MICRODIGITAL

TK color computer

Manual de Operación

TK
color computer

Proyecto y ejecución editorial:

Departamento de Publicaciones Técnicas de MICRODIGITAL ELETRÔNICA LTDA.

MICRODIGITAL ELETRÔNICA LTDA.

CAJA POSTAL 54.121

SÃO PAULO — BRASIL

Impreso en Brasil — Printed in Brazil
1ª edición — 1987

ÍNDICE

INTRODUCCIÓN

Este manual contiene las siguientes secciones:

— Características técnicas. Sirve a la descripción general del aparato y a las referencias a equipos accesorios y software disponible.

— Instalación. Comprende todas las informaciones y explicaciones necesarias para el perfecto funcionamiento del micro.

— Defectos y Probables Causas. Resumen de los eventuales problemas de funcionamiento del aparato, con la indicación de las providencias que se pueden tomar en cada caso.

— Teclado. Explica cómo se debe manejar el teclado, según la tarea que será realizada.

— Disposición de la Pantalla de TV. Enseña como operar con la pantalla de textos y da detalles sobre la pantalla de gráficos. - El Ordenador como Calculadora ... Efectos Sonoros. Capítulos dedicados al BASIC de su ordenador.

Nota: Si ya posee razonables conocimientos sobre el lenguaje BASIC, podrá dispensar la lectura de esta parte del manual, utilizándola sólo como fuente de consulta para esclarecer alguna duda o para saber cómo usar la función UDG.

— Memoria. Contiene informaciones generales sobre el microprocesador del aparato y el mapa de la RAM.

— Código Máquina. Da los conceptos básicos sobre programación en código máquina y describe las funciones utilizadas para ejecutar rutinas en lenguaje máquina, en programas BASIC.

— Apéndices. Abarcan programas, mensajes del ordenador, variables del sistema, código de caracteres, descripción de los interfaces serial y paralelo, y modo de utilización del light pen.

PRESENTACIÓN

Su micro emplea un lenguaje muy difundido entre los microordenadores tanto por su simplicidad como por su versatilidad, éste es el lenguaje BASIC. Sin embargo, su ordenador posee comandos exclusivos, tales como: INK, PAPER, BORDER, BRIGHT, UDG, SOUND y TRACE.

Mediante el comando INK, Ud. determina el color de los caracteres que aparecerán en la pantalla o el color del trazo que pretende crear.

Con la palabra-clave PAPER, elige el color de fondo de la pantalla.

Por medio del comando BORDER, selecciona el color del margen de la pantalla, independientemente del color de la zona central.

Todavía con la instrucción BRIGHT, realza los ocho colores disponibles, haciendo más claro el colorido de sus producciones.

Su ordenador es capaz de generar el alfabeto entero, en minúsculas y mayúsculas, y aún puede producir caracteres acentuados para los idiomas portugués y castellano, mediante la función UDG. Esta función permite también, que use su creatividad para generar nuevos caracteres y aplicarlos a sus programas.

Si quiere crear música o efectos sonoros en programas, siéntase a gusto: su microordenador dispone del comando SOUND, que permite el control de la tonalidad y del tiempo de duración de las notas, enviando las señales sonoras a través del altavoz de su TV.

Para facilitar la interconexión de su micro a la grabadora, fue adoptado el sistema "easy-load" (carga fácil). Este sistema asegura mayor eficiencia en lectura y grabación de cintas magnéticas, además de presentar, en su pantalla, un "feedback" de lo que sucede durante la operación de carga.

Tan importante, o más aún, es la posibilidad ofrecida por el modo TRACE, que le permite seguir la evolución del ordenador a la hora de ejecutar un programa BASIC. Esta condición lo ayuda a detectar posibles errores lógicos en su programa.

Finalmente, otra característica digna de hacerse notar, es la de poder insertar varios comandos en una misma línea de programa. Para eso, basta separarlos por dos puntos. Este recurso es bastante utilizado en ordenadores de mayor porte, lo que pone aún más en evidencia que su micro es una máquina poderosa y compacta.

Características Técnicas

CARACTERISTICAS TÉCNICAS

1. Unidad Central de Proceso (CPU)

La CPU de su ordenador es el microprocesador Z80A, de 8 bits, con reloj (clock) de sonido de 3,58MHz.

2. Memoria

Su micro posee 16 Kbytes de memoria permanente (ROM), donde se hallan el interpretador BASIC y los controles de video, cassette y equipos accesorios.

La memoria programable (RAM) es de 48 Kbytes.

3. Entradas y Salidas (I/O)

Teclado tipo "qwerty", teclas con repetición automática, señalizadas sonoramente cuando son pulsadas; salida de sonido por el altavoz de la TV; entrada y salida para grabadora cassette; enchufe de TV de blanco y negro o color, sistema PAL-N; enchufe para ampliación de memoria, interface serial, impresora, light pen, mother board, etc.; interface para joystick incorporado.

4. Corriente Eléctrica

Su microordenador puede conectarse a redes eléctricas de 110 ó 220 V CA. La Fuente de Alimentación debe ajustarse según la tensión de la red local.

5. Pantalla

El TK puede enviar señales de vídeo en modo normal o inverso (TRUE VIDEO e INV. VIDEO). Presenta recursos de parpadeo (FLASH); brillo (BRIGHT); control de color de cada carácter (INK), de color de fondo (PAPEL) y de color de borde (BORDER).

La pantalla de textos se compone de 24 líneas X 32 columnas. La pantalla de gráficos permite la presentación de 256 X 192 puntos, en alta resolución.

Su micro dispone de ocho colores, en pantallas de textos y de gráficos:

0 - negro	2 - rojo	4 - verde	6 - amarillo
1 - azul	3 - magenta	5 - cyan	7 - blanco

6. Teclado

El teclado está encargado del envío de caracteres y acceso de funciones y BASIC, según la distribuición que sigue:

— 80 comandos, instrucciones y funciones BASIC preprogramadas

— 14 funciones matemáticas
— 55 caracteres alfanuméricos
— 26 caracteres alfanuméricos en mayúsculas y minúsculas
— 8 caracteres gráficos

Mediante la función UDG, Ud. aún podrá obtener los caracteres a continuación:

UDG Ø/1 — 38 caracteres acentuados para los idiomas portugués y español

UDG 2 — 21 caracteres gráficos definidos por el usuario

7. Cursor

El TK opera en diversos modos, que se indican por el cursor:

— K: para acceder a comandos y algunas funciones BASIC (palabras-clave)
— L: para generar caracteres alfanuméricos y accionar determinadas funciones BASIC
— C: para acceder sólo a caracteres en versal (mayúsculas)
— E: para acceder a la mayoría de las funciones BASIC
— G: para acceder a caracteres gráficos

8. Equipos y Dispositivos Accesorios

La salida de expansión del micro posibilita la conexión de varios equipos y dispositivos accesorios disponibles en el mercado.

IMPRESORA

Al ordenador se puede acoplar una impresora paralela norma Centronics o una impresora serial.

Los comandos que actúan sobre este equipo son: LPRINT, LLIST, TAB y COPY.

LÁPIZ ÓPTICO

El lápiz óptico (light pen) permite que se detecten los puntos luminosos emitidos por la pantalla de TV y se los envíen directamente al ordenador. Posibilita crear gráficos y figuras, y elegir opción, en programas que contienen menú.

Nota: Consulte el Apéndice E, para saber cómo manejar este equipo.

OTROS

Además de los equipos citados, pueden conectarse otros, entre ellos:

- Serial Interface RS232-C (interface serial).

  Permite la conexión del micro a sistemas, vía teléfono, y a equipos con entrada-salida serial. Consulte el Apéndice G para obtener más detalles sobre esta placa.

- Parallel Printer Interface (interface paralelo).

- Posibilita la conexión del micro a una impresora paralela. Vea, en el Apéndice F, una descripción más detallada de esta placa.

- Game Control

  Joystick - teclas correspondientes: 6, 7, 8, 9 y 0.

- Mother Board

  Permite que se conecte más de un dispositivo accesorio en el enchufe de expansión del micro.

9. Lenguajes

El TK emplea el BASIC, el más popular de los lenguajes de programación. El BASIC (Beginner's All-purpose Symbolic Instruction Code) es un código de aprendizaje muy simple, que combina instrucciones en inglés con símbolos matemáticos. Es un lenguaje de alto nivel, pues su lógica está cercana a la estructura lógica del lenguaje humano.

Su ordenador además puede operar con lenguajes como el Assembler y el Logo, entre otros.

10. Software

En el mercado hay una considerable cantidad de programas desarrollados para su ordenador. Son softwares que dispensan las tareas de programación, dando al usuario verdaderas "herramientas" de trabajo o instrumentos de entretenimiento ya listos para uso.

Puede disponer, entre otros, de los tipos de programas a continuación:

— Profesionales y comerciales

  - Multifile (base de datos) . Multitext (procesador de textos)
  - Softcalc (planilla electrónica de cálculos)
  - Control Comercial
  - Cálculo de Estructuras
  - Engecalc

— Utilitarios

- Assembler Z80 (ensamblador)
- TK BUG (monitor desensamblador)
- Compilador BASIC

— Educativos

— Juegos animados e inteligentes

11. Publicaciones

La lista de publicaciones sobre su micro es muy larga. Puede contar con toda la literatura que trata de la línea Sinclair y con libros específicos sobre el TK, además de revistas y otros tipos de publicaciones internacionales. Además, hay en el mercado una vasta literatura sobre el microordenador ZX-Spectrum (el cual es compatible con su TK Color Computer).

Instalación

INSTALACIÓN

Para que su ordenador quede listo para funcionar, hay que instalarlo adecuadamente, haciéndose las conexiones y ajustes conforme a las instrucciones a continuación:

1. Conexiones

Es necesario identificar todos los enchufes de su micro, antes que lo conecte a cualquier cosa. Abajo están enumerados todos estos, indicándose a qué se destinan:

ENCHUFES		DESTINACIÓN
VERSIÓN PROFESIONAL	VERSIÓN STANDARD	
⎓	DC	conexión de la fuente de alimentación
	EXPANSION	conexión de placas y dispositivos accesorios
	JOYSTICK	conexión del Game Control
	MIC	conexión para grabación de cintas cassette
	EAR	conexión para lectura de cintas cassette
	TV	conexión de la TV

2. Conexión del Micro a la Corriente Eléctrica y la TV

2.1. Fuente de Alimentación

La fuente de alimentación es un aparato encargado de la captación y transmisión de energía eléctrica para funcionamiento del ordenador.

Este aparato tiene que estar de acuerdo con las especificaciones que siguen:

a) Estar ajustado a la red eléctrica local (110 ó 220 V).

b) Tener capacidad para suministrar la energía total necesaria al micro cuando éste está conectado a otros equipos.

c) Ser controlado por un estabilizador de voltaje, en sitios donde hay mucha oscilación en el flujo de corriente alternada.

Para adecuar la Fuente a la red eléctrica del sitio en donde instalará su ordenador, basta que ponga el conmutador de voltaje en 110 ó 220 V.

Su micro está preparado para conectarse a equipos accesorios compatibles existentes en el mercado. Entre dichos equipos, puede haber algunos que consuman más energía de la que es suministrada por la fuente de alimentación normal. Así, si desea hacer conexión de equipos a su micro, consulte el servicio técnico autorizado y sepa qué tipo de fuente deberá usar.

En caso de que sea preciso utilizar un estabilizador de voltaje, se aconseja consultar el servicio técnico autorizado.

Si ya se ha cerciorado de que está con la fuente de alimentación adecuada y ajustada en el voltaje de la red eléctrica local, siga las instrucciones a continuación:

- Introduzca el pequeño conectador del cable de la fuente de alimentación en el enchufe del micro destinado a la fuente.
- Ponga la llave conecta/desconecta de la fuente de alimentación en OFF.
- Conecte el conectador del otro cable de la fuente en un enchufe CA.

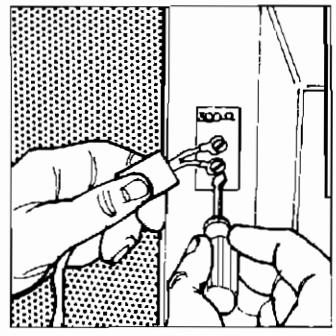
2.2. TV

La TV es el principal dispositivo de salida que su ordenador utiliza para comunicarse con Ud. Por medio de la pantalla de la televisión, Ud. toma conocimiento de las actividades del ordenador y recibe las informaciones más inmediatas sobre los datos procesados o en proceso.

Este aparato puede ser de blanco y negro o color, en el sistema PAL-N.

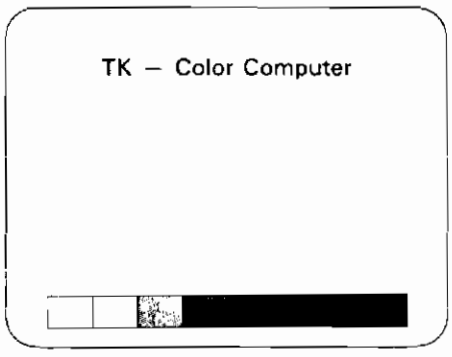
Para conectar la TV al ordenador y sintonizarla correctamente, siga las instrucciones abajo:

- Desconecte la antena de TV y, en cambio de ésta, instale el cable de vídeo.



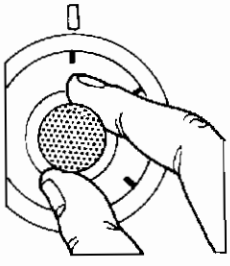
- Introduzca el otro conectador del cable de vídeo en el enchufe del micro destinado a la TV.
- Conecte la TV y sintonícela en el canal 3.
- Conecte el ordenador, poniendo la llave de la fuente de alimentación en ON.

Si la sintonía de la frecuencia de la TV está captando la señales del ordenador, obtendrá la siguiente pantalla de presentación:



En caso de que la imagen no haya sido obtenida, se debe ajustar la sintonía fina del televisior:

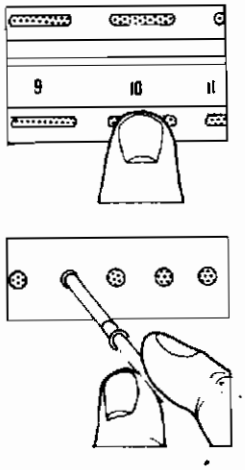
SELETOR TRADICIONAL



- Gire el mando selector de canales hacia un determinado sentido, hasta que obtenga imagen.

  Si es preciso, invierta el sentido e intente nuevamente.

SISTEMA PUSH-BUTTON



- Busque el botón correspondiente al canal 3 y gírelo hacia la derecha e izquierda hasta que obtenga la pantalla de presentación.

Si tiene una TV en color, verá en la pantalla los ocho colores que el TK produce:

blanco	amarillo	cyan	verde	magenta	rojo	azul	negro

- Busque los mandos de color, brillo y contraste de su TV, y ajústelos hasta obtener la mejor presentación. (La tonalidad de los colores puede variar conforme el aparato de TV utilizado.)

Si su TV es de blanco y negro, será presentada una escala de tonalidades que varía del gris claro al negro.

3. Grabadora Cassette

La grabadora cassette le permite guardar sus programas en cinta magnética, de forma que no los pierda cuando desconecte el ordenador. Siempre que desee, podrá cargarlos desde la cinta hacia la memoria del ordenador.

Por medio de la grabadora podrá todavía cargar en la memoria del ordenador los software en cinta disponibles en el mercado.

Para la grabación de cintas, se debe conectar la grabadora al micro de la manera que sigue:

- Conecte el cable de lectura/grabación al enchufe MIC de la grabadora y al enchufe del TK destinado a la grabación de cintas.

Para cargar el contenido de las cintas hasta la memoria del TK, conecte la cassette al micro del modo que sigue:

- Conecte el cable de lectura/grabación al enchufe EAR de la grabadora y al enchufe del TK destinado a la lectura de cintas cassette.

4. Equipos Accesorios

Para conectar equipos al microordenador, hay que seguir las instrucciones de los manuales de instalación de esos aparatos.

Los equipos se conectan a su ordenador a través del enchufe de expansión. En la versión profesional del TK, dicho enchufe está protegido por una tapa, la cual quitará cuando quiera usarlo.

4.1. Señales del Enchufe de Expansión

Este enchufe permite el paso de 56 señales en el orden siguiente:

1 - A14
2 - A12
3 - 5 Volts
4 - 9 Volts
5 - guía
6 - GND
7 - no utilizado
8 - Clock
9 - AO
10 - A1
11 - A2
12 - A3
13 - IORQUE
14 - GND
15 - GND
16 - no utilizado
17 - no utilizado
18 - no utilizado
19 - BUSREQ
20 - RESET
21 - A7
22 - A6
23 - A5
24 - A4
25 - ROMCS
26 - BUSACK
27 - A9
28 - A11

56 - A15
55 - A13
54 - D7
53 - SOUND IN
52 - guía
51 - D0
50 - D1
49 - D2
48 - D6
47 - D5
46 - D3
45 - D4
44 - INT
43 - NMI
42 - HALT
41 - MREQ
40 - IOREQ
39 - RD
38 - WR
37 - 5 Volts
36 - WAIT
35 - 12 Volts
34 - 12 Volts
33 - M1
32 - RFSH
31 - A8
30 - A10
29 - no utilizado

5. Defectos y Probables Causas

En caso de que haya problemas durante la operación de su TK, consulte el cuadro a continuación. Si no consigue resolver el problema, diríjase al servicio técnico autorizado.

DEFECTOS	CAUSAS PROBABLES	PROVIDENCIAS
La pantalla de presentación no aparece cuando conecta el ordenador.	No hay alimentación de corriente alternada.	Verifique la conexión del conectador de la fuente al enchufe del TK.
	No fueron obedecidas las instrucciones para conectar el ordenador y sus accessorios.	Hay que conectar todos los accesorios antes de conectar el ordenador.
	Accesorio no conectado correctamente.	Verifique la conexión.
	Su televisión no está ajustada.	Verifique el contraste, el brilho y la sintonía fina.
	El cable del vídeo no fue conectado correctamente.	Conéctelo según las instrucciones de este manual.
Ud. no consigue cargar su programa en cinta cassette.	La conexión de la cassette no fue hecha correctamente.	Verifique las instrucciones de conexión de la grabadora.
	El volumen de su grabadora está muy bajo.	En este caso las franjas indicadoras de carga (b carga (BORDER) no aparecerán. Aumente el volumen de su grabadora.
	El cabezal de su grabadora puede estar desalineado.	Para alinearlo, coloque una cinta de programas y presione "PLAY" (o tecla correspondiente) de la grabadora. Con una herramienta apropiada (destornillador o destornillador Philips), gire el tornillo de ajuste del cabezal hacia la derecha o la izquierda hasta obtener el sonido más estridente y alto posible. Este tornillo suele hallarse del lado izquierdo del cabezal de grabación/lectura de su grabadora.
El ordenador se "traba" durante la operación, siendo necesario desconectar el aparato	Hubo fluctuación de energía.	Cerciórese de que la alimentación está bien.
	Conexión de la fuente con defecto o instalada erróneamente.	Verifique si todos los cables de conexión están conectados o si ellos no están dañados.
	Error de programación.	Revea el programa.

Teclado

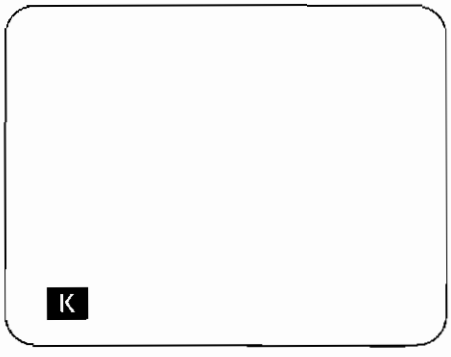
TECLADO

El teclado es la vía por la cual registra caracteres para hacer la introducción de datos en la pantalla y la memoria del ordenador, y es el medio por el cual pide tareas de proceso al aparato.

1. Modos de Comandar el Teclado

El teclado comprende caracteres alfabéticos (mayúsculas y minúsculas), numéricos, simbólicos y gráficos, comandos y funciones. A los comandos (palabras-clave) se accede al oprimir las teclas en las cuales están escritos. Ej.: POKE, PRINT, LIST son comandos directos del teclado. No es preciso teclearlos letra por letra.

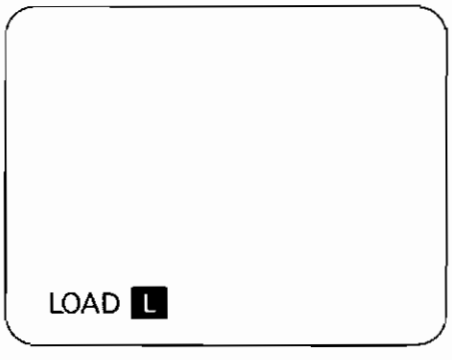
El ordenador opera en diversos modos que se indican por el cursor. Este, representado inicialmente por una letra **K** , en la parte inferior izquierda de la pantalla, informa que el micro está esperando un comando. El cursor también muestra la posición del próximo carácter que será tecleado.



Las palabras-clave aparecen siempre en el comienzo de una línea de programación o después del comando THEN.

Vamos a usar el comando LOAD (contenido en la letra J) a ver qué sucede.

Repare que, después que se introduce un comando, el cursor cambia automáticamente a **L** (ahora a la derecha de la palabra-clave que ha introducido).



La letra **L** dice que el ordenador está aguardando un carácter, y no un comando o función. Teclee cualquier cosa para observar el comportamiento del teclado.

Pruebe a mantener una tecla cualquiera oprimida, durante un rato. Notará que su ordenador está dotado de un repetidor automático de caracteres. Ello facilita la tarea de teclear.

Después de estas experiencias con el teclado, su pantalla debe estar repleta de caracteres sin significado. Es la hora de utilizar otro modo de comando.

2. Borrador — DELETE

Borre los caracteres impresos en la pantalla, utilizando el comando DELETE.

teclado standard — oprima <CAPS SHIFT> y <0>, simultáneamente
teclado profesional — pulse la tecla <DELETE>

3. Caracteres en Mayúscula — CAPS SHIFT y CAPS LOCK

Mantenga <CAPS SHIFT> oprimida y teclee caracteres alfabéticos (de A a Z): está produciendo letras mayúsculas.

Suelte la tecla <CAPS SHIFT> para que los caracteres aparezcan en minúsculas, de manera similar a que sucede en una máquina de escribir.

Nota: Los signos < y > se utilizan para identificar una tecla, por lo tanto no serán tecleados. Una instrucción del tipo <ENTER> le dice que oprima sólo la tecla ENTER.

Ahora pruebe a entrar al modo **C** . En este modo, los caracteres se imprimen solamente en mayúsculas. Para acceder al modo **C** , haga así:

teclado standard — oprima <CAPS SHIFT> y <2>, simultáneamente
teclado profesional — pulse <CAPS LOCK>

Importa que se observe que CAPS LOCK sólo actúa sobre los caracteres, y no sobre los comandos y funciones.

Borre todo, usando el comando DELETE o desconectando el ordenador.

Nota: Para desactivar cualquier tecla de cambio de modo, vuelva a presionar la misma tecla que pulsó para activar.

4. SYMBOL SHIFT

Repare que hay palabras-clave y símbolos escritos en la parte superior derecha de la mayoría de las teclas. Para generarlos, es preciso que oprima esas teclas en conjunto con la tecla <SYMBOL SHIFT>.

En la versión standard del TK, estos signos están escritos en rojo.

5. Espacios Blancos

Teclee, por ejemplo, 2 + 2. Para obtener espacios entre los caracteres, use la tecla que funciona como la barra de espacio de una máquina de escribir:

teclado standard — <SPACE>
teclado profesional — <espacio>

6. Modo Extendido — EXTENDED MODE

En la mayoría de las teclas se encuentran varias otras funciones. En la versión profesional esas funciones están escritas en la cara frontal de cada tecla, dispuestas en una o dos hileras. En la versión standard, están escritas en azul, encima de cada tecla, o en rojo y abajo de éstas. Dichas funciones pertenecen al modo extendido de comando. El cursor que representa este modo usa la letra **E** .

Para acceder al modo extendido (**E**), proceda así:

teclado standard — presione <CAPS SHIFT> y <SYMBOL SHIFT>, simultáneamente
teclado profesional — pulse <EXTENDED MODE>

Para seleccionar la función deseada, desde el modo extendido teclee:

teclado standard
función en azul → sólo la tecla correspondiente
función en rojo → <SYMBOL SHIFT> + tecla correspondiente

teclado profesional
función de la hilera superior → sólo la tecla correspondiente
función de la hilera inferior → <SYMBOL SHIFT> + la tecla correspondiente

7. Modo Gráfico — GRAPHICS

El cursor usa **G** para representar el modo gráfico. Hay ocho caracteres gráficos, contenidos en las teclas numéricas (1-8). Otros caracteres gráficos pueden ser definidos por el usuario, como se enseña en el capítulo "El Conjunto de Caracteres".

Para activar el modo gráfico:

teclado standard — oprima <CAPS SHIFT> y <9>, conjuntamente
teclado profesional — presione <GRAPHICS>.

El modo gráfico se desactiva mediante las mismas teclas usadas para accionarlo.

8. Video Invertido — INVERSE VIDEO y TRUE VIDEO

El comando INVERSE VIDEO sirve para invertir el carácter tecleado, cambiando el color de éste con lo del fondo:

teclado standard - pulse <CAPS SHIFT> y <4>, simultáneamente
teclado profesional — presione <INV. VIDEO>.

Para volver al video normal, se usa el comando TRUE VIDEO:

teclado standard - oprima <CAPS SHIFT> y <3>, en conjunto
teclado profesional — presione <TRUE VIDEO>.

9. Modo Programado

Es necesario saber que, para utilizar el ordenador de manera programada, y no en el modo directo, se deben organizar las instrucciones en una secuencia lógica, inserta en líneas numeradas en orden creciente.

Cuando el ordenador está enseñando el cursor K , además de comandos del tipo palabra-clave, acepta también numeración de líneas de programación.

Luego de teclear número, el cursor queda en K para la introducción de una palabra-clave.

Haga una prueba, tecleando primero el número 10. A continuación pulse la letra P. El comando PRINT aparecerá en la pantalla. Tras el comando, el cursor pasa automáticamente a L.

10. ENTER

La tecla <ENTER> indica al ordenador que la línea de comando o instrucción de proceso acabó. Después que presiona <ENTER>, el ordenador examina lo que fue escrito. Si la instrucción es inteligible, será aceptada; si es algo incomprensible, aparecerá, en el lugar del error, un signo de interrogación (?) y, eventualmente, un mensaje de error. Si sucede ello, la línea de comando tiene que ser corregida o borrada.

El signo ? puede indicar aún la omisión de algun carácter necesario a la instrucción.

11. Modo Edición — EDIT

El modo edición está explicado detalladamente en el capítulo "Programación BASIC". Se accede a dicho modo mediante el comando EDIT, permitiendo la edición de la línea del programa en donde se encuentra el cursor:

teclado standard — pulse <CAPS SHIFT> y <1>, simultáneamente
teclado profesional — presione <EDIT>.

En este modo, las cuatro teclas flechas (< ← > < → > < ↑ > < ↓ >) se utilizan para mover el cursor, desplazándolo en sentido y dirección indicados por las flechas.

Disposición
da la Pantalla de TV

DISPOSICIÓN DE LA PANTALLA DE TELEVISIÓN

1. Pantalla de Textos

La pantalla tiene 24 líneas y 32 columnas.

