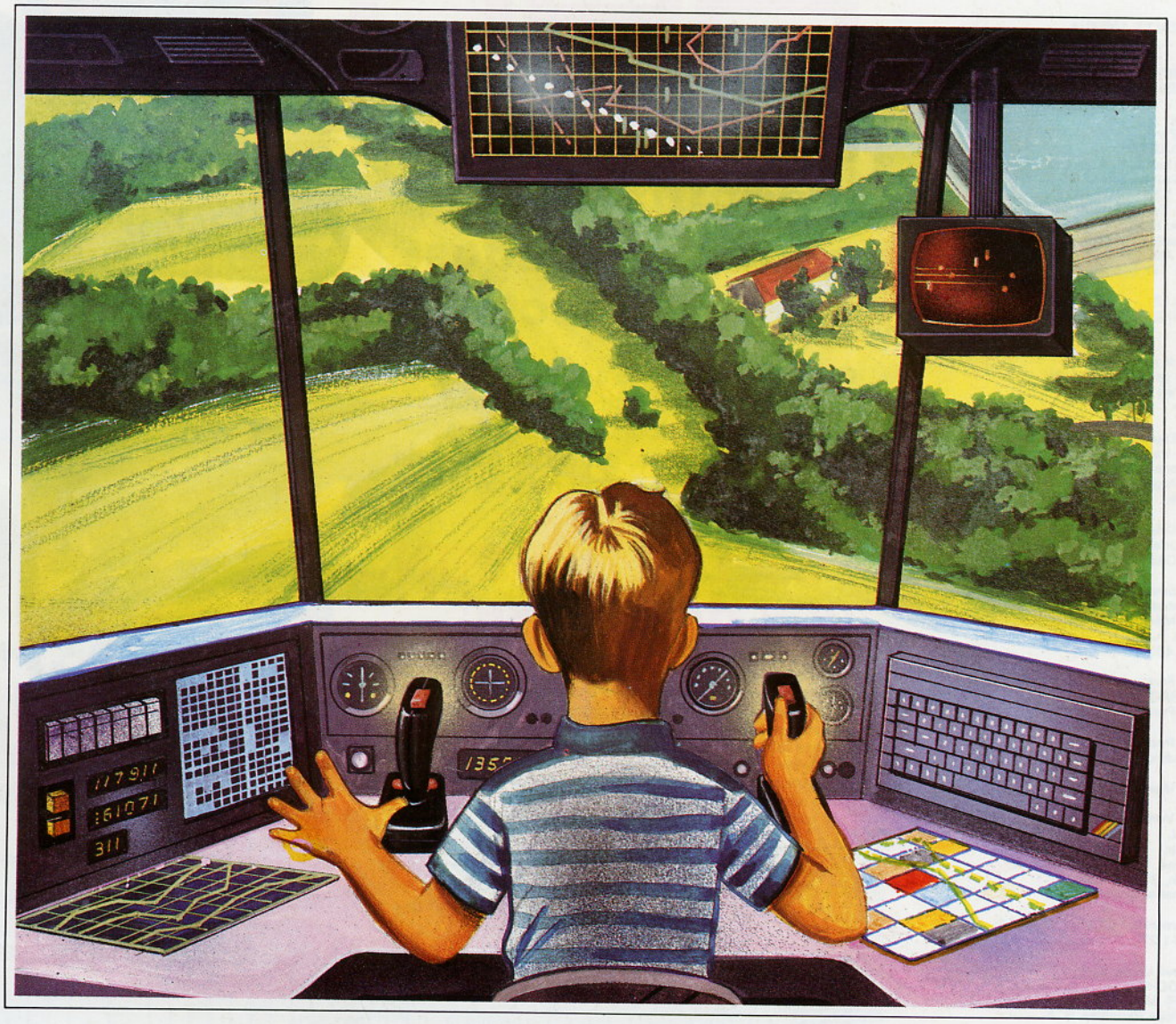
6

150pts.

RUN

Enciclopedia Práctica del Spectrum



Nueva Lente/Ingelek

UN POCO DE PRACTICA

VAMOS a aplicar los conocimientos adquiridos, para confeccionar un programa que nos permita calcular la nota media obtenida en una determinada asignatura, suministrando al ordenador el número total de calificaciones y cada una de estas en particular.

Comenzamos preparando el Spectrum por medio de la instrucción **NEW** para aceptar este nuevo programa y tecleamos:

```
 10 REM - CALCULO DE NOTA MEDIA -
J.M.LOPEZ MARTINEZ
 20 LET T=0
 30 LET I=0
 40 CLS
 50 INPUT "Numero total:";N
 60 PRINT "Numero total: ";N
 70 PRINT
 80 PRINT "--- CALIFICACIONES OBTENIDAS ---"
 90 INPUT C
100 PRINT C,
110 LET T=T+C
120 LET I=I+1
130 IF I<N THEN GO TO 90
140 PRINT
150 LET M=T/N
160 PRINT "Nota media..: ";M
170 INPUT "Fin de programa ?";F$
180 IF F$="NO" THEN GO TO 20
190 IF F$<>"SI" THEN GO TO 170
```

REM

La palabra clave BASIC **REM** no nos ha sido todavía presentada. Para remediar este olvido diremos que en inglés significa comentario **(REMARK)** y no es, por lo tanto, una instrucción para que el ordenador la ejecute, sino que nos sirve de ayuda para incluir comentarios clarificadores de lo que el programa hace, en los lugares que nos interese, o simplemente para indicar la utilidad del programa al principio de éste, como es el caso del ejemplo que nos ocupa:

10 REM - CALCULO DE NOTA MEDIA - J.M. LOPEZ MARTINEZ

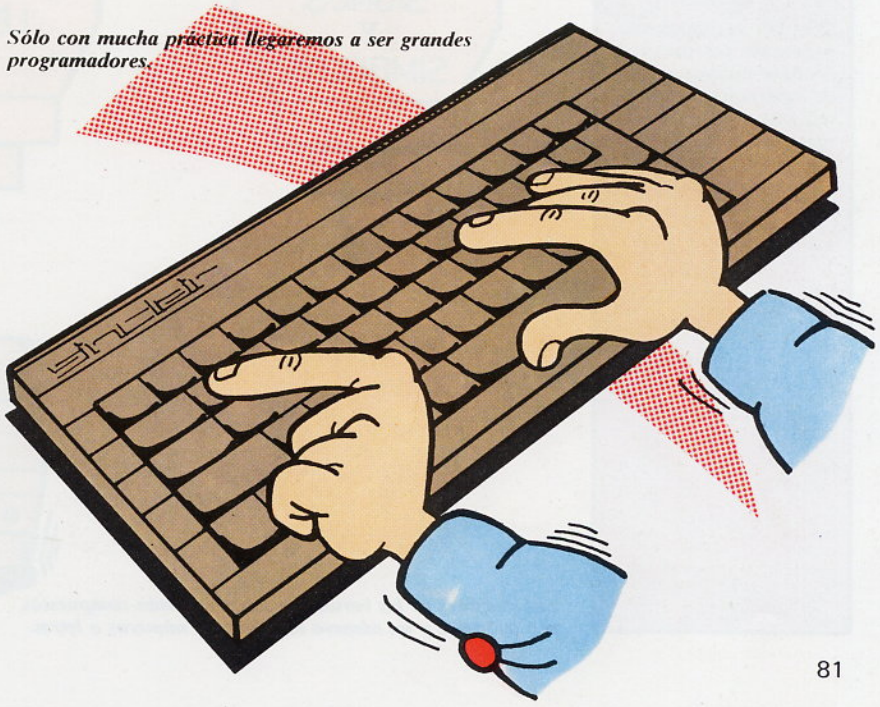
Todo lo que se encuentra a la derecha de la palabra **REM** no es interpretado como una instrucción por el ordenador. Debido a esta particularidad, es una técnica de uso común entre los programadores, anteponer la plabra clave **REM** a una instrucción que queremos que el ordenador no considere, sin suprimirla definitivamente del programa:

20 REM LET T=0

Cuando estamos realizando pruebas con un determinado programa, y pensamos que sería necesario eliminar una instrucción, pero no estamos completamente seguros de que con ello vayamos a solucionar nuestros problemas, antes de suprimir la instrucción debemos anteponerle un **REM.**

Hecho esto, volvemos a ejecutar el programa y comprobamos sus resultados en la seguridad de que, si ahora las cosas marchan correctamente, nos basta con borrar definitivamente la instrucción tecleando su número seguido de **ENTER,** como ya vimos anteriormente.

Si, por el contrario, la instrucción debe permanecer aún en el programa, no tendremos que volver a teclearla, como en el caso de que la hubié-



Sólo con mucha práctica llegaremos a ser grandes programadores.

¡!

La palabra BASIC **REM** sirve para introducir comentarios en los programas, y no es ejecutada como una instrucción por el ordenador.

*

El efecto de cualquier instrucción precedida de **REM** queda absolutamente anulado, por lo que suele ser empleada en la depuración de programas.

