3), al iluminar un pixel pequeño se encienden también los 15 adyacentes (es decir, todo el pixel grande).

Apéndice G

Anomalías de funcionamiento

Síntoma	Causa posible	Remedio
No se enciende el piloto de alimentación	Interruptor de alimentación apagado	Encienda el interruptor (está en el panel derecho)
	Cable de alimentación no conectado	Compruebe que están bien conectados los dos cables del adaptador de c.a.
	Fusible fundido	Haga que un distribuidor autorizado lo cambie
No hay sonido ni imagen	Canal incorrecto o mal sintonizado	Sintonice en UHF
	Cable de video mal encajado	Compruebe la conexión del cable en ambos extremos
No hay sonido	Volumen demasiado bajo	Ajuste el volumen en el televisor
Baja calidad de la imagen o ausencia de color	Niveles de color y contraste mal ajustados	Ajuste los mandos de color y contraste del televisor

Apéndice H

Generador programable de sonido (PSG)

Además de la orden **PLAY**, con la que se puede hacer que el **SVI•728** interprete notas musicales, disponemos también de la orden **SOUND** para controlar más directamente el generador de sonido. La sintaxis de **SOUND** es la siguiente:

SOUND ⟨*registro del PSG*⟩,⟨*valor*⟩

donde ⟨*registro del PSG*⟩ es el número de uno de los 13 registros del generador y ⟨*valor*⟩ es un número del margen de 0 a 255. La misión de los registros es la que se indica en la siguiente tabla:

Control de	Registro n.º	Efecto
Generador de tono	0	Canal 1. Frecuencia, ajuste fino. 0–255.
	1	Canal 1. Frecuencia, ajuste aproximado. 0–15
	2	Canal 2. Frecuencia, ajuste fino. 0–255.
	3	Canal 2. Frecuencia, ajuste aproximado. 0–15.
	4	Canal 3. Frecuencia, ajuste fino. 0–255
	5	Canal 3. Frecuencia, ajuste aproximado. 0–15.
Generador de ruido	6	Frecuencia predominante en el ruido (periodo de ruido). 0–31.
Selección de canales	7	Selección de canales para tonos y ruido
Amplitud	8	Canal 1. Amplitud. 0–15 (16=envolventes)
	9	Canal 2. Amplitud. 0–15 (16=envolventes)
	10	Canal 3. Amplitud. 0–15 (16=envolventes)
Generador de envolventes	11	Periodo de las envolventes, ajuste fino. 0–255
	12	Periodo de las envolventes, ajuste aproximado. 0–15
	13	Forma de las envolventes

CONTROL DEL GENERADOR DE TONO

El PSG tiene tres canales de tono: A, B y C. La frecuencia está relacionada con el contenido de los registros mediante las siguientes fórmulas:

```
Número = 3579545/(16*Frecuencia)
Bajo = Número AND 255
Alto = Número/256
```

Canal	Reg. bajo	Reg. alto
A	0	1
B	2	3
C	4	5

Ejemplo:

```
10 INPUT "Frecuencia: ";F
20 N=3579545#/(16*F)
30 A=N/256
40 B=N AND 255
50 SOUND 0,B
60 SOUND 1,A
70 SOUND 8,15:PRINT"Control de
   volumen del canal A"
80 SOUND 7,254:PRINT"11111110 binario
   para activar canal A"
90 END
```

CONTROL DE AMPLITUD

En el ejemplo anterior hemos utilizado el registro 8 para controlar el volumen del canal A. Con los registros 9 y 10 se controla el volumen de los otros dos canales. En cada canal se puede elegir un nivel comprendido entre 0 (mínimo) y 15 (máximo).

Ejemplo:

```
10 SOUND 0,100
20 SOUND 1,0
30 SOUND 7,254:REM Activar canal A
40 FOR N=15 TO 0 STEP -1
50 SOUND 8,N
60 FOR M=1 TO 200:NEXT M:REM Retardo
70 NEXT N
```

Este programa genera por el canal A un tono de volumen decreciente (de 15 a 0).

Poniendo en el registro de control de amplitud (8, 9 o 10) el número 16 se entrega el control de la amplitud de ese canal a una envolvente, cuyas características dependerán del contenido de los registros de control de envolventes (11–13).