Los comandos introducidos aparecen en la margen inferior del vídeo. Cuando es editada una línea de comando y Ud. usa <ENTER>, ésta pasa de la margen inferior de la pantalla, para la parte superior.

El ordenador va a disponer cada instrucción en una lista ordenada, siempre en orden creciente de números de línea.

El resultado de las operaciones, como también las salidas de los programas, aparecen siempre en la parte superior del video, a partir de la última línea ocupada, si ésta no ha sido borrada.

Cuando la pantalla es ocupada hasta la vigésimo segunda línea, el ordenador se encargara de mover toda la imagen una línea para arriba (scroll) sin alterar la margen inferior. Si este proceso implica en la pérdida de una línea que Ud. puede no haber tenido la oportunidad de ver, el ordenador interrumpe el proceso, preguntando si Ud. lo autoriza para que mueva el contenido de la pantalla. La pregunta aparece como "scroll?", en el ángulo inferior izquierdo del vídeo. Si la respuesta es <N>, <SPACE> o <STOP>, el programa o cualquier actividad en la pantalla, será interrumpida. En el borde inferior aparecerá el mensaje D BREAK-CONT repite Ø:1. Si la respuesta no es una de estas 3 teclas, el "scroll" se efectuará.

1.1 Margen Inferior

La margen inferior comprende dos líneas y es reservada para la entrada de datos (INPUT), líneas de programa, mensajes de error y comandos. Cuando hay necesidad, el mensaje inferior avanza para arriba, adaptándose al tamaño de su contenido. La última línea cronológicamente introducida en el programa se llama línea corriente y es indicada por el símbolo >.

2. Pantalla de Gráficos

Para el trazado de gráficos, se puede considerar la pantalla de éste ordenador como una matriz dividida en 296 x 192 elementos de imagen de alta resolución.

Su ordenador posibilita la creación de trazados gráficos con detalles y con aprovechamiento de los ocho colores disponibles.

El modo de tratamiento de la pantalla permite la combinación de gráficos y textos de manera que se puede dispensar la utilización de "ventanas".

En el capítulo "Gráficos" se encuentran las instrucciones BASIC usada para producir gráficos.

El Ordenador como Calculadora

EL ORDENADOR COMO CALCULADORA

Puede utilizar su ordenador como una calculadora. Para eso no hay ninguna complicación: basta que emplee el llamado modo inmediato del lenguaje BASIC.

En este capítulo mostraremos cómo hacer para que el ordenador efectúe las operaciones algebraicas mediante el modo inmediato. En los capítulos sobre programación BASIC, Ud. encontrará explicaciones sobre operaciones matemáticas más complejas.

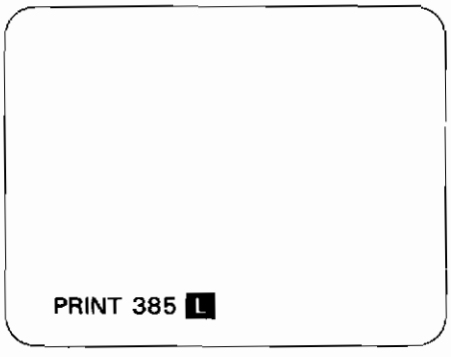
1. Modo Inmediato

Modo inmediato es el modo del proceso que el ordenador realiza cuando Ud. emplea el lenguaje BASIC directamente, o sea, sin utilizar este lenguaje en un programa. En este modo, también denominado modo directo, el TK procesa inmediatamente las instrucciones, y la pantalla presenta el resultado de la tarea que solicitó a su ordenador.

Efectúe dos pequeños ejercicios:

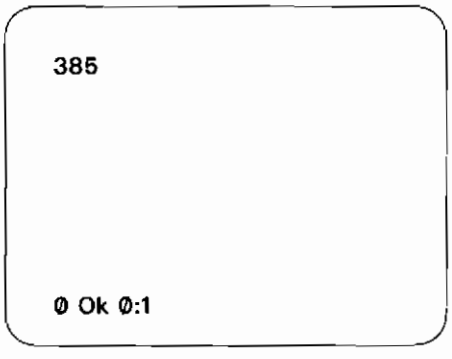
a) Pulse el comando <PRINT> y el número 385.

La pantalla presentará:



b) Presione ahora la tecla <ENTER>.

La pantalla mostrará:



2. Los Cálculos

El lenguaje BASIC usa una simbología propia, para expresar las operaciones algebraicas. Observe el cuadro que sigue:

TIPO DE OPERACIÓN	SÍMBOLO EMPLEADO
SUMA	+
SUBSTRACCIÓN	–
MULTIPLICACIÓN	*
DIVISIÓN	/
POTENCIACIÓN	↑

En el modo inmediato, los cálculos algebraicos son realizados mediante el comando PRINT más el símbolo matemático correspondiente en el BASIC, a cada tipo de operación. Solamente la radicación se realiza de otra forma, como veremos más adelante.

Pruebe a sumar 8 y 2:

Para introducir: **PRINT 8 + 2**

Teclee: **<PRINT> <8> <SYMBOL SHIFT> y <K> <2> <ENTER>**

El resultado (10) aparecerá en el ángulo superior izquierdo de la pantalla, como ocurrió cuando digitó sólo PRINT 385. Si no desconectó el ordenador desde entonces y no efectuó ninguna otra instrucción, el número 10 aparecerá abajo del 385, en la segunda línea.

Trate de obtener una división, por ejemplo 25/2 (vea en el cuadro arriba el símbolo usado en BASIC para representar la división).

Debe haber tecleado <PRINT> <25> </> <2> <ENTER> y obtenido 12.5, debajo del número 10.

Practique las operaciones, variando números, símbolos algebraicos e incrementando parcelas, multiplicadores, etc.

Si intenta una división por cero, obtendrá un mensaje de error.

Nota: Mensajes de error serán tratados específicamente en el Apéndice B.

2.1. Potenciación

Para elevar un número a una potencia, deberá introducir:

<PRINT> (base) < ↑ > (exponente), presionando <ENTER> al final.

Nota: Se obtiene ↑ al oprimir <SYMBOL SHIFT> y <H>. Este símbolo no se hay de confundir con la tecla-flecha hacia arriba.

Ejemplo:

Elevaremos 4 a la quinta potencia:

<PRINT> 4 < ↑ > <5> <ENTER>

Veremos el resultado en la primera línea disponible de la parte superior de la pantalla.

2.2. Radicación

La radicación es la operación matemática por la cual calculamos la potencia fraccionaria de un número. Siendo así, para calcular la raíz de un número, deberá elevarlo a la potencia de exponente igual a 1 dividido por el radical.

Ejemplo:

Calcularemos la raíz quinta de 9:

<PRINT> 9 < ↑ > <1/5> <ENTER>

Después del <ENTER> veremos la respuesta: 1.5518456

2.3. Raiz Cuadrada

Para extraer la raíz cuadrada de un número, tiene dos opciones:

a) Podrá teclear el comando <PRINT>, la función SQR , y poner el radicando después de SQR. Finalice con <ENTER>.

Extraeremos la raíz cuadrada de 16:

teclado standard - **<PRINT> <CAPS SHIFT> y <SYMBOL SHIFT> <H> 16**

teclado profesional - **<PRINT> <EXTENDED MODE> <H> 16**

El resultado aparecerá en la pantalla, debajo de la última línea utilizada.

b) Puede proceder como en el punto 2.2:

PRINT 16↑ (1/2), que equivale a SQR 16.

3. Expresiones Algebraicas

Su ordenador realiza las operaciones algebraicas de acuerdo con un orden de precedencia establecido:

1º) potenciación
2º) multiplicación/división
3º) suma/substracción

Si quiere cambiar este orden, deberá emplear paréntesis en lugar de los símbolos de asociación.

Ejemplos:

Calcularemos 25 + 10-5 * 2-75/3

Obtendremos 0 como resultado, pues el ordenador seguirá el orden establecido.

Calcularemos 25 + (10-5) * 2-75/3, el resultado será 10.

Para resolver [25 + (10-5)] * 2-75/3, sustituya los corchetes por paréntesis (25 + (10-5)) * 2-75/3 y obtendrá 35.

Como en el ejemplo anterior, sustituya las llaves y los corchetes de la expresión

25 + [10-[5 * 2-(75/3)]] por paréntesis:

25 + (10-(5 * 2-(75/3)))

No se olvide del comando <PRINT>, en el inicio, y de <ENTER> en el final. El resultado será 50.

Los Números en el Ordenador

LOS NÚMEROS EN EL ORDENADOR

1. Notación Científica

La notación científica es un recurso empleado por el ordenador, para resumir la grafía de números muy extensos. En esta notación los números tienen el siguiente formato:

número E ± ee, donde:

número = base exponencial
E = * 10↑ (multiplicado por 10 elevado al exponente)
± = signo positivo o negativo del exponente
ee = exponente

NOTACIÓN CIENTÍFICA	NOTACIÓN TK	OPERACIÓN
2345679 X 10	2.345679E+11	2345679*10 11
5746389 X 10	5.746389E+26	5746389*10 26

2. Limites Numéricos

Su micro hace operaciones con los números reales comprendidos en la franja de −1.7E38 a 1.7E38. Intentos de cálculo o impresión de números fuera de este límite producirán error de sintaxis, con el signo ? indicando el límite.

3. Forma de los Números

Se emplea punto en vez de coma en los números fraccionarios.

El cero a la izquierda del punto decimal no es considerado.

El cero a la derecha del último número significativo después de la coma (punto) decimal es eliminado.

Números con más de 8 guarismos entre las partes entera y decimal son redondeados a más (último guarismo ⩾ 5) o menos (<5).

Números muy próximos a cero (entre -3E-39 y 3E-39) son redondeados.

Programación Basic

PROGRAMACIÓN BASIC

1. Conceptos

Hasta ahora Ud. tuvo que repetir varias veces los comandos para que el ordenador resolviera sus ejercicios. Los comandos o instrucciones pueden ser introducidos en líneas de programas para evitar que tenga que repetirlos cada vez que desee la ejecución de una operación.

Al conjunto de instrucciones con las cuales se informa al ordenador para que realice las tareas, se le da el nombre de "programa".

Los programas deben ser escritos en un lenguaje que el ordenador pueda entender. Su micro comprende el lenguaje BASIC, por lo tanto, Ud. empleará este lenguaje, para comunicarse con él.

Los Datos

En lenguaje BASIC se distinguen dos tipos principales de datos: los numéricos y los alfanuméricos. Cada uno de ellos recibe un tratamiento diferente del ordenador.

Los datos numéricos son representados, normalmente, por guarismos (de 0 a 9). Por ejemplo:

23 y .01234 son datos numéricos representados por guarismos arábigos.

Variables

Las variables son almacenadores de valores a los que se les atribuyen determinados contenidos. El contenido de una variable puede ser de dos tipos: un valor numérico o una secuencia de caracteres entre comillas, denominada cadena (string). Siendo así, el valor 5736 puede ser el contenido de una variable numérica y "DIEGO" el de una variable string. Cualquier contenido entre comillas es considerado un string aunque se utilice una secuencia de caracteres numéricos, tal como "5.736". En este caso los guarismos no representan un valor, sino una cadena de caracteres. La única excepción para el contenido de un string son las comillas. Un string no puede valer comillas. Por lo tanto, los espacios en blanco son considerados caracteres, entonces " " (un espacio entre comillas) puede ser atribuido como contenido de una variable alfanumérica.

5736
variable numérica

DIEGO
variable string

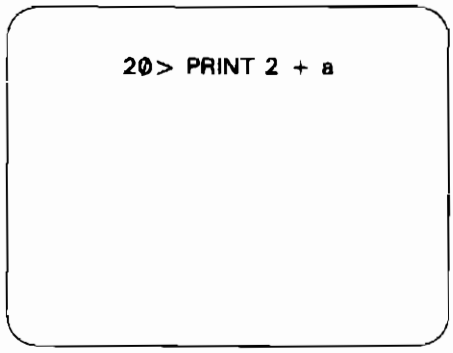
2. Comandos

PRINT

Ya tuvo la oportunidad de experimentar el comando PRINT en el modo inmediato. Pruébelo ahora en un programa. Teclee:

PRINT 2 + a <ENTER>

De esta forma, en vez de ser ejecutado el comando, éste será almacenado en una línea de programa.



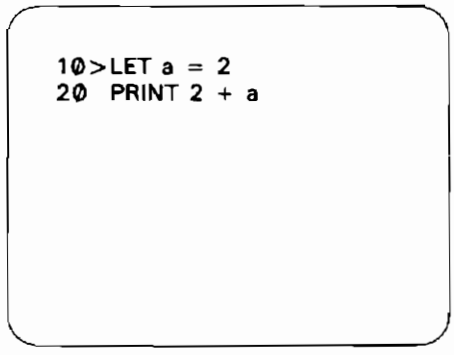
LET

Introduzca una nueva línea con numeración inferior a aquélla de la pantalla:

10 LET a = 2

El comando LET atribuye un valor a una variable. A la variable después del LET le es atribuido el valor 2 expreso a la derecha del signo de igualdad.

Cuando presione <ENTER>, la línea 10 pasará del margen inferior a la posición correcta que ocupa en el programa, pues el orden creciente es obedecido, no importando en que momento la línea del programa es introducida. Su pantalla deberá contener el listado del programa:

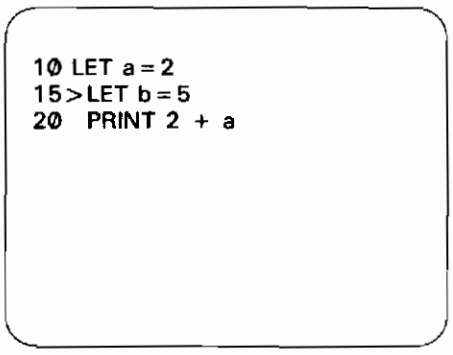


Una tercera línea será introducida, esta vez con numeración intermedia:

15 LET b = 5

Nota: Siempre enumere sus programas de forma tal que permitan la introducción de nuevas líneas intermedias. Se aconseja la numeración de 10 en 10. Las líneas pueden usar la numeración de 1 a 9999.

Presione <ENTER> y el listado será:



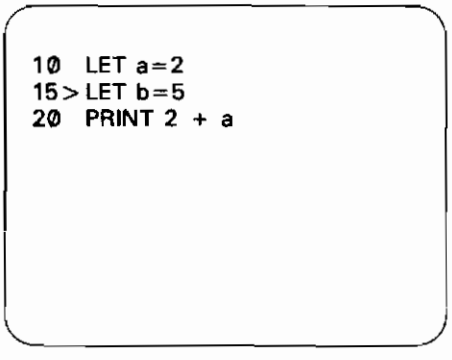
Con la introducción de la línea 15, la línea 20 puede ser modificada, pues ahora tenemos una nueva variable (b) y podemos introducirla en el cálculo.

EDIT

Podría teclear nuevamente la línea 20 con el fin de transformarla, pero existe una forma más práctica, principalmente indicada en el tratamiento de líneas más extensas, ya que la nueva introducción se haría fastidiosa y lenta. Esta forma más práctica de modificar una línea, es editarla, y ello se hace por medio del comando EDIT

EDIT permite la edición (corrección de la línea-corriente en el margen inferior de la pantalla. La línea-corriente es indicada por un cursor representado por el signo ▄ .

Accione el comando EDIT y note que el cursor ahora está indicando la línea 15, la última línea a ser introducida en el programa.

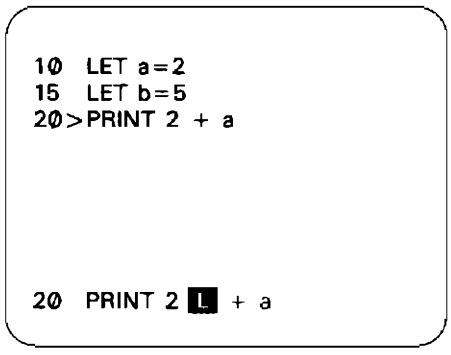


Mueva el cursor de la línea corriente, usando las teclas-flecha < ↑ > o < ↓ >, y colóquelo en la línea que desea editar. En este momento quiere editar la línea 20, por lo tanto basta mover el cursor de línea corriente una posición hacia abajo.

A continuación accione EDIT. Obtiene entonces una copia de la línea 20 en el margen inferior de la pantalla. Obtiene además un cursor K , esperando que Ud. lo desplace hacia la derecha del carácter a ser modificado.

Para mover el cursor en el sentido horizontal, utilice las teclas-flecha < → > y < ← >, que no alteran el contenido de la línea, sólo pasan el cursor por sobre los caracateres.

Usando la tecla-flecha a la derecha, mueva el cursor hacia la posición entre el número 2 y el signo + :



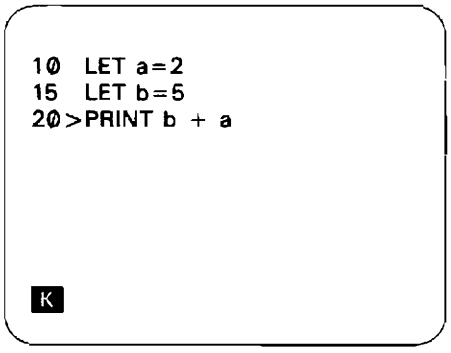
Nota: En el modo de edición, el cursor salta automáticamente los caracteres del comando — pruebe a moverlo hacia a trás y hacia adelante del comando PRINT. Espacios adecuados van siendo creados si introduce nuevos caracteres en la línea que está siendo editada.

DELETE

El cursor está a la derecha del número 2, para que cambie este número por la letra b. Si simplemente digita b, este carácter será agregado a la línea, pero el número 2 permanecerá. Hay la necesidad de eliminar este guarismo, antes o después de insertar b. Es más práctico eliminar antes de introducir, para tener una visión mejor de lo que está siendo editado y no necesitar volver a colocar el cursor en posición.

Para eliminar el 2, use el ya conocido comando DELETE. Si utiliza DELETE sólo una vez, elimina el número y ya tiene el cursor en posición para introducir la letra b. Hágalo y finalice la operación con <ENTER>.

Después de este proceso, su ordenador mostrará:



Por ahora, lo que fue hecho no produjo resultados prácticos. Tenemos un programa, pero hasta ahora no obtuvimos respuestas para el cálculo que éste contiene.

RUN

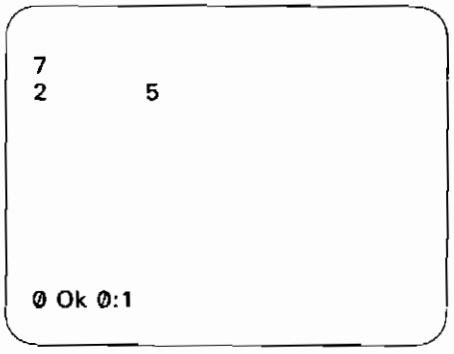
La ejecución de un programa requiere que Ud. dé un orden al ordenador. Este orden es el comando <RUN> seguido de <ENTER>. La palabra-clave RUN acciona la ejecución del programa. Ejecútelo y obtendrá en la pantalla el resultado (7) del cálculo.

Nota: RUN puede ser usado seguido por un número de línea. En este caso el programa será ejecutado a partir de esa línea.

Haga que el ordenador pruebe que todavía "recuerda" las variables que fueron creadas por su programa. Introduzca, en modo inmediato:

PRINT a,b

En la pantalla aparecerán, además del 7 ya existente, los números 2 y 5, en la siguiente disposición:



Ya debe haber percibido que el número 5 apareció en la misma línea que el 2, pero a algunas columnas adelante (columna 16). Esto sucedió porque utilizó la coma entre las variables, en el comando PRINT. Haga la experiencia con otro PRINT, pero esta vez use punto y coma, así:

<PRINT> a;b <ENTER>

Lo que Ud. ve en la pantalla no es el número 25, y sí los números 2 y 5 ocupando columnas adyacentes. Pruebe:

<PRINT> a <ENTER>

y a continuación:

<PRINT> b <ENTER>

Las dos respuestas usarán dos líneas distintas y subsiguientes. Lo mismo sucede si usa PRINT a'b. El apóstrofo genera una línea entre los elementos que separa.

CLS

Limpie la pantalla con <CLS> y <ENTER>. Hecho esto, obtenga el listado del programa, para introducir más alteraciones.

LIST

El comando LIST ejecuta la operación de listar.

Nota: De ahora en adelante, no será más necesario decirle que presione <ENTER>, pues ya debe saber cuándo necesita hacerlo.

Puede utilizar LIST para hacer saltar el cursor de la línea corriente directamente a la línea que quiere editar.

En listados muy largos no sería conveniente mover el cursor línea por línea. Es más fácil comandar:

<LIST> (nº de línea)

y después **<EDIT>**

Vamos a crear otras líneas en su programa para ilustrar lo que ha acabado de aprender. Introduzca (no se olvide del <ENTER>):

```
30 <PRINT>
40 <PRINT>
50 <PRINT>
60 <PRINT>
```

En la pantalla tendrá:

```
10 LET a=2
15 LET b=5
20 PRINT b+a
30 PRINT
40 PRINT
50 PRINT
60>PRINT
```

Fíjese en la posición del cursor de línea corriente. Supongamos que lo quiera en la línea 30. En vez de subirlo con la flecha, pulse <LIST> 30. Además de aparecer el cursor en la línea deseada, sucederá otro hecho: Las líneas anteriores a 30 desaparecerán en el nuevo listado. Esto no tiene importancia, pues ellas continuarán en la memoria del ordenador, y un próximo LIST las traerá de vuelta. Mientras tanto, no necesitamos de ellas.

La pantalla se comporta como una ventana a través de la cual Ud. puede ver las partes del programa almacenado en la memoria del TK.

Lo que su pantalla presenta ahora es:

```
30>PRINT
40  PRINT
50  PRINT
60  PRINT




0 Ok 0:1
```

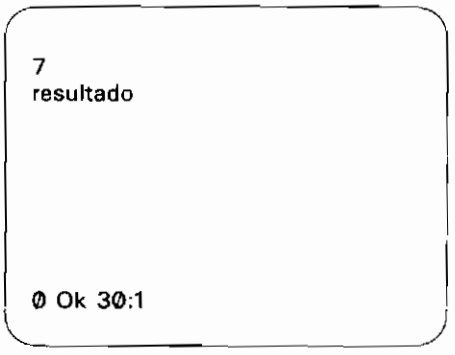
Está todo listo para usar EDIT y traer una réplica de la línea 30 para la edición.

Aproveche la edición de la línea 30 para agregar a ella nuevos elementos.

Coloque el cursor **L** a la derecha de PRINT (use < → >). Teclee "resultado". No se olvide de las comillas. Finalice la edición con <ENTER>. El listado completo de su programa reaparecerá en su pantalla.

Vamos a eliminar las líneas 40, 50 y 60, pues no forman parte del objetivo del programa. Para eliminar una línea, teclee el número de ésta, seguido del <ENTER>. Cuidado con esta operación, pues es irreversible.

Quedaron cuatro líneas en el programa (10, 15, 20 y 30). Ejecútelo. El resultado obtenido no será muy lógico:



Borre todo con CLS y haga un nuevo listado del programa.

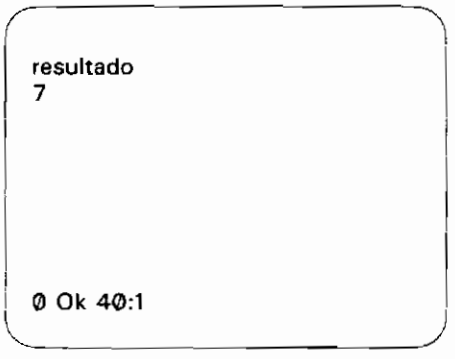
Nota: Para eliminar a la vez los caracteres del margen inferior, sin borrar toda la pantalla y sin usar DELETE, utilice EDIT. Así que aparezca la línea editada en el fondo de la pantalla, presione <ENTER> y ésta volverá al programa, habiendo vaciado la zona inferior de la pantalla.

Pruebe a editar la línea 20 y cambiar su numeración para 40.

Si el cursor de línea-corriente no está apareciendo, liste la línea 20 o accione la tecla de movimiento < ↑ >. Esto lo hará aparecer.

Después de corregir la línea 20, transformando su número en 40, cierre la edición y liste el programa. Notará que la línea 20 permanece en el listado, y que la línea 40 es una copia de ella. Tendremos que eliminar la 20 de la manera simple que ha visto un poco más arriba: tecleando 20 <ENTER>.

Ahora está todo listo para intentar nuevamente. Presione <RUN>. La salida del programa está con un aspecto más lógico:



No se puede decir lo mismo en lo que se refiere a su estética. Para una pantalla con 22 líneas y 32 columnas disponibles, el aprovechamiento del espacio deja mucho que desear.

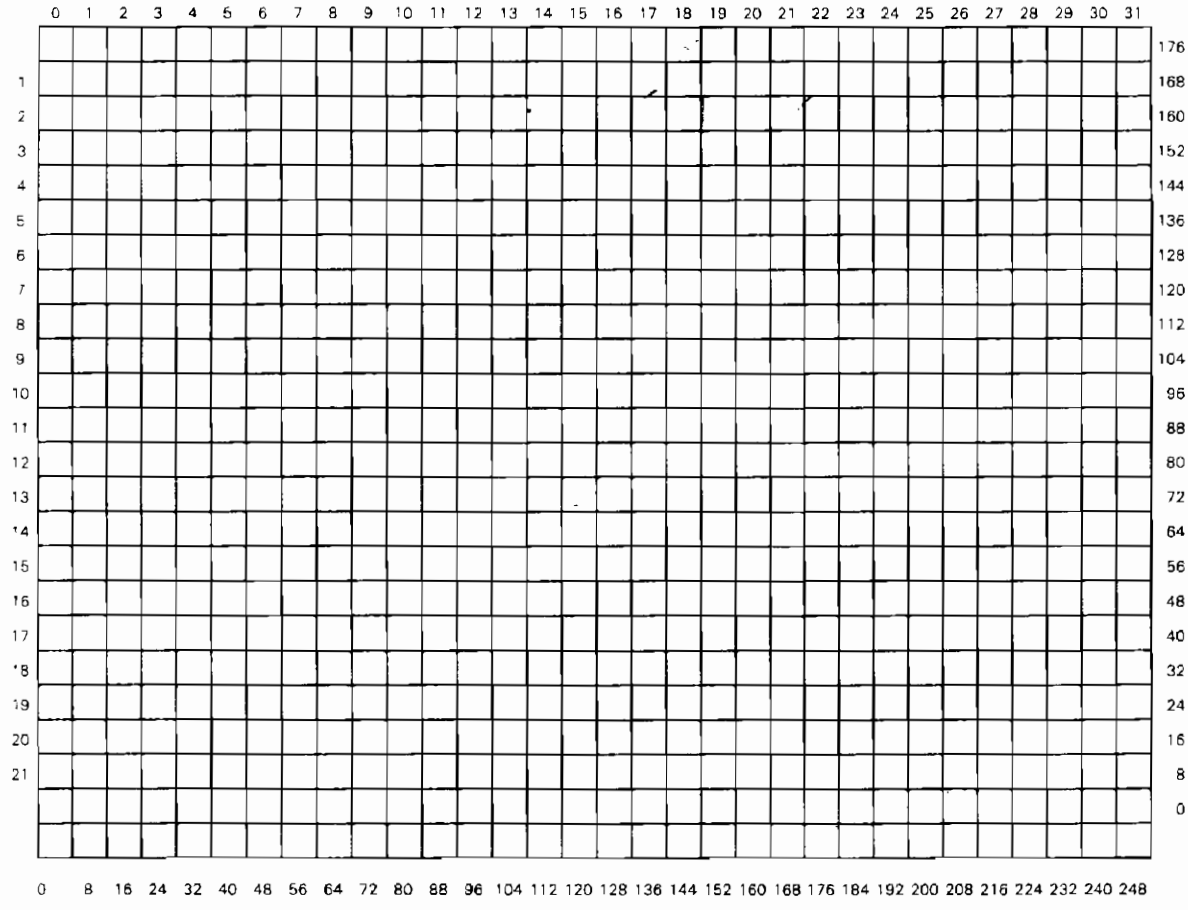
AT

Puede controlar la disposión espacial aprendiendo a usar la función AT, que viene precedida de PRINT. Su forma es:

PRINT AT (línea), (columna); (contenido)

Introduzca, en el modo inmediato: <PRINT><AT>11,10;"(su nombre)" <ENTER>

Su nombre aparecerá más o menos en el centro de la pantalla. Puede ajustar la centralización, seleccionando los números de línea y columna con ayuda de la matriz de pantalla de texto que sigue:



En lo que se refiere a la presentación visual de su programa, vamos a hacer alguna mejoría:

Limpie la pantalla y liste el programa.