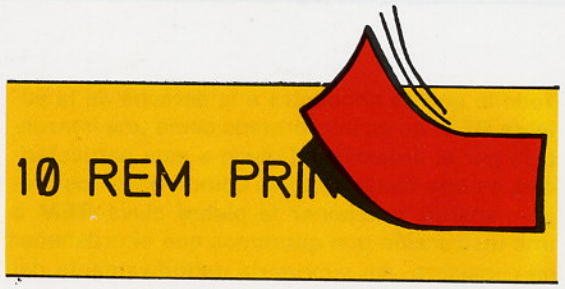
*

En una instrucción de asignación **LET**, siempre se calcula el valor de la expresión a la derecha del símbolo igual (=), y se asigna a la variable situada a la izquierda del mismo.

*

Con las variables numéricas se pueden realizar cualquier tipo de operaciones matemáticas, pero no con las alfanuméricas.

ramos suprimido directamente, sino que nos bastará con desproveerla de su **REM** para que el programa vuelva a quedar en las mismas condiciones que al principio. Digamos, por tanto, que **REM** tiene el efecto de «anular» las instrucciones a las que se antepone.



El REM *se utiliza para anular el efecto de las instrucciones a que se antepone.*

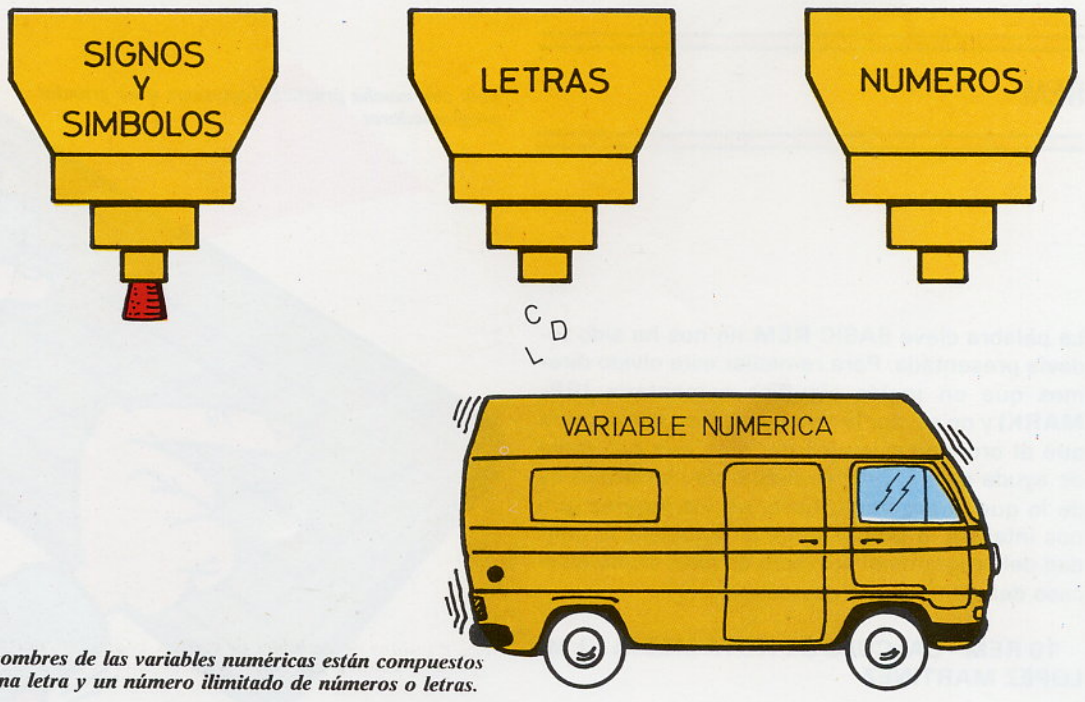
LET

La instrucción **LET** asigna valores a las variables BASIC. La estructura de esta sentencia es siempre la misma. La instrucción comienza por un **LET**, que va inmediatamente seguido por el nombre de la variable a que afecta y el signo igual (=). A la derecha del símbolo de igualdad se encuentra el nombre de otra variable, una constante o una expresión que el ordenador va a calcular para determinar el valor de la variable especificada por **LET**:

```
20 LET T=0
30 LET I=0

110 LET T=T+C
120 LET I=I+1
150 LET M=T/N
```

En el caso de las líneas 20 y 30, igualamos las variables especificadas por **LET** a otras variables (del mismo tipo), situadas a la derecha del signo igual (=). Este es el sistema más simple de asignación.
En las instrucciones 110, 120 y 150, la asignación se efectúa por medio de una fórmula. El ordenador debe calcular, en estos casos, el resultado de la expresión a la derecha del símbolo igual (=), para dar después el valor obtenido a la variable correspondiente.
A la hora de asignar valores a las variables, debemos poner especial atención en el tipo de variables a que nos estamos refiriendo. Ya hemos dicho que el Spectrum es capaz de manejar tanto números como letras; sin embargo, el ordenador da el mismo tratamiento a los caracteres numéricos que a los alfabéticos. Esta distinción es lógica porque, si pensamos un poco, nos damos cuenta en seguida de que con los números podemos realizar cálculos, pero no con las letras. Con más propiedad, deberíamos decir que los dos tipos de variables que existen son las «numéricas» y las «alfanuméricas».



Los nombres de las variables numéricas están compuestos por una letra y un número ilimitado de números o letras.

Las variables numéricas deben contener forzosamente números y se emplean para realizar cálculos de tipo numérico. Se denominan con una serie de caracteres alfanuméricos, el primero de ellos alfabético. Con ellas se pueden realizar, por lo tanto, las operaciones matemáticas básicas, o de cualquier tipo más complejo.

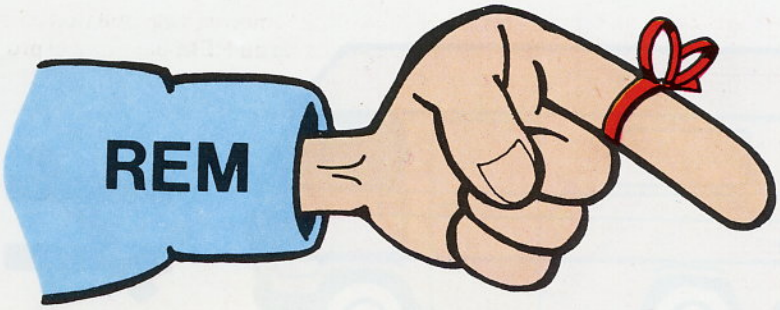
Las variables alfanuméricas pueden contener números, letras o caracteres especiales, es decir, cualquiera de los accesibles por medio del teclado, incluyendo símbolos gráficos. Estas variables se denominan con una letra más el símbolo dólar **($)**, y no pueden ser sometidas a las operaciones matemáticas, pero si admiten la CONCATENACION y el TRUNCAMIENTO.

La concatenación consiste en la «suma» de dos o más cadenas de caracteres para dar lugar a una nueva, mientras que el truncamiento nos permite obtener una subcadena a partir de una cadena de caracteres dada; es decir, utilizar sólo una parte de la cadena original. Estas dos operaciones con cadenas serán estudiadas con detalle más adelante.

Debemos tomar buena nota de que no podemos mezclar en una misma instrucción **LET** los dos tipos diferentes de variables, pues provocaríamos un error. A los dos lados del símbolo de igualdad deben existir siempre expresiones del mismo tipo:

```
110 LET T=T+C
```

En la línea 110, tanto la variable **T** como la variable **C** son numéricas, por lo tanto, la asignación por medio de **LET** es correcta. No lo sería, sin embargo:



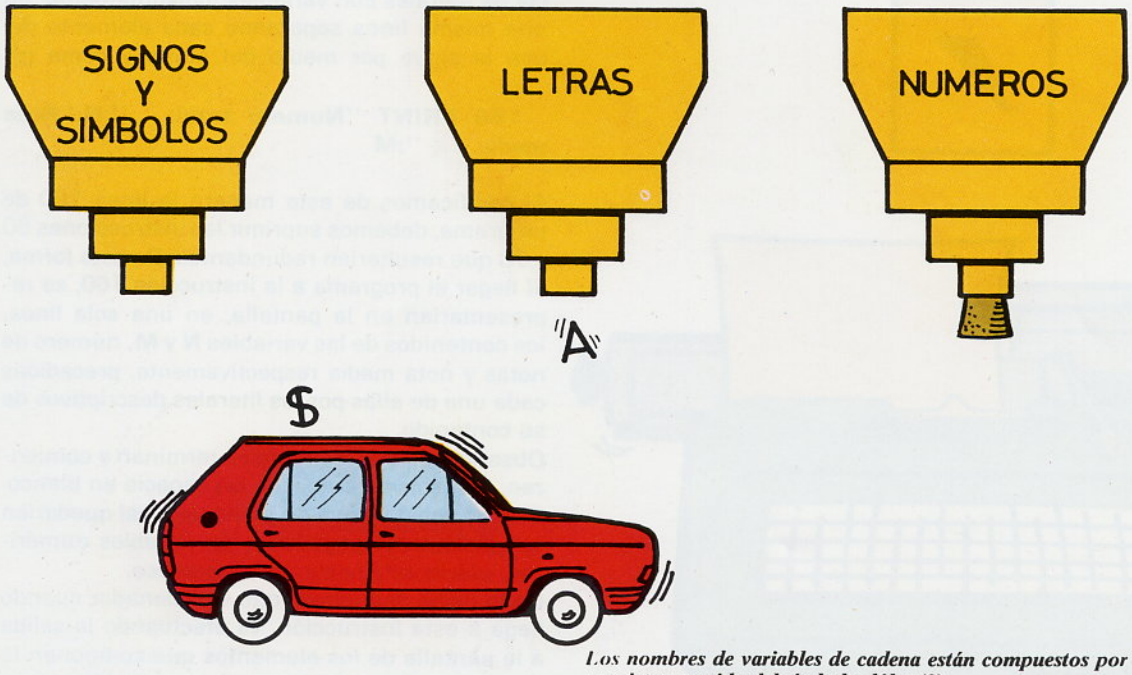
La sentencia **REM** *se emplea para incluir comentarios en los programas.*

```
110 LET T$=T+C
```

Esta instrucción produciría un error, ya que la variable **T$** es de cadena (alfanumérica), y no podemos asignarle un valor numérico, como sería el resultado de la operación de suma entre las variables **T** y **C**.