SELECCIÓN DE CANALES

El registro 7 realiza la selección de canales de tono y ruido:

Bit n.°:	7 6	5 4 3	2 1 0
Ruido/tono		Ruido	Tono
Canal:	— —	C B A	C B A
Bit a 1: canal desactivado. Bit a 0: canal activado.			

Por ejemplo,

```
SOUND 7,&B11111110
```

activa el tono en el canal A, mientras que

```
SOUND 7,&B11110110
```

activa tanto el ruido como el tono en el canal A.

CONTROL DE ENVOLVENTES

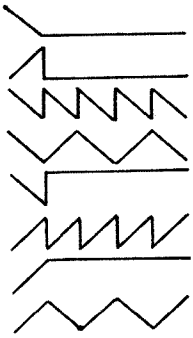
Los registros 11 y 12 funcionan como un solo registro de 16 bits, mediante el cual se puede controlar la frecuencia de las envolventes. Las siguientes fórmulas dan la relación entre la frecuencia y el contenido de estos dos registros:

```
Número = 3479545/(256*frecuencia)
Bajo = Número AND 255
Alto = Número/256
```

Ejemplo:

```
 10 SOUND 0,100
 20 SOUND 1,0:REM Canal A
 30 SOUND 7,&B11111110:REM Activar A
 40 SOUND 8,16:REM Valor 16 para activar
    registro E/P
 50 SOUND 13,14:REM Selección de forma
 60 F=.5:REM 0.5 Hz
 70 N=3579545#
 80 B=N/(256*S) AND 255
 90 A=N/(256*S)/256
100 SOUND 11,B
110 SOUND 12,A
120 END
```

El registro 13 permite seleccionar las diversas formas de envolvente disponibles:

Contenido del registro 13	Forma de la envolvente
0, 1, 2, 3 o 9	
4, 5, 6, 7 o 15	
8	
10	
11	
12	
13	
14	

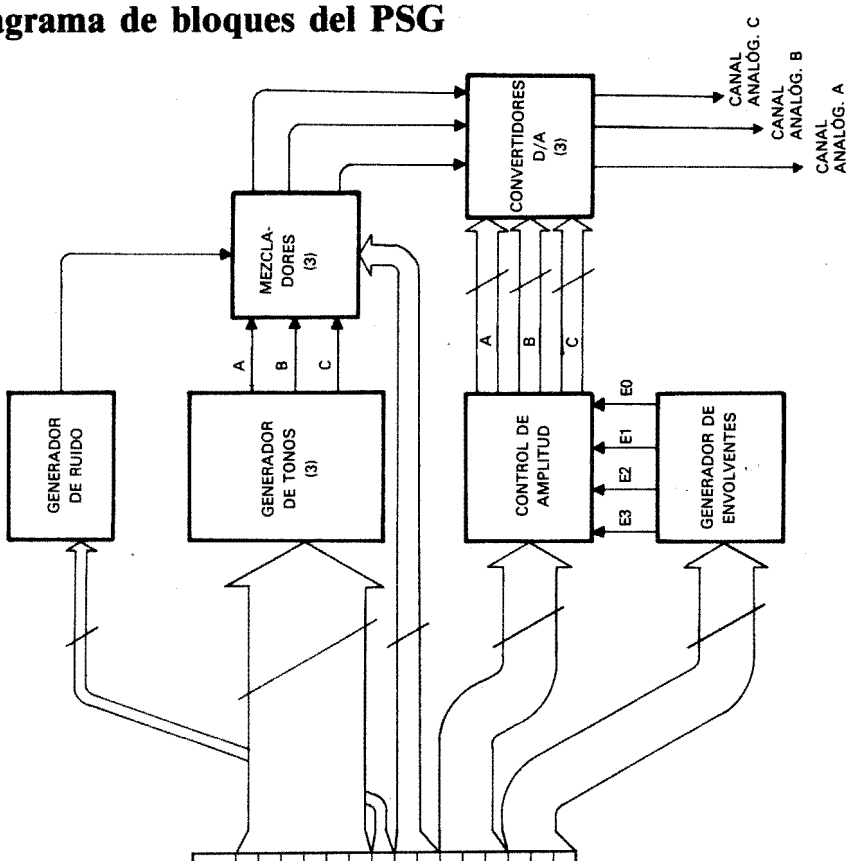
→ ← Periodo de la envolvente

Por ejemplo,

SOUND 13,14

crea en el canal A un tono modulado en amplitud creciente y decreciente, a condición de que en el registro 8 se haya puesto el valor 16 (**SOUND 8,16**); el periodo de la modulación depende de los valores depositados en los registros 11 y 12.

