

# MICROHOBBY

REVISTA INDEPENDIENTE PARA USUARIOS DE ORDENADORES SINCLAIR

ESPECIAL

Nº4-350 PTS

APRENDE A MOVER  
TUS GRAFICOS  
UTILIZANDO SPRITES

Los tecnodelincuentes,  
la nueva generación del robo

Visitamos la  
escuela de  
informática

EDIGRAF

Para archivar, copiar  
gráficos y mejorar las  
posibilidades  
del Melbourne Draw

MICROMIRON

Un programa para  
abrir programas

Y ADEMÁS

El mas completo diccionario  
de código máquina

Y ADEMÁS




GUIA DE  
JOYSTICKS

Con todos los  
modelos del mercado

# UNA JUGADA MAESTRA

**T**odo sobre el baloncesto americano en fascículos. BASKET USA.

**L**os gigantes de la cancha y sus técnicas.

**C**ómo se hace un campeón.

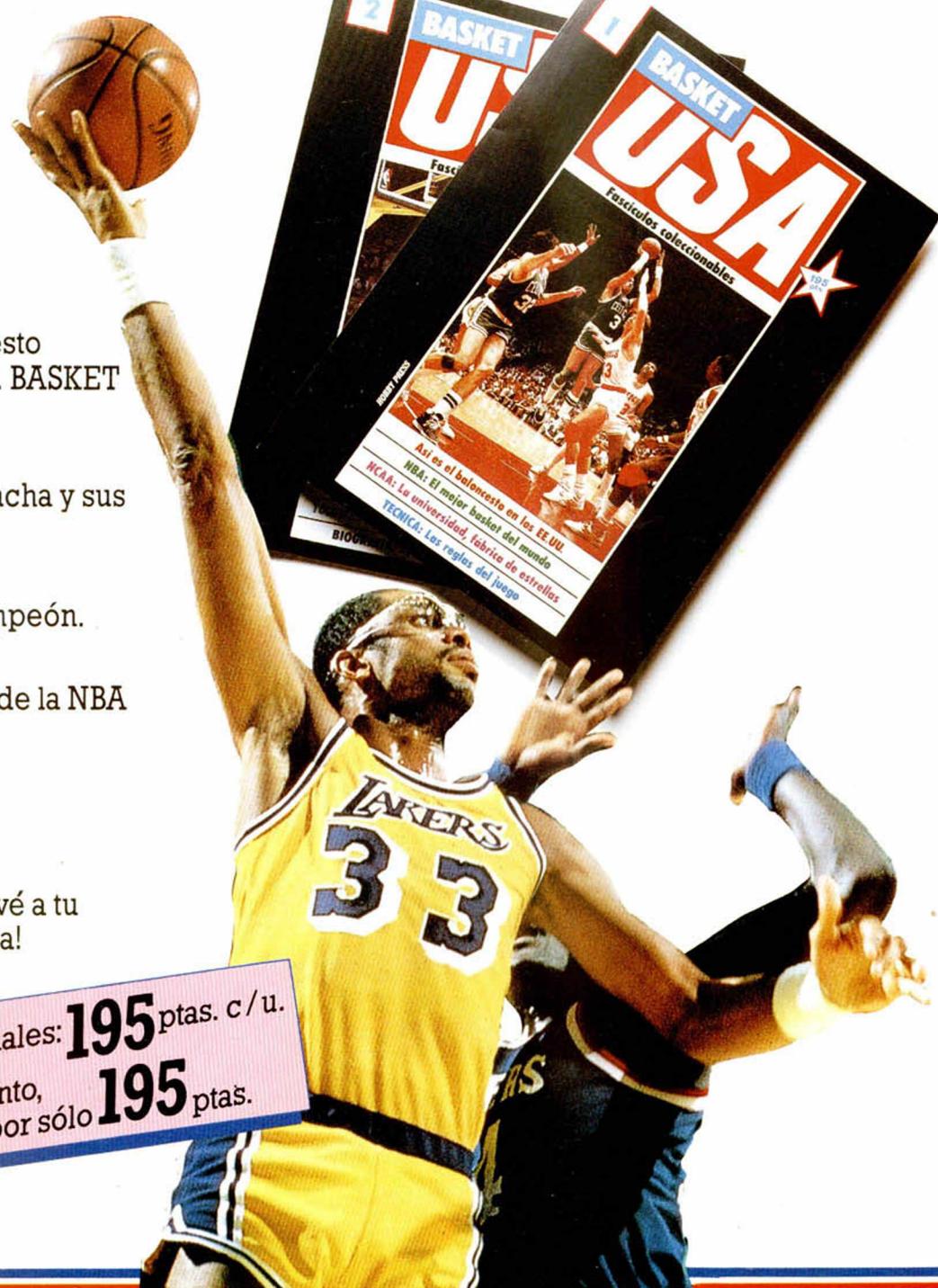
**L**os grandes equipos de la NBA y la liga Amateur...

**B**ASKET USA.

**H**az la mejor jugada, vé a tu quiosco... ¡Mételo en casa!

52 fascículos semanales: **195** ptas. c/u.