CLS Y PRINT

La palabra clave **CLS** nos permite borrar el contenido de la pantalla. La siguiente instrucción



Los nombres de variables de cadena están compuestos por una letra seguida del símbolo dólar **($)**.

¡!

A una variable numérica nunca le podemos asignar resultados alfanuméricos, ni a una de este tipo un valor numérico.

*

Las únicas operaciones que se pueden efectuar con cadenas son la concatenación y el truncamiento.

*

El apóstrofe se utiliza en el Spectrum para forzar un «retorno de carro» en combinación con una instrucción **PRINT**.

Nunca se pueden realizar operaciones de igualdad entre expresiones de distinto tipo.

PRINT que coloquemos inmediatamente después de **CLS**, se situará al comienzo de la primera línea de la misma:

```
40 CLS
60 PRINT ''Numero total:    '';N
```

En nuestro programa, el literal ''Numero total: '' aparecerá en la primera línea de la pantalla.
La instrucción **PRINT** es, sin duda, una de las más populares del vocabulario BASIC. Permite dar salida a la pantalla tanto de literales, que son transcritos en la forma exacta en que los incluimos en el programa, como de variables, o resultados de cálculos ejecutados por el ordenador:

```
160 PRINT ''Nota media..:    '';M
```

Hubiera sido igualmente válido, ahorrándonos además el uso de la variable intermedia **M**, escribir:

```
160 PRINT ''Nota media..:    '';T/N
```

Su uso más elemental se reduce a producir una línea en blanco en la pantalla, como es el caso de la línea 70 de nuestro programa:

```
70 PRINT
```

Sin embargo, el empleo más potente y generalizado de la instrucción se da con los caracteres de puntuación BASIC.
Podemos imprimir una lista de variables, o mezcla de literales con variables indistintamente, en una misma línea separando cada elemento del que le sigue por medio del punto y coma (;):

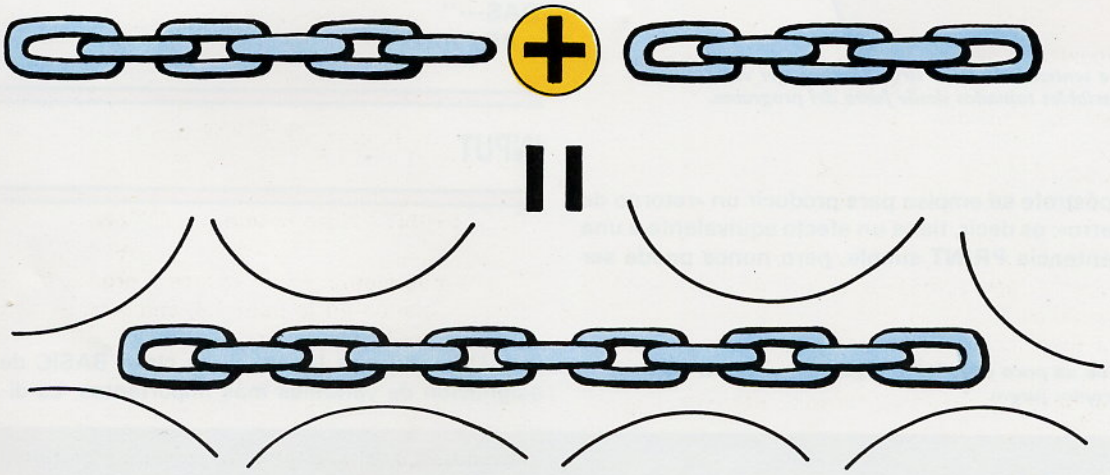
```
160 PRINT ''Numero total:    '';N;''Nota media..:    '';M
```

Si codificamos de esta manera la línea 160 de programa, debemos suprimir las instrucciones 60 y 70 que resultarían redundantes. De esta forma, al llegar el programa a la instrucción 160, se representarían en la pantalla, en una sola línea, los contenidos de las variables **N** y **M**, número de notas y nota media respectivamente, precedidas cada una de ellas por los literales descriptivos de su contenido.
Observamos que los literales terminan y comienzan, respectivamente, con un espacio en blanco. Esto es debido a que de no hacerlo así quedarían completamente pegados a las variables numéricas, efecto éste bastante antiestético.
De lo dicho deducimos que el ordenador cuando llega a esta instrucción, va efectuando la salida a la pantalla de los elementos que componen la línea, tanto los literales como las variables, unos

El apóstrofe tiene el efecto de un «retorno de carro» en combinación con una sentencia **PRINT**.

a continuación de otros, hasta que se detecta el final de la instrucción.
Como el último elemento a imprimir en nuestro ejemplo (la variable **M**), no va seguido de un punto y coma (;), el BASIC producirá automáticamente un «retorno de carro», similar al de una máquina de escribir eléctrica cuando pulsamos el cambio de línea, escribiéndose las siguientes sentencias **PRINT**, si existen, al comienzo de la siguiente línea de la pantalla.
Otro carácter de puntuación que empleamos en nuestro programa de ejemplo con la sentencia **PRINT** es la coma (,):

```
100 PRINT C,
```



La operación de concatenación consiste en la «suma» de cadenas.

Para entender el uso de este separador como carácter de tabulación, debemos suponer que el ancho de la pantalla (32 columnas) está dividido en dos zonas de 16. Si escribimos:

```
PRINT A,B
```

Obtenemos el valor de la variable **A** al comienzo de la primera mitad de la línea, y la variable **B** al comienzo de la segunda mitad de ésta. Hay que añadir que la coma (,) al final de una instrucción no produce un cambio de línea, por lo que, por sí sola, tiene un efecto equivalente a:

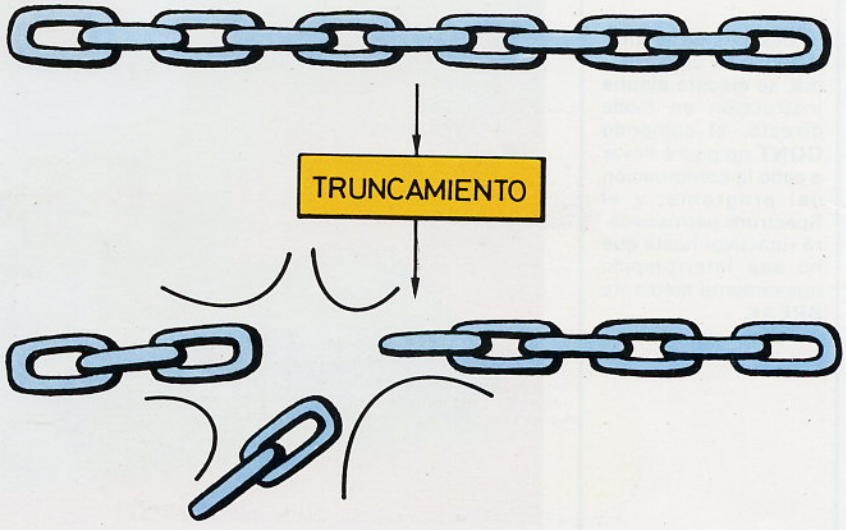
```
100 PRINT C,;
```

Por cada nuevo valor que suministramos a la variable **C** por medio de la instrucción **INPUT** de la línea 90, obtenemos la impresión de la variable en la siguiente «mitad de la pantalla» que le corresponda.
Tenemos como inconveniente que el número de calificaciones **(C)** puede ser impar y, en ese caso, la instrucción 140 se imprimiría en la segunda mitad de una línea de la pantalla, en contra de nuestros deseos. Para resolver ese problema, empleamos una sentencia **PRINT**, cuya única finalidad es que produzca un «retorno de carro» (cambio a la siguiente línea de la pantalla). Como además queremos que exista una línea en blanco, que separe las últimas calificaciones introducidas de la línea en que se imprime el resultado de la nota media, debemos incluir otra sentencia **PRINT**:

```
140 PRINT '
```

Esta instrucciíon nos da pie a introducir un último signo de puntuación que modifica los resultados de la sentencia **PRINT**: el apóstrofe. El

El truncamiento es la «extracción» de determinada parte de una cadena.