Diagrama de bloques del PSG



Matriz de registros (14 registros de lectura/escritura)

Registro	Bit	7	6	5	4	3	2	1	0
R0	Periodo tono, canal A	Ajuste fino en 8 bits							
R1		Ajuste aprox. en 4 bits							
R2	Periodo tono, canal B	Ajuste fino en 8 bits							
R3		Ajuste aprox. en 4 bits							
R4	Periodo tono, canal C	Ajuste fino en 8 bits							
R5		Ajuste aprox. en 4 bits							
R6	Periodo ruido	Ajuste en 5 bits							
R7	Activar	E/S		Ruido			Tono		
		E/S B	E/S A	C	B	A	C	B	A
R8	Amplitud, canal A			M	L3	L2	L1	L0	
R9	Amplitud, canal B			M	L3	L2	L1	L0	
R10	Amplitud, canal C			M	L3	L2	L1	L0	
R11	Periodo de envolventes	Ajuste fino en 8 bits							
R12		Ajuste aprox. en 8 bits							
R13	Formas de envolventes				CONT	ATAQ	ALT	SOST	

Apéndice I

Demostración de BASIC

Para manejar con eficacia el **SVI•728** hay que conocer gran número de sentencias y técnicas de programación. Sin embargo, si es usted como casi todos los usuarios, estará impaciente por ver el **SVI•728** en acción ya, sin esperar a recorrer todas las fases del aprendizaje. Por eso hemos nos disponemos a ofrecerle unos cuantos ejemplos que le darán una idea de lo que puede esperar del ordenador y le abrirán el apetito para seguir leyendo este manual.

Damos por supuesto que usted ha leído los dos primeros capítulos y tiene el equipo correctamente instalado. Encienda primero el televisor o monitor, después el ordenador, y espere hasta que aparezca el mensaje de presentación de BASIC, con la indicación del espacio de memoria libre y el mensaje **Ok**. Este último indica que el **SVI•728** está esperando sus instrucciones.

Empezaremos por la orden de BASIC que controla el color de la pantalla y de los caracteres que aparecen en ella.

COLOR

La orden **COLOR** nos permite elegir los colores de los diversos elementos de la pantalla. Hay un total de 16 colores, numerados del 0 al 15. Por ejemplo, escriba:

```
COLOR 6,1
```

y luego pulse [ENTER]. Verá que la pantalla cambia de su color inicial (letras blancas sobre fondo azul) a letras naranja sobre fondo verde. Experimente con otros colores escribiendo diferentes números tras la palabra **COLOR**. Cuando haya terminado, apague y vuelva a encender el ordenador. De esta manera se «inicializa» completamente la máquina y, por consiguiente, se vuelve a los colores normales.

Hay otras formas menos drásticas de hacer lo mismo, según explicaremos más adelante en el manual.

CLS

Ahora vamos a introducir un programa que escribirá una palabra en el extremo superior izquierdo de la pantalla. La primera línea es la siguiente:

```
10 CLS
```

Escriba **RUN** y pulse la tecla [ENTER]. La pantalla se borra y aparece el mensaje **Ok** en la línea 1, con el cursor (rectángulo blanco) debajo. La orden **CLS** (abreviatura de *clear screen*) borra la pantalla siempre que el ordenador la encuentra en un programa.

Introduzca esta línea:

```
20 PRINT "SPECTRAVIDEO"
```

Escriba **RUN** y pulse [ENTER]. El ordenador ha escrito la palabra 'SPECTRAVIDEO' en la primera línea, el mensaje **Ok** en la segunda y el cursor en la tercera. Acaba usted de escribir un programa de dos líneas. Ahora vamos a corregirlo (en lenguaje informático, a «editarlo»). Antes de nada, escriba **LIST** y pulse [ENTER]. Esta orden hace que BASIC escriba en la pantalla el «listado» (lista de instrucciones) del programa que tenemos almacenado en la memoria del ordenador. Sirviéndose de las 'teclas del cursor', lleve el cursor a la letra 'S' de 'SPECTRAVIDEO'. Pulse la tecla [INS/PASTE] y observe que la altura del cursor se ha reducido a la mitad. Esto indica que estamos en el 'modo de inserción'. Escriba su nombre:

```
20 PRINT "NOMBRE
   APELLIDOSPECTRAVIDEO"
```

El cursor sigue estando sobre la 'S' de 'SPECTRAVIDEO'. Pulse la tecla [DEL/CUT]: la 'S' ha desapa-

recido y el cursor está sobre la 'P'. Pulse la misma tecla unas cuantas veces más, hasta llegar al signo de 'cerrar comillas', y luego pulse [ENTER].