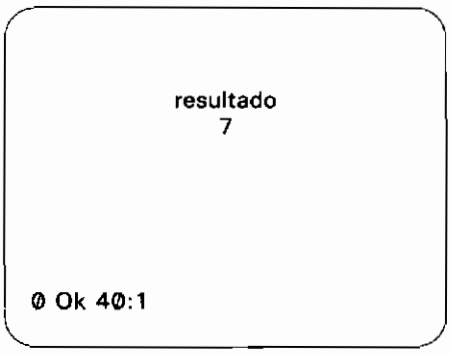
Edite la línea 30 y modifíquela:

30 PRINT AT 9,12;"resultado"

Haga lo mismo con la 40, variando línea y columna:

40 PRINT AT 12,16;b + a

Ejecute el programa y obtendrá:

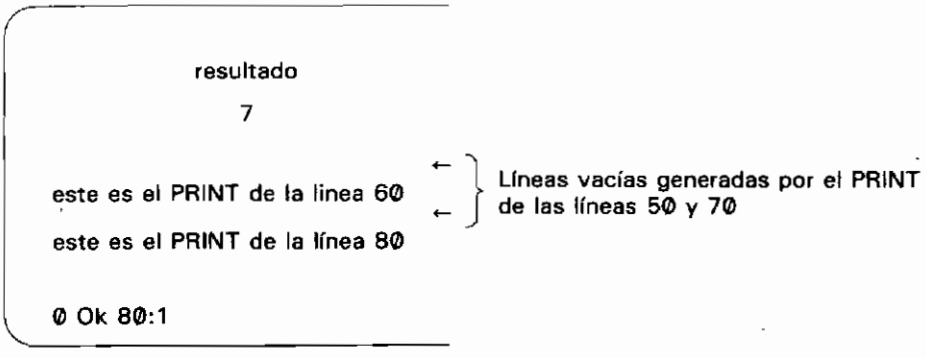


El comando PRINT, por sí solo, encierra una instrucción de saltar una línea. Pruebe esto introduciendo en el programa nuevas líneas:

```
50 PRINT
60 PRINT "este es el PRINT de la linea 60"
70 PRINT
80 PRINT "este es el PRINT de la linea 80"
```

Nota: El PRINT que está entre comillas tendrá que ser escrito letra por letra, pues en este caso es sólo una palabra y no un comando.

Al ejecutar este programa obtendrá:



TAB

Establece la posición la columna especificada para uso del PRINT. Su mayor valor en una línea corresponde a TAB 32 (última columna). Caso de que este valor sea ultrapasado, los datos que serán imprimidos se desplazarán hacia la línea siguiente. Ejemplo:

```
TAB 33 corresponde a TAB 1 = 33-32
TAB 38 corresponde a TAB 6 = 38-32
```

Uso práctico:

```
PRINT TAB 6;"columna 6";TAB 10;"columna 10"
```

NEW

Modifique su programa a gusto, aplicando todos los comandos que aprendió hasta ahora. Cuando desee borrar todo el programa de la memoria, utilice un comando que evita que tenga que desconectar el ordenador para borrar el contenido de la RAM: el comando NEW, que actúa hasta el RAMTOP.

NEW ''vacía'' la memoria del ordenador, destruyendo todas las instrucciones y variables que fueron acumuladas desde que Ud. lo conectó, o desde el último NEW.

Es necesario asegurarse siempre de que realmente desea eliminar el contenido de la memoria RAM, pues es imposible recuperarlo, a no ser que haya sido ''salvado'' en una cinta cassette (como verá en el capítulo específico sobre el asunto — ''Almacenamiento y Carga — Grabadora Cassette''). Después de teclear <NEW>, el ordenador asumirá la actitud de comienzo, como si hubiese conectado recién el aparato.

Para hacer desaparecer el mensaje de presentación, presione <ENTER>. A continuación aparecerá el cursor K .

REM

Si desea insertar en el listado de su programa algún comentario o mensaje, hágalo después de la instrucción REM, como en el ejemplo:

```
10 CLS
20 REM esta linea no altera el programa
30 PRINT ''mi nombre es''
40 PRINT ''(su nombre)'':REM teclee el nombre
50 REM aqui termina el programa
60 PRINT ''fin''
```

Esta instrucción hace que el programa ignore todo lo que insertó en la línea, después del REM:

: (Dos puntos)

Reparó en los puntos (:) de la línea 40? Los dos puntos separan dos comandos del programa. No hubo necesidad de abrir otra línea de programa para la introducción del segundo comando. Observe que el cursor estaba preparado para identificar esta situación, tanto es así que pasó de L a K después de la entrada de los dos puntos. En realidad, puede escribir varios comandos en una misma línea, separándolos por dos puntos. De esta forma economizará espacio en la memoria de su ordenador. Puede hacerlo incluso en el modo inmediato.

Ejecute el programa anterior. Después introduzca la siguiente línea:

```
55 PRINT: PRINT: PRINT
```

Ejecute nuevamente el programa, para observar el efecto. Como era previsto, fueron introducidas tres líneas vacías entre su nombre y ''fin''.

Pruebe a usar dos puntos en otros programas y habitúese a introducir REM en programas cuyo listado quiere preservar. Este recurso le ayudará mucho en la identificación de lo que el programa hace y cómo lo hace.

GOTO

Es muy común que encontremos en los programas BASIC la instrucción GOTO. Este comando hace que el programa se desvíe hacia una determinada línea.

Introduzca una línea en su programa, conteniendo esta instrucción. Primero colóquela con numeración 25, como en el ejemplo:

```
10 CLS
20 REM esta linea no altera el programa
25 GOTO 60
30 PRINT "mi nombre es"
40 PRINT "(su nombre)": REM teclee su nombre
50 REM aqui termina el programa
55 PRINT: PRINT: PRINT
60 PRINT "fin"
```

Nota: Si especifica una línea inexistente después de GOTO, el ordenador accederá a la próxima línea existente. Si no hay ninguna otra línea, el programa parará.

Ejecute el programa. Observe el resultado: al alcanzar la línea 25, el programa salta directamente a la línea 60, ignorando las líneas intermedias.

Puede colocar el GOTO en otras líneas del programa, para verificar como éste actúa. Trate de colocarlo en la línea 70, eliminando la línea 25, de la siguiente forma:

```
70 GOTO 30
```

Ejecute el programa.

Solamente saldrá del bucle cuando utilice BREAK o responda negativamente al "scroll?".

Puede usar GOTO (nº línea) en el modo inmediato cuando esté con un programa en la memoria. Es diferente de RUN (nº de línea), pues éste anula las variables y GOTO no las altera.

INPUT

Abandone el programa y limpie la memoria con <NEW>. A continuación teclee:

```
10 PRINT "teclee una palabra"
20 INPUT a$
30 INPUT "teclee un numero"
40 INPUT b
50 PRINT: PRINT "a$ = ";a$,"b = ";b
60 PRINT: GOTO 10
```

En este programa aparece un nuevo comando denominado INPUT. INPUT, así como LET, permite la asociación entre un nombre de variable y su valor. Sin embargo, INPUT detiene el programa hasta que se introduzca el dato que está siendo pedido.

A cada pasaje por el INPUT, puede sustituir el contenido de la variable, sea ésta numérica o string (alfanumérica). Cuando el ordenador se encuentre a la espera del valor de una variable numérica, el cursor permanecerá en **L** , y se colocará en el margen inferior izquierdo de la pantalla.

Cuando Ud. teclee el valor y <ENTER>, la variable asumirá el valor que haya introducido. Cuando la variable sea del tipo "string", el cursor en el modo **L** estará localizado en el mismo lugar, pero entre comillas. Esto es coherente con la necesidad de colocar comillas en los valores "string". El ordenador se encarga de colocarlas automáticamente en el INPUT.

El INPUT puede incluir un comentario, mensaje o cualquier "string", pero es necesario separar tal "string" de la variable por un punto y coma, en la línea del programa. Este "string" aparecerá en el margen inferior de la pantalla, precediendo al cursor y tendrá por función informar sobre el tipo de dato que ha que ser introducido en ese momento. Por ejemplo:

```
<NEW>
10 CLS
20 INPUT "teclee una palabra"; a$
30 INPUT "teclee un numero";b
40 PRINT "a$ = ";a$,"b = ";b
50 GOTO 20
```

INPUT LINE

Una variación muy útil del comando INPUT es la instrucción INPUT LINE. Este comando permite la entrada de strings, que serán asociados a las respectivas variables. La diferencia entre este INPUT y lo que ya ha sido presentado es que, con LINE, el ordenador omite las comillas que siempre acompañan el INPUT. Esto le permite introducirlas en el string, tratándolas como cualquier carácter.

El formato de esta instrucción es:

INPUT LINE variable string

Se puede usar INPUT LINE también en el modo inmediato.

Nota: INPUT LINE no acepta interrupción por medio del comando STOP.

STOP

Para detener un programa en el momento del INPUT, use STOP (<SYMBOL SHIFT> + <A>). Si el INPUT está a la espera de un "string", elimine las comillas o mueva el cursor hacia la izquierda de ellas. Si no lo hace, el ordenador entenderá el STOP como el "string" que está atribuyendo a la variable del programa, y no lo interrumpirá.

CONT

Si pretende continuar la ejecución del programa después de una interrupción del tipo BREAK, STOP o negativa al "scroll?", use <CONT> (continúe). Esta instrucción hace que el ordenador vuelva a la ejecución del programa.

Cuando el comando CONT es accionado después del mensaje "D BREAK-CONT repite", el recomienzo se iniciará en el mismo punto en que paró, repitiendo la última acción ejecutada.

Después del mensaje "L BREAK-CONT prosigue", el recomienzo se iniciará en la primera instrucción después de la última acción ejecutada. La instrucción CONT actúa de manera peculiar cuando responda "n" a la pregunta "scroll?", y el programa se detenga.

Pruebe a teclear 25 líneas de programa conteniendo sólo "REM", así:

```
1 REM
2 REM
3 REM
4 REM
.
.
.
25 REM
<LIST> <ENTER>
```

A la vez que en la pantalla caben sólo 22 líneas, aparecerá el mensaje "scroll?". Responda <n>. Otro mensaje aparecerá. Esta vez contiene:

D BREAK-CONT repite Ø:1

Responda con CONT <ENTER>

El margen inferior de la pantalla desaparece. El ordenador está en bucle, porque CONT repite la primera instrucción de la línea de comando: en este caso, la instrucción LIST. CONT será ejecutada indefinidamente hasta que sea interrumpida. Para hacerlo, utilice BREAK. Recibirá el mensaje:

L BREAK-CONT prosigue Ø:1

Nada de esto ocurre si colocamos dos puntos (:) antes de LIST. Verifique:

: LIST <ENTER>

Cuando aparezca "scroll?", pulse <n>. Después del mensaje, introduzca CONT.

Esta vez aparecerá:

Ø Ok Ø:1

porque la instrucción CONT salta a la segunda instrucción de la línea, que es el fin. Si usa:

::LIST

CONT saltará a la tercera (e inexistente) instrucción de la línea, generando el mensaje:

N Comando perdido Ø:2

De estas dos maneras se evita el bucle.

Antes de pasar al próximo capítulo, haga algunos programas, intentando usar todos los comandos que aprendió hasta aquí.

Decisiones

DECISIONES

1. IF ... THEN ... STOP

Entre las características más significativas del ordenador, se destaca su capacidad de tomar decisiones.

Hasta ahora ha visto el ordenador seguir precisamente las instrucciones que programó, sin una participación activa en la resolución de algún problema. Sin embargo, él es capaz de ser mucho más útil, pues consigue resolver, además de problemas matemáticos, cuestiones que impliquen lógica. Basta que lo programe para eso.

Un programa lógico podría tener la siguiente estructura:

```
10 Limpie la pantalla
20 Reciba una información Colóquela en la pantalla
30 Reciba otra información, colóquela en la pantalla
40 Si la primera es igual a la segunda, entonces pase a la línea 60.
50 Escriba (1ª información) diferente de la (2ª información): Pare.
60 Escriba (1ª información) = (2ª información)
70 Pare
```

en BASIC:

```
<NEW>
10 CLS
20 INPUT "teclee informacion ";a$:PRINT AT 6,8;"1a.información;a$
30 INPUT "digite informacion ";b$:PRINT AT 8,8;"2a.informacion ";b$
40 IF a$=b$ THEN GOTO 60
50 PRINT AT 12,10;a$;" < > ";b$:STOP
60 PRINT AT 10,8,a$;" = ";b$
70 STOP
```

Observe que en la línea 40 el programa fuerza al ordenador á comparar los strings a$ y b$, y decidir:

Si a$ es igual a b$, el programa es desviado para la línea 60. Si a$ es diferente (< >) de b$, el programa continúa en la misma secuencia, o sea, en la línea inmediatamente posterior (50).

La función del STOP de la línea 50 es bien clara, pues si no fuese colocado ahí, el programa continuaría hasta el fin, sin respetar las condiciones del IF.

Es evidente que su ordenador puede trabajar con programas mucho más difíciles y complejos, involucrando strings y/o valores a ser comparados, pero el principio del método de comparación, es similar al que acabamos de ver.

El formato del IF es siempre IF (condición) THEN (instrucción).

Trate de reemplazar los strings por números o expresiones algebraicas, sin olvidarse de cambiar las variables a numéricas.

Elabore un programa con mayores dificultades y comprobará la eficiencia de su ordenador como herramienta de trabajo.

2. Comparando Datos

A continuación tiene un cuadro de operadores matemáticos y lógicos, para comparación de datos.

SÍMBOLO BASIC	MATEM.	APLICACIÓN
>	>	Es mayor que
<	<	Es menor que
=	=	Es igual a
< =	⩽	Es menor o igual a
> =	⩾	Es mayor o igual a
< >	≠	Es diferente de

OPERADORES LÓGICOS	
AND (y)	relación AND otra relación
OR (o)	relación OR otra relación
NOT (no)	relación NOT

Nota: **< =, > = y < >, así como AND, OR y NOT tienen teclas propias, por tanto no es necesario teclear carácter por carácter.**

Estos operadores se aplican tanto a la manipulación de datos numéricos como también de strings. Recuerde, no obstante, que no es posible comparar un string con un número y viceversa.

2.1. Comparaciones Numéricas

Cuando compara números, el ordenador sigue los principios de la matemática. En el programa siguiente verá como:

```
<NEW>
 10 INPUT "teclee valor a ";a
 20 CLS
 30 INPUT "teclee valor b ";b
 40 PRINT AT 6,5;"a= ";a
 50 PRINT AT 8,5;"b= ";b
 60 IF a>b THEN GOTO 100
 70 IF a=b THEN GOTO 120
 80 PRINT AT 10,5;a;" es menor que ";b
 90 GOTO 10
100 PRINT AT 10,5;a;" es mayor que ";b
110 GOTO 10
120 PRINT AT 10,5;a;" es igual a ";b
130 GOTO 10
```

En las comparaciones con los signos <= y >=, basta que una sola condición sea satisfecha, para que la respuesta sea considerada positiva, y sea cumplida la instrucción del IF.

Por ejemplo:

```
IF x<= 0 THEN PRINT "OK"
```

Para satisfacer la condición del IF, da lo mismo que a la X le sea atribuido un valor mayor o igual a cero.

Las relaciones pueden ser combinadas mediante uso de operadores lógicos AND, OR y NOT, en los siguientes formatos:

una relación AND otra relación

```
IF X < 0 AND A = 100 THEN GOTO 120
```

La instrucción después del THEN sólo es ejecutada si ambas condiciones son satisfechas.

una relación OR otra relación

```
IF y = 0 OR m = 100 THEN STOP
```

La instrución STOP será obedecida siempre que una de las condiciones sea satisfecha.

NOT relación

```
IF NOT P = 0 THEN PRINT P
```

El PRINT será ejecutado siempre que la relación sea falsa, o sea, siempre que P sea distinto de cero.

2.2. Comparaciones Alfanumericas

No sólo podemos comparar números, sino también strings. Al comparar cadenas de caracteres, debemos distinguir el significado que los signos tienen para los strings del que tienen para los números. Los signos "mayor que" (>) y "menor que" (<) distinguen dos strings por su orden, según el Código de Caracteres (Apéndice D), por lo tanto:

```
"A" < "AA"          "BILLÓN"<"MILLÓN"
"AB" < "ABA"        "DIEZ" > "CIEN"
"AC" > "AB"
```

y así sucesivamente.

Los signos <= ,>=, <> e = actúan de la misma forma.

Ejemplos que podrán estar contenidos en programas:

```
IF x$ = "SI" THEN GOTO 30
IF n$ <> "CARLOS" THEN PRINT N$
IF s$> "FZZZZ" AND f$="FIN" THEN STOP
```

Instrucción For ... Next

INSTRUCCIÓN FOR ... NEXT

La instrucción FOR ... NEXT comprende dos comandos: el comando FOR y el comando NEXT. Esta instrucción hace que sean repetidas, un cierto número de veces, las instrucciones que se encuentran entre los órdenes FOR y NEXT.

Limpie da memoria del ordenador con <NEW> y acceda al comando CAPS LOCK. A continuación teclee:

```
10 CLS
20 FOR I=1 TO 5
30 PRINT "TK"
40 NEXT I
<RUN>
```

Como puede observar, el nombre "TK" se presenta cinco veces en la pantalla.

Vamos a analizar el formato general de esta instrucción:

```
FOR variable = valor inicial de la variable TO valor final
.
.
.
líneas de instrucción a ser repetidas
.
.
.
NEXT variable
```

Analicemos cómo funciona el programa que acabó de introducir:

El ordenador coloca el valor inicial en la variable (la variable de una instrucción FOR...NEXT tiene que ser una única letra).

El ordenador verifica si el valor de la variable "I" es menor que el valor final (5).

Si es menor, el programa prosigue normalmente hasta encontrar el comando NEXT I (línea 40). En este momento, el valor inicial de la variable es incrementado en una unidad.

El nuevo valor de I (ahora I + 1) es nuevamente comparado con el valor final de la variable y, si es aún menor, el programa salta a la línea siguiente a la del FOR.

El ciclo se repite hasta que el valor final de I se iguale a 5. A continuación sucede el último ciclo, pues al pasar por NEXT, I pasará a valer 6, y el ordenador al comparar 6 con el valor final determinado para I, hará que el bucle sea abandonado, prosiguiendo el programa en la primera instrucción, después de NEXT.

Notas: 1) **Coloque siempre después del NEXT la misma variable que inició el ciclo (la que fue definida por el FOR). Use siempre variables numéricas.**

2) **Si usa inadecuadamente la instrucción FOR...NEXT, recibirá uno de los siguientes mensajes de error: ''1 NEXT sin FOR'', ''2 Variable inexistente '' o ''I FOR sin NEXT''. Consulte el Apéndice B, para saber el significado de dichos informes.**

Se puede insertar una instrucción FOR ... NEXT en una única línea, tal como:

```
20 FOR I=0 TO 100: PRINT I: NEXT I
```

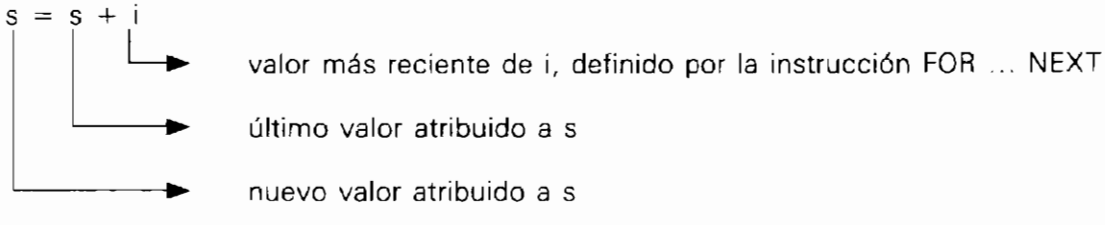
1. Uso de FOR ... NEXT como Artificio Matematico

El potencial de esta instrucción es muy grande. En este ítem explicaremos la aplicación de FOR ... NEXT como artificio matemático, dentro de un programa.

El programa ejemplo que vamos a describir, tiene por función computar la suma de los enteros positivos entre 1 y N. Digite:

```
<NEW>
 5 REM calculo de la suma de enteros positivos entre 1 y n
10 CLS
20 INPUT ''n= '';n
30 LET s=0
40 FOR i=1 TO n
50 LET s=s+i
60 NEXT i
70 PRINT ''la suma es '';s
80 GOTO 20
```

La variable ''s'' (que en este caso indica suma) es usada en las líneas 30, 50 y 70. En la línea 30, que es procesada antes del comando FOR, el valor cero es atribuido a ésta. La línea 50, entre los comandos FOR y NEXT, es ejecutada para i=1, i=2, i=3 y así en adelante, hasta i=n. El nuevo valor de ''s'' pasa a ser:



Por ejemplo, suponga que se atribuya el valor 3 a la variable ''n'', entonces ''s'' tendrá los valores:

línea 30	s = 0
1ª ejecución de la línea 50	s = s+i = 0+1 = 1
2ª ejecución de la línea 50	s = s+i = 1+2 = 3
última ejecución de la línea 50	s = s+i = 3+3 = 6

Ejecute el programa diversas veces, variando siempre el valor de n.

2. STEP

Esta instrucción hace que se cambie el ''paso'' de la instrucción FOR ... NEXT de 1, para el número que desee.

Recuerde que cada vez que el programa pasaba por NEXT, el valor de la variable indicada en el comando FOR era incrementado en una unidad, hasta que se igualase al valor final. Si junto a FOR hay la orden STEP, a cada instrucción NEXT, la variable pasará a ser aumentada con el valor indicado después del orden STEP. Este valor puede ser positivo o negativo.

Observe el ejemplo:

```
<NEW>
10 CLS
20 FOR j=0 TO 10 STEP 2
30 PRINT j
40 NEXT j
<RUN>
```

Cuando el incremento es negativo, el programa continúa el ciclo mientras la variable determinada por el FOR sea mayor o igual al límite.

Se debe tener cuidado al realizar dos o más bucles FOR...NEXT al mismo tiempo, o sea, ''bucles anidados''. El siguiente programa muestra eso:

```
<NEW>
10 FOR m=0 TO 10
20 FOR n=0 TO m
30 PRINT m;":";n;":";
40 NEXT n
50 PRINT
60 NEXT m
```

El bucle n está íntegramente adentro del bucle m, por lo tanto, están encuadrados de manera correcta. Es imperativo que los bucles no se crucen, pues si esto sucede, el programa podrá no funcionar adecuadamente.

Trate de invertir las variables, solamente en las líneas 40 y 60, por ejemplo. Obtendrá un resultado bien diferente, ya que la lógica del programa se alterará.

Subrutinas

SUBRUTINAS

A veces, diferentes partes de un programa tienen que ejecutar la misma tarea, repitiéndose las instrucciones innecesariamente. Puede evitar la repetición de estas líneas al escribir el programa, convirtiéndolas en subrutinas y usándolas cuando sea necesario.

Para conseguir este efecto, use los comandos GOSUB y RETURN, en el siguiente formato:

```
10 GOSUB n
20
.
.
REM principio de la subrutina n
.
.
RETURN
```

Esta instrucción significa: desvíe hacia la línea n; ejecute todas las instrucciones hasta encontrar RETURN. Vuelva a la línea inmediatamente siguiente a GOSUB.

Entenderá mejor como funciona GOSUB y RETURN en los siguientes programas:

```
<NEW>
 10 INPUT "introduzca un numero ";a
 20 IF a > 10 THEN GOSUB 100
 30 PRINT AT 10,0;"secuencia normal del programa":PRINT
 40 PRINT "ejecutando la linea 40"
 50 PRINT:PRINT "linea 50-fin-":STOP
 60 REM principio de la subrutina
100 CLS
110 PRINT AT 5,0;a;" > 10
115 PRINT "hubo desvio hacia la linea 100"
120 FOR c = 1 TO 500: NEXT c
125 CLS
130 RETURN
```

En el ejemplo que vamos a describir se demuestra cómo una subrutina puede desviarse hacia otra subrutina. Trate de entender la lógica antes de ejecutar el programa.

Limpie la memoria del ordenador mediante <NEW> y acceda al comando CAPS LOCK.

A continuación, teclee:

```
 15 PRINT AT 7,2;"EL GOSUB DESVIA EL PROGRAMA"
 20 PRINT TAB 2;"HACIA UNA SUBRUTINA LA CUAL"
 25 GOSUB 500
 30 GOSUB 100
 35 PRINT TAB 2;"AL ENCONTRAR A LA INSTRUCCION"
 40 PRINT TAB 2;"RETURN."
 45 STOP
100 REM ******** SUBRUTINA 100
105 PRINT TAB 2;"ES PROCESADA RETORNANDO"
110 PRINT TAB 2;"A LA LINEA SIGUIENTE AL GOSUB"
115 GOSUB 500
120 RETURN
500 REM ******** SUBRUTINA 500
505 FOR X=1 TO 700: NEXT X
510 RETURN
```

Modo Trace:
Recurso de Depuración

MODO TRACE RECURSO DE DEPURACIÓN

Utilizado como apoyo en la elaboración de programas, este comando es una poderosa herramienta de depuración.

Cuando se trabaja con programas complejos, con muchas condiciones y desvios hacia subrutinas, la posibilidad de errores de lógica se hace mayor, como consecuencia de las diversas opciones del lenguaje BASIC. El comando TRACE posibilita la monitoración de las líneas ejecutadas por el programa, haciendo así posible la detección de eventuales errores y, en general, la verificación de la ejecución del programa según las reglas lógicas planeadas.

FORMATO: **TRACE 1** (activado) **TRACE 0** (desactivado)

Nota: El número de la línea se imprime antes de la ejecución. En líneas con más de un comando (separados por :), el número de la línea se imprimira para cada comando. TRACE sin el argumento equivale a TRACE 0.