¡!

La interrupción de un programa durante la ejecución de un **INPUT** no se puede conseguir mediante **BREAK, sino con STOP.**

*

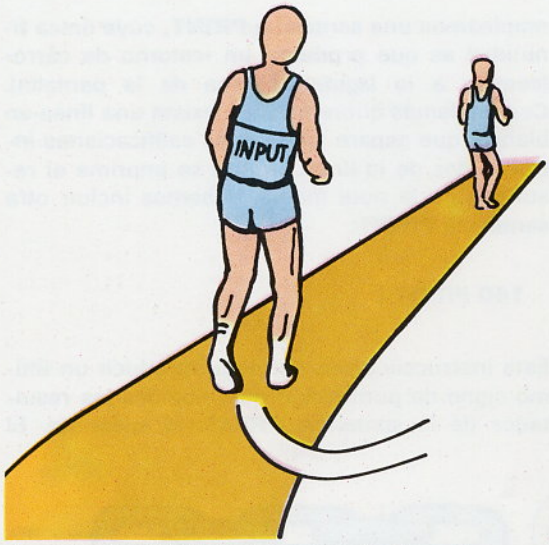
Para que el **STOP** sea admitido en un **INPUT** de variable de cadena, es necesario suprimir en primer lugar las comillas (") que caracterizan este tipo de datos.

*

Para continuar la ejecución de un programa que ha sido detenido mediante **BREAK** o **STOP** se utiliza el comando **CONT.**

*

Si después de la interrupción de un programa, se ejecuta alguna instrucción en modo directo, el comando **CONT** no podrá llevar a cabo la continuación del programa, y el Spectrum permanecerá «inactivo» hasta que no sea interrumpido nuevamente mediante **BREAK.**



La sentencia **INPUT** *sirve para recoger valores de variables tomados desde fuera del programa.*

empleado al comienzo de una instrucción, pues no se trata de una palabra clave BASIC. La utilización del apóstrofe nos permite ahorrar espacio; de no ser por él, para conseguir el efecto de dos retornos de carro en la línea 140, habríamos tenido que utilizar dos instrucciones en vez de una:

```
140 PRINT '
                =
140 PRINT
140 PRINT
```

Y lo mismo habría sucedido en la línea 80:

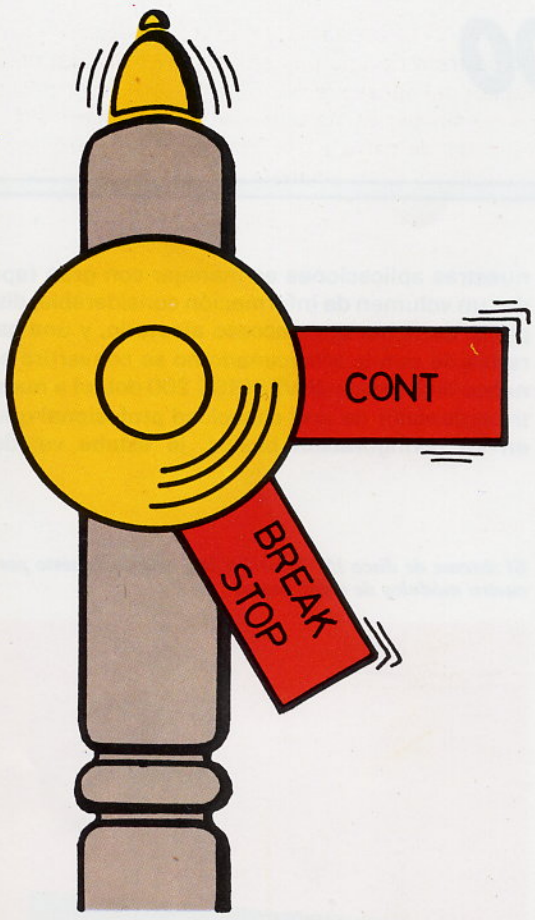
```
80 PRINT "---CALIFICACIONES OBTENIDAS---" '
                =
80 PRINT "---CALIFICACIONES OBTENIDAS---"
85 PRINT
```

INPUT

apóstrofe se emplea para producir un «retorno de carro»; es decir, tiene un efecto equivalente a una sentencia **PRINT** simple, pero nunca puede ser

Con un poco de práctica llegaremos a realizar nuestros propios juegos.

INPUT y **LET** son las palabras clave BASIC de asignación de variables más importantes. La di-



Gracias al comando **CONT** *podemos continuar la ejecución de un programa que ha sido detenido mediante* **BREAK** *o* **STOP.**

rica o de cadena. La diferencia la encontramos en el momento de la puesta en marcha del programa, ya que en la línea 170, por tratarse de una variable de *string* (cadena de caracteres), el Spectrum nos muestra el cursor **L** entre comillas, mientras que el **INPUT** numérico (línea 90) lo hace sin ellas.

Si deseamos interrumpir el programa cuando nos encontramos dentro de una sentencia **INPUT,** podemos hacerlo por medio de la tecla **STOP** en lugar de **BREAK.** En el caso de que el **INPUT** sea numérico, **STOP** funcionará sin más dificultades, pero si se trata de una variable de cadena, debemos eliminar previamente las comillas avanzando una posición el cursor y pulsando dos veces **DELETE** para, finalmente, pulsar **STOP** y **ENTER.**

Desde luego **STOP** no implica necesariamente una detención definitiva del programa. Siempre que el ordenador se encuentra con la instrucción **STOP,** antes de detenerse «recuerda» exactamente en qué punto del programa se quedó. Cuando depuramos errores en un programa, nos puede ser muy útil interrumpirlo por medio de **STOP** para preguntar el valor de determinadas variables y comprobar resultados; después podremos continuar la ejecución del programa mediante el comando **CONT.**

¡!

Para borrar toda la línea tecleada en un **INPUT,** antes de la pulsación de **ENTER,** se puede emplear la tecla **EDIT (CAPS SHIFT + 1).** En caso de **INPUTs** de cadenas, las comillas que los caracterizan también desaparecerán por este sistema.

La sentencia **LET** *asigna valores a las variables desde dentro del programa.*

ferencia fundamental entre ambas sentencias es que **INPUT** nos permite incluir el valor desde «fuera» del programa, decidiéndolo en el mismo momento de la ejecución, a petición del propio programa, que efectúa una parada para que se le suministre el valor que debe asignar a la variable. **INPUT** permite la introducción de variables numéricas:

```
50 INPUT "Numero total:";N
```

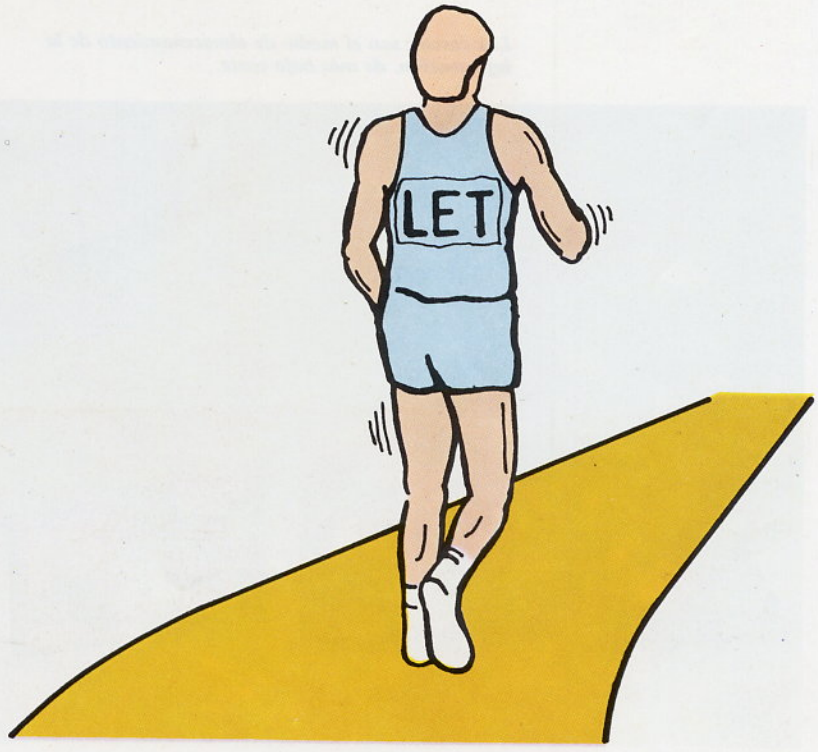
O variables de cadena:

```
170 INPUT "Fin de programa?";F$
```

Asimismo, el mensaje que hemos incluido de petición de variable en ambas líneas, es opcional y podemos eliminarlo escribiendo:

```
90 INPUT N o 170 INPUT F$
```

Aparentemente no existe diferencia alguna entre que el **INPUT** corresponda a una variable numé-