La línea modificada ha quedado almacenada en la memoria del ordenador. Compruebe que el listado del programa es el siguiente:

```
10 CLS
20 PRINT "NOMBRE APELLIDO"
```

Ejecute el programa (con **RUN**[ENTER]) y verá cómo el obediente ordenador escribe su nombre en la primera línea de la pantalla. ¡Enhorabuena! Ya ha escrito su primer programa, y además ha utilizado el 'editor de pantalla' del **SVI•728** para corregirlo.

GRÁFICOS

Un área en la que rápidamente se descubre las amplias posibilidades del **SVI•728** es la de los gráficos. Para empezar, introduzca el siguiente programa (pulsando [ENTER] al final de cada línea):

```
10 CLS
20 SCREEN 2
30 CIRCLE (128,80),60,11
40 PAINT (128,80),11
50 GOTO 50
```

Escriba **RUN**[ENTER] y verá cómo el ordenador dibuja una circunferencia con tinta amarilla y la rellena con el mismo color. Para entender cómo funciona el programa, estudiemos sus líneas una a una:

```
10 CLS
```

Como ya hemos visto antes, **CLS** borra la pantalla. Da un toque de profesionalidad poner esta línea al principio de todos los programas.

```
20 SCREEN 2
```

Esta línea activa la pantalla número de 2, que es una de las pantallas gráficas del **SVI•728**. (La pantalla con la que ha estado trabajando hasta ahora es una de las pantallas de texto, la número 1; ésta es la pantalla que el ordenador selecciona automáticamente cuando lo encendemos.)

```
30 CIRCLE (128,80),60,11
```

CIRCLE es la orden que dibuja circunferencias. Los números 128 y 80 son las coordenadas del centro de la circunferencia (columna número 128, empezando a contar por la izquierda, y fila número 80, contando desde arriba).

El número 60 es el radio de la circunferencia. El 11 especifica el color (amarillo).

```
40 PAINT (128,80),11
```

Esta orden hace que el ordenador tome un pincel, lo sitúe en el punto (128,80) y empiece a pintar con el color 11 hasta que encuentre un límite de ese mismo color.

```
50 GOTO 50
```

Para que no desaparezca la pantalla gráfica inmediatamente, hemos incluido la línea 50, de la cual el programa no puede salir mientras no lo interrumpamos pulsando [CTRL][STOP]. Al hacer esto, volvemos a la pantalla de texto, la cual está vacía y con el cursor en el extremo superior izquierdo.

Experimente con este programa: cambie la posición, el radio y el color del círculo.

De los dos números que forman las coordenadas del centro de la circunferencia, el primero (coordenada *X*) puede estar entre 0 y 255. El segundo (coordenada *Y*) puede estar entre 0 y 191.

RECTAS Y RECTÁNGULOS

Dibujar una recta en el **SVI•728** es muy sencillo. Basta suministrarle a la orden **LINE** las coordenadas de los extremos. Escriba **NEW**[ENTER] para borrar el programa anterior e introduzca el siguiente:

```
10 CLS
20 SCREEN 2
30 LINE (50,40)-(200,150)
40 GOTO 40
```

Cuando ejecute este programa verá cómo el ordenador dibuja una recta que desciende de izquierda a derecha. La orden responsable es la de la línea 30:

```
30 LINE (50,40)-(200,150)
```

LINE es, como decíamos, la orden de BASIC que dibuja rectas. Las dos parejas de números que figuran entre paréntesis especifican las coordenadas de los puntos extremos de la recta.