Teclee el programa:

```
<NEW>
 10 INPUT "pulse enter "; LINE e$:CLS :PRINT AT 5,8;"teclee 100,200 o 300"
 20 INPUT a
 30 IF a = 100 THEN GOTO 100
 40 IF a = 200 THEN GOTO 200
 50 IF a = 300 THEN GOTO 300
 60 STOP
100 PRINT AT 8,8;"esta es la linea 100"
110 GOTO 10
200 PRINT AT 8,8;"esta es la linea 200"
210 GOTO 10
300 PRINT AT 8,8;"esta es la linea 300"
310 GOTO 10
```

Antes de ejecutar el programa analícelo. Este es un programa hecho a propósito en forma simple, para que pueda seguir sus desvíos sin usar TRACE. Ejecútelo de la manera habitual. Hecho ello, teclee TRACE 1.

Ejecute el programa nuevamente y observe que ahora los números de las líneas ejecutadas van siendo mostrados después del signo #. De esta forma, puede saber exactamente adonde los desvíos conducen la ejecución de las instrucciones.

Imagine que tuviese que analizar un programa complicado y repleto de desvíos condicionales y de subrutinas. En verdad, no conseguiría seguir su ejecución y localizar algún error eventual. Con TRACE activado, la tarea será mucho más cómoda.

Para un mejor aprovechamiento, colóquese en el lugar del ordenador, concluya cuál deve ser la próxima línea a ser ejecutada y confirme con lo que el comando TRACE está indicando.

TRACE puede ser activado y desactivado tanto en el modo inmediato, como en línea de programa.

Lista de Datos

LISTA DE DATOS

1. READ...DATA

Cuando un programa requiere la introducción de varios datos o informaciones, el uso del comando INPUT se hace tedioso y lento. Éste es reemplazado, con ventaja, por la instrucción READ ... DATA.

Las instrucciones DATA y READ son normalmente usadas en conjunto. Si un programa contiene una o más instrucciones DATA, deberá tener por lo menos una instrucción READ.

Generalmente, por una cuestión de organización, la instrucción DATA se incluye en el inicio o en el final del programa, sin embargo, nada impide que se la coloque en cualquier otro punto del programa. Vea los ejemplos que siguen:

Aquí aparece en el inicio:

```
10 DATA 10,20,30,40,50
20 FOR v= 1 TO 5
30 READ a
40 PRINT AT 10,0;"yo tengo ahora ";a;" bananas"
50 FOR i=0 TO 300 NEXT i
60 NEXT v
```

Resultado:

```
yo tengo ahora 10 bananas
yo tengo ahora 20 bananas
yo tengo ahora 30 bananas
yo tengo ahora 40 bananas
yo tengo ahora 50 bananas
```

En este programa DATA fue introducida en el final:

```
10 FOR v= 1 TO 5
20 READ a$
30 PRINT AT 10,0;"yo tengo ";a$;" bananas"
40 FOR i=0 TO 300: NEXT i
50 NEXT v
60 DATA "pocas","muchas","quince","naranjas y","una docena de"
```

Resultado:

```
yo tengo pocas bananas
yo tengo muchas bananas
yo tengo quince bananas
yo tengo naranjas y bananas
yo tengo una docena de bananas
```

2. RESTORE

Dijimos que el comando READ retira, mientras lee, los datos de la instrucción DATA. Cuando la lectura termine, todos los datos habrán sido usados, por lo tanto, el número de datos disponibles debe ser igual al número de veces que indicó, en el programa, que el ordenador efectuase la lectura (y la retirada). Si la lectura prosigue, se producirá un mensaje de error, "E Sin datos",a menos que reconstituya la lista de datos, usando el comando RESTORE.

RESTORE (número de línea)

Rehabilita el conjunto de datos en la memoria a partir de un determinado punto (devuelve el puntero a un punto específico). Cada vez que es ajecutado, los datos de DATA vuelven a ser leídos desde el principio, por la próxima instrucción READ.

Ejecute este programa:

```
<NEW>
10 DATA "un TK Color Computer", "manual ","grabadora","television"
20 CLS
30 FOR v=1 TO 4
40 READ a$
50 PRINT AT 10,0;"yo tengo ";a$
60 FOR i=0 TO 350: NEXT i
70 NEXT v
80 GOTO
20 <RUN>
```

Agregue la línea 75 al programa anterior. Teclee:

```
75 RESTORE
<RUN>
```

La línea 75 hace que los datos de DATA sean rehabilitados, permitiendo una nueva lectura.

La instrucción RESTORE puede ser usada para la reutilización de una línea específica de datos. Basta, para ello, que se indique el número de la línea.

Ejemplo:

```
<NEW>
 10 REM estaciones del ano
 20 DATA "primavera","verano","otono","invierno"
 30 REM puntos cardinales
 40 DATA "norte","sur","este","oeste"
100 REM lectura de los puntos cardinales
110 RESTORE 40
120 FOR i = 1 TO 4
130 READ a$
140 PRINT "punto cardinal ";i;": ";a$
150 NEXT i
200 REM lectura de las estaciones del ano
210 RESTORE 20
220 FOR i = 1 TO 4
230 READ a$
240 PRINT "estacion del ano ";i;": ";a$
250 NEXT i
300 STOP
```

20 definición de los datos referentes a las estaciones del año.

40 definición de los datos referentes a los puntos cardinales.

110 selecciona DATA de la línea 40 (RESTORE número de línea).

120 a 150 rutina de lectura de puntos cardinales e impresión en la pantalla.

210 selecciona DATA de la línea 20 (RESTORE número de línea).

220 a 250 rutina de lectura de estaciones del año e impresión en la pantalla.

Obs.: En este programa la utilización del RESTORE número de línea realza la versatilidad de este comando en la gestión de datos específicos contenidos en DATAS distintos.

Fíjese que, en este caso, inicialmente fue utilizado el comando RESTORE número de línea (110 RESTORE 40 y 210 RESTORE 20) antes de la rutina de lectura, permitiendo así la selección previa de los datos a ser leídos.

En las líneas 120, 220, note que fueron definidos bucles de 4 ciclos, pues cada DATA contiene 4 datos.

Aparecerá el mensaje de error "E Sin datos x:y", caso de que el número de ciclos definidos en los bucles FOR...NEXT sea mayor que la cantidad de datos contenidos en los DATAs.

Variables

VARIABLES

Las variables, tanto numéricas como strings, tienen algunas normas de utilización:

Las variables strings deben ser representadas por una única letra, seguida por un $.

Las variables numéricas pueden usar cualquier letra o dígito (con tal que el primero sea una letra). Pueden ser incluidos espacios, pero éstos no serán considerados parte del nombre. Se puede usar tanto mayúsculas como minúsculas.

Ejemplos de nombres válidos:

NUMÉRICAS	STRINGS
x	x$
VARIABLE1	
TOTAL	Q$
tl	
mes12	L$

Ejemplos de nombres incorrectos:

1000 (es un número)
5 meses (empieza con número)
A*N*O (* no es letra ni dígito)
Arco-Iris (- no es letra ni dígito)

A las variables numéricas les pueden ser atribuidos valores (según las reglas explicadas anteriormente), que pueden ser representados incluso por expresiones u otras variables ya estipuladas dentro del programa.

Ejemplo:

X = 2*a+total **mes1 = tot1/4*ano2** **AA = suma+total**

Los valores para variables string deben estar siempre entre comillas:

A$ = ''banana'' **n$ = ''nombre''** **C$ = ''cuenta de los dividendos''**

Cuando se quiera escribir un string conteniendo comillas, se debe escribir la palabra que será destacada entre dobles comillas.

R$ = ''este es un string conteniendo dobles''''comillas''''
e$ = ''este''''string''''es un ejemplo''

Manejo de Cadenas (Strings)

MANEJANDO CADENAS (STRINGS)

Se entiende por string una cadena de caracteres consecutivos dispuestos entre comillas en una determinada secuencia. Si tomamos sólo una parte de esta secuencia, tendremos un substring. Vea los ejemplos que siguen:

STRING	SUBSTRING
"1234567890"	"12345" "234" "567890"
"casa de perro"	"cas" "casa de" "de perro"

1. Determinando un "Corte"

Cada substring representa un corte del string original. Es posible determinar un substring, a partir de un string mayor, aplicando la función TO (no confundir con otras aplicaciones de la función TO de otros comandos).

Esta función puede ser representada por una notación muy distinta, en otras versiones de BASIC (por ejemplo, MID$, LEFT$ y RIGHT$). El comando TO puede ser representado en el siguiente formato:

"secuencia de caracteres"(x TO y)

Ejemplo:

STRING	CORTE	SUBSTRING
"soy el string original"	**(3 TO 9)**	**"y el st"**

En el ejemplo que vimos, el corte efectuado puede ser esquematizado así:

```
soy el string original
123456789
```

Nota: Observe que el espacio blanco es considerado como un carácter.

Vea una aplicación de este recurso:

```
10 LET a$ = "soy el string original"
20 PRINT a$
30 PRINT :PRINT a$(3 TO 9)
```

Ejecute y obtendrá "y el st".

Si varía el contenido de los paréntesis en la línea 30:

para a$ (TO 9) "soy el st"

para a$ (5 TO) "el string original"

para a$ (TO) "soy el string original"

a$ (TO) equivale a a$ (1 TO final del string). Lo mismo ocurre con a$() y a$.

Existe, aún una última forma en la cual el TO es omitido:

soy el string original"(9) = "t" pues "t" ocupa la novena posición en la secuencia.

Trate, por ejemplo, de substituir el índice y de la línea 30 por 23.

```
30 PRINT:PRINT a$(3 TO 23)
```

Cuando el parámetro después del TO sea mayor que el tamaño del string, aparecerá el mensaje de error: 3 indice errado.

Modifique nuevamente la línea 30:

```
30 PRINT:PRINT a$(10 TO 5)
```

En este ejemplo, el resultado obtenido es un string vacío, que puede ser representado por "".

Caso de que uno de los índices (x o y) sea negativo, se producirá otro mensaje de error, diciendo: "Entero excede limite".

2. Mezclando Strings

Además de se poder "cortar un pedazo" de un string, también es posible "insertar" un substring en otro o en un string original, creándose los más diversos arreglos.

Teclee:

```
<NEW>
10 LET a$ = "yo soy un ordenador"
20 PRINT a$
30 LET a$ (5 TO 7) = "* * *"
40 PRINT a$
<RUN>
```

En su pantalla debe aparecer:

```
yo soy un ordenador
yo s* * * *un ordenador
```

Observe que los tres asteriscos tomaron el lugar de los caracteres originales desde la posición 5 hasta la 7 del string.

Trate de aumentar el número de asteriscos entre las comillas, pero sin modificar los índices entre paréntesis:

```
30 LET a$ (5 TO 7)="* * * * * *"
```

Parece que nada cambió:

```
yo soy un ordenador
yo s* * * *un ordenador
```

Lo que ocurre aquí es una adaptación del substring "insertado" al espacio delimitado por los índices x y y. Había seis asteriscos, para ocupar sólo tres posiciones, entonces, tres de éstos fueron eliminados. Es interesante saber que el ordenador los eliminó a partir de la derecha.

Coloque "123456" en el lugar de los asteriscos y verá que "456" no aparecerá en el nuevo string. Si el límite estipulado por los índices x e y es mayor que el substring que será insertado, las posiciones restantes son automáticamente ocupadas por espacios blancos. Altere nuevamente la línea 30:

```
30 LET a$(3 TO 10)="12345"
```

Ejecute el programa y verá:

```
yo soy un ordenador
yo12345 ordenador
```

Observe que los espacios blancos son caracteres significativos.

3. Sumando Strings

Es perfectamente viable que sumemos strings, con tal que el término "sumar" sea interpretado como una concatenación de caracteres de un determinado string (o substring) con otro.

Pruebe lo siguiente:

```
<NEW>
10 LET a$= "yo soy "
20 LET b$= "un ordenador"
30 PRINT "a$ = ";a$:PRINT "b$ = ";b$ : PRINT
40 PRINT "a$+b$ = ";a$+b$
<RUN>
```

En su pantalla verá:

```
yo soy un ordenador
a$+b$ = yo soy un ordenador
```

Teclee este programa y ejecútelo:

```
<NEW>
10 LET a$ = "yo soy un ordenador"
20 LET b$ = a$ + a$(11 TO)
30 PRINT a$: PRINT
40 PRINT "a$ + a$(11 TO) = ": PRINT b$
```

La línea 30 imprime a$ y una línea vacía.

La línea 40 imprime lo que está entre comillas y, en el comando siguiente, el valor de b$, que es a$ + "ordenador", ya que cortamos este substring del string original a$, desde la 11ª posición hasta el fin. Por este motivo, "ordenador" aparece dos veces. Como no colocamos espacio ni en el final del string a$, y ni b$ en el inicio del substring, las palabras fueron impresas sin intervalos. El orden de los factores, en este caso, muda el resultado.

Invierta el orden de los comandos en la línea 20 del programa y observe el resultado.

4. Manejando la Longitud del String - LEN

La instrucción LEN hace que el ordenador indique el número de caracteres de una cadena.

Se puede usar LEN de dos maneras:

LEN"(string)"
o **LEN (nombre de variable string)**

Ejemplo:

```
PRINT LEN "abeja"

LET a$ = "abeja"
PRINT LEN a$
```

Como puede ver, en los dos formatos el resultado es el mismo, o sea, la presentación del número 5 en la pantalla, dado que "abeja" posee 5 caracteres.

Introduzca el siguiente programa en su ordenador:

```
<NEW>
10 REM cuenta de caracteres
20 INPUT "cual es la palabra?";x$
30 LET n = LEN x$
40 PRINT "el numero de letras es ";n
50 PRINT
60 GOTO 20
<RUN>
```

Note que, en este ejemplo, el número de caracteres del string fue atribuido al parámetro "nombre de variable string" de la instrucción LEN.

Nota: Para que el programa pare, mueva el cursor hasta la izquierda de las comillas y accione STOP, conforme aprendió anteriormente.

Funciones

FUNCIONES

Aquí son presentadas algunas otras funciones:

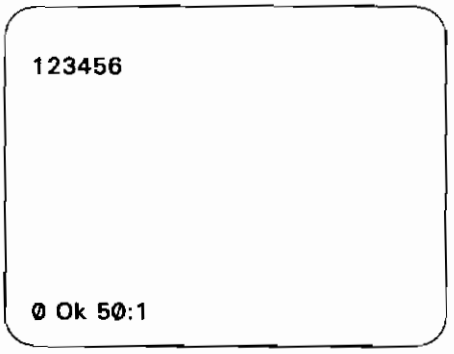
1. STR$ (número)

Como indica el signo $, esta función convierte un número en string.

Ejemplo:

```
<NEW>
10 LET a=123
20 LET b=456
30 LET a$=STR$(a)
40 LET b$=STR$(b)
50 PRINT a$+b$
```

Después del comando RUN tendremos:



Fíjese que los datos 123 y 456 no fueron tratados como números (cuya suma sería 579), ellos fueron tratados como strings. El signo + hizo que se encadenasen.

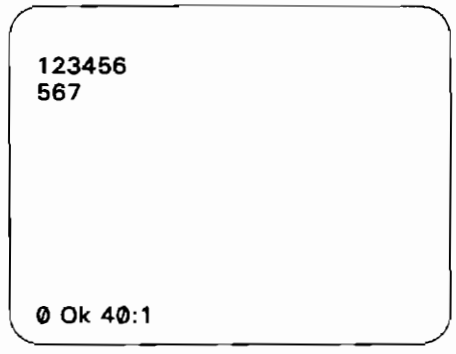
2. VAL variable string o VAL "string"

La instrucción VAL ejecuta la función inversa a la instrucción STR$.

Observe el siguiente ejemplo:

```
<NEW>
10 LET a$="123"
20 LET b$="456"
30 PRINT a$+b$
40 PRINT VAL a$+VAL b$
<RUN>
```

Después de RUN serán presentados:



Observe que, mientras la suma de los strings "123" + "456" resulta "123456", la suma del VAL de éstos resulta la suma verdadera de sus valores numéricos.

3. VAL$ string

A pesar de ser muy parecida a VAL, esta instrucción no tiene la misma utilidad. El argumento de VAL$, es también un string o variable string, sin embargo, su resultado permanece siendo un string, y no un valor numérico.

Ejemplo:

```
<NEW>
10 LET a$ ="TK90X"
15 LET c$ ="a$"
20 PRINT "VAL$";""""c$"""";" =";VAL$ "c$"
25 PRINT
30 PRINT "VAL$ c$" = "; VAL$ c$
<RUN>
```

Siga las instrucciones a continuación:

Primero introduzca los valores de las variables string, tecleando en el modo inmediato:

```
<NEW>
LET A$ =""""TK""""          y          LET B$ = "TK"

MEMORIA                  "TK"          TK
                          A$            B$
```

Nota: No se olvide que, en la memoria del ordenador, la variable A$ contiene "TK" con comillas por causa de las dobles comillas, y que B$ contiene sólo TK, pues las comillas simples son simbólicas.

Luego solicite al ordenador que dé una serie de variaciones de esos strings que están en la memoria. Se obtienen los resultados que colocamos al lado de cada comando:

COMANDO	RESULTADO	COMENTARIO
PRINT A$ PRINT B$	"TK" TK	son los strings en la forma original
PRINT VAL$ A$ PRINT VAL$ B$	TK C Sintaxis error 0:1	VAL$ quita las comillas de los strings originales. Como el string B$ queda sin las comillas, se torna sintácticamente erróneo.
PRINT "A$"	A$	Este PRINT imprime el string que está entre comillas: los caracteres A y $.
PRINT VAL$ "A$"	"TK"	Nuevamente VAL$ quita las comillas, pero esta vez PRINT se refiere a la variable A$, por lo tanto, a PRINT, a$ que es = "TK".

Como vió en los ejemplos arriba, VAL$ puede diversificar el uso de una misma variable alfanumérica. Esto puede servir en la selección de elementos de matrices multidimensionales y hasta para la accesión a las propias matrices. El uso del VAL$ tiene efectos más palpables en programación avanzada.

4. DEF FN

Esta instrucción permite la definición de funciones inexistentes en el ordenador, creadas por el propio usuario. El formato de la instrucción es el siguiente:

DEF FN f(x) = fórmula

Donde:

f = nombre de la función (cualquier letra del alfabeto)
x = variable de la función (cualquier letra del alfabeto)
fórmula = cualquier función matemática que utilice variable definida

La utilización de esta función permite definir las más diversas ecuaciones matemáticas, lo que posibilita la variación del argumento x en el intervalo que se desee.

Por el manejo de esta instrucción, podremos incluso establecer el gráfico característico de las ecuaciones utilizadas.

El ejemplo que sigue aplica la fórmula de la recta:

```
 5 DEF FN r (x) = 2*x+b
10 PRINT "formula de la recta"
20 PRINT AT 2,12;" 2*x+b"
25 INPUT "entre valor de x = ";x
30 INPUT "entre valor de b = ";b
40 PRINT "FN r(x)= ":";FN n(x)
45 INPUT "prosigue (s/n)?"";"elija la opcion y <enter>";p$
50 IF p$ <>"s" THEN STOP
55 CLS GOTO 10
```

Para la definición de funciones utilizando strings, use el siguiente formato:

DEF FN f$(x$)= substring de x$

donde:

f$ = nombre de la función (cualquier letra seguida del signo $)

substring de x$ = cualquier función que, aplicada al string, resulte un substring.

Ejemplo:

DEF FN f$(x$)= x$ (7 TO)

Utilizando la fórmula arriba, suponga que el string x$ sea:

"soy el ordenador TK"

Teclee

```
<NEW>
 1 LET x$ ="soy el ordenador TK"
 5 DEF FN f$(x$) = x$(7 TØ)
1Ø PRINT FN f$(x$)
<RUN>
```

El programa concluye en el substring:

ordenador TK

En el ejemplo a continuación se utiliza una "fórmula" que genera siempre la mitad de cualquier string que sea introducido por medio del INPUT:

```
<NEW>
 5 DEF FN f$(x$) = x$(LEN x$/2+1 TO)
1Ø INPUT "introduzca un nombre ";x$
2Ø PRINT FN f$(x$)
25 INPUT "prosigue (s/n)?";"elija la opcion y <enter> ";p$
3Ø IF p$ <>"s" THEN STOP
35 CLS: GOTO 1Ø
```

5.SGN — argumento

La función SGN indica el signo del argumento:

PRINT SGN 123 **PRINT SGN** -1.ØØ34

El resultado será:

1 si el signo es positivo
– 1 si el signo es negativo
Ø si el argumento es cero

6. ABS argumento

Es otra función que es usada solamente para argumentos numéricos. Convierte el argumento en un número positivo, sin considerar el signo que lo precede:

PRINT ABS – 30.5

o

PRINT ABS 30.5

Tendrán como resultado un único número positivo: 30.5

7.INT argumento

Tiene por finalidad obtener sólo la parte entera de un número (positivo o negativo), ignorando la parte fraccionaria:

PRINT INT 30.9 resultará 30,

sin embargo

PRINT INT -30.9 tendrá como resultado -31.

Debe de haber percibido que INT siempre redondea siempre a menos, es decir, dá el próximo número entero menor que el argumento.

Funciones Matemáticas

FUNCIONES MATEMÁTICAS

1.EXP

El uso de la matemática en áreas como la Ingeniería y la Física requiere la utilización de exponentes naturales que tienen el número e (2,7182818) como base.

Si e elevado a x es igual a y, entonces podemos usar la siguiente fórmula para el cálculo de y:

y = e ↑ x

Entonces podemos expresar

y = EXP(x)

2. LN

Logaritmo natural (base e = 2,71828182...). La función inversa de un exponencial es la función logarítmica:

Ejemplo:

PRINT LN 2/LN 10
Respuesta: 0.30103

Si desea calcular el logaritmo de un número en otra base, la fórmula a ser usada es:

log a = LNa/LNb

En el ejemplo anterior calculamos log 2 (base 10).

3. PI

La función PI del TK (obtenida por la tecla <M>, en el modo extendido), es de gran valor cuando se trabaja con cálculos que utilizan este número. El ejemplo más común es el cálculo de la circunferencia o perímetro de un círculo. Para efectuar el cálculo en su micro introduzca:

LET P=d*PI

donde:

P = el perímetro que Ud. busca
d = el diámetro que informará al ordenador
PI = la función que representa el número 3,1415927

4. Funciones Trigonométricas

SIN	COS	TAN	ASN	ACS	ATN
SENO	COSENO	TANGENTE	ARCOSENO	ARCOCOSENO	ARCOTANGENTE

Obs.: Las funciones trigonométricas trabajan en radianes, no en grados. Para efectuar las conversiones, use el siguiente esquema:

de radianes a grados	r/PI*180
de grados a radianes	g/180*PI

Ejemplo:

```
<NEW>
10 LET r= 360/6.28
20 FOR i= 0 TO 360 STEP 30
30 LET y= SIN (i/r)
35 LET x= COS (i/r)
40 LET z= TAN (i/r)
45 PRINT"""SIN ";i;" = ";y
50 PRINT"COS " ;I;" = ";x
60 PRINT"TAN " ;i;" = ";z
65 NEXT I
```

Números Aleatorios

NÚMEROS ALEATORIOS

Su ordenador trabaja con dos instrucciones distintas en el tratamiento de números aleatorios (randómicos): RND y RAND.

1. RND

Es similar a otras funciones, dispensando, sin embargo, cualquier argumento. Use <PRINT>+ <SYMBOL SHIFT>+ <T>. RND genera un número aleatorio entre 0 y 1, cada vez que la usa. Puede obtener incluso el número 0, pero nunca el número 1.

Trate de ejecutar:

```
10 PRINT RND,
20 GOTO 10
```

Obtiene varios resultados. Fíjese que es imposible determinar una norma común entre estos números. Podemos decir que RND simula un número randómico, pues, en verdad, genera un pseudorandómico, ya que sigue una secuencia fija de 65.536 números.

A pesar de que RND genera números sólo en la franja de 0 a 1, puede ampliar esta zona de 0 a 5, por ejemplo, ejecutando 5*RND.

Para obtener números enteros, use INT, sin olvidar que INT siempre redondea a menos:

```
10 PRINT INT (6*RND),
20 GOTO 10
```

Con este programa, obtiene un bucle que genera números aleatorios entre 0 y 5.

2. RAND

Aunque es parecido con RND, RAND tiene otra aplicación. Es, en realidad, un comando y no una función.

Puede usar RAND para inicializar RND en diversos lugares de la secuencia, tecleando RAND y un número entre 1 y 65535, y <ENTER>.

En el ejemplo:

```
<NEW>
10 RAND 1
20 FOR t=1 TO 10
30 PRINT RND,
40 NEXT t
50 PRINT
60 GOTO 10
```

Cada vez que pasa por la línea 10, la secuencia de RND recomienza con 0.0022735596.

Si utiliza RAND sin argumento (o con argumento cero, RAND 0), el efecto será diferente del RAND de la línea 10, con argumento 1. Verifique esto en el programa:

```
<NEW>
10 RAND
20 PRINT RND
30 GOTO 10
```

Aquí no se obtiene una secuencia aleatoria, pues RAND está usando la cuenta desde que el ordenador fue conectado hasta la ejecución del programa (y durante éste). Ocurre que cada ejecución siguiente a la primera tomará de la secuencia números muy próximos, perdiendo así el carácter aleatorio.

Para que la elección del número se haga más aleatoria, modifique la línea 30 por:

```
30 GOTO 20
```

El uso de esta función asociada a comandos del tipo IF...THEN permite generar programas con comportamiento aleatorio, como es requerido en juegos, simulaciones estadísticas y otras aplicaciones.

Matrices

MATRICES

Matriz es un conjunto o lista de variables bajo el mismo nombre.

Cada elemento de la matriz es identificado por un único índice numérico, que figura entre paréntesis, después del nombre de la matriz.

Ejemplo:

a(12) = 245

a = nombre de la matriz
12 = índice que identifica un elemento de la matriz a
245 = el elemento de la matriz

Cuando los elementos de la matriz sean datos alfanuméricos, el nombre de ésta tendrá que ser seguido por el signo $.

Ejemplo:

b$(5) = "L"

b$ = nombre de la matriz alfanumérica
5 = índice
"L" = elemento

1. DIM

Antes de que podamos usar cualquier matriz, debemos reservarle un espacio en la memoria del ordenador. Para ello, utilizamos la instrucción DIM (dimensionamiento).

Suponga que quiera montar una matriz conteniendo un conjunto de diez variables. Emplee DIM así:

DIM a(10) — para variables numéricas
o
DIM a$(10) — para variables alfanuméricas

Usaremos como ejemplo una matriz numérica a conteniendo diez elementos [de a(1) a a(10)]. El programa que sigue, además de dimensionar la matriz en cuestión, atribuye a cada elemento de ésta un valor del x. La operación es coordenada por un bucle de diez ciclos.

```
<NEW>
10 DIM a(10)
20 FOR x=1 TO 10
30 LET a(x)=x
40 NEXT x
```

Para cerciorarse que su matriz contiene los elementos de 1 a 10, puede solicitar cualquiera de éstos o todos:

```
PRINT a(5) — deberá resultar 5
     o
50 FOR y=1 TO 10
60 PRINT "a(";y;")=";a(y),
70 NEXT y
```

Nota: No use cero como el valor inicial de un FOR...NEXT, si está trabajando con matrices, pues este número no sirve de índice de elementos.

2. Matrices Multidimensionales

En matrices multidimensionales se pueden asociar dos valores a una misma variable.