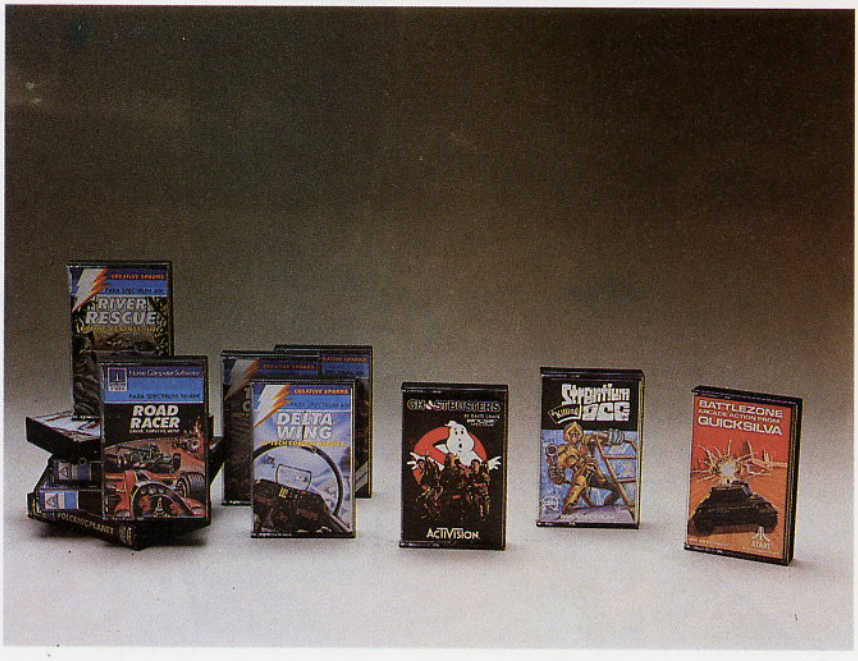
INVESDISK 200

¿CUÁNTAS veces todos aquellos que manejamos habitualmente el Spectrum, nos desesperamos cuando al grabar o cargar un programa almacenado en cinta, tenemos que aguardar varios minutos hasta completar la tarea; es más, puede que recibamos con cara de pocos amigos la noticia de que ese programa que tanto nos costó realizar, ha quedado reducido a un exasperante mensaje en la parte inferior de la pantalla: **R Tape loading error.**

Por desgracia, estos problemas se suceden con una relativa frecuencia. Desde luego, las cintas tienen sus ventajas; son baratas y como medio de almacenamiento masivo de datos y programas son una buena solución. Pero la transmisión de la información es lenta, y para un tratamiento «profesional» de datos están bastante lejos de ser el medio ideal.

La unidad de disco comercializada por INVESTRONICA, viene a completar la amplia gama de periféricos disponibles para el Spectrum. Desde luego, si nuestro ordenador tan sólo nos sirve como base de lanzamiento de misiles, los discos estarán de más; pero si lo que necesitamos en nuestras aplicaciones es manejar con gran rapidez un volumen de información considerable, disponer de ficheros de acceso aleatorio, y una garantía de que lo almacenado no se convertirá en nubes de humo, el INVESDISK 200 dotará a nuestro ordenador de una capacidad profesional que, en su configuración básica, le estaba vetada.

Los casetes son el medio de almacenamiento de la información, de más bajo coste.



El sistema de disco INVESDISK 200 está compuesto por cuatro módulos de color negro.

DESCRIPCION DEL SISTEMA

El sistema consta de cuatro módulos del mismo color que el ordenador: INTERFACE, ALIMENTADOR DE CORRIENTE, CONTROLADOR y UNIDAD DE DISCOS.

El interface (el más pequeño de los cuatro elementos), se conecta por una parte a la tarjeta de expansión del Spectrum, y por otra al controlador, mediante un conector del tipo D con 15 terminales. En su parte superior, incorpora un botón de *reset,* que permite en cualquier momento restaurar la situación inicial del ordenador (con la consiguiente pérdida de la información de la memoria).

El objetivo del alimentador es suministrar la co-

¡!

El controlador es un elemento fundamental en el sistema INVESDISK 200; en él reside el sistema operativo TOS (Timex Operating System), y es el encargado de controlar todas las operaciones relacionadas con el disco. Para llevar a cabo su tarea se auxilia en un microprocesador Z-80 y una memoria interna independiente de la del Spectrum.

*

Los disquetes utilizados por INVESDISK 200 poseen dos tipos de protecciones: una contra el borrado y escritura accidental por el usuario, consistente en unos obturadores deslizantes (uno por cada cara); y otra contra el deterioro físico del soporte magnético, que se encuentra protegido por una placa metálica que impide su exposición al medio ambiente, y encerrado en una caja rígida para evitar que se dañe al ser doblado.

¡!

La cabeza de lectura/escritura de la unidad de disco fundiona por una sola cara (la superior del disquete que se inserta), debido a ello no se pueden acceder simultáneamente más que a 160 K de las 320 del disquete.

*

El sistema INVESDISK 200 es incompatible con el ZX INTERFACE 1, y por tanto, con los Microdrives, interface RS-232 y red de área local.

rriente necesaria para el perfecto funcionamiento del sistema. Un interruptor en su parte posterior, marcado con 0-1, permite activarlo. De esta unidad parten cuatro cables; uno para la conexión a la red, otro que la une al controlador y finalmente, dos cables para las unidades de disco. Si sólo disponemos de un *drive* (unidad de disco), uno de estos cables quedará libre. Por el contrario, si deseáramos conectar tres o cuatro unidades, necesitaríamos una segunda fuente de alimentación.

El controlador tiene la misión de procesar todas las instrucciones que recibe desde el ordenador con destino a las unidades de disco. Este elemento del sistema INVESDISK puede ser considerado como un «periférico inteligente», ya que tiene su propio microprocesador Z-80 y su memoria de acceso independiente, por lo que se puede decir que

El CONTROLADOR posee su propio microprocesador Z-80 para tratar los datos procedentes del ordenador.

El INTERFACE es el más pequeño de los cuatro componentes del equipo.

es una reducidísima versión de un microordenador, cuya única función es controlar las unidades de disco.

Cada vez que el controlador recibe una orden, la almacena en su memoria interna, se encarga de ejecutarla y, además, no ocupa memoria del Spectrum. Es en este módulo donde reside el TOS (Timex Operating System), sistema operativo para

disco de la firma TIMEX, necesario para realizar cualquier tarea que se le encomiende al disco. Un pulsador de *reset* permite el borrado de la memoria interna de la unidad, y la posibilidad de que, si en el disquete insertado en la unidad existe un programa con el nombre START, éste se cargue y ejecute automáticamente.

El último módulo del sistema es la unidad de disco. En su frontal se encuentra situada la ranura por donde introducir el disquete. Un indicador luminoso de color rojo, se enciende cada vez que el disco entra en funcionamiento. Si es la primera ocasión en que se conecta el disco, o hemos realizado un *reset* en el controlador, el citado indicador permanecerá parpadeante hasta completar la carga del sistema operativo (TOS). Si ahora efectuamos un *reset* en el interface, además del habitual mensaje de presentación de Sinclair, obtenemos en la pantalla la inscripción: **(C) 1984 TIMEX-TOS VA.1**.