Esta misma orden, con una pequeña modificación, puede dibujar rectángulos:

```
30 LINE (50,40)-(200,150),10,B
```

El número 10 especifica el color del borde del rectángulo. La letra 'B' hace que **LINE** dibuje un rectángulo en lugar de una recta. Si además queremos que **LINE** rellene el rectángulo con el mismo color del borde, basta con que añadamos la letra **F**:

```
30 LINE (50,40)-(200,150),10,BF
```

MÚSICA

PLAY

El **SVI•728** contiene un complejo sintetizador de música y efectos sonoros que podemos utilizar a pleno rendimiento con simples órdenes de BASIC. La orden fundamental es **PLAY**. Por ejemplo,

```
PLAY "CDE"ENTER
```

hace sonar tres notas a través del altavoz del televisor o monitor. Igual resultado se obtiene incluyendo la orden **PLAY** en un programa:

```
10 PLAY "CDE"
20 GOTO 10
```

Son muy numerosos los efectos que podemos conseguir con el sintetizador del **SVI•728**. En este apéndice vamos a dar sólo unos ejemplos muy sencillos, partiendo del sencillo programa que tenemos en este momento en la memoria del ordenador.

O (OCTAVA)

Modifique la línea 10:

```
10 PLAY "O1CDE"
```

Cuando ejecute la nueva versión del programa observará que las notas son ahora de tono mucho más grave que antes. Esto se debe a que con **O1** hemos seleccionado la octava número 1. Esta cláusula nos permite elegir entre las 8 octavas disponibles (numeradas de 1 a 8). Añada esta línea:

```
11 PLAY "O4CDE"
```

Al ejecutar ahora el programa se puede oir el segundo grupo de notas mucho más agudo que el primero.

T (TEMPO)

Cambie la línea 10:

```
10 PLAY "T32O1CDE"
```

El programa interpreta ahora las mismas notas, pero a velocidad mucho más lenta. La cláusula **T32** ha especificado ese tempo. Los valores posibles van desde 32 (lento) hasta 255 (rápido).

Observe que también las notas generadas por la línea 11 son más lentas. Una vez establecido un tempo, queda en vigor mientras no se especifique otro distinto. Cambie la línea 11:

```
11 PLAY "T255O4CDE"
```

Las notas de la línea 11 son ahora mucho más rápidas que las de la línea 10.

L (DURACIÓN)

La cláusula **L** (de *length*, 'longitud') controla la duración de las notas. Compruébelo cambiando la línea 10:

```
10 PLAY "T255O1CDL1E"
```

Hemos hecho la nota E mucho más larga que las otras dos. Este cambio ha afectado también a las notas de la línea 11. Los valores especificables tras **L** van desde 1 (máxima duración) hasta 64 (mínima).

Hay otras dos cláusulas, **S** y **M**, que determinan las cualidades tonales de las notas, es decir, controlan la *envolvente* del sonido. Una misma nota suena de forma diferente en los diversos instrumentos musicales; lo que caracteriza los instrumentos es el *timbre*. **S** y **M** nos permiten generar notas con diferentes timbres.

S (FORMA DE LA ENVOLVENTE)

Mediante la cláusula **S** podemos seleccionar la forma de la variación del volumen de la nota con respecto al tiempo, es decir, la envolvente de volumen.

Como ejemplo, cambie la línea 10:

```
10 PLAY "S1O4CDE"
```

y suprima la línea 11 (escribiendo **11**[ENTER]). Ejecute el programa y observe el sonido.

M (MODULACIÓN)

M especifica el periodo de la envolvente, es decir, cuánto tiempo va a estar sonando la envolvente elegida. Pruebe la siguiente versión de la línea 10:

```
10 PLAY "S10M5000O4CDE"
```

Está claro que esta cláusula ha cambiado drásticamente el sonido. **M** puede ir seguida de cualquier valor comprendido entre 1 y 65535, aunque los valores más útiles están entre 100 y 2000.

R (SILENCIO)

Mediante la cláusula **R** podemos intercalar silencios entre las notas. Cambie la línea 10:

```
10 PLAY "O4CR1DR10E"
```

Hemos intercalado un silencio entre la primera nota y la segunda, y un silencio de mayor duración entre las dos últimas. La duración de los silencios se especifica de la misma manera que la de las notas (**L**).

V (VOLUMEN)

La última cláusula que vamos a estudiar aquí es la que controla la intensidad de las notas: **V**. Cambie la línea 10:

```
10 PLAY "O4V5CV10DV15E"
```

En este ejemplo cada nota va precedida de una cláusula de volumen. El parámetro de **V** puede estar entre 0 (silencio) y 15 (intensidad máxima).