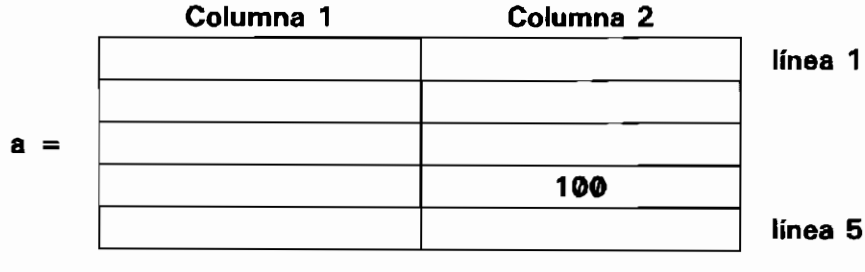
El programa a continuación enseña cómo se usa una matriz bidimensional numérica, y nos da su resultado:

```
10 DIM a(5,2)
20 FOR c=1 TO 5
30 FOR f=1 TO 2
35 INPUT "introduzca un numero ";a(c,f)
40 NEXT f: PRINT AT 2,2;"insercion ";c
45 NEXT c
50 CLS
55 FOR v=1 TO 5
56 PRINT "conjunto ";v
60 FOR g=1 TO 2
65 PRINT "a("; v; ",";g;")";" = ";a(v,g)
70 NEXT g: PRINT: NEXT v
```

Las líneas de 10 a 45 corresponden a la inserción de datos numéricos en una matriz bidimensional.

Las líneas de 55 a 70 indican cómo los datos son identificados.

En el ejemplo tenemos la matriz representada en cinco líneas por dos columnas:



El elemento a(4,2) es 100 (suponiéndose que haya sido introducido cuando c valía 4, y f valía 2).

El ejemplo a continuación enseñará la utilización de una matriz bidimensional alfanumérica:

```
<NEW>
10 DIM a$(5,10)
20 FOR c=1 TO 5
35 INPUT "introduzca un dato ";a$(c)
40 PRINT AT 2,2;"insercion ";c
45 NEXT c
50 CLS
55 FOR v=1 TO 5
56 PRINT "indice ";v
65 PRINT "a$(";v;")";" =";a$(v)
70 PRINT: NEXT v
```

Al dimensionar una matriz alfanumérica, se debe definir (x,y), donde:

x corresponde al número de elementos que serán introducidos en la matriz
y corresponde a la cantidad de caracteres para cada elemento de la matriz

La atribución de dos elementos a una variable se hace a partir del primer carácter hasta la cantidad de posiciones determinada por el segundo índice (y).

En el esquema que sigue está representada una matriz alfanumérica z$, conteniendo tres string y dimensionada así:

DIM z$(3,8)

	1	2	3	4	5	6	7	8	
1	N	U	M	E	R	O			
2	T	E	L	E	G	R	A	F	O
3	V	E	R	D	E				

Si accede a los elementos, obtiene:

z$(1)	NUMERO
z$(1,4)	E
z$(3,7)	(espacio blanco)
z$(2,9)	(mensaje 3 Indice errado 0:1)
z$(2,5 TO 8)	GRAF

De donde se concluye que:

El acceso al elemento que utiliza sólo un índice corresponde a un string entero; considerándose la matriz constituida de tres strings.

El acceso a un elemento con dos números como índice (x y) corresponde a una unidad (y) del string (x), que es un carácter.

Los espacios que el string no ocupa en su campo son complementados con espacios blancos.

El string que sobrepasa los límites establecidos es cortado a partir de la derecha.

z$(2,5 TO 8) = z$(2) (5 TO 8) = GRAF. Este proceso se llama "slicing" y consiste en el corte de un elemento de la matriz.

3. Matrices Tridimensionales

```
<NEW>
10 DIM a$(5,2,20)
20 FOR c=1 TO 5
30 FOR f=1 TO 2
35 PRINT AT 2,2;"insercion "; c
40 INPUT "introduzca un nombre "; a$(c,f)
45 NEXT f: NEXT c
50 CLS
55 FOR v=1 TO 5
60 PRINT
65 PRINT "conjunto ";v
70 FOR g= 1 TO 2
75 PRINT "a$(";v;",";g;")";"= ";a$(v,g)
80 NEXT g: NEXT v
```

En la línea 10 se halla el dimensionamiento de la matriz:

```
DIM a$(5,2,20)
```

Donde:

El primer índice (5) determina que la matriz está dispuesta en cinco líneas.

El segundo índice (2) informa que existen dos columnas.

El último índice (20) establece la cantidad máxima de caracteres que pueden ser introducidos en cada elemento de la matriz.

Nota: Caso de que una matriz alfanumérica sea dimensionada con un nombre de variable ya utilizando - sea para una variable simple, sea para otra matriz - la última matriz prevalecerá, y las anteriores serán destruidas.

Colores

COLORES

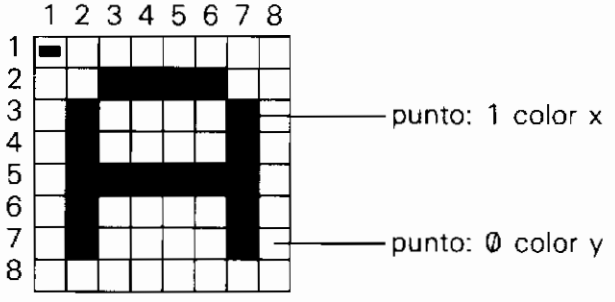
El TK puede producir ocho colores distintos y dos niveles de brillo. Caso de que la televisión sea de blanco y negro, estos colores servirán para generar tonalidades que variarán del gris claro al negro.

A continuación la lista de los ocho colores:

0 BLACK	**(NEGRO)**
1 BLUE	**(AZUL)**
2 RED	**(ROJO)**
3 MAGENTA	**(MAGENTA)**
4 GREEN	**(VERDE)**
5 CYAN	**(CYAN)**
6 YELLOW	**(AMARILLO)**
7 WHITE	**(BLANCO)**

Estos colores siguen, respectivamente, las teclas numéricas.

La imagen activa está formada por un área de 768 posiciones, o sea, 24 líneas por 32 columnas, creándose, así 768 posiciones. Cada una de estas posiciones se imprime en la pantalla como un cuadrado de 8 X 8 puntos. Ello permite que, en cada posición, pueda ser representado cualquier carácter.



1. INK y PAPER

El punto 1 representa un punto encendido en la matriz. Su color es definido de dos maneras:

- Al conectarse el ordenador.
- Inmediatamente después del comando INK.

En el primer caso, cualquier punto encendido aparecerá en la pantalla en el color norma (negro-0). En el segundo caso, el punto aparecerá en el color definido por la última instrucción INK.

El punto Ø representa un punto apagado en la matriz. Su color también es definido de dos maneras:

- Al conectarse el ordenador; con el color blanco (7).
- Inmediatamente después del comando PAPER, con el color definido por el usuario.

Ejemplo:

```
INK 2
PAPER 6
PRINT "TK90X"
```

En el ejemplo de arriba, todos los puntos 1 asumen el color 2 (rojo), y los puntos Ø, el color 6 (amarillo). Tanto INK como PAPER pueden ser definidos por el usuario, en la franja de Ø a 7. Un valor superior atribuido, ocasionará un mensaje de error.

Existen otros medios de usar las instrucciones INK y PAPER.

Teclee:

```
<NEW>
10 PRINT PAPER 6;"amarillo";: PRINT " sol"
15 PRINT
20 PRINT INK 2;"rojo";:PRINT " fuego"
<RUN>
```

Se puede observar en el resultado de este programa, que tanto INK como PAPER vuelven a su valor anterior en el comando PRINT, después de los dos puntos.

Este método también puede ser usado con el comando INPUT, teniendo el mismo resultado que PRINT:

```
<NEW>
10 INPUT INK 3; "que color es este?";c$
15 IF c$< > "lila" THEN GOTO 10
20 INPUT PAPER 6; "le gusto el color?";g$
<RUN>
```

2. BRIGHT

Aumenta la intensidad del color:

BRIGHT Ø = desconectado

BRIGHT 1 = conectado - aumento de la intensidad

Teclee y ejecute el siguiente programa:

```
<NEW>
10 PRINT PAPER 5;AT 2,10;"programa prueba"
20 PRINT AT 4,0;"observe el fondo de la pantalla"
30 PRINT "despues el comando BRIGHT"
35 PRINT INK 3;"vea en el listado a continuacion:"
40 FOR x= 1 TO 1000NEXT x
50 BRIGHT 1
60 PRINTLIST
65 BRIGHT 0
```

Debe de haber notado como el color de fondo del listado aumentó su brillo.

3. FLASH

Artificio usado para realzar un carácter, conjunto de caracteres o incluso la pantalla entera, donde:

FLASH 0 = desactivado

FLASH 1 = activado

Ejemplo:

```
<NEW>
10 FLASH 1
20 PRINT PAPER 6;AT 10,10;"parpadeo";PRINT PAPER 7;"-";PAPER 5;"parpadeo"
30 FLASH 0
```

Caso de que se omita la línea 30 de este programa, FLASH no será desactivado, y el propio listado sufrirá el efecto de éste, cuando sea efectuado LIST. Aunque se desconecte FLASH (FLASH 0), el contenido de la impresión continuará parpadeando. Se debe limpiar la pantalla para interrumpir el efecto de FLASH, en este caso.

4. BORDER

Esta instrucción tiene como finalidad modificar los colores del borde de la pantalla. BORDER 0 a 7 es su límite. Ejemplo:

```
<NEW>
10 FOR x=0 TO 7
20 BORDER x
25 PRINT INK 7;PAPER x;AT 10,2;"border color ";x
30 FOR t= 0 TO 500: NEXT t
35 NEXT x
40 GOTO 10
```

5. INVERSE

Instrucción que posibilita la inversión de colores de los caracteres, en relación a color del fondo (PAPER), donde:

INVERSE 0 = desactivado

INVERSE 1 = activado

Al activarse el modo inverso, todos los caracteres contenidos en la pantalla sufren alteración de color, incluso el listado del programa a continuación:

```
<NEW>
10 INK 3
20 INVERSE 1
25 PRINT AT 10,2;"caracter blanco/fondo magenta"
30 FOR t=0 TO 1000: NEXT t
40 INVERSE 0
50 PRINT AT 10,2;"caracter magenta/fondo blanco"
55 FOR t=0 TO 1000: NEXT t
60 GOTO 20
```

Fíjese que, al ejecutarse un listado, las líneas de programa también son invertidas.

Para volver al modo normal, introduzca:

```
INK 0
INVERSE 0
LIST
```

Utilizando los caracteres de control también se obtienen las modificaciones necesarias.

A continuación una lista de caracteres de control:

CHR$	INSTRUCCION
16	INK
17	PAPER
18	FLASH
19	BRIGHT
20	INVERSE

Nota: Lea las explicaciones referentes a CHR$ en el próximo capítulo.

Cada uno de los caracteres de control visto anteriormente es seguido por otro carácter, que indica un color con el código:

```
<NEW>
10 PRINT CHR$ 17+CHR$ 4;"color verde"
<RUN>
```

Otra aplicación de los caracteres de control es su uso en líneas de instrucciones, donde se puede dar énfasis a una línea o subrutina del pograma.

Siga las instrucciones:

Defina una línea de programa — 10, por ejemplo. Vamos a hacer un bucle de 10 ciclos usando FOR...NEXT, siendo esta línea (10) colocada en el listado, en color verde. Entonces:

Teclee 10.

Accione el modo extendido (cursor E).

Defina el color verde, presionando <CAPS SHIFT> y <4>.

Ahora basta teclear la línea:

FOR x = 0 TO 10

La línea deberá estar en verde. Para volver al color anterior:

Accione el modo extendido.

Defina el color anterior <CAPS SHIFT> y <nº> <ENTER>

Este es el método para colorear las líneas de programas. Supongamos que quiera enfatizar un mensaje, hacer que éste permanezca parpadeando en una línea de programa después de su impresión:

Haga, por ejemplo, la línea 20 en el color cyan (5), usando la misma regla anterior:

Teclee: 20 PRINT''

Después de haber tecleado la línea 20 (20 PRINT''), accione el modo extendido (cursor E).

Presione <CAPS SHIFT> <9> (conecta el parpadeo).

Entre en modo extendido nuevamente.

Defina el color amarillo: <CAPS SHIFT> <6>.

Escriba el mensaje: FELICIDAD.

Después de haber escrito el mensaje, se debe desconectar el parpadeo. Si no lo hace, las líneas posteriores recibirán el mismo efecto. Para hacerlo:

Entre en el modo extendido (cursor E).

Accione <CAPS SHIFT> <8> (desconecta el parpadeo).

No se olvide de cambiar al color deseado, pues en caso contrario, la próxima línea recibirá el último color definido. Para hacerlo:

- Entre en el modo extendido.
- Defina el color anterior <CAPS SHIFT> <nº>.
- Cierre las comillas.

<ENTER>

Ahora, en la línea 30, cierre el bucle <NEXT X> con el color magenta (3), usando el método inicial (línea 10).

Ejecute el programa: <RUN>.

Obs.: Los números de las líneas, deben ser tecleados antes de cualquier proceso de coloración.

6. OVER

Utilizado para sobreponer literalmente strings numéricos, alfanuméricos o caracteres aislados, el comando OVER (1 activa, 0 desactiva) es un interesante artificio en la programación. Teclee y ejecute el siguiente programa:

```
<NEW>
10 PRINT AT 20,1;"presione ENTER"
15 LET a$=" OVER 0 prueba"
20 LET b$=" OVER 1 prueba"
25 LET c$=" <=>"
30 PRINT INK 1;AT 10,8;OVER 0;c$
35 INPUT e$
40 PRINT INK 2;AT 10,8;OVER 0;a$
45 INPUT e$
50 PRINT INK 3;AT 10,8;OVER 0;b$
55 INPUT e$
60 PRINT INK 1;AT 10,8;OVER 1;c$
65 INPUT e$:GOTO 10
```

15 a 25 — Definen los strings usados en el ejemplo.

30 — Imprime en el centro de la pantalla (AT 10,8), el contenido de c$, en color amarillo, con OVER 0 (desactivado).

40 — Imprime el contenido de a$ en la misma posición de c$, en color verde, y con OVER 0 (desactivado). Observe que a$ "ocupó" la posición de c$, eliminándolo.

50 — Imprime el contenido de b$, también ocupando el lugar del último mensaje.

60 — Imprime el contenido de c$ en la misma posición de b$, pero con OVER 1 (activado). Nótese que cuando utilizamos OVER 1, c$ se sobrepone a b$, pero conserva su contenido en la pantalla, originando una fusión de los dos string.

Nota: El INPUT e$ de las líneas 35, 45, 55, 65 funciona sólo como una especie de temporizador, pues el ordenador aguarda su autorización para proseguir.

7. ATTR

Nos da un número que revela el estado de una posición de la pantalla — ATTR (línea, columna) — originado por la suma de cuatro números que corresponden a:

128, si FLASH 1 ó 0 si FLASH 0 +
64, si BRIGHT 1 ó 0 si BRIGHT 0 +
8 por el número del color definido con PAPER +
0 número del color definido con INK

Ejemplo:

```
PRINT AT 0,0;FLASH 1;BRIGHT 1;PAPER 6;INK 1;"$";ATTR(0,0)
```

El resultado obtenido es 241, y se refere a:

FLASH 1;BRIGHT 1;PAPER 6;INK 1

```
 V     V      V     V
128 + 64 + (8*6) + 1 = 241
```

8. INVERSE VIDEO y TRUE VIDEO

Estos comandos insertan códigos de control en líneas de programa, para producir colores invertidos o normales. INVERSE VIDEO provoca inversión de colores, y TRUE VIDEO hace cancelar la inversión.

Al accionarse INVERSE VIDEO, la impresión de los caracteres siguientes sufre un cambio: se invierte el color del fondo y de los caracteres. Así, por ejemplo, los caracteres que, normalmente, aparecerían en negro sobre fondo blanco pasan a mostrarse blanco sobre fondo negro.

El Conjunto de Caracteres

EL CONJUNTO DE CARACTERES

El conjunto de caracteres utilizado por este ordenador es, en su gran maioria, el mismo utilizado en otros ordenadores, los caracteres del código ASCII.

Además de los caracteres considerados "norma", tales como los alfabéticos (en mayúscula y minúscula), los numéricos de (Ø a 9) y los simbólicos comunes ($, < >, % etc.), el conjunto de caracteres del TK incluye las palabras-clave (PRINT, LIST, LOAD, etc.). Pero, esto no es todo. Entre los 256 caracteres diponibles, su ordenador posee dos conjuntos de caracteres especiales, utilizados ampliamente en los idiomas castellano y portugués. Y, para mayor comodidad del usuario, el TK dispone todavía de una rutina ubicada en la ROM, que posibilita crear caracteres inéditos.

A continuación veremos como tener acceso a todos esos recursos.

1. CHR$ número

Esta función normalmente está precedida del comando PRINT, y es empleada para obtener el carácter correspondiente al número que le sirve de argumento. Teclee por ejemplo:

```
PRINT CHR$38
```

El ordenador presentará:

```
&
```

Si verifica en la lista de códigos del Apéndice D, verá que el símbolo & corresponde al número 38.

CHR 35 = # **CHR 36 = $**
CHR 37 = % **CHR 38 = &**
CHR 39 = ' **CHR 40 = (**

El siguiente programa genera una lista de caracteres (de 32 a 255). Los primeros (de Ø a 31) son "invisibles", pues sólo desempeñan funciones:

```
5 FOR X=32 TO 255
10 PRINT,,"CHR$ $";x;" = ";CHR$ x
20 NEXT x
```

2. CODE "string"

La función CODE se aplica a un string, volviendo el código del primer carácter de la secuencia alfanumérica.

Ejemplo:

PRINT CODE "&abcdef"

Nos da 38 como resultado. Ya vimos que 38 es el código del carácter &, que es el primero de la secuencia del ejemplo.

3. Caracteres del Modo Gráfico

El modo gráfico **G** da acceso a los caracteres gráficos inscritos en las teclas 1 a 8. Para obtenerlos, basta teclear (en modo **K**):

PRINT "

Cambiar al modo gráfico y presionar la tecla correspondiente al dibujo en ella inscrito; volver al modo **L** y cerrar las comillas.

Si <CAPS SHIFT> o <SYMBOL SHIFT> se accionan, se obtiene el inverso del dibujo, totalizando, así, 16 caracteres gráficos diferentes.

4. UDG

La función UDG es el medio por el cual se da acceso a los dos conjuntos de caracteres especiales del portugués y del español.

Antes de nada, teclee las letras de A a U, en modo gráfico (comience con PRINT", en modo **K**, y cerre las comillas en el modo **L**):

PRINT"ABCDEFGHIJKLMNOPQRSTU"

El resultado será la impresión de esas letras en minúscula.

Poco interesaría si el conjunto obtenido en **G** fuese idéntico al conseguido mediante <CAPS SHIFT>. En realidad, estas letras permiten ser reemplazadas por otros símbolos gráficos que Ud. mismo va a crear, como verá más adelante.

Antes de crear nuevos caracteres, vamos a dar acceso a los caracteres de la lengua portuguesa. Teclee:

UDG 0 <ENTER> (se obtiene UDG con <SYMBOL SHIFT> y <X>).

Ahora repita PRINT", entre en modo **G** y teclee cada letra de A a U nuevamente (deje un espacio entre ellas para mejorar la visión). Esta vez, el resultado cambió bastante:

Á É Í Ó Ú á é í ó ú À à Â Ê â ê õ ã Ç ç

Para acceder al conjunto de caracteres del idioma castellano, teclee UDG 1. Repita la operación PRINT, y tendrá en la pantalla:

Á É Í Ó Ú á é í ó ú ô ñ Ñ Ã õ Ü ü

Nota: Los diez primeros caracteres son comunes a los dos conjuntos.

Ahora puede escribir correctamente ''PortuguêS'' o ''Español''. No se preocupe si en el listado los caracteres salieran cambiados, lo importante es el resultado.

Acceda a UDG 0 y programe:

10 UDG 0: PRINT ''Português'' (para obtener ê, use G en el modo gráfico)

Antes de teclear la próxima línea, acceda a:

UDG 1

A continuación:

20 UDG 1: PRINT ''Español'' (para obtener ñ, use **L** en el modo gráfico)

En el momento que presione <ENTER>, el listado saldrá confuso:

10 UDG 0: PRINT ''Portuguüs'' (la ê salió como ü porque cambió a UDG1)
20 UDG 1: PRINT ''Español''

Cuando ejecute el programa, todo saldrá conforme fue planeado:

Português
Español

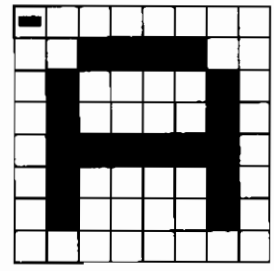
Antes de seguir las instrucciones del próximo punto, desconecte y conecte el ordenador para que se borre de la RAM el código UDG 1 (o UDG 0). El comando NEW no es suficiente para hacerlo, pues esta parte de la memoria está protegida (hablaremos sobre la memoria del ordenador en los capítulos finales del manual).

5. Definiendo Caracteres

Vamos a ampliar el uso de la función UDG.

Teclee: <UDG> <2> <ENTER>

Acaba de habilitar una función utilitaria. En su pantalla verá una matriz de 8x8 elementos, conteniendo la letra A y, abajo de la matriz, tres líneas como en la figura a continuación:



ABCDEFGHIJKLMNOPQRSYU
ABCDEFGHIJKLMNOPQRSTU
A=A

La primera línea representa las letras (A — U) del teclado, disponibles para que el usuario defina qué carácter gráfico recibirá cada tecla. La segunda línea representa el contenido actual de cada tecla. Este contenido puede ser cambiado, pero siempre que el ordenador sea conectado, el contenido norma (el de la figura) aparecerá.

Si hace una retrospectiva, verá que en el rubro UDG, obtuvo el contenido norma de las teclas A a U en modo **G**, cuando digitó:

<PRINT> <"> G <A><B><C> ... L <">

No se confunda con los caracteres que obtuvo con UDG Ø y UDG 1. Aquellos no deben ser modificados, a no ser que sea necesario.

La tercera línea representa el carácter actual, o sea, la tecla que está siendo manejada, en este caso, la tecla A. Haga una prueba: para cambiar de la tecla actual a F, sólo teclee <F>.

La matriz ahora muestra la "estructura" del carácter F ampliada 8 veces, y la tercera línea muestra O como tecla actual. Esta tercera línea muestra el carácter definido en el tamaño real.

Fíjese que la matriz tiene el fondo de color blanco y los cuadraditos llenos en negro. Son los colores norma para INK y PAPER.

Si determina (antes de acceder a UDG 2) INK 2 Y PAPER 6, por ejemplo, obtendrá la matriz y el carácter en ella contenido, en color rojo (2); el fondo de la pantalla en amarillo (6) (color del PAPER), y las tres líneas de abajo de la matriz en los colores norma, o sea en negro sobre fondo blanco. Estas líneas estarán siempre en estos colores.

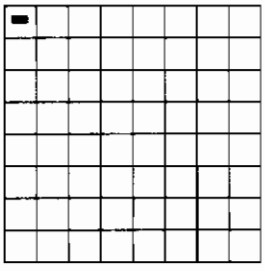
Supongamos que quiera modificar el contenido de la tecla F. La primera cosa a hacer es poner esta tecla en evidencia teclando <F>. La estructura de F aparecerá en la matriz, y, en la tercera línea, Ud. tendrá F=F. Pero, no quiere F=F. Vamos a suponer que quiera que mediante la tecla F, en modo **G**, aparezca el signo ≈. Tendrá que crearlo, pues no existe en el ordenador.

Primeramente accione EDIT para limpiar la matriz y iniciar la construcción de la estructura de ≈.

Suceden tres cosas:

- La matriz queda vacía, con excepción de un pequeño punto.
- Al lado de F=, en la tercera línea, no hay nada más.
- Hay un espacio blanco en la segunda línea, informando que la tecla O no tiene representación gráfica en este momento.

Dijimos que en la matriz permaneció un punto. Este punto es el cursor de gráfico: un indicador de la posición en que la matriz será llenada o borrada.

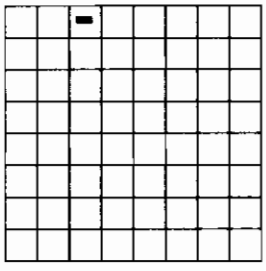


```
ABCDEFGHIJKLMNOPQRSTU
ABCDEFGHIJKLMNPQRSTU
F=
```

Para mover el cursor dentro de los límites de la matriz (éste jamás sobrepasará estos límites), use las siguientes teclas:

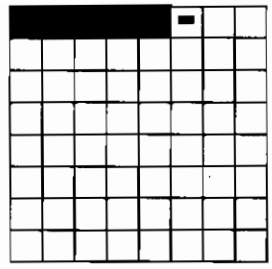
<5> izquierda <6> abajo

<8> derecha <7> arriba



```
ABCDEFGHIJKLMNOPQRSYU
ABCDEFGHIJKLMN OPQRSTU
F =
```

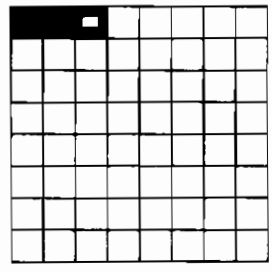
Si utiliza las teclas-flechas, moverá el cursor, llenando el cuadrado donde éste se encuentra con el color determinado anteriormente por INK. En caso de que no haya sido determinado, como fue dicho, el color será negro (Ø).



→
<CAPS SHIFT>

ABCDEFGHIJKLMNOPQRSTU
ABCDE-GHIJKLMN-PQRSTU
F = -

Para borrar los cuadraditos ya llenos, use el mismo sistema de las teclas 5, 6, 7 y 8, pero manteniendo la tecla <SYMBOL SHIFT> presionada.



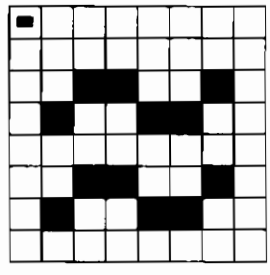
←
<SYMBOL SHIFT>

ABCDEFGHIJKLMNOPQRSTU
ABCDE-GHIJKLMN-PQRSTU
F = -

Fíjese que al mismo tiempo que ocurren cambios en la matriz, estos van apareciendo al lado de la letra O, en la tercera línea, y en la posición de esta letra en la segunda otros caracteres.

Ahora que ya conoce los principios de creación, arme en la matriz la ≈.

Siga el modelo de la figura de abajo, o mejórela:



ABCDEFGHIJKLMNOPQRSTU
ABCDE≈GHIJKLMN PQRSTU
F = ≈

Para salir de la rutina UDG 2, use DELETE.

De ahora en adelante, obtendrá siempre el carácter (o caracteres) que creó en UDG 2. Para esto, basta oprimir la tecla correspondiente en modo **G**.

Use el símbolo ≈ , que acabamos de construir, aprovechándolo en el programa.

```
<CLS>
10 FOR r=0 TO 233
20 PRINT " ≈ ";:REM usando la letra "O" en modo grafico
30 NEXT r
<RUN>
```

El resultado deberá ser una pantalla llena de los nuevos símbolos.

Nota: Utilice la rutina UDG2 luego que el ordenador haya sido conectado o antes de haber utilizado los comandos UDG0 y/o UDG1. Después de la lectura de uno de estos códigos por la RAM, no tendrá más acceso al código libre para definiciones. La rutina UDG2 también funciona en este caso, pero trae a la RAM los caracteres del último código accedido.

Verá en el capítulo "Almacenamiento y Carga — Grabadora Cassette" cómo guardar los símbolos gráficos creados, para que pueda usarlos cuando los necesite.