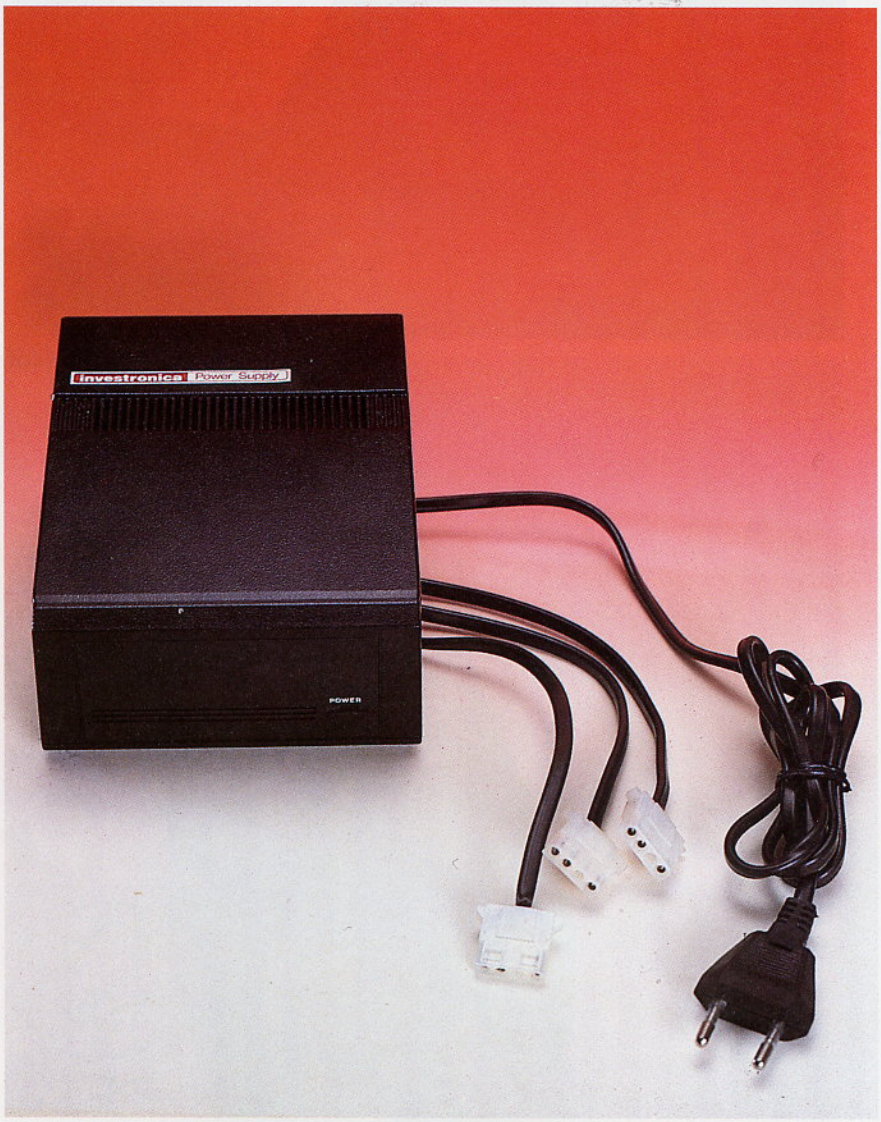
LOS DISQUETES

Los disquetes utilizados son de tres pulgadas, con sus dos caras disponibles para nuestro uso. La capacidad de almacenamiento de información por cada cara es de 160 K, aunque en realidad, tras la inevitable operación **FORMAT** que más adelante estudiaremos, no quedarán más que 140 Kbytes aprovechables. Esto se debe a que el sistema operativo ocupa aproximadamente 20 K, y es del todo necesario que se encuentre grabado en el disquete, antes de poder realizar cualquier operación con él.

Los disquetes disponen físicamente de dos protecciones contra el posible deterioro de la información almacenada en ellos. La primera de ellas está constituida por dos pequeños dispositivos deslizantes de plástico rojo; según la posición en que éstos se encuentren, la cara del disquete a que afectan podrá ser sometida o no a operaciones de escritura o borrado, quedando siempre abierta la posibilidad de lectura.

Este mecanismo es similar al de los casetes o cartuchos para microdrives Sinclair, con la diferencia de que en este caso su efecto es fácilmente reversible: basta con deslizar nuevamente el obturador plástico en sentido contrario, sin tener necesidad de «parchear» el disquete con cinta adhesiva u otros medios, como sucede con los cartuchos y los casetes, e incluso en otros tipos de disquetes.

La segunda protección que poseen estos *floppies* (nombre por el que se conoce a los disquetes con soporte magnético flexible), no es ya contra el borrado accidental, sino contra el posible deterioro físico que pueda sufrir el dispositivo. La plancha de material magnético está encerrada en una caja rígida y la zona en la que se sitúa la cabeza de lectura/escritura del disco, se encuentra protegida por una placa metálica que evita la exposición directa de la superficie magnética al medio ambiente. Gracias a estas características se evitan los dos problemas más frecuentes con este tipo de soportes de información: doblar el disquete o tocar con los dedos la superficie magnética. El sistema por el cual el disquete queda expuesto a la acción de la cabeza de lectura/escritura



Al contrario que el ZX Microdrive, el disco distribuido por INVESTRONICA precisa de una fuente de alimentación independiente de la del Spectrum.

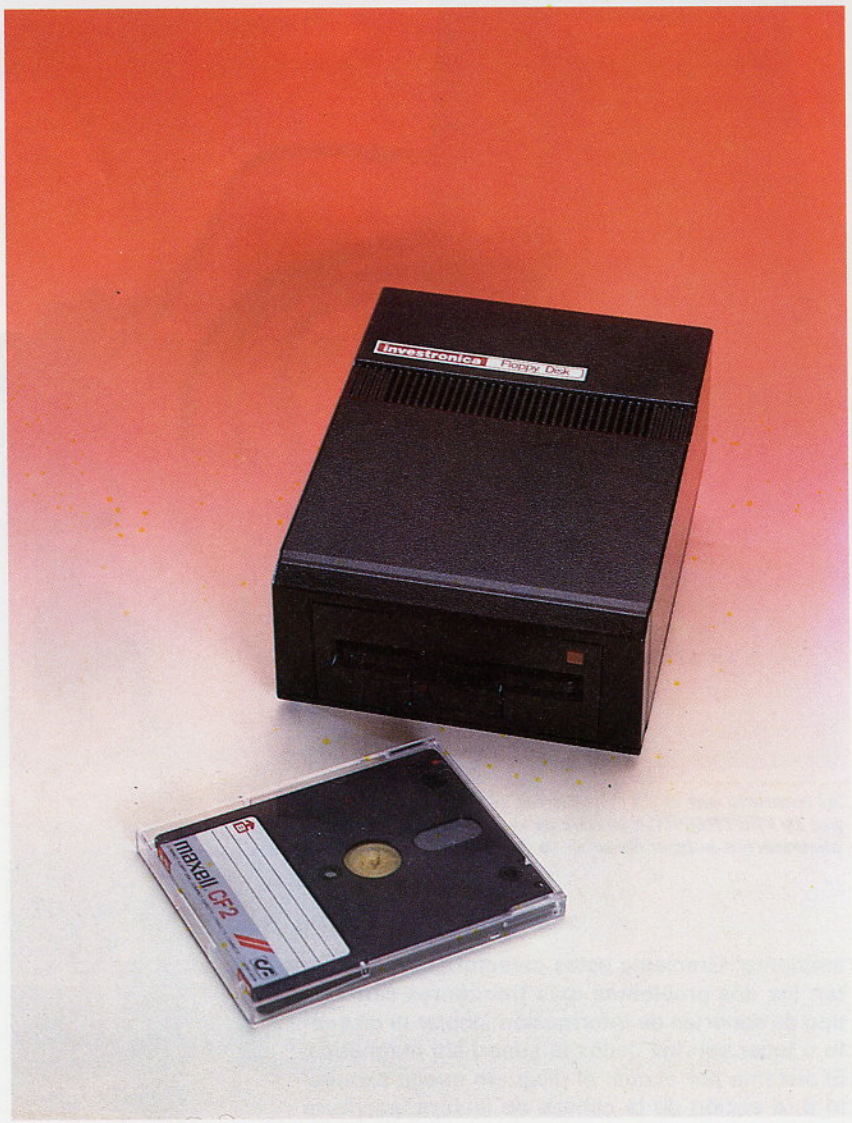
El disquete empleado por INVESDISK 200 es un floppy de tres pulgadas, utilizable por las dos caras, con una capacidad de 160 K (antes del «formateado») por cada una de ellas. Para la protección del disquete contra la escritura se dispone de dos pequeños elementos deslizantes de color rojo, que se encuentran ubicados en la parte delantera del mismo. Cuando el disquete se inserta en la unidad de disco, la placa metálica que habitualmente protege la superficie magnética se desplaza para permitir las operaciones de lectura y escritura.

del disco es bastante ingenioso: una palanca de plástico blanco situada en un costado del mismo, se encuentra unida a la placa metálica de protección antes mencionada; conforme el disquete se va insertando en la unidad de disco, la palanca se va desplazando, dejando al descubierto la superficie magnética para que pueda ser accedida libremente por la cabeza del disco.

Aunque antes hemos visto que la capacidad de cada disquete es de 160 K por cada cara (140 K útiles), no hemos de pensar que disponemos de 320 K por disquete, puesto que la cabeza de la unidad de disco no puede trabajar por las dos caras. Esto quiere decir que adquirir un disquete con destino al INVESDISK 200, es como comprar a un mismo tiempo 2 disquetes de 160 K, pero no debemos engañarnos pensando que podemos acceder, sin darle la vuelta al disquete, a sus 320 K de información. Debido a esto, cuando se realiza una operación **FORMAT** para preparar el disquete virgen para su uso, ésta no se realiza sobre las dos caras.

En todo caso, su capacidad de almacenamiento es bastante considerable (casi el doble que la de un cartucho de microdrive), aunque su punto fuerte se encuentra en las posibilidades de manejo del disco, más que en su capacidad de almacenamiento de información.

CARACTERISTICAS DEL DISCO



Al igual que el ZX INTERFACE 1, el sistema operativo TOS utiliza el símbolo asterisco (*) para identificar los comandos destinados al disco. Así, palabras BASIC como **GOSUB, ATTR** o **MOVE,** adquieren un significado adicional al acoplar el sistema INVESDISK 200 a nuestro ordenador.

La unidad de disco se acompaña de un disquete con programas de demostración.