ARMONÍA

En todos los ejemplos que hemos visto hasta ahora no ha sonado más que una nota al mismo tiempo. Sin embargo, el **SVI•728** dispone de tres canales de sonido independientes, que podemos combinar para producir acordes (varias notas simultáneas). Pruebe el programa con esta versión de la línea 10:

```
10 PLAY "O1CDE","O3EFC","O5GAB"
```

Las tres notas suenan al mismo tiempo, una en cada canal, produciendo un acorde. También es posible, por ejemplo, programar una melodía en un canal y un acompañamiento en los otros.

En este apéndice sólo hemos dado una pequeña muestra de lo que el **SVI•728** puede hacer por usted. Para empezar a conocerlo a fondo, lea los capítulos de este manual y practique con los ejemplos.

Apéndice J

Glosario y léxico inglés-español

A pesar de que la mayor parte de los términos que figuran en este glosario no aparecen en el manual, los hemos incluido para que usted pueda consultarlos cuando los necesite en el futuro.

access time **tiempo de acceso**

Tiempo empleado en buscar una palabra en la memoria.

accumulator **acumulador**

Uno de los registros del microprocesador; en él se almacena temporalmente el resultado de las operaciones.

address **dirección**

Número utilizado por la UCP para especificar una posición de memoria

ALU **UAL**

Arithmetic and Logic Unit, unidad aritmética-lógica. Es la parte de la UCP que realiza las operaciones aritméticas (suma, resta, desplazamientos) y las lógicas (AND, OR, etc.).

alphanumeric **alfanumérico**

Hace referencia a los caracteres que pueden ser letras o cifras.

architecture **arquitectura**

Estructura y organización de un sistema informático.

array **matriz**

Lista de variables lógicamente interrelacionadas que el ordenador distingue por uno o varios números de índice.

ASCII **ASCII**

American Standard Code for Information Interchange, código estadounidense estándar para el intercambio de la información. Código de 8 bits mediante el que se representa un conjunto de 128 caracteres (letras, cifras, signos de puntuación, etc.).

assembler **ensamblador**

Programa que convierte sus instrucciones, los llamados 'mnemónicos', en códigos de máquina.

BASIC **BASIC**

Beginners' All-purpose Symbolic Instruction Code, código de instrucciones simbólicas de uso general para principiantes. Lenguaje de programación de alto nivel, adecuado para principiantes.

baud **baudio**

Unidad de medida de la velocidad de transmisión de señales. Un baudio es aproximadamente un bit por segundo.

binary **binario**

Sistema de numeración de base 2 en el que los números se representan como combinaciones de dos elementos: el 0 y el 1.

bit **bit**

BInary digiT, dígito binario. Uno de los dos símbolos (el 0 y el 1) con que se representa los números en el sistema de base 2.

boolean logic **lógica de Boole**

Procesos matemáticos lógicos basados en un sistema algebraico desarrollado en el siglo XIX por el matemático inglés George Boole.

bootstrap **autocarga**

Técnica o dispositivo que carga las primeras palabras o instrucciones de una rutina en la memoria. Estas instrucciones cargan a su vez el resto del programa.

branch **salto**

Técnica utilizada para transferir el control del programa a una instrucción distinta de la siguiente a la actual.

bug **error de programación**

Fallo no previsto en un programa; puede consistir en que el programa no funciona como se esperaba o en que el programa entra en una situación tal que se pierde el control de la máquina.

bus **bus**

Grupo de conexiones, dispuestas en paralelo, por las que se transmite datos o señales entre las diferentes partes de un sistema informático.

byte **byte**

Grupo de 8 bits tratado por la UCP como unidad de información.

clock **reloj**

Sistema temporizador que controla y sincroniza las secuencias de operaciones del microprocesador.

COBOL **COBOL**

Common Business Oriented Language, lenguaje común orientado a las aplicaciones de gestión. Lenguaje de alto nivel utilizado en muchas aplicaciones de gestión.

command **orden**

Instrucción para el ordenador.

compiler **compilador**

Programa que convierte las instrucciones de un lenguaje de alto nivel en códigos directamente utilizables por el microprocesador.

controller **controlador**

Interfaz que controla las operaciones de entrada/salida entre la UCP y un periférico.