6. SCREEN$

El formato de la función SCREEN$ es el siguiente:

```
SCREEN$ (l,c)
```

SCREEN$ da el carácter que ocupa determinada posición en la pantalla. Si este carácter es un carácter gráfico definido por el usuario, mediante UDG, SCREEN$ dará un string vacío. Por ejemplo:

```
<CLS>
10 PRINT AT 5,8; "A"
20 PRINT
30 PRINT SCREEN$ (5,8)
<RUN>
```

Al ejecutarse el programa, éste enseñará, en la posición 5,8, dada por la instrucción SCREEN$, la letra A, que será repetida dos líneas abajo.

Manejando La Memoria

MANEJANDO LA MEMORIA

1. POKE dirección, dato

La instrucción POKE sirve para introducir datos directamente en la memoria del ordenador. Acuérdese de que los datos sólo podrán ser introducidos en la memoria RAM, ya que el contenido de la ROM es inalterable. Para conseguir tal entrada deberá indicar:

1) La dirección de la memoria RAM a la cual desea enviar el dato.
2) El dato que será introducido.

Ejemplo:

```
POKE 23609,20
     decimal decimal
```

POKE 23609,20 indica la entrada del valor 20 en la dirección 23609. Estos números son decimales, pero serán almacenados por el ordenador bajo la forma binaria.

Los datos son almacenados en bytes (8 bits). Siendo así, el dato que será introducido directamente en la memoria, debe estar comprendido entre 0 y 255. Si es intentada la entrada de un número fuera de estos límites, el ordenador presentará el mensaje:

```
B Entero excede limite
```

El mismo mensaje será presentado, si la dirección es superior a 65.535, ya que ésta es la dirección límite de la memoria considerada por la CPU del TK.

2. PEEK dirección

PEEK efectúa la lectura de una determinada dirección de la memoria, retornando el dato que ahí se encuentra:

Ejemplo:

```
10 LET K=PEEK 23609
20 PRINT K
```

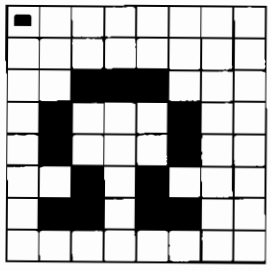
Al ejecutarse el programa, el contenido de la dirección 23609 se presenta en la pantalla (caso de que haya introducido 20, éste es el resultado del PEEK).

3. BIN

Dijimos que cada dato es compuesto por 8 bits, o sea, un byte. Para representar un carácter, por ejemplo, tendríamos que ''traducir'' su forma en bits. El ordenador usa el lenguaje binario y sólo entiende los números Ø y 1. Si el bit está con valor Ø, está desconectado; con el valor 1, conectado.

Para entender mejor lo que fue explicado, verifique la composición de los caracteres en el capítulo anterior. La matriz de 8 x 8 que conoció mediante la función UDG 2, sirve perfectamente para ilustrar un byte y su contenido.

La figura siguiente completará la explicación:



A cada cuadradito lleno se asocía el valor 1, y a cada vacío, el valor cero.

Para representar toda la secuencia, tendríamos:

ØØØØØØØØ
ØØØØØØØØ
ØØ111ØØØ
Ø1ØØØ1ØØ
Ø1ØØØ1ØØ
ØØ1Ø1ØØØ
Ø11Ø11ØØ
ØØØØØØØØ

No puede, todavía, introducir esta secuencia en la forma en que ésta se presenta, pues está trabajando en BASIC, y no en sistema binário. Para utilizar el POKE, tendríamos que obtener los números en decimal, como fue dicho en el inicio del capítulo. Para ello se utiliza la función BIN que tiene la forma siguiente:

BIN secuencia binaria

Ejemplo:

```
PRINT BIN ØØØ11100
```

El ordenador devolverá:

28
o
POKE 23609, BIN 00011100

El ordenador introducirá 28 en la dirección especificada (23609).

Fíjese que este POKE hace que el ordenador emita sonido al presionarse cualquier tecla. Si incrementa el valor, aumentará la duración del sonido.

Para introducir el byte representado por la matriz de la figura anterior, tendríamos que transformar en decimal toda la secuencia binaria, colocando la función BIN antes de cada conjunto de 8 bits. Pero, qué dirección usaríamos para el POKE ?

Vamos a asociar el ''carácter'' que está en la matriz, a una letra del teclado: A, por ejemplo.

A continuación utilizamos la función USR, teniendo A como argumento. Significa que el ''carácter'' que creamos será almacenado a partir de la dirección A (USR''A'', USR ''A''+1, USR''A''+2...USR''A''+7).

Nota: **La función USR se usa aquí para convertir, en el correspondiente carácter gráfico a ser alterado, un argumento tipo string, ubicado en la dirección del primer byte en la memoria.**

Ejecute el programa:

```
<NEW>
10 FOR x=0 TO 7
15 READ s
20 POKE USR "a"+x,s
25 NEXT x
30 DATA 0,0
35 DATA BIN 00111000,BIN 01000100
40 DATA BIN 01000100,BIN 00101000
45 DATA BIN 01101100,0
```

Nota: **Una secuencia de ceros en binario puede ser representada directamente por un cero en decimal, sin ser precedido por BIN (líneas 30 y 45).**

Para confirmar que el nuevo ''carácter'' fue introducido en la RAM, entre en modo gráfico y use A. Caso de que quiera ver mejor, entre en UDG 2 y ponga ''A'' en evidencia.

La función USR puede estar seguida de un número como argumento. El uso de esta forma será visto posteriormente, en el capítulo ''Código Máquina''.

Gráficos

GRÁFICOS

1. PLOT x,y

Esta instrucción tiene como objetivo colocar un punto en cualquier lugar especificado de la pantalla.

PLOT usa coordenadas de (0,0) a (255, 175) ofreciendo, por lo tanto, 45.056 puntos en la pantalla, proporcionando una mejor definición de gráficos y de dibujos.

Cuando los límites admisibles sean sobrepasados, habrá un mensaje de error:

VÁLIDO	NO VÁLIDO
PLOT 255,175	**PLOT 255,255**

Introduzca la siguiente instrucción:

```
PLOT INK 1;127,87.
```

Se observa un punto azul en el centro de la pantalla.

INK 1 definió el color del punto
(127,87) definición de columna, línea

Las coordenadas (0,0) se localizan en el ángulo inferior izquierdo de la pantalla, por lo tanto, no coinciden con las coordenadas 0,0 del PRINT. Introduzca y ejecute el programa:

```
<NEW>
10 FOR x=0 TO 255
20 PLOT INK 2; x,0
30 NEXT x
```

Nótese que la recta tiene su inicio en el extremo inferior izquierdo y se extiende hasta la columna 255 (valor máximo de x).

2. DRAW x,y

Dibuja un punto o una secuencia de puntos a partir de la coordenada definida por el último PLOT (o DRAW). En caso de que PLOT (o DRAW) no haya sido utilizado anteriormente, el punto inicial de DRAW será (0,0).

Ejemplo:

```
<NEW>
10 DRAW 80,80
<RUN>
```

El punto inicial antes de usar DRAW era (0,0).

Agregue esta línea y ejecute el programa: 20 DRAW 80,0

DRAW ahora incrementó 80 puntos para el primer parámetro (columna) y mantuvo el segundo parámetro (línea) del DRAW anterior.

DRAW también disminuye ambos parámetros, con tal que los valores atribuidos a éstos sean negativos.

Agregue esta línea:

```
30 DRAW - 80,-80
40 DRAW - 80,0
<RUN>
```

Las coordenadas especificadas no deben sobrepasar los límites (0,0) a (255,175), en caso contrario, aparecerá un mensaje de error.

2.1. DRAW x,y,z

Existe un tercer parámetro, usado en DRAW, que tiene como finalidad trazar un arco con medidas en radianes. Si el valor de este parámetro es negativo, el arco se describirá en el sentido de las agujas del reloj; si es positivo, en el sentido contrario.

Ejemplo:

```
<NEW>
10 PLOT 127,87
20 DRAW 0,20,5
30 DRAW 0,-20,-4
```

La línea 10 coloca un punto en el centro de la pantalla. Por lo tanto, el punto inicial del primer DRAW será en este mismo lugar (127,87).

La línea 20 traza un arco a partir de la última coordenada, con el tercer parámetro definiendo el grado de abertura del arco en radianes.

La línea 30 traza un arco a partir de la última coordenada, pero con el tercer parámetro negativo, haciendo el arco en el sentido de las agujas del reloj.

Obs.: El tercer parámetro muestra la fracción de circunferencia que debe ser trazada. Una circunferencia completa mide 2 * π radianes. Si este parámetro es igual a π , será dibujada media circunferencia; 0.5*PI corresponderá a 1/4 de circunferencia y así sucesivamente.

3. POINT x,y

Indica la situación de un punto especificado por las coordenadas x,y.

Si el resultado es 0, entonces el punto en la coordenada definida tiene el mismo color del fondo (pantalla). Si el resultado es 1, el punto tiene otro color (carácter con INK definido).

Ejemplo:

```
<NEW>
10 PRINT POINT (0,0)
20 PLOT 127,87
30 PRINT POINT (127,87)
```

4. CIRCLE x,y,r

Dibuja una circunferencia completa, con tal que le sean atribuidas coordenadas específicas al centro y al radio.

Ejemplo:

```
CIRCLE 127,87,20
```

Las instrucciones PLOT, DRAW y CIRCLE pueden ser usadas en conjunto con las instrucciones INK y PAPER, para definición de color. Ejemplo:

```
<NEW>
10 PLOT INK 1; 127,87
20 CIRCLE INK 2; 127,87,40
30 DRAW INK 3; 50,-20
```

Pruebe a usar PAPER en lugar de INK.

Caso de que quiera usar INVERSE o FLASH en estas instrucciones, úselas del mismo modo que INK y PAPER, pero siguiendo sus reglas:

```
INVERSE 0 ó INVERSE 1

FLASH 0 ó FLASH 1
```

Tiempo y Movimiento

TIEMPO Y MOVIMIENTO

1. PAUSE n

Frecuentemente es preciso interrumpir temporalmente un programa o una rutina de programa. Para este propósito se utiliza el comando PAUSE.

Ejemplo:

```
30 PAUSE 400
```

n puede ser un número entero hasta 65535, lo que da cerca de 18 minutos de paralización. Si n = 0, la pausa es, teóricamente, infinita.

El tiempo de duración debe ser obtenido de la siguiente forma:

$$t = \frac{n}{q}$$

- t = tiempo en segundos
- n = parámetro definido para PAUSE
- q = es igual a 60 en países donde la frecuencia de la red es de 60 Hz, y es igual a 50 donde la frecuencia es de 50 Hz

Se puede interrumpir la ejecución del comando aun antes del tiempo especificado. Basta que se presione cualquier tecla en el transcurso de la paralización (excepto <SHIFT>).

Ejemplo:

```
<PAUSE> 1000 <ENTER>
```

Presione cualquier tecla para interrumpir.

Ahora ejecute este programa:

```
<NEW>
10 FOR x=1 TO 10
20 PRINT AT 10,14;"((";x;"))"
40 PRINT PAPER 4;AT 2,12;"p a u s e"
45 PAUSE 80
50 CLS
60 NEXT x
```

Fíjese como el tiempo es controlado por la línea 45. Pause n equivale a un tiempo, como lo que fue especificado arriba, pero el proceso también lleva tiempo. En el caso de la línea 45, hay una paralización de un poco más de un segundo <PAUSE 80>.

2. INKEY$

La función INKEY$ lee el teclado y verifica cuál tecla fue accionada. Caso de que el teclado no haya sido accionado, INKEY$ es igual a un string vacío ("").

La función INKEY$ es similar al comando INPUT, pero con dos marcadas diferencias:

- La función INPUT permite la entrada de varios caracteres, que se consideran, en conjunto, como el dato solicitado. La función INKEY$ permite la entrada de sólo 1 carácter, y solamente este carácter representa el dato necesario a ser insertado.

- En la función INKEY$, al contrario de INPUT, no se oprime <RETURN> en la introducción del dato, pues ésta se hace automáticamente.

Ejemplo:

```
<NEW>
 10 PRINT INK 2,AT 2,0;"quiere ejecutar el programa ? (s/n)"
 20 IF INKEY$ = "" THEN GOTO 20
 30 IF INKEY$ = "n"THEN GOTO 200
 35 IF INKEY$ <> "s" AND INKEY$ <> "n" THEN GOTO 20
 40 CLS
 45 PRINT PAPER 5;AT 2,0;"INKEY$";" recibe solo un caracter"
 50 PRINT PAPER 4;AT 5,0;"numerico o alfanumerico"
 60 STOP
200 CLS
210 PRINT INK 3;AT 10,6;"entonces es el fin"
```

La línea 20 espera que el teclado sea accionado. Si esto no ocurre, habrá un desvío hacia la misma línea.

Las líneas 30 y 35 imponen una condición: las teclas que serán presionadas deben ser <s> o <n>. En caso contrario, nuevamente habrá desvío hacia la línea 20.

Los desvíos serán hechos de acuerdo con la condición del INKEY$. Si es <n> desvía hacia la línea 200. Si es <s>, el programa no será desviado, ejecutandose, así, las líneas subsiguientes.

Efectos Sonoros

EFECTOS SONOROS

Su micro es capaz de generar señales sonoras por medio del altavoz de su televisión, mediante sintetizador de sonido, manejado por software. Para obtener este efecto, basta utilizar un comando muy simple, que funciona tanto en el modo inmediato como en el programado. Nos referimos al comando SOUND, obtenido por la tecla <Z> en modo extendido **E** , siendo presionada <SYMBOL SHIFT>. El formato es el siguiente:

SOUND d,f

d = la duración de la emisión de sonido (+ o − 0.00191 a 10)
f = la frecuencia o nota a ser emitida (−60 a 60)

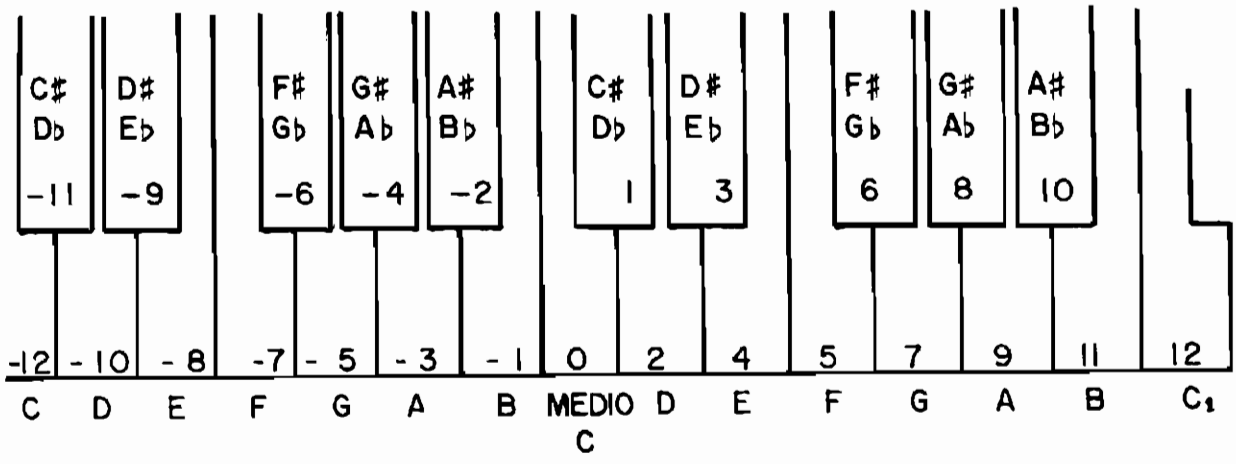
Nota: d puede ser todavía menor, pero el sonido emitido se hace inaudible.

Para que tenga una idea mejor de la equivalencia de la duración del sonido del TK y el tiempo de las notas musicales, observe el esquema de la figura:

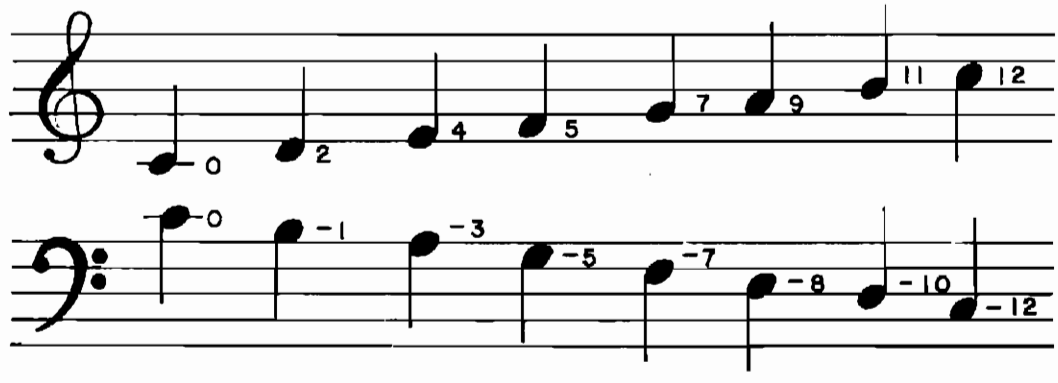
NOTA	TIEMPO	COMANDO
𝅝	4	SOUND 1,nota
𝅗𝅥	2	SOUND .5,nota
♩	1	SOUND .25,nota
♪	1/2	SOUND .125,nota
𝅘𝅥𝅯	1/4	SOUND .062,nota
𝅘𝅥𝅰	1/8	SOUND .031,nota
𝅘𝅥𝅱	1/16	SOUND 0.16,nota

En las figuras que ve a continuación, están representadas las notas musicales y su equivalencia con los valores de f del comando SOUND. Observe que las figuras no incluyen todas las escalas que el TK alcanza, estando la representación restringida a sólo dos escalas.

En el teclado:



En el pentagrama:



en el programa:

```
<NEW>
10 SOUND .25,0: SOUND .25,2: SOUND .25,4: SOUND .25,5: SOUND .25,7:
   SOUND .25,9: SOUND .25,11: SOUND .25,12
20 SOUND .25,0: SOUND .25,-1: SOUND .25,-3: SOUND .25,-5: SOUND .25,-7:
   SOUND .25,-8:SOUND .25,-10: SOUND .25,-12
```

Es simple obtener una nota en otra escala, como Ud. debe haber notado. Para eso es suficiente que se sume o se substraiga 12 de la nota en cuestión.

Si lo desea, puede obtener sonidos intermedios, usualmente omitidos de las escalas musicales por ser inferiores a semitonos. Esto se consigue, atribuyéndole a f valores fraccionarios.

Teclee y ejecute el programa a continuación. Éste presenta sólo una pequeña noción de los efectos del comando SOUND:

Do-re-mi

Limpie la memoria del ordenador con <NEW> y acceda al comando CAPS LOCK.

```
 1 PAPER 3: BORDER 7: CLS: PRINT PAPER 5;AT 7,9;"P R O G R A M A"
 2 PRINT PAPER 6;AT 10,12;"DO-RE-MI"
 3 PAUSE 150: PAPER 1: CLS
 4 PRINT PAPER 5;AT 10,7;"VA A SER EJECUTADO": PAUSE 150: PAPER 5: CLS
 5 PRINT PAPER 7; INK 2;AT 10,3;"PRESIONE BREAK PARA PARAR"
10 FOR I=1 TO 64
20 READ A
30 SOUND .2,A
40 NEXT I
50 DATA -24,-22,-20,-19,60,-19,60,-19,-24,-22,-24,-22,60,-22,60,-22
        -24,-17,-19,-20,60,-20,60,-20,-24,-22,-20,-19,60,-19,60,-19
60 DATA 60,60,0,5,4,2,0,2,0,0,0,2,0,2,4,5,60,60,0,5,4,2,0,2,0,0,0,2,0,2,4,5
70 RESTORE
80 GOTO 10
```

Almacenamiento y Carga - Grabadora Cassette

ALMACENAMIENTO Y CARGA - GRABADORA CASSETTE

1. Easy-Load

El TK está dotado del sistema "easy-load" para almacenamiento de informaciones en cinta magnética. Este sistema es sensiblemente superior a todo lo que existe en materia de uso de cintas cassette con ordenadores. Además de un "feedback" (retorno) visual del proceso de la operación de carga, permite indexar sus ficheros, con búsqueda automática por el nombre y todavía es casi independiente del ajuste de volumen de la grabadora, siempre que se usen equipos y cintas de buena calidad.

2. SAVE "programa"

Cuando se desconecta el TK, se pierde todo el programa y las variables almacenadas en la memoria. Una de las maneras de preservar programas, es instruir al ordenador para grabarlos en cinta cassette, pudiendo, así, ser posteriormente recuperados.

La instrucción SAVE tiene la finalidad de grabar programas y variables definidas. Esta instrucción es seguida por un string que es dado por el usuario, o sea, el nombre del programa.

Procedimiento para la grabación:

- Teclee este programa:

```
<NEW>
10 FOR x=0 TO 154 STEP 4
20 FOR z=50 TO 100
25 PLOT INK 4;x,z
30 NEXT = z
35 DRAW z,5,3
40 DRAW INK 2;-z,5,3
50 NEXT X
```

- Conecte el cable de lectura/grabación conforme a las instrucciones del capítulo "Instalación".
- Bobine la cinta hasta pasar la parte no magnetizada.
- Teclee:

<SAVE>"programa 1" (el nombre debe tener como mínimo un carácter)

- Teclee: <ENTER>

Aparecerá en la pantalla el mensaje:

CONECTE EL GRABADOR Y TECLEE <ENTER>

- Acuérdese que la grabadora debe estar conectada en el modo de grabación (PLAY/REC).

- Después de haber tecleado <ENTER>, aparecerán en los bordes de la pantalla bandas horizontales. Con la primera señal se produce la grabación del nombre del programa; con la segunda, cuando las bandas son más estrechas y en colores azul y amarillo, el propio programa está siendo grabado.

- Espere hasta que aparezca en la pantalla el mensaje:

  0 Ok 0:1

- Pare la grabadora: el programa está grabado.

3. LOAD "programa"

Esta instrucción recupera los programas grabados en cinta por la instrucción SAVE. El procedimiento de la instrucción LOAD es el siguiente:

- Se busca en la cinta. Si existen otros programas grabados antes del que debe ser cargado, surgirán en pantalla sus respectivos nombres. Cuando el programa deseado es encontrado, comienza a ser transferido a la memoria del ordenador.

- El nombre que esté entre comillas deberá ser idéntico al del programa deseado; de lo contrario, el sistema no encontrará el programa.

- También se puede dispensar el nombre entre las comillas:

  LOAD" "

En este caso, el primer programa encontrado será trasladado a la memoria.

- Al ejecutarse la instrucción LOAD, se observan en los bordes de la pantalla bandas finas horizontales. Las acompaña una señal sonora aguda y constante. Los efectos visuales y sonoros de la cinta son semejantes a los de la grabación.

Para que se pueda hacer una transferencia del programa de la cinta hacia el ordenador, se debe:

- Conectar el cable de lectura/grabación, siguiendo las instrucciones del capítulo "Instalación".

- Ajustar el volumen de sonido de la grabadora al máximo de agudos.

- Accionar el comando LOAD"" o LOAD"nombre"

- Apretar <ENTER>.

- Accionar PLAY en la grabadora.

- Esperar el mensaje

  "0 Ok 0:1"

4. SAVE...LINE

Estas dos instrucciones conjuntas tienen la misma finalidad que la instrucción SAVE, pero cuando termina la lectura del programa por la instrucción LOAD éste es inmediatamente ejecutado a partir del número de la línea especificada después del LINE.

Procedimento para grabación en el Modo SAVE/LINE: Se define:

SAVE "nombre del programa" LINE número de la línea.

Siga las instrucciones del modo SAVE vistas anteriormente.

5. SAVE...DATA

También se pueden guardar datos de una matriz, en cinta cassette. Esto se hace con dos instrucciones conjuntas: SAVE y DATA.

Ejemplo:

SAVE" nombre del programa" DATA nombre de la matriz ( )

El nombre de la matriz indica cuál matriz es la que se quiere grabar. Podrá ser sólo de una letra o de una letra seguida por $ (en caso de matriz alfanumérica) siendo obligatorio el uso de los paréntesis al final; de lo contrario, se incurre en error de sintaxis.

A continuación un ejemplo, donde SAVE/DATA está en línea de programa (nada impide que sean usadas en el modo inmediato).

Prepare la grabadora para grabación y ejecute:

```
<NEW>
10 DIM a$(5,10)
20 FOR x=1 TO 5
30 INPUT "INTRODUZCA NOMBRE !";a$(x)
35 NEXT x
40 SAVE "prueba" DATA a$()
```

La línea 10 del programa, define que podrán ser inscritos en la matriz cinco strings, con diez caracteres cada uno.

Para facilitar la entrada de datos, fue utilizado un bucle FOR...NEXT, de cinco ciclos, que corresponden a la cantidad de datos que serán introducidos en a$.

La línea 40 graba todos los datos de a$(1) hasta a$(5), pero no graba el programa, sólo los nombres introducidos por el INPUT de la línea 30. Si quiere grabar el programa y el contenido de la matriz, debe usar SAVE "nombre" (en el formato normal).

6. VERIFY...DATA

Después de haber grabado los datos de una una matriz usando las instrucciones SAVE DATA, se puede comparar si lo que fue grabado corresponde fielmente a las informaciones existentes en la memoria. VERIFY DATA tiene esa finalidad.

Ejemplo:

- Rebobine la cinta hasta el punto donde comenzó la grabación de los datos de la matriz.
- Conecte el cable de lectura/grabación.
- Ajuste el volumen.
- Introduzca:

VERIFY "prueba" DATA a$( )

Serán presentados en la pantalla el nombre atribuido a la matriz y las bandas horizontales características. Una señal sonora aguda y constante será emitida.

- Espere el mensaje:

0 Ok 0:1 o

R Error de lectura 0:1 (si la grabación no fue satisfactoria, debiendo ser repetida)

La instrucción VERIFY fue usada para verificar grabaciones de datos de matrices, sin embargo, puede también ser usada con otros tipos de SAVE. Verifíquelo en la lista de instrucciones.

7. LOAD...DATA

Después de haber grabado los datos de una matriz, puede recuperarlos, usando instrucciones conjuntas LOAD/DATA. El procedimento de lectura es idéntico al del LOAD normal, visto anteriormente.

El formato de la instrucción es:

LOAD "prueba" DATA a$( )

Limpie la memoria:

<NEW>

Teclee el programa a continuación:

```
10 REM este programa recupera las matrices
20 PRINT "prepare la grabadora para lectura"
25 PRINT AT 4,2: PRINT "cuando OK teclee algo"
30 INKEY$ = IF""THEN GOTO 30
40 LOAD "prueba" DATA a$( )
45 CLS
50 FOR c = 1 TO 5
60 PRINT a$(c)
70 NEXT c
```

Ejecute el programa.

LOAD...DATA puede ser usado en modo inmediato (sin número de línea).

Nota: Recuerde que cuando se carga una matriz de datos alfanuméricos, se borra cualquier matriz o cadena simple con el mismo nombre, si hay esta coincidencia.

8. SAVE...CODE

SAVE/CODE hace posible el almacenamiento de bloques de bytes en cinta. Los bytes grabados pueden representar una forma gráfica generada por un programa en modo inmediato, o cualquier carácter que esté en la pantalla en el momento del comando.