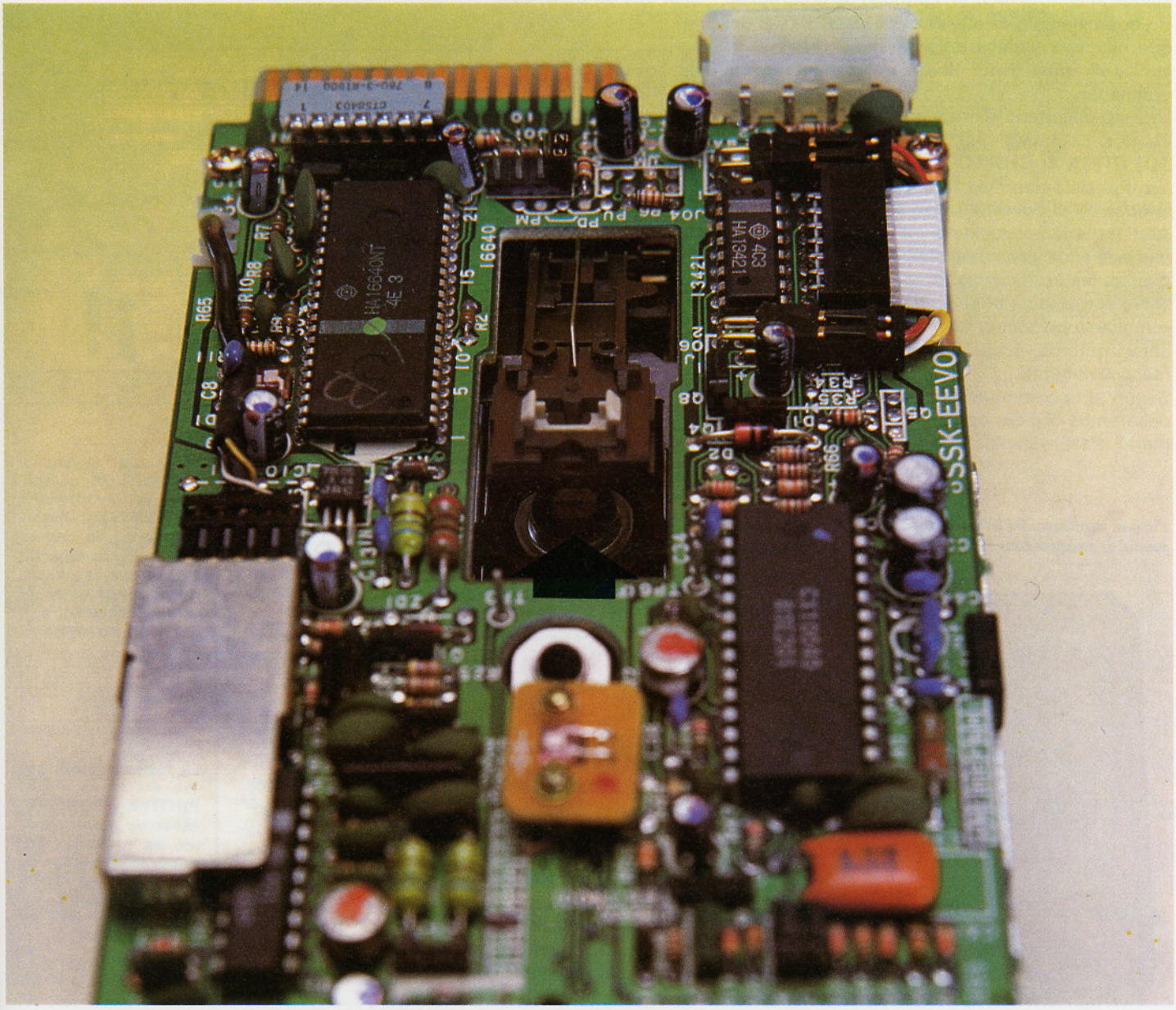
Desde luego, el manejo del casete no es en ningún momento entorpecido por el disco, gracias a lo cual podremos disponer simultáneamente de dos tipos diferentes de almacenamiento de datos: el disco para el trabajo diario que precisa de rapidez y agilidad en el tratamiento, y el casete para las baratas y siempre necesarias copias de seguridad **(BACK-UP)**.

Quizás el único inconveniente que podamos encontrar en la incompatibilidad INTERFACE 1-INVESDISK es que nos priva de dos herramientas asociadas al uso del interface de Sinclair: el interface serie RS-232 y la interconexión en red de área local **(NET)**.

Como último dato añadiremos que el sistema operativo pone a nuestra disposición dieciséis canales diferentes por los que circularán los datos. Los canales del 1 al 4 son rápidos, y del 5 al 16, lentos. Esto se debe a que cada canal rápido tiene reservado un *buffer* de 512 bytes exclusivamente para él. Los lentos comparten un único *buffer* común, de forma que cada vez que entra una nueva información en él, se destruye la que hubiera almacenada con anterioridad.

Unas páginas más adelante finalizaremos el examen del INVESDISK 200, estudiando más a fondo la capacidad y características de su sistema operativo.

En la zona central de la foto podemos apreciar la cabeza de lectura/escritura de la unidad de disco. Al introducir un disquete, la superficie magnética queda automáticamente expuesta a la acción de la cabeza de lectura/escritura del disco.

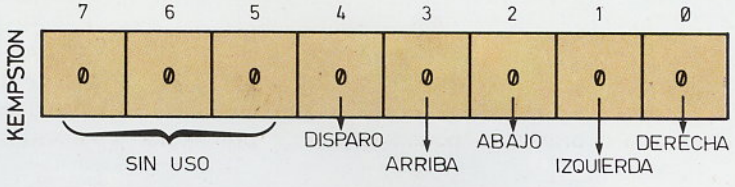


TELE-SKETCH

EL objetivo de este programa no es otro que el de acercarnos al empleo del *joystick* en nuestros propios programas, con el fin de que la palanca de juego no se convierta en un instrumento de poca utilidad, que sólo empleamos para abatir naves interestelares o masacrar marcianitos. La utilización de este periférico ayuda en muchas ocasiones al control de gran cantidad de programas. Un buen ejemplo de ello es el **TELE-SKETCH** del que nos ocuparemos a continuación, gracias al cual podremos trazar líneas en la pantalla del ordenador.

Una vez ejecutado el programa, aparece un menú de dos opciones: **JOYSTICK KEMPSTON** y **JOYSTICK & CURSORES.** Como ya sabemos, los dos sistemas más difundidos de control del *joystick* en el Spectrum, son el **KEMPSTON** y el **AGF** (también conocido como **CURSOR**). El primero de ellos, considera absolutamente independientes las acciones sobre la palanca de juego y sobre el teclado. Por el contrario, el sistema **AGF** «emula» la pulsación de las teclas 5, 6, 7, 8 y 0, para izquierda, abajo, arriba, derecha y disparo, respectivamente.

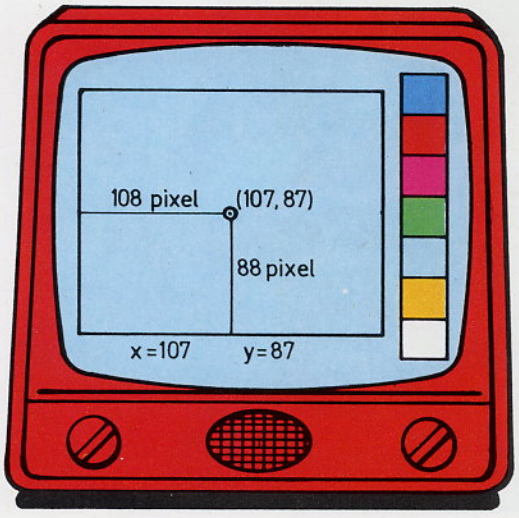
Según el tipo de interface del que dispongamos, deberemos decidirnos por una u otra opción del menú. Para aquellos que no poseamos un *joystick,* el segundo punto del menú nos permitirá manejar el programa en virtud a las teclas 5, 6, 7, 8 y 0, antes mencionadas. En todo caso, hay que insistir en que este programa no es importante por el trabajo que realiza, sino porque sirve como ejemplo de la forma de controlar la palanca de juego en nuestros propios programas, por lo que será de mayor interés el estudio del mismo, que su utilización.



El resultado de la lectura del* PORT *empleado por el interface* KEMPSTON *(IN 31), debe ser interpretado según los datos expuestos en el gráfico.

Una vez que hayamos seleccionado una opción del menú, aparecerá en el margen izquierdo de la pantalla una zona de dibujo acotada por un rectángulo blanco, y a su derecha, una representación de los siete colores que podemos emplear (el fondo de la pantalla es negro). En la parte inferior, se imprimen las coordenadas del punto de dibujo: **X** para el eje horizontal e **Y** para el vertical.

Para el comienzo del trazado de la línea se ha escogido el punto de coordenadas 107,87.



MANEJO DEL PROGRAMA

Para trazar cualquier línea, sólo tenemos que utilizar el *joystick* o el teclado. Cuando deseemos cambiar el color de escritura, pulsaremos el botón de disparo (o la tecla **0** si hemos elegido la opción de teclado). Acto seguido, una punta de flecha recorrerá cíclicamente el menú de colores que se encuentra a la derecha de la pantalla; cuando el indicador apunte al color que deseamos utilizar, hemos de pulsar nuevamente el botón de disparo, y la tinta quedará fijada para los próximos trazados.

Al analizar la lectura del **PORT** 31, controlado por un interface **KEMPSTON**, debemos tener en cuenta que la lógica seguida es positiva, es decir, las acciones sobre la palanca colocan a 1 los bits correspondientes.

La lógica seguida por los sistemas **AGF, PROTEK** y **CURSOR,** así como por el teclado, es negativa. Por tanto, la pulsación de una tecla, o la acción sobre el **joystick,** sitúan a 0 los bits correspondientes.