Oferta de lanzamiento,  
los números 1 y 2 por sólo **195** ptas.



**HOBBY PRESS. Para gente inquieta.**

Director Editorial  
José I. Gómez-Centurión

Director  
Gabriel Nieto

Director de Microhobby  
Domingo Gómez

Redactora Jefe  
Afrida Pérez-Tolosa

Diseño  
Carlos Catalán

Redactores  
Amaio Gómez,  
Pedro Pérez

Secretaría Redacción  
Carmen Santamaría

Colaboradores

Alejandro Júlvez, Marcos Ortiz,  
Víctor Prieto, José M. Lazo,  
J. J. García Quesada, Marita Chacón,  
Paco Martín, Carlos Belver

Fotografía  
Carlos Candel,  
Chema Sacristán

Dibujos

F. L. Frontán, J. Igual,  
M. López Moreno, A. Luis González Romero,  
Vital Garia, Horacio

Edita  
HOBBY PRESS, S.A.

Presidente  
María Andriño

Consejero Delegado  
José I. Gómez-Centurión

Jefe de Publicidad  
Mar Lumbreiras

Publicidad Barcelona  
José Galán Cortés  
Tels.: 303 10 22 - 313 71 76

Secretaría de Dirección  
Pilar Aristizábal

Suscripciones  
M.ª Rosa González  
M.ª del Mar Calzada

Redacción, Administración  
y Publicidad  
Ctra. de Irún  
km 12,400 (Fuencarral)  
Tel.: 634 70 12  
Telex: 49480 HOPR

Dto. Circulación  
Carlos Peropadre

Distribución  
Coedis, S. A. Valencia, 245  
Barcelona

Imprime  
Rotedic, S. A. Ctra. de Irún,  
km 12,450 (MADRID)

Fotocomposición  
Novocomp, S. A.  
Nicolás Morales, 38-40

Fotomecánica  
Antón  
C/ Azcóna, 33

Depósito Legal:  
M-36.598-1984

Representante para Argentina,  
Chile, Uruguay y Paraguay. Cia.  
Americana de ediciones, S.R.L.  
Sud América 1.532. Tel.: 21 24 64,  
1209 BUENOS AIRES (Argentina)

MICROHOBBY no se hace  
necesariamente solidaria de las  
opiniones vertidas por sus  
colaboradores en los artículos  
firmados. Reservados todos los  
derechos.

Solicitado control:  
OJD

# MICROHOBBY

## ESPECIAL

ESPECIAL MICROHOBBY • AÑO II • N.º 4 OCTUBRE 1986

4

**ENTREVISTA.** El auge de la Escuela  
Universitaria de Informática visto  
por José Gabriel Zato, su jefe de Es-  
tudios.

12

**HACKER.** Micromirón, para listar,  
analizar y alterar programas Basic.

18

**PROGRAMA.** La Fuga.

24

**CÓDIGO MAQUINA.** Sprites para el  
Spectrum.

32

**GRAFICOS.** Edigraf, un programa  
para crear gráficos.

40

**PROGRAMACION.** Los UDG.

44

**LENGUAJES.** Doce nuevos comandos  
para añadir al Basic mediante un  
programa en Código Máquina: am-  
pliación del Basic.

50

**ESPECIAL.** Guía de comandos.

60

**UTILIDADES.** Procesadores de tex-  
tos.

66

**INFORME.** Los Tecnodelincuentes, la  
nueva generación delictiva.

72

**PROGRAMACION.** Estructura de da-  
tos.

78

**Joystick para todos**  
**los gustos.**



SUMARIO

Alejandro JULVEZ Y Marcos ORTIZ

DE TODOS ES SABIDO EL AUGE QUE LA INFORMATICA ESTA TENIENDO EN NUESTRO PAIS, Y POR ENDE LA APARICION DE DIVERSOS CENTROS DE ENSEÑANZA EN LOS CUALES SE IMPARTEN TEMAS RELACIONADOS CON ESTA CIENCIA. UNO DE ELLOS ES LA ESCUELA UNIVERSITARIA DE INFORMATICA PERTENECIENTE A LA UNIVERSIDAD POLITECNICA DE MADRID, Y ES SU JEFE DE ESTUDIOS, JOSE GABRIEL ZATO, QUIEN NOS HABLA DEL TEMA.