El formato de la instrucción SAVE CODE es:

SAVE''nombre del programa''CODE dirección, número de bytes

El programa que sigue define formas en alta resolución, utilizando todos los colores de que se dispone en el TK. Al final de la ejcución del programa, graba solamente la pantalla, usando SAVE/CODE.

```
<NEW>
 5 LET a=40: LET f=0: LET k=0
 6 LET t=50
10 FOR x=k TO 10+k
20 FOR z=t TO a
25 PLOT INK f;x,z
30 NEXT z
35 LET a=a+8
40 NEXT x
41 LET f=f+1: IF f=7 THEN LET f=0
45 LET a=60: LET k=k+15
50 LET t=t-5: IF t=5 THEN LET t=50
55 IF x>=240 THEN GOTO 70
60 GOTO 10
70 PRINT AT 2,2;''teclee algo para grabar''
80 IF INKEY$=''''  THEN GOTO 80
85 PRINT AT 2,2;'' observe el dibujo ''
90 SAVE ''pantalla'' CODE 16384,6912
```

Observe que después del término del dibujo, una orden de grabación es ejecutada así que sea autorizada (línea 70), y todo lo que hay en la pantalla (solamente en la pantalla) es grabado. Para eso se definió la dirección (16384) y la cantidad de bytes que se quiere grabar. En este caso, 16384 corresponde a la dirección del primer byte del fichero del display (imagen de la TV), y 6912 es el número de bytes que existen en el fichero.

Las direcciones y cantidades de bytes son definidas de acuerdo a la necesidad del usuario, respectándose los límites admisibles.

Para almacenar, por ejemplo, el área de los caracteres gráficos definidos por el usuario, puede usar:

SAVE ''graficos'' CODE USR ''a'', 21*8

9. LOAD...CODE

Esta instrucción recupera las informaciones grabadas por SAVE/CODE: imágenes, dibujos, gráficos y programas en lenguaje máquina.

La instrucción tiene los siguientes formatos:

LOAD "pantalla" CODE 16384
o
LOAD "pantalla" CODE

Cuando la dirección en que se quiere cargar los datos de la cinta es diferente del original, se usa el primer formato. Si la dirección y el tamaño son mantenidos, la posición y tamaño originales serán asumidos.

Ejemplo:

LOAD "pantalla" CODE 16384, 5000

10. LOAD...SCREEN$

Para facilitar el proceso de lectura a ser operado por LOAD/CODE, se usa la instrucción LOAD SCREEN$. Así, en lugar de definirse los valores del primer byte y la cantidad de bytes, se usa la instrucción SCREEN$, que equivale a los códigos que serán definidos en el SAVE.

Formato:

LOAD"pantalla"SCREEN$

La grabación (SAVE/CODE) también puede ser alterada para SAVE"pantalla"SCREEN$, sin necesidad de definir los valores de dirección y cantidad de bytes.

Formato:

SAVE"pantalla"SCREEN$

11. MERGE

Hace posible la carga de un programa almacenado en cinta, sobreponiéndolo al que ya se encuentra en la memoria. Caso de que exista coincidencia de líneas de los programas, prevalecerá la línea recién cargada, por lo tanto se debe tener cuidado para que no suceda la pérdida de una línea importante del programa.

El mensaje "4 Sin memoria" aparecerá si se intenta cargar más programas de los que caben en la memoria del ordenador. Cuando eso ocurra, grabe el programa que se encuentra en la memoria, aunque esté incompleto, y límpiela. Procure reducir todo el conjunto de instrucciones para poder obtener una fusión definitiva y completa de las líneas de programación.

La instrucción MERGE es muy útil cuando se quiera recuperar programas largos, que se grabaran por partes. MERGE junta todas estas partes para formar un solo programa.

Teclee este programa:

```
<NEW>
 1 LET a=0: LET k=0: LET l=0
 4 CLS
 7 FOR c=0 TO 120 STEP 20
10 FOR x=0 TO 10
20 PLOT c+x,140
30 DRAW INK a;0,10
35 SOUND .05,x
40 NEXT x
45 LET a=a+1
50 NEXT c
53 IF k=1 THEN GOTO 77
55 PRINT "prepare la grabadora": PRINT "para grabar este programa"
60 PRINT "digite algo si tudo ok"
65 IF INKEY$="" THEN GOTO 65
75 SAVE "prog1" LINE 5 77 IF l=1 THEN GOTO 83
<RUN>
```

Despues de haber seguido las instrucciones del programa y de haberlo grabado en cinta, límpielo de la memoria:

```
<NEW>
```

Ahora teclee este otro programa:

```
 80 LET k=0
 83 LET a=0
 85 FOR r=0 TO 120 STEP 20
 90 FOR x=0 TO 20
 95 PLOT x,120-r
100 DRAW INK a;x,10
105 SOUND .05,x+10
110 NEXT x
115 LET a=a+1
120 NEXT r
125 INK 4
128 IF k=1 THEN STOP
130 PRINT "prepare la grabadora"
135 PRINT "para leer el programa anterior"
140 PRINT "digite algo cuando ok"
145 IF INKEY$="" THEN GOTO 145
190 MERGE "prog1"
195 LET a=0: LET l=1: LET k=1: GOTO 4
```

Antes de ejecutar este programa, devuelva la cinta al inicio del programa grabado anteriormente (1er programa). A continuación introduzca:

```
<RUN>
```

12. LISTA DE INSTRUCCIONES

A continuación la lista de instrucciones vistas en este capítulo:

INSTRUCCIONES	FORMATO	FINALIDAD
SAVE	SAVE''nombre''	Almacenar en cinta magnética un programa que se encuentra en la RAM.
SAVE LINE	SAVE''nombre''LINE n	Lo mismo que SAVE, pero determina la ejecución automática del programa, a partir de la línea n, después de su lectura.
SAVE DATA	SAVE''nombre''DATA x( )	Almacena en cinta magnética los datos de una matriz numérica X o alfanumérica X$.
SAVE CODE	1-SAVE''nombre''CODE n1, n2	1-Salva n2 bytes, comenzando en la dirección n1.
	2-SAVE ''pantalla'' CODE 16384	2-Salva la imagen de la pantalla con la dirección del primer byte (16384) y cantidad de bytes que grabará (6912).
SAVE SCREEN$	SAVE''nombre''SCREEN$	Lo mismo que SAVE/CODE, teniendo SCREEN$ representando la dirección y tamaño norma (16384,6912).
LOAD	LOAD''nombre''	Carga un programa grabado en cinta magnética en la memoria RAM del ordenador.
LOAD DATA	LOAD''nombre'' DATA ...( )	Carga desde una cinta hasta la RAM, datos de una matriz numérica X o alfanumérica X$.
LOAD CODE	LOAD''nombre'' CODE n1,n2	Carga un programa grabado por SAVE/CODE en una LOAD''nombre'' CODE dirección especificada o no. No es necesario que en LOAD''nombre'' CODE se defina la cantidad de bytes.
LOAD SCREEN$	LOAD''nombre''SCREEN$	Lo mismo que LOAD/CODE, pero utilizando la dirección y tamaño del área de vídeo en forma implícita.
MERGE	MERGE''nombre''	Ejecuta la sobreposición de un programa grabado en cinta en otro que ya se encuentra en la memoria.
VERIFY	VERIFY''nombre''	Compara la grabación de un programa con el programa original, que aún se encuentra en la memoria RAM.
VERIFY DATA	VERIFY''nombre''DATA	Lo mismo que VERIFY, pero comparando datos de una matriz, grabados en cinta, con los de una matriz que aún está en la memoria.
VERIFY CODE	VERIFY''nombre''CODE n1,n2	Compara los bloques de datos almacenados en la cinta con el correspondiente en la memoria entre las direcciones n1 y n1+n2.

Memoria

MEMORIA

La unidad central de proceso (UCP o CPU) de su ordenador es el microprocesador Z80.

El Z80 se encarga de procesar todos los datos e informaciones, utilizando un lugar intermedio de almacenamiento llamado memoria.

El almacenamiento de informaciones en la memoria se hace bajo la forma de bytes, y el lugar que cada byte ocupa se llama "dirección".

El Z80 es capaz de manejar hasta 2^8 16 (65536) bytes.

1. Tipos de Memoria

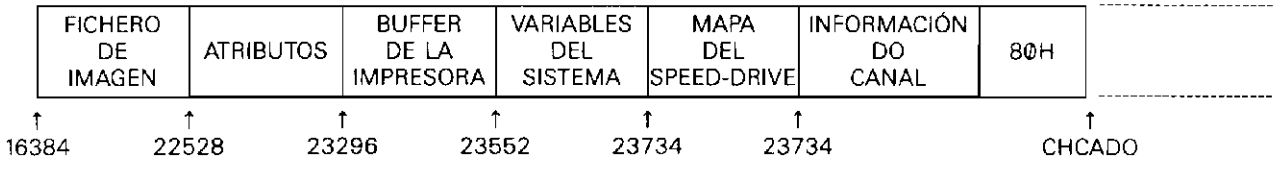
Al conectar su ordenador, éste empieza inmediatamente a funcionar, presentando la pantalla inicial. Ello ocurre porque contiene un programa "pregrabado" llamado monitor, encargado de acatar y ejecutar sus comandos y programas. Este programa está grabado en una memoria permanente llamada ROM (Read-Only Memory), ocupando las direcciones de 0 a 16.383, y no se desgraba nunca, ni cuando desconecta su ordenador.

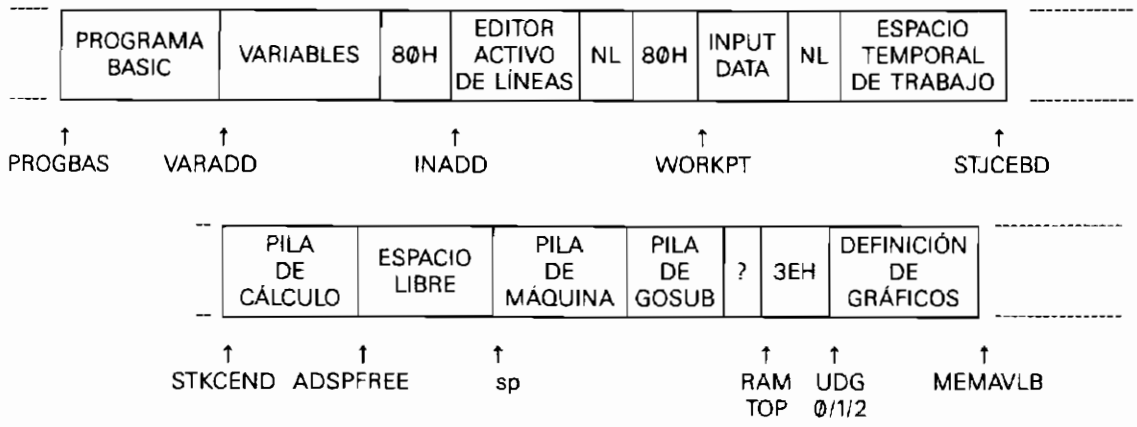
En las direcciones de memoria superiores a 16.383 existe otro tipo de memoria, la memoria RAM (Random Acess Memory), que por sus características de fabricación es volátil, es decir, puede ser leída y grabada cuantas veces se quiera, borrandose cuando se desconecta el ordenador.

2. MAPA DE LA RAM

El Sistema Operacional del TK atribuye a las áreas de la RAM diferentes funciones, de acuerdo con el tipo de informaciones a ser almacenadas en cada área.

A continuación un diagrama del mapa de la RAM del TK:



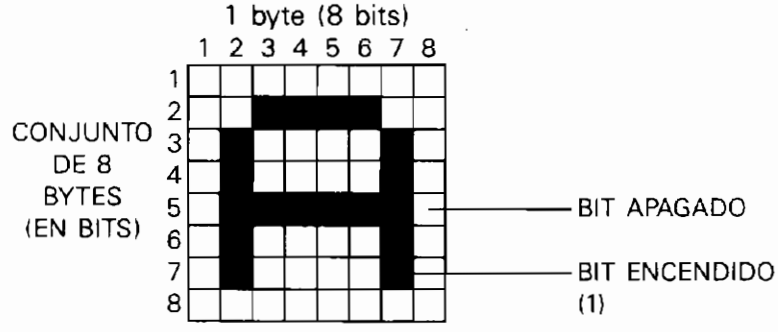


2.1. Fichero de Imagen

Existe un área de la RAM, denominada Fichero de Imagen, que tiene como objetivo guardar la imagen presente en la pantalla.

Cada una de las 768 posiciones de la pantalla está formada por un cuadrado de 8 X 8 puntos, y cada punto (bit) puede ser 1 ó Ø (encendido o apagado).

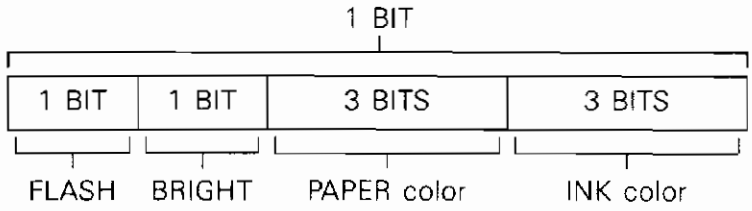
Observe la figura que sigue:



El Fichero de Imagen contiene exactamente 6144 bytes (ó 768 * 8 bytes). Como la resolución de la pantalla es de 256 X 192, se puede concluir, entonces, que cada punto de la pantalla corresponde a un bit en la memoria. El bit puede ser conectado o desconectado para componer la imagen deseada.

2.2. Atributos

En esta área de 768 bytes, son almacenados, dentro de cada byte, informaciones de INK, PAPER, BRIGHT y FLASH de cada una de las 768 posiciones de la pantalla.



2.3. Buffer de la Impresora

Área de almacenamiento intermedio de todos los caracteres destinados a la impresora.

2.4. Variables del Sistema

Los bytes de la memoria desde 23552 hasta 23733 se reservan para uso específico del sistema. Al hacerse una lectura de estas direcciones, mediante PEEK, es posible obtener varias informaciones sobre el sistema.

Las variables del sistema son representadas por nombres mnemónicos que hacen más fácil su identificación. Éstas no deben ser confundidas con las variables creadas por un programa BASIC. Están descritas con mayores detalhes en el Apéndice C.

2.5. Informacíon del Canal

Esta área contiene información sobre las unidades de entrada y salida de informaciones a partir del Sistema Operacional.

2.6. Area del Sistema BASIC

Area disponible para la entrada de datos y de programas. Es flexible en la medida en que nuevos datos sean insertados, o que datos existentes sean eliminados. Así:

Si en la memoria hay un programa en el cual se inserta una nueva línea, se crea un espacio para ésta mediante el desplazamiento de todo lo que esté arriba de dicha línea, restringiéndose, entonces, el espacio libre.

Al ser eliminada una línea, el área de programación BASIC se retrae, ampliándose el espacio libre.

2.7. Pila de Cálculo

Los números con los cuales el calculador opera están contenidos en su mayoría en esta área.

2.8. Pila de Máquina

Es la pila usada por el procesador Z8Ø, para guardar direcciones de retorno y otras informaciones. Su puntero es el propio "sp" del microprocesador.

2.9. Pila de GOSUB

Dirección del primer ítem en la Pila de Máquina, para retorno tras GOSUB.

2.10. Definición de Gráficos

Área reservada para los caracteres acentuados de los idiomas portugués y español (UDG 0/1) y gráficos definidos por el usuário (UDG2).

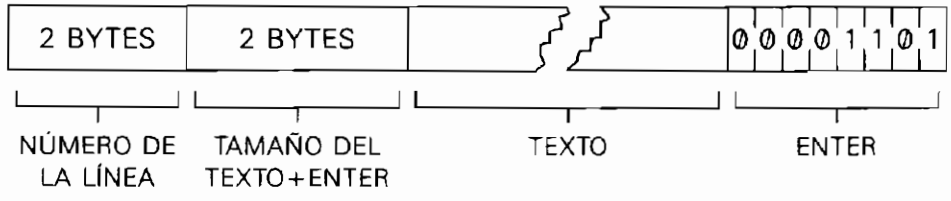
Esta área podrá ser invadida por el BASIC en casos de programas muy extensos. Para protegerla, se debe utilizar CLEAR dirección, definiéndose así la mayor dirección que el BASIC podrá alcanzar.

3. Línea de Programa en Memoria

Las líneas de programa siguen una norma de almacenamiento en la memoria. Por ejemplo:

10 PRINT 9 <ENTER>

Ésta es retratada de la siguiente forma en la memoria:



Al contrario de todos los otros casos de números de 2 bytes en el Z80, el número de la línea se guarda con su byte más significativo en primer lugar.

4. RAMTOP

Tope de la memoria RAM.

Cuando el ordenador se conecta, éste hace una prueba para verificar la cantidad de memoria, colocando la pila de la máquina exactamente en el tope. Se guarda entonces la dirección del primer byte no existente. El comando NEW no alcanza las informaciones que están arriba del RAMTOP, como, por ejemplo, los caracteres gráficos definidos por el usuario, que se encuentran en esta posición de memoria.

5. CLEAR

Se puede alterar la dirección del RAMTOP utilizando la instrucción CLEAR juntamente con la nueva dirección de la RAMTOP. Ejemplo:

CLEAR 65535

Resultado:

CLEAR cambia a cero todas las variables del BASIC y el Fichero de Imagen.

La posición del PLOT pasa a ser Ø,Ø.

Ejecuta un RESTORE, limpia la pila del GOSUB y, naturalmente, coloca el nuevo valor de RAMTOP, permaneciendo entre la pila del calculador y el final físico de la RAM.

RUN también ejecuta un CLEAR (limpia variables), pero no modifica al RAMTOP.

Código Máquina

CÓDIGO MÁQUINA

El desarrollo de programas en código máquina es un tanto más elaborado y requiere estudio apropiado. Este capítulo está dirigido a la programación avanzada.

En este capítulo serán vistos algunos conceptos básicos de este método eficiente de programación.

1. Código Binario

Recordando lo que fue visto en el capítulo "Manejando la Memoria", el Z80 entiende sólo signos binarios, lenguaje de bajo nivel, dirigida a la máquina.

Es muy trabajosa la elaboración de programas en formato binario. Para facilitar el entendimiento y el desarrollo de programas en esa forma, se hizo una convención de signos que representasen los binarios de una forma mas inteligible. El conjunto de estos signos se llama lenguaje Assembly (ensamblador).

Cuando se trabaja con el lenguaje BASIC el ordenador se encarga de ejecutar estas instrucciones de alto nivel por intermedio de un Interpretador.

Mediante de un programa utilitario (Monitor Assembler), se pueden desarrollar programas en lenguaje máquina sin ayuda del interpretador BASIC, haciendo así mas ágil el proceso.

Una vez concluido un programa en lenguaje máquina, el próximo paso será introducirlo en el ordenador con el auxilio del Monitor Assembler, a partir de una dirección predeterminada.

Existen áreas en la memoria reservadas para el uso del propio sistema operacional del ordenador (vea el Mapa de Memoria). Caso de que la dirección inicial especificada para el programa corresponda al utilizado por el Sistema Operativo, podrán ser producidos resultados inesperados. Hay que tener especial cuidado en la definición de la dirección inicial del programa en Assembly, que debe estar entre el BASIC y los caracteres definidos por el usuario.

Se puede trabajar con el BASIC conjuntamente con rutinas creadas en lenguaje máquina. Para ello, se debe definir el área que deberá ser ocupada por la rutina en Assembly, atribuyendo nuevo valor a RAMTOP. Ejemplo:

CLEAR 65Ø67

Ello hará que el espacio reservado para este fin tenga 3ØØ bytes. Recordando que el área de definición de gráficos comienza en 65368.

Vamos a ver un ejemplo de como quedaría un programa utilizando el Monitor Assembler y también el mismo programa en BASIC:

ASSEMBLER	BASIC
LD bc, 20	5 DATA 1,20,0,201
	10 LET e=32300 (ou 65068 para 48K)
	15 FOR x= 1 TO 4
ret	20 READ c: POKE e,c
	25 LET e=e+1:NEXT x

El programa inserta 20 en el par de registradores bc.

Tenemos entonces en lenguaje máquina los respectivos valores: 01, 20, 00 y 201.

01 - código de LD bc, (valor cargado en el registrador c)
20 - valor cargado en el registrador b
00 - valor cargado en el registrador b
201 - código ret (retorno de la subrutina en código máquina)

Utilizamos, en BASIC, el comando POKE, para insertar, a partir de la dirección 32299, los respectivos códigos.

2. USR Número

Rutinas en lenguage máquina pueden ser ejecutadas en un programa BASIC, si se usa la función USR. El argumento de esta función es la dirección del inicio de la subrutina y su resultado es un entero de dos bytes, sin signo, o con el signo del par de registradores bc en retorno.

La dirección de retorno para el BASIC se mantiene en la forma normal, siendo el retorno efectuado por medio de una instrucción "ret".

Vea el ejemplo:

PRINT USR 65068

El resultado presentado será 20.

El "ret" es la instrucción de retorno al BASIC después de la ejecución del programa en código máquina, cuya dirección está en la pila en forma convencional.

Para almacenar en cinta un programa en código máquina se procede de la siguiente forma:

SAVE "nombre del programa" CODE dirección inicial, cantidad de bytes

Para lectura:

LOAD "nombre del programa" CODE dirección inicial, cantidad de bytes

Use SAVE...LINE para ejecución inmediata después de la lectura.

Para aclarar mejor el tema abordado, sugerimos consultar la literatura técnica sobre el lenguaje Assembly.

3. IN — OUT

Las funciones IN y OUT se utilizan en programación avanzada. De forma análoga a las funciones PEEK y POKE, IN y OUT ejecutan un papel de lector (IN) y determinador (OUT) de los valores que se encuentran en los puertos de I/O (entrada/salida).

Forma IN y OUT:

IN dirección
OUT dirección, valor

Ejemplos:

```
PRINT IN 200
OUT 16384,20
```

Los puertos de entrada y salida son usados por el procesador para comunicarse con periféricos adicionales y teclado. Estos puertos pueden ser controlados por IN y OUT para recibir o enviar datos de estos dispositivos.

APÉNDICE.A

APÉNDICE A

PROGRAMAS:

Este apéndice está destinado a presentar programas razonablemente simples para demostrar algunas habilidades de su ordenador.

PAISAJE

El ejemplo es apropiado para que se tenga una idea de la alta resolución gráfica de su microordenador y de la nitidez de los colores que produce.

Su ejecución genera un bello amanecer marino.

Sugestión: grabar sólo la pantallla con SAVE "nombre" SCREEN$.

```
  5 PAPER 1: BORDER 0: CLS
  6 GOSUB 120
  8 INK 0
 10 FOR c=1 TO 129 STEP 3
 15 PLOT 0+c,80
 20 DRAW 45-c,40,-1
 25 PLOT 50+c,80
 30 DRAW 30-c,20,-1
 40 NEXT c
 65 FOR c=1 TO 48 STEP 2
 70 PLOT 178+c,80
 75 DRAW 20-c,8,-.01
 80 PLOT 206+c,80
 85 DRAW 10-c,6,-1
 90 NEXT c
100 FOR k=2 TO 0.01 STEP -.05
105 PLOT INK 6;118,81
110 DRAW INK 6;60,0,-k
115 NEXT k
117 STOP
120 FOR s=174 TO 85 STEP -1
125 PLOT INK 5;0,s
130 DRAW PAPER 5; INVERSE 1; INK 0;255:,0: DRAW PAPER 5; INVERSE 1; INK 0;-255,-1
135 NEXT s
150 RETURN
```

Los programas siguientes no tienen pretensiones científicas. Son sólo ilustrativos, sin embargo puede aprovechar sus fundamentos para obtener gráficos bastante útiles:

Curva del SIN

```
<NEW>-<ENTER>
 5 BRIGHT 1: BORDER 5: CLS
10 PRINT AT 0,16;"**curva de sin**"
15 FOR x=0 TO 255
20 PLOT INK 1;x,87
25 PLOT INK 1;127,175-x
30 NEXT x
35 FOR m=-20 TO 20 STEP .10
40 PLOT INK 2;m*6+127,87+20*SIN m
45 NEXT m
```

Curvas SIN — COS — SQR

```
<NEW>-<ENTER>
 5 BRIGHT 1: BORDER 5: PAPER 7: CLS
10 INK 0
15 FOR v=-1 TO 254
20 PLOT 0,174-v: PLOT v,87.5
25 NEXT v
30 PRINT INK 2;AT 20,2;"sqr";AT 0,2;"cos";AT 9,2;"sin"
35 FOR n=5 TO 250
40 INK 4: PLOT n,90+80*SIN (n/128*PI)
45 INK 1: PLOT n,80**SQR (n/64)
50 INK 3: PLOT n,90+80*COS (n/128*PI)
55 NEXT n
```

Música

Green Leaves to a Ground

```
<NEW>-<ENTER>
 5 DIM n(75): DIM t(75)
10 DATA 2,5,7,9,10,9,7,4,0,2,4,5,2,2,1,2,4,1,-3,2,5,7,9,10,9,7,4,0,2,4,5,4,2,1,-1, 1,2,
   69.84,12,12,12,11
11 DATA 9,7,4,0,2,4,5,2,2,1,2,4,1,-3,69.84,12,12,12,11,9,7,4,0,2,4,5,4,2,1,-1,1,2,69.84
15 DATA .3,.6,.3,.6,.1,.1,.6,.3,.6,.1,.1,.6,.3,.6,.1,.1,.6,.3,.6,.3,.6,.3, .6,.1,.1,.6,.3,.6,.1,.1,
   .3,.3,.3,.3,.3,.3,.6.,6,.6,.3,.3
16 DATA .3,.3,.6,.3,.3,.3,.3,.6,.3,.3,.3,.3,.6,.3,.6,.6,.6,.3,.3,.3,.3,.6,
   .3,.3,.3,.3,.3,.3,.3,.3,.3,.3,1,.6
20 FOR x=1 TO 75: READ n(x): NEXT x
25 FOR x=1 TO 75: READ t(x): NEXT x
26 FOR a=1 TO 2
30 FOR s=1 TO 75: SOUND t(s),n(s): NEXT s
35 NEXT a
```

Advine

Ejecute este programa, y diviértase tratando de adivinar lo que éste pide.

Limpie la memoria del ordenador con <NEW> y acceda al comando CAPS LOCK.

A continuación teclee:

```
 1 BORDER 1: PAPER 6
 2 CLS
 4 PRINT PAPER 2; INK 7;"# # # # # # # ADIVINE SI PUEDE # # # # # # #"
 5 LET N=INT (RND*100)
10 FOR C=10 TO 1 STEP -1
15 PRINT AT 8,1;"UD.TIENE OPORTUNIDAD PARA "
16 PRINT AT 8,11; INK 0;C
17 IF C=1 THEN SOUND .2,-10: SOUND .2,-10: SOUND .2,-10: SOUND 1,-14
20 PRINT AT 9,5;"ADIVINAR EL NUMERO "; INK 2;""""N""""
30 INPUT "INDIQUE (0 A 100): ";P
35 IF P>N THEN PRINT AT 12,7;P;" ES MAYOR QUE "; INK 1;""""N"""";"      "
40 IF P<N THEN PRINT AT 12,7;P;" ES MENOR QUE "; INK 1;""""N"""";"      "
45 IF P=N THEN PRINT INK 2; FLASH 1;AT 12,1;"FELICITACIONES!!! UD. ACERTO":
   GOTO 200
46 SOUND .3,-12
50 NEXT C
60 IF C=0 AND N <> P THEN GOSUB 300
100 CLS
105 PRINT AT 20,1;"QUIERE JUGAR NUEVAMENTE (S/N)"
106 PAUSE 20:FOR T=1 TO 60 STEP 10:SOUND .1,T NEXT T
110 IF INKEY$="" THEN GOTO 110 115 IF INKEY$="S" THEN GOTO 2
120 CLS: PRINT PAPER 4;"FIN": STOP
200 FOR T=1 TO 8
210 SOUND .1,10: SOUND .1,20: SOUND .1,30: NEXT T
220 GOTO 100
300 FOR T=-8 TO -20 STEP-1:SOUND .08,T: NEXT T
305 RETURN
```

APÉNDICE.B

APÉNDICE B

MENSAJES

El TK está dotado de un sistema que informa al úsuario, por medios de mensajes, de cualquier situación que ocasione una parada. La interrupción puede ser causada por un error de programación, por un comando de parada del tipo BREAK o STOP, o por la simple conclusión de la ejecución del programa.