Cuando se selecciona un nuevo color, nuestra posición actual se destaca por un círculo blanco que a continuación desaparece, permitiéndonos continuar el trazado del dibujo. Finalmente, podremos utilizar la tinta negra, igual a la de fondo, para «hacer invisibles» determinadas líneas del gráfico. Esto lo conseguiremos pulsando la tecla **B.**

Sólo nos queda recordar que, como ya estudiamos anteriormente, el Spectrum no permite que dos líneas de distinto color coexistan en una misma posición de carácter. Al intentarlo, podremos apreciar cómo la línea trazada con anterioridad adopta el color de la recién llegada.

Una vez introducido el programa, podemos grabarlo en cinta mediante **SAVE "JOY" LINE 1,** si deseamos que se autoejecute al cargarlo; o bien **SAVE "JOY",** para que se realice simplemente la carga.

Para conseguir interpretar el estado del **joystick** con el interface **KEMPSTON,** es necesario realizar la lectura del **PORT** 31 (decimal), lo cual se consigue mediante el comando BASIC **IN 31.** El resultado obtenido debe interpretarse de la siguiente manera: DERECHA = 1, IZQUIERDA = 2, ABAJO = 4, ARRIBA = 8 y DISPARO = 16. Si se desean estudiar resultados combinados, debemos descomponer el movimiento en sus sentidos principales, y sumarlos. Por ejemplo, la diagonal

superior derecha implica el resultado 9, puesto que el desplazamiento hacia arriba produce un valor 8 y el de la derecha un valor 1.

Si obtenemos la forma binaria del resultado, su análisis será mucho más sencillo. De los ocho bits que componen el byte, los tres bits de la izquierda no nos deben importar, ya que sólo son posibles cinco estados simples (ARRIBA, ABAJO, IZQUIERDA, DERECHA y DISPARO). De los cinco bits restantes (los de la derecha), cada uno corresponde a uno de los estados. Concretamente, interpretándolos de derecha a izquierda: DERECHA, IZQUIERDA, ABAJO, ARRIBA y DISPARO. Si el bit estudiado está a 1, indica que el movimiento correspondiente ha sido realizado, en caso contrario, estará a 0.

Veamos un ejemplo. Si el jugador ha realizado un movimiento en la diagonal superior derecha, y al mismo tiempo ha pulsado el botón de disparo, los bits 0, 3 y 4 (contando de derecha a izquierda y empezando por el cero), estarán a 1 (lógicamente, el resto serán ceros). Por tanto, el byte tendría esta forma en binario: 000 11001, lo cual traducido a decimal supone un valor 25 **(PRINT BIN 00011001)**. Efectivamente, descomponiendo el movimiento en sus sentidos principales, obtendríamos: 1 (ARRIBA) + 8 (DERECHA) + 16 (DISPARO) = 25.

Si el interface empleado es el de los cursores **(AGF** o **PROTEK),** podremos seguir procedimientos parecidos a los anteriores. Para estudiar resultados simples, es decir, los que no son combinación de varias acciones, utilizaremos la función BASIC **INKEY$,** de la siguiente forma: DERECHA = "5", ABAJO = "6", ARRIBA = "7", DERECHA = "8" y DISPARO = "0".

Por el contrario, para analizar movimientos compuestos, deberemos recurrir en esta ocasión a estudiar las formas binarias de dos direcciones del **PORT.** El bit 4 del **IN 63486,** indicará el estado de la tecla "5" (DERECHA) y los bits 0, 2, 3 y 4 (contando a partir de cero, y de derecha a izquierda) del **IN 61438,** responderán a los movimientos siguientes: DISPARO, DERECHA, ARRIBA y ABAJO, respectivamente. Para terminar, hemos de destacar algo muy importante: cuando se realizan lecturas del teclado, los bits a 1 indican teclas no pulsadas, y los bits a 0, teclas pulsadas; es decir, la lógica inversa a la utilizada con el interface **KEMPSTON.**

```
  5 REM ************************
  6 REM * J.M.MAYORAL SERRANO *
  7 REM ************************
  8 GO SUB 2090
  9 POKE 23658,8: LET K=9: PAPER 0: INK 9: BORDER 0:
CLS
 10 FOR P=0 TO 6
 20 FOR N=0 TO 2
 30 PRINT  PAPER P+1;AT P*3+N,28;"
 40 NEXT N
 50 NEXT P
 80 FOR Y=175 TO 7 STEP -24
 90 PLOT 255-33,Y
100 DRAW 27,0
110 NEXT Y
120 FOR X=255-33 TO 255 STEP 27
130 PLOT X,8
140 DRAW 0,166
150 NEXT X
160 DATA 215,0,0,153,-215,0,0,-153,74,46,77,46,77,65
,89,79,82,65,76,32,32,32,49,57,56,53
170 PLOT 0,12
180 FOR N=1 TO 4
190 READ A,B
200 DRAW A,B
210 NEXT N
220 FOR N=1 TO 17
225 READ C
230 PRINT AT 0,4+N;CHR$ C
240 NEXT N
250 REM
300 REM MOVIMIENTO CURSOR
305 REM
310 LET X=107: LET Y=87
320 PLOT X,Y
330 PRINT AT 21,5;"X=";X;"  ","Y=";Y;"  "
335 IF E$="2" THEN  GO TO 2180
340 LET I=IN 31
342 IF INKEY$="B" THEN  LET K=0
343 IF I>=16 THEN  GO SUB 1000
345 IF I=0 THEN  LET X=PEEK 23677: LET Y=PEEK 23678:
GO TO 340
350 LET R=LN I/LN 2
360 IF R-INT R<>0 THEN  GO TO 340
370 GO SUB R*100+400
380 GO TO 340
395 REM DERECHA
403 IF X=213 THEN  LET X=X-1: RETURN
405 PLOT  INK K;X,Y
410 LET X=X+1
415 GO SUB 900
420 RETURN
495 REM IZQUIERDA
500 IF X=2 THEN  RETURN
505 PLOT  INK K;X,Y
510 LET X=X-1
515 GO SUB 900
520 RETURN
595 REM ABAJO
600 IF Y=14 THEN  RETURN
605 PLOT  INK K;X,Y
610 LET Y=Y-1
615 GO SUB 900
620 RETURN
695 REM ARRIBA
700 IF Y=163 THEN  RETURN
705 PLOT  INK K;X,Y
710 LET Y=Y+1
715 GO SUB 900
720 RETURN
795 REM PINTA
900 PRINT AT 21,7;X;"  "
910 PRINT AT 21,18;Y;"  "
920 RETURN
1000 REM COLOR
1020 LET F=1
1030 PRINT  OVER 1;AT F,31;"<"
1080 BEEP .2,20: BEEP .3,21
1090 PRINT  OVER 1;AT F,31;"<"
1091 IF E$="1" THEN  GO TO 1095
1093 IF INKEY$="0" THEN  GO TO 2000
1094 GO TO 1100
1095 IF IN 31>=16 THEN  GO TO 2000
1100 LET F=F+3
1105 IF F>=20 THEN  LET F=1
1110 GO TO 1030
1900 DATA 1,1,2,4,3,7,4,10,5,13,6,16,7,19
2000 RESTORE 1900
2010 FOR N=1 TO 7
2020 READ A,B
2030 IF F=B THEN  LET K=A
2040 NEXT N
2050 FOR N=1 TO 4
2055 CIRCLE  OVER 1;X,Y,1
2070 NEXT N
2080 RETURN
2090 REM MENU
2100 FOR N=1 TO 2
2110 PRINT  FLASH 1;AT N*7,3;N
2120 NEXT N
2130 PRINT AT 7,5;"JOYSTICK KEMPSTON"
2140 PRINT AT 14,5;"JOYSTICK & CURSORES"
2150 LET E$=INKEY$
2160 IF CODE E$<49 OR CODE E$>50 THEN  GO TO 2150
2170 RETURN
2180 REM J.CURSOR
2190 IF INKEY$="5" THEN  GO SUB 500
2200 IF INKEY$="6" THEN  GO SUB 600
2210 IF INKEY$="7" THEN  GO SUB 700
2220 IF INKEY$="8" THEN  GO SUB 400
2230 IF INKEY$="0" THEN  GO SUB 1000
2240 IF INKEY$="B" THEN  LET K=0: GO TO 2190
2250 GO TO 2185
```

¡!

Para el control del trazado de la línea, el programa utiliza la variable del sistema **COORDS.** La dirección decimal 23677, contiene la coordenada **X** del último punto trazado, y 23678 la coordenada **Y** del mismo.

*

El sistema **KEMPSTON** considera independientes las acciones sobre el teclado y el *joystick*. Por el contrario, los sistemas **AGF, PROTEK** y **CURSOR,** emulan la pulsación de las teclas 5, 6, 7, 8 y 0, para las acciones sobre el *joystick*.

*

El sistema **KEMPSTON** emplea una sola dirección del **PORT,** la 31 (decimal). Por el contrario, los emuladores de los cursores, precisan de la lectura de dos direcciones: la 63486 y la 61438.

En el análisis de la fila superior del teclado, el bit 4 del IN 63486 corresponde a la tecla "5"; los bits 0, 2, 3 y 4 del IN 61438, se ven afectados por la pulsación de las teclas "0", "8", "7" y "6", respectivamente.