CPU **UCP**

Central Processing Unit, unidad central de proceso. Núcleo del sistema informático que interpreta las instrucciones y hace que sean obedecidas.

CRT **CRT, TRC**

Cathode Ray Tube, tubo de rayos catódicos. Tubo de vacío en el que se representa la información gráfica.

cursor **cursor**

Símbolo móvil que indica en qué lugar de la pantalla aparecerá el próximo carácter que se envíe a ella.

data **datos**

Elementos de información con los que trabaja el programa.

data bus **bus de datos**

Conjunto de líneas por las que se transmite los datos.

debugging **depuración**

Proceso de detección y corrección de errores de un programa.

DMA **DMA, ADM**

Direct Memory Access, acceso directo a la memoria. Acceso a la memoria sin intervención de la UCP.

disk, disc **disco, disco flexible**

Hoja de plástico circular, fina y plana, recubierta de material magnetosensible, que se utiliza como medio de almacenamiento de la información.

DOS **DOS, SOD**

Disk Operating System, sistema operativo de disco. Programas que controlan las operaciones de la unidad de disco.

dump **volcado**

Transferencia de la información almacenada en lugar del sistema informático a otro; generalmente, hacia la pantalla o la impresora.

editor **editor**

Programa que ayuda a crear y modificar ficheros de texto.

execute **ejecutar**

Hacer que el programa realice las tareas para las que ha sido diseñado.

expression **expresión**

Fórmula que especifica una manera de combinar y procesar la información.

fetch **lectura**

Hace referencia a la lectura de una instrucción o un dato en una posición de la memoria.

file **fichero**

Colección de datos, generalmente homogéneos, grabados en cinta o disco.

floppy disk drive **unidad de disco flexible**

Dispositivo físico que mueve los discos flexibles y lee y escribe en ellos. Es un dispositivo de entrada/salida.

flowchart **ordinograma**

Representación esquemática de la evolución de las acciones de un programa.

format **formato**

Método de organización de los datos en un medio de almacenamiento o visualización (disco, pantalla, impresora, etc.).

FORTRAN **FORTRAN**

FORmulae TRANslator, traductor de fórmulas. Lenguaje de programación de alto nivel, de aplicación primordialmente científica.

gate **puerta lógica**

Dispositivo consistente en una o varias entradas y una salida, diseñado de modo que la salida está determinada por el estado de las entradas.

hardware **equipos, máquinas, circuitos**

Componentes electrónicos o mecánicos de un sistema informático.

hexadecimal (hex) **hexadecimal (hex)**

Sistema de numeración de base 16. Los dígitos de este sistema son las cifras del 0 al 9 y las letras de la A a la F.

high level language **lenguaje de alto nivel**

Lenguaje de programación en el que las instrucciones están más próximas al lenguaje ordinario que al código de máquina. Ejemplos: BASIC, FORTRAN, PASCAL, PL-1.

I/O devices **dispositivos de E/S**

Dispositivos de entrada/salida: unidad de disco, cassette de datos, teclado, pantalla, impresora, etc.

instruction **instrucción**

Orden que manda al ordenador que realice alguna operación o tarea. Un conjunto de instrucciones forma un programa.

interface **interfaz**

Vía de comunicación entre el ordenador y los periféricos, o entre dos ordenadores.

interpreter **intérprete**

Un programa que convierte una a una las instrucciones de alto nivel en instrucciones directamente utilizables por la UCP.

K **K**

1024 bytes.

keyboard **teclado**

Matriz de interruptores alfanuméricos a través de la cual se introduce información en el ordenador.

kilobyte **kilobyte**

1024 bytes.

library **biblioteca**

Colección de programas o ficheros de datos.

load **cargar**

Introducir programas o ficheros en la memoria del ordenador.

location **posición, celda de memoria**

Porción de la memoria en la que se almacena una palabra de datos o una instrucción.

logic **lógica**

Ciencia que trata de la verdad o falsedad de las proposiciones.

loop **bucle**

Grupo de instrucciones que se repite reiteradamente hasta que se cumple cierta condición.