En el sistema existen 27 mensajes, precedidos por un código numérico de Ø a 9, o una letra de A a R. Además del código y del texto del informe, aparecen también dos números separados por dos puntos. El primer número informa cuál fue la última línea del programa que el ordenador ejecutó. El segundo indica cuántos comandos de la última línea fueron ejecutados.

Por ejemplo:

9 STOP ejecutado 3Ø:3

significa que el programa fue ejecutado hasta la línea 3Ø y que paró después de haber sido efectuada la tercera instrucción. Cada mensaje tiene características relativas a la situación que lo generó.

A continuación tiene una lista de todos los mensajes, su significado y la circunstancia en la cual el mensaje puede aparecer.

Nota: X y Y serán usadas para simbolizar los números que aparecen en el final de los mensajes, separados por dos puntos.

Ø Ok X:Y

Término de una ejecución bien ejercida o desvío hacia un número de línea superior al más alto número de línea de un determinado programa.

1 NEXT sin FOR X:Y

La variable que controla el número de bucles no fue definida por una instrucción FOR, pero existe una variable común con el mismo nombre de la que está en el NEXT.

2 Variable inexiste X:Y

Tentativa de utilizar variable común no definida en el programa. La variable debería haber sido introducida por instrucciones de entrada de datos del tipo LET, READ o INPUT, cargada de una cinta o definida en un FOR. El mensaje también aparece cuando una variable indexada fue utilizada antes de haber sido dimensionada una matriz por DIM.

3 Indice errado X:Y

Índice de variable fuera de las dimensiones de la matriz, indexación inadecuada del dimensionamiento de una matriz.

4 Sin memoria X:Y

Tentativa de reservar o ocupar más espacio del que la memoria dispone. Es necesario eliminar la línea que está siendo editada en caso de que el ordenador quede paralizado. A veces es necesario eliminar una o dos líneas del programa, se debe utilizar el comando CLEAR para obtener más espacio, y entonces volver a introducir las líneas eliminadas.

5 Area-video excedida X:Y

Se produce el mensaje cuando hay una tentativa de utilizar la 23ª línea de la pantalla, en un comando del tipo PRINT AT 23,...

6 Numero excede limite X:Y

Cálculo resultando un número superior a 10 digitos aproximadamente.

7 RETURN sin GOSUB X:Y

Ejecución de un RETURN sin haber sido accedido por un GOSUB.

8 Fin de archivo X:Y

Mensaje de algunos periféricos.

9 STOP ejecutado X:Y

Interrupción realizada por la presencia de una instrucción STOP en una línea de programa. Para proseguir la ejecución de eventuales líneas subsiguientes, hay que usar el comando CONT.

A Argumento ilegal X:Y

Introducción de un argumento, que de alguna forma es impropio para una de las siguientes funciones: SQR, LN, ACS o USR.

B Entero excede limite X:Y

Aparece cuando un entero es solicitado y el valor introducido (eventualmente redondeado) está más allá del límite permitido.

C Sintaxis error X:Y

El texto del argumento (string) no constituye expresión válida. El argumento está relacionado a VAL o VAL$.

D BREAK-CONT repite X:Y

Accesión del BREAK durante el funcionamento de algún periférico o respuesta negativa (N) al scroll?. Si se usa CONT después del mensaje, el último comando será repetido.

E Sin datos X:Y

Tentativa de usar el comando READ, más veces de lo que permite el número de datos existentes en DATA.

F Nombre ilegal X:Y

Cuando se usa SAVE y un nombre de fichero que excede de 10 caracteres o que no tiene ningún carácter (string vacío).

G Memoria ocupada

No hay más lugar para acomodar una nueva línea de programa.

H STOP en INPUT X:Y

Introducción de STOP durante un INPUT. Si se usa CONT, el INPUT será repetido.

I FOR sin NEXT X:Y

Cuando el valor determinado para la variable de control del FOR no permite que ningún bucle sea realizado, y también cuando no fue encontrado el NEXT correspondiente a tal variable. Por ejemplo:

FOR X= 2 TO 1:INSTRUCCIONES: NEXT m

J Periferico ilegal

Dispositivo de entrada/salida no válido.

K Color ilegal X:Y

El número escogido está fuera del límite de los colores existentes. Se produce también cuando se determinan otros números que no es 0 ó 1 para FLASH, BRIGHT, INVERSE u OVER, o después de uno de los carateres de control correspondientes.

L BREAK-CONT prosigue X:Y

Se produce cuando se presiona BREAK en la ejecución de un programa, entre dos comandos. En este caso X y Y se refieren a la línea y a la instrucción ejecutadas antes del BREAK. Si CONT es accionado, el programa proseguirá desde el punto en que paró, sin repetir ningún comando.

M RAMTOP ilegal X:Y

Se produce cuando se determina un número muy grande o muy pequeño para RAMTOP, mediante el comando CLEAR (a veces también con RUN).

N Comando perdido X:Y

Tentativa de desvío hacia un comando inexistente. Se produce generalmente con las instrucciones RETURN, NEXT y CONT.

El Canal ilegal

Mensaje de periféricos.

P FN sin DEF X:Y

Cuando hay tentativa de usar FN sin que haya sido ejecutado un DEF FN previamente.

Q Error de parametro X:Y

Durante el uso de FN, la determinación de los argumentos ha sido incorrecta en la cantidad o en el tipo (string en lugar de número o viceversa).

R Error de lectura X:Y

Por algún motivo no fue posible leer los datos de una cinta.

APÉNDICE.C

APÉNDICE C

VARIABLES DEL SISTEMA

DIRECCION	NOMBRE	CONTENIDO
23552	KBDWORK	Lectura del teclado.
23560	KEYPRS	Acumula la última tecla presionada.
23561	RPTDLAY	Auto REPEAT, 1/60 de segundo para que una tecla presionada comience a repetir. Valor inicial: 35, pudiendo alterarse.
23562	RPTCCLE	1/60 de segundo de intervalo entre las repeticiones sucesivas de una tecla presionada Valor inicial: 5.
23563	PT DEF	Dirección de los argumentos de una función definida por el usuario. Valor inicial: 0.
23565	K CLR	Acumula el segundo byte de control de color vía teclado.
23566	TV CLR	Acumula bytes de los canales conectados al sistema.
23606	PTBLCHR	Menos 256 de la dirección del conjunto de caracteres (comienza con el espacio y va hasta el símbolo delta). Normalmente en ROM, pero puede ser definido en RAM y accedido por PTBLCHR.
23608	BUZCCLE	Intervalo de la Señal sonora.
23609	KCLICK	Intervalo de la señal sonora del teclado
23610	ERRCD	Inicia en 255 (para -1) de modo que el contenido de PEEK 23610 es 255. Uno menos que el código de mensaje.
23611	SFLAG0	Diversos FLAGS de control del sistema BASIC.
23612	SFLAG1	FLAGS asociado a la televisión.
23613	P ERR	Dirección del elemento de la pila de la máquina que debe ser usado como regresso del error.

DIRECCION	NOMBRE	CONTENIDO
23615	P LIS	Dirección de retorno del listado automático.
23617	CURSOR	Especifica cursor: K , L , C , E ó G
23618	LNJMP	Línea-destino del salto.
23620	INSTRNR	Número de la instrucción en la línea-destino del salto. Haciendo un POKE primero para LNJMP y luego para INSTRNR, fureza un salto hacia una instrucción específica de una línea.
23621	EXCLINE	Número de línea de la instrucción que está siendo ejecutada.
23623	SUBLEXC	Número dentro de la línea de instrucción que está siendo ejecutada.
23624	BORCLR	Número de color definido con BORDER mùltiplicado por 8, más los atributos de la parte inferior de la pantalla.
23625	CURLINE	Número de la línea-corriente (con cursor de programa).
23627	VARADD	Dirección de las variables.
23629	XVARADD	Dirección de destino de la variable a ser modificada.
23631	CHCADD	Dirección de los datos del canal.
23633	IOADD	Dirección de la información que está siendo utilizada para entrada/salida.
23635	PROGBAS	Dirección del programa BASIC.
23637	NEXEXC	Dirección de la próxima línea del programa.
23639	ENDDATA	Dirección del indicador del último elemento DATA.
23641	INADD	Dirección de la instrucción que será introducida vía teclado.
23643	CURADD	Dirección del cursor.
23645	CHNXADD	Dirección del próximo carácter que será interpretado: el carácter después del argumento PEEK o NEW LINE en el final de la instrucción POKE.
23647	SYCHADD	Dirección del carácter después de ?.
23649	WORKPT	Dirección del espacio de trabajo temporal.
23651	STKCEND	Dirección final de la pila del calculador

DIRECCION	NOMBRE	CONTENIDO
23653	ADSPFREE	Dirección del inicio del espacio libre.
23655	BREGCAL	Registro b del calculador.
23656	MEMCADD	Dirección del área usada para la memoria del calculador (generalmente MENSPCAL).
23658	SFLAG2	+ n FLAGS.
23659	SIZE	Número de líneas de texto (incluyéndose una vacía) en la parte inferior de la pantalla
23660	LISTNR	Número inicial de las líneas de un programa para lista automática
23662	CONTJMP	Número de la línea destino después de CONT.
23664	CONTNR	Número de instrucciones dentro de la línea destino después de CONT.
23665	SFLAG3	n FLAGS.
23666	STRVLEN	Largo de la variable que será modificada.
23668	SYTADD	Direcci fon del próximo elemento de la tabla de sintaxis.
23670	INITRND	El primer número para la intrucción RND. Esta variable es inicializada por RAND.
23672	TVCOUNT	3 bytes, contador de imágenes (el primero menos significativo). Incrementando todos los 20 ms.
23675	UDGRAPH	Dirección del primer carácter gráfico definido por el usuario.
23677	LSTPLOT	Coordenada X del último punto de un PLOT.
23678	COORDS	Coordenada Y del último punto de un PLOT.
23679	POSIMPR	Número hasta 33 para posicionar columna en la impressora.
23680	PRTADD	Byte menos significativo de la dirección de la próxima posición en que la LPRINT debe escribir (en el buffer de la impresora).
23681		No es usado.

DIRECCION	NOMBRE	CONTENIDO
23682	HVBFFIN	Número de columna (hasta 33) y líneas (hasta 24) de la memoria de entrada.
23684	DFPSPRT	Dirección, en el Fichero de Imagen, de la posición PRINT.
23686	DPSPRTL	La misma función de DFPSPRT, pero para la parte inferior de la pantalla.
23688	HV POS	Número de columnas (hasta 33) para la posición de PRINT.
23689		Número de líneas (hasta 24) para la posición de PRINT.
23690	HV POSL	Como HV POS, pero para la parte inferior.
23692	SCRINC	Contador para scroll: es siempre 1 menos que el número de scrolls que serán hechos antes de la pregunta scroll?. Si introduce continuamente en esta dirección un número mayor que 1, la pantalla continuará haciendo scroll ininterumpidamente.
23693	ATCLR P	Situación de la posición permanente de la pantalla, en cuanto al color, etc.
23694	MASKCLRP	Usado para los colores transparentes, etc. muestra que el bit de atributo correspondiente no es tomado de ATCLR P, sino que aquél que ya está en la pantalla.
23695	ATCLR T	Situación de una posición de la pantalla.
23696	MASKCLRT	Lo mismo que MASKCLRP, pero temporal.
23697	SFLAG4	n FLAG.
23698	MEMSPCAL	Área de memoria del calculador usada para guardar números que no pueden ser colocados convenientemente en la pila del calculador.
23728	NMIVCT	Usado en la interrupción del NMI.
23730	RAMTOP	Dirección del último byte del área del sistema BASIC.
23732	MEMAVLB	Dirección del último byte de la RAM física.

APÉNDICE.D

APÉNDICE D

CÓDIGO DE CARACTERES

CÓDIGO	CARÁCTER	HEXA	ASSEMBLY Z80	DESPUÉS DEL CB	DESPUÉS DEL ED
0	UDG	00	nop	rlc b	
1	TRACE	01	Ld bc,N	rlc c	
2	No usado	02	Ld (bc),a	rlc d	
3	No usado	03	inc bc	rlc e	
4	No usado	04	inc b	rlc h	
5	No usado	05	dec b	rlc l	
6	PRINT coma	06	Ld,b,N	rlc (hl)	
7	EDIT	07	rica	rlc a	
8	cursor hacia la izquierda	08	ex af,af'	rrc b	
9	cursor hacia la derecha	09	add hl,bc	rrc c	
10	cursor hacia abajo	0A	Ld a,(bc)	rrc d	
11	cursor hacia arriba	0B	dec bc	rrc e	
12	DELETE	0C	inc c	rrc h	
13	ENTER	0D	dec c	rrc l	
14	Número	0E	Ld c,N	rrc (hl)	
15	No usado	0F	rrca	rrc a	
16	INK control	10	djnz DIS	rl b	
17	PAPER control	11	Ld de,NN	rl c	
18	FLASH control	12	Ld (de),a	rl d	
19	BRIGHT control	13	inc de	rl e	
20	INVERSE control	14	inc d	rl h	
21	OVER control	15	dec d	rl l	
22	AT control	16	Ld d, N	rl (hl)	
23	TAB control	17	rla	rl a	
24	No usado	18	jr DIS	rr b	
25	No usado	19	add hl,de	rr c	
26	No usado	1A	Ld a,(de)	rr d	
27	No usado	1B	dec de	rr e	
28	No usado	1C	inc e	rr h	
29	No usado	1D	dec e	rr l	
30	No usado	1E	Ld e,N	rr (hl)	
31	No usado	1F	rra	rr a	
32	Espacio	20	jr nz,DIS	sla b	
33	!	21	Ld hl,NN	sla c	
34	"	22	Ld (NN),hl	sla d	
35	#	23	inc hl	sla e	
36	$	24	inc h	sla h	
37	%	25	dec h	sla l	
38	&	26	Ld h,N	sla (hl)	
39	'	27	daa	sla a	
40	(	28	jr z,DIS	sra b	
41	)	29	add hl,hl	sra c	
42	*	2A	Ld hl,(NN)	sra d	

CÓDIGO	CARÁCTER	HEXA	ASSEMBLY Z80	DESPUÉS DEL CB	DESPUÉS DEL ED
43	+	2B	dec hl	sra e	
44	,	2C	inc l	sra h	
45	-	2D	dec l	sra l	
46	.	2E	Ld l,N	sra (hl)	
47	/	2F	cpl	sra a	
48	Ø	3Ø	jr nc,DIS		
49	1	31	Ld sp,NN		
5Ø	2	32	Ld (NN),a		
51	3	33	inc sp		
52	4	34	inc (hl)		
53	5	35	dec (hl)		
54	6	36	Ld (hl),N		
55	7	37	scf		
56	8	38	jr c,DIS	srl b	
57	9	39	add hl,sp	srl c	
58	:	3A	Ld a,(NN)	srl d	
59	;	3B	dec sp	srl e	
6Ø	<	3C	inc a	srl h	
61	=	3D	dec a	srl l	
62	>	3E	Ld a,N	srl (hl)	
63	?	3F	ccf	srl a	
64	@	4Ø	Ld b,b	bit Ø,b	in b, (c)
65	A	41	Ld b,c	bit Ø,c	out (c),b
66	B	42	Ld b,d	bit Ø,d	sbc hl,bc
67	C	43	Ld b,e	bit Ø,e	Ld (NN),bc
68	D	44	Ld b,h	bit Ø,h	neg
69	E	45	Ld b,l	bit Ø,l	retn
7Ø	F	46	Ld b,(hl)	bit Ø,(hl)	im Ø
71	G	47	Ld b,a	bit Ø,a	Ld i,a
72	H	48	Ld c,b	bit 1,b	in c,(c)
73	I	49	Ld c,c	bit 1,c	out (c),c
74	J	4A	Ld c,d	bit 1,d	adc hl,bc
75	K	4B	Ld c,e	bit 1,e	Ld b,c(NN)
76	L	4C	Ld c,h	bit 1,h	
77	M	4D	Ld c,l	bit 1,l	reti
78	N	4E	Ld c,(hl)	bit 1,(hl)	
79	O	4F	Ld c,a	bit 1,a	Ld r,a
8Ø	P	5Ø	Ld d,b	bit 2,b	in d,(c)
81	Q	51	Ld d,c	bit 2,c	out (c),d
82	R	52	Ld d,d	bit 2,d	sbc hl,de
83	S	53	Ld d,e	bit 2,e	Ld (NN),de
84	T	54	Ld d,h	bit 2,h	
85	U	55	Ld d,l	bit 2,l	
86	V	56	Ld d,(hl)	bit 2,(hl)	im1
87	W	57	Ld d,a	bit 2,a	Ld a,i
88	X	58	Ld e,b	bit 3,b	in e,(c)
89	Y	59	Ld e,c	bit 3,c	out (c),e
9Ø	Z	5A	Ld e,d	bit 3,d	adc hl,de
91	[	5B	Ld e,e	bit 3,e	Ld de,(NN)
92	/	5C	Ld e,h	bit 3,h	
93	]	5D	Ld e,l	bit 3,l	
94	↑	5E	ld e,(hl)	bit 3,(hl)	im 2
95	—	5F	Ld e,a	bit 3,a	Ld a,r
96	Σ	6Ø	Ld h,b	bit 4,b	in h,(c)

CÓDIGO	CARÁCTER	HEXA	ASSEMBLY Z80	DESPUÉS DEL CB	DESPUÉS DEL ED
97	a	61	Ld h,c	bit 4,c	out (c),h
98	b	62	Ld h,d	bit 4,d	sbc hl,hl
99	c	63	Ld h,e	bit 4,e	Ld (NN),hl
100	d	64	Ld h,h	bit 4h	
101	e	65	Ld h,l	bit 4,l	
102	f	66	Ld h,(hl)	bit 4,(hl)	
103	g	67	Ld h,a	bit 4,a	rrd
104	h	68	Ld l,b	bit 5,b	in l,(c)
105	i	69	Ld l,c	bit 5,c	out (c),l
106	j	6A	Ld l,d	bit 5,d	adc hl,hl
107	k	6B	Ld l,e	bit 5,e	Ld hl,(NN)
108	l	6C	Ld l,h	bit 5,h	
109	m	6D	Ld l,l	bit 5,l	
110	n	6E	Ld l,(hl)	bit 5,(hl)	
111	o	6F	Ld l,a	bit 5,a	rld
112	p	70	Ld (hl),b	bit 6,b	in f,(c)
113	q	71	Ld (hl),c	bit 6,c	
114	r	72	Ld (hl),d	bit 6,d	sbc hl,sp
115	s	73	Ld (hl),e	bit 6,e	Ld (NN),sp
116	t	74	Ld (hl),h	bit 6,h	
117	u	75	Ld (hl),l	bit 6,l	
118	v	76	halt	bit 6,(hl)	
119	w	77	Ld (hl),a	bit 6,a	
120	x	78	Ld a,b	bit 7,b	in a,(c)
121	y	79	Ld a,c	bit 7,c	out (c),a
122	z	7A	Ld a,d	bit 7,d	adc hl,sp
123	{	7B	Ld a,e	bit 7,e	Ld sp,(NN)
124	|	7C	Ld a,h	bit 7,h	
125	}	7D	Ld a,l	bit 7,l	
126	~	7E	Ld a,(hl)	bit 7,(hl)	
127	△	7F	Ld a,a	bit 7,a	
128	□	80	add a,b	res 0,b	
129	▗	81	add a,c	res 0,c	
130	▖	82	add a,d	res 0,d	
131	▄	83	add a,e	res 0,e	
132	▝	84	add a,h	res 0,h	
133	▐	85	add a,l	res 0,l	
134	▚	86	add a,(hl)	res 0,(hl)	
135	▟	87	add a,a	res 0,a	
136	▘	88	adc a,b	res 1,b	
137	▚	89	adc a,c	res 1,c	
138	▌	8A	adc a,d	res 1,d	
139	▙	8B	adc a,e	res 1,e	
140	▀	8C	adc a,h	res 1,h	
141	▜	8D	adc a,l	res 1,l	
142	▛	8E	adc a,(hl)	res 1,(hl)	
143	■	8F	adc a,a	res 1,a	
144	(a) UDG	90	sub b	res 2,b	
145	(b) UDG	91	sub c	res 2,c	
146	(c) UDG	92	sub d	res 2,d	
147	(d) UDG	93	sub e	res 2,e	
148	(e) UDG	94	sub h	res 2,h	
149	(f) UDG	95	sub l	res 2,l	
150	(g) UDG	96	sub (hl)	res 2,(hl)	

CÓDIGO	CARÁCTER	HEXA	ASSEMBLY Z80	DESPUÉS DEL CB	DESPUÉS DEL ED
151	(h) UDG	97	sub a	res 2,a	
152	(i) UDG	98	sbc a,b	res 3,b	
153	(j) UDG	99	sbc a,c	res 3,c	
154	(k) UDG	9A	sbc a,d	res 3,d	
155	(l) UDG	9B	sbc a,e	res 3,e	
156	(m) UDG	9C	sbc a,h	res 3,h	
157	(n) UDG	9D	sbc a,l	res 3,l	
158	(o) UDG	9E	sbc a,(hl)	res 3,(hl)	
159	(p) UDG	9F	sbc a,a	res 3,a	
160	(q) UDG	A0	and b	res 4,b	Ldi
161	(r) UDG	A1	and c	res 4,c	cpi
162	(s) UDG	A2	and d	res 4,d	ini
163	(t) UDG	A3	and e	res 4,e	outi
164	(u) UDG	A4	and h	res 4,h	
165	RND	A5	and l	res 4,l	
166	INKEY$	A6	and (hl)	res 4,(hl)	
167	PI	A7	and a	res 4,a	
168	FN	A8	xor b	res 5,b	Ldd
169	POINT	A9	xor c	res 5,c	cpd
170	SCREEN$	AA	xor d	res 5,d	ind
171	ATTR	AB	xor e	res 5,e	outd
172	AT	AC	xor h	res 5,h	
173	TAB	AD	xor l	res 5,l	
174	VAL$	AE	xor (hl)	res 5,(hl)	
175	CODE	AF	xor a	res 5,a	
176	VAL	B0	or b	res 6,b	Ldir
177	LEN	B1	or c	res 6,c	cpir
178	SIN	B2	or d	res 6,d	inir
179	COS	B3	or e	res 6,e	otir
180	TAN	B4	or h	res 6,h	
181	ASN	B5	or l	res 6,l	
182	ACS	B6	or (hl)	res 6,(hl)	
183	ATN	B7	or a	res 6,a	
184	LN	B8	cp b	res 7,b	Lddr
185	EXP	B9	cp c	res 7,c	cpdr
186	INT	BA	cp d	res 7,d	indr
187	SQR	BB	cp e	res 7,e	otdr
188	SGN	BC	cp h	res 7,h	
189	ABS	BD	cp l	res 7,l	
190	PEEK	BE	cp (hl)	res 7,(hl)	
191	IN	BF	cp a	res 7,a	
192	USR	C0	ret nz	set 0,b	
193	STR$	C1	pop bc	set 0,c	
194	CHR$	C2	jp nz,NN	set 0,d	
195	NOT	C3	jp NN	set 0,e	
196	BIN	C4	call nz,NN	set 0,h	
197	OR	C5	push bc	set 0,l	
198	AND	C6	add a,N	set 0,(hl)	
199	<=	C7	rst 0	set 0,a	
200	>=	C8	ret z	set 1,b	
201	<>	C9	ret	set 1,c	
202	LINE	CA	jp z,NN	set 1,d	
203	THEN	CB	set 1,e		
204	TO	CC	call z,NN	set 1,h	

CÓDIGO	CARÁCTER	HEXA	ASSEMBLY Z80	DESPUÉS DEL CB	DESPUÉS DEL ED
205	STEP	CD	call NN	set 1,l	
206	DEF FN	CE	adc a,N	set 1,(hl)	
207	CAT	CF	rst 8	set 1,a	
208	FORMAT	D0	ret nc	set 2,b	
209	MOVE	D1	pop de	set 2,c	
210	ERASE	D2	jp nc,NN	set 2,d	
211	OPEN #	D3	out (n),a	set 2,e	
212	CLOSE #	D4	call nc,NN	set 2,h	
213	MERGE	D5	push de	set 2,l	
214	VERIFY	D6	sub N	set 2,(hl)	
215	SOUND	D7	rst 16	set 2,a	
216	CIRCLE	D8	ret c	set 3,b	
217	INK	D9	exx	set 3,c	
218	PAPER	DA	jp c,NN	set 3,d	
219	FLASH	DB	in a,(N)	set 3,e	
220	BRIGHT	DC	call c,NN	set 3,h	
221	INVERSE	DD	util. inst. prefijos ix	set 3,l	
222	OVER	DE	sbc a,N	set 3,(hl)	
223	OUT	DF	rst 24	set 3,a	
224	LPRINT	D0	ret po	set 4,b	
225	LLIST	E1	pop hl	set 4,c	
226	STOP	E2	jp po,NN	set 4,d	
227	READ	E3	ex(sp),hl	set 4,e	
228	DATA	E4	call po,NN	set 4,h	
229	RESTORE	E5	push hl	set 4,l	
230	NEW	E6	and N	set 4,(hl)	
231	BORDER	E7	rst 32	set 4,a	
232	CONT	E8	ret pe	set 5,b	
233	DIM	E9	jp (hl)	set 5,c	
234	REM	EA	jp pe,NN	set 5,d	
235	FOR	EB	ex de,hl	set 5,e	
236	GOTO	EC	call pe,NN	set 5,h	
237	GOSUB	ED	set 5,l		
238	INPUT	EE	xor N	set 5,(hl)	
239	LOAD	EF	rst 40	set 5,a	
240	LIST	F0	ret p	set 6,b	
241	LET	F1	pop af	set 6,c	
242	PAUSE	F2	jp p,NN	set 6,d	
243	NEXT	F3	di	set 6,e	
244	POKE	F4	call p,NN	set 6,h	
245	PRINT	F5	push af	set 6,l	
246	PLOT	F6	or N	set 6,(hl)	
247	RUN	F7	rst 48	set 6,a	
248	SAVE	F8	ret m	set 7,b	
249	RAND	F9	Ld sp,hl	set 7,c	
250	IF	FA	jp m,NN	set 7,d	
251	CLS	FB	ei	set 7,e	
252	DRAW	FC	call m,NN	set 7,h	
253	CLEAR	FD	util. inst. prefijos iy	set 7,l	
254	RETURN	FE	cp N	set 7,(hl)	
255	COPY	FF	rst 56	set 7,a	