LSI **LSI, IGE**

Large Scale Integration, integración a gran escala. Tecnología de fabricación de circuitos integrados que se caracteriza por el hecho de que en una sola pastilla de silicio se incorporan miles de componentes electrónicos.

machine code **código de máquina**

Lenguaje de programación que es directamente comprensible y utilizable por el microprocesador.

memory **memoria**

Zona del ordenador en la que se almacena las instrucciones y los datos. Cada instrucción está en una posición de memoria, identificada por su dirección.

menú **menú**

Lista de opciones que el programa ofrece al usuario.

microprocessor **microprocesador**

Circuito integrado que controla el funcionamiento del ordenador.

mnemonics **mnemónicos**

Nombres abreviados de las instrucciones, elegidos de forma que sean fáciles de recordar para el programador.

modem (MOdulator/DEModulator) **modulador/demodulador, modem**

Dispositivo que conecta el ordenador a una línea telefónica o a una red de transmisión en serie de la información.

octal **octal**

Sistema de numeración de base 8. Los dígitos de este sistema son las cifras del 0 al 7.

off line **fuera de línea**

Situación en que se encuentra un periférico cuando no está activamente conectado al ordenador.

on line **en línea**

Término opuesto a off line/fuera de línea.

operating system **sistema operativo**

Programa que gestiona el manejo de la memoria, los procesos de entrada y salida, las interrupciones, etc. Representa un interfaz entre la máquina y los programas del usuario.

output **salida**

Todo lo que sale del ordenador como resultado de algún proceso o cálculo.

page **página**

Grupo de posiciones de memoria; en los ordenadores de 8 bits, una página está formada normalmente por 256 bytes.

peripheral **periférico**

Dispositivo externo al ordenador mediante el cual se realizan tareas tales como la impresión, el almacenamiento de datos, las comunicaciones con otras máquinas, etc.

pixel (picture element) **punto gráfico, pixel**

La menor área de la pantalla que puede ser controlada con independencia de las demás.

pointer **puntero**

Registro de la UCP que contiene la dirección de memoria.

program **programa**

Conjunto de instrucciones que dirigen el funcionamiento del ordenador.

program counter **contador de programa**

Registro de la UCP que contiene la dirección de la próxima instrucción que se va a ejecutar.

prompt **mensaje inductor**

Grupo de caracteres que recuerda al usuario que el ordenador está esperando una entrada por el teclado.

RAM **RAM**

Random Access Memory, memoria de acceso aleatorio. Memoria en la que se puede leer en cualquier dirección y en cualquier orden. Su contenido se pierde cuando se apaga la máquina.

register **registro**

Circuito de la UCP en el que se puede almacenar o procesar bits o bytes de datos.

ROM **ROM**

Read Only Memory, memoria de 'sólo lectura'. Memoria cuyo contenido no se puede modificar ni borrar.

routine **rutina**

Una parte del programa que realiza una tarea específica.

software **programas, software**

Programa o conjunto de programas que hace que el ordenador funcione.

source program **programa fuente**

Programa escrito en un lenguaje fácil de comprender para el usuario pero no utilizable directamente por el ordenador.

sprite **agente gráfico móvil, sprite**

Objeto gráfico diseñado por el programador aprovechando las funciones gráficas de la máquina.

subroutine **subrutina**

Véase routine/rutina.

syntax **sintaxis**

Reglas que gobiernan la correcta utilización del lenguaje de programación. Su violación provoca en BASIC el mensaje "Syntax error".

terminal **terminal**

Dispositivo de entrada/salida generalmente formado por la combinación de teclado y pantalla.

time sharing **tiempo compartido**

Sistema en el que una UCP realiza dos tareas simultáneamente o es compartida por varios usuarios.

truth table **tabla de verdad**

Tabla que da los posibles resultados de una operación lógica (AND, OR, etc.) en función del valor de los operandos.

utility program **programa de ayuda**

Un programa con el que se realiza una operación común a varias aplicaciones; por ejemplo, ordenar datos o copiar ficheros.

variable **variable**

Un dato o grupo de datos que puede ser identificado mediante un nombre; su valor puede variar como consecuencia de la ejecución del programa.

volatile storage **memoria de almacenamiento volátil**

Memoria cuyo contenido se pierde cuando se apaga la máquina; por ejemplo, la RAM.

window **ventana**

La pantalla se puede subdividir en ventanas, en cada una de las cuales se visualiza información generada por una tarea diferente de las demás.

Índice


SVI
SA
ESPAÑA