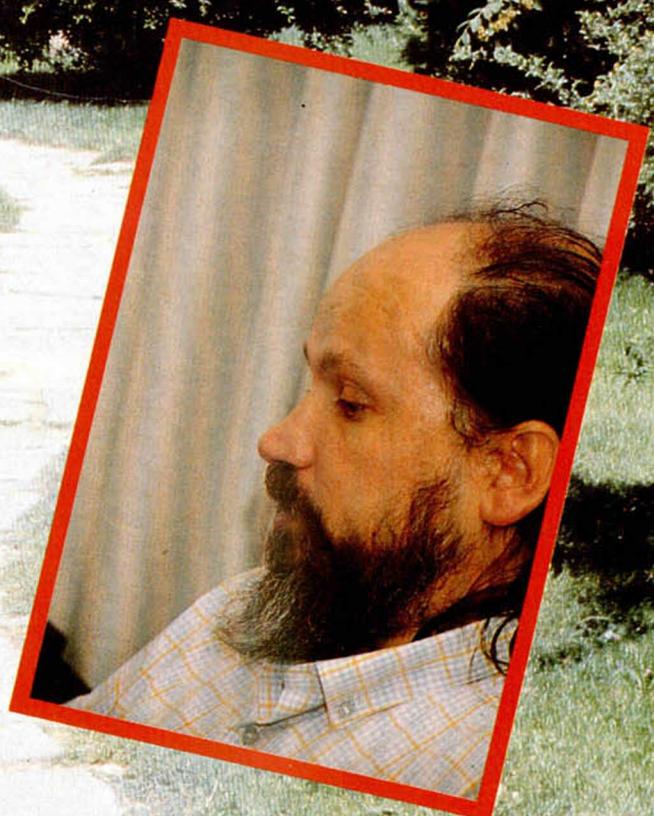
# Escuela de INFORMATICA

«Como su nombre indica, la E.U.I. es una carrera técnica, de las llamadas de ciclo corto. Hay que hacer notar que diplomas en Informática hay muchos, como son los que dan las academias, instituciones públicas o privadas y otros, pero aquí se da el título de diplomado de informática por la Escuela Universitaria de Informática de la Universidad Politécnica de Madrid, que ya marca unas diferencias, y además, no se da en esta escuela sólo programación, sino que también se imparte algo de análisis, por lo que yo la llamaría escuela de ingeniería técnica de Software o ingeniería técnica de sistemas, que sería más adecuado viendo las asignaturas que se imparte-

ten. Creo que se conforma una carrera muy adecuada a las necesidades de la sociedad a un cierto nivel, un nivel que yo llamaría universitario.»

Hay que decir que en la E.U.I. se imparten tres lenguajes: Pascal, Fortram y Cobol. También se da una amplia formación matemática, física y electrónica, y asignaturas tales como Traductores e Intérpretes, Estructura de Ordenadores, Bases de Datos y Sistemas Operativos, entre otras.

Viendo el programa de estudios de la escuela observamos que la especialización, o elección de especialidad, no se produce hasta el último curso (3.º) y preguntamos al señor Zato sobre el tema:



*«Estoy completamente de acuerdo en que la especialización llega demasiado tarde, además de que tendría que haber muchas más ramas entre las cuales poder escoger. Actualmente sólo hay dos especialidades: Sistemas Lógicos y Sistemas Físicos.*

*Este año hemos tenido una primera experiencia con la introducción de algunos elementos de Inteligencia Artificial (lenguajes LISP y PROLOG) y algo de Robótica, pero todo ello a nivel superficial, no como especialidad.*

*Yo creo que la elección de especialidad debería comenzar en segundo. Se podría diseñar una carrera corta en la que habría un primer curso básico en el que se deberían impartir conocimientos tales como Física, Electrónica, Matemáticas, Lógica de los Sistemas Digitales, etc... Luego, un segundo curso en el que se podrían ya ramificar las especialidades.*

*Podrían ser, especialidad de Gestión, desarrollar más la especialidad de Sistemas Físicos que tenemos ahora, que está poco desarrollada debido a que hay pocas empresas que se dedican a Mantenimiento; Inteligencia Artificial sería otra especialidad, se podría dar más de Algoritmos, Sistemas Operativos, Planificación y explotación de sistemas informáticos y otras asignaturas que se imparten ahora; en fin, cabría dar un mayor empuje a muchas de las ramas que están mínimamente diseñadas.*

*Pero hay un problema de tipo económico, ya que habría que dotar a la escuela de laboratorios, etc..., y además de una enseñanza teórica, habría que dar una amplia enseñanza experimental.*

**“En esta escuela formamos ingenieros técnicos en una ingeniería que no está reconocida en España, pero sí en EE.UU. ☺”**





—Sr. Zato, ¿qué ofrece esta escuela que no ofrecen las diversas academias que imparten cursos de informática?

—*Primeramente, la mayor parte de las academias se limitan a la enseñanza de uno o dos lenguajes, generalmente BASIC y COBOL. Esto puede servir como una primera información en el arte de programación, pero la informática no es sólo programar. Aquí, en esta escuela, se pretende formar ingenieros técnicos en una ingeniería que no está reconocida en España, pero sí en EE.UU. que es lo que se llama, como he dicho antes, ingeniería de Software. Nuestros diplomados salen capacitados para resolver problemas concretos que tengan los usuarios, cosa que no ocurre con los que sólo saben programar que sólo resolverán problemas que ya estén más o menos resueltos previamente, pero no pueden modificar paquetes de software, ni tampoco pueden poner en práctica aplicaciones, para lo cual se necesitan más conocimientos de informática como los que se dan en esta escuela.*

—Ya que estamos con el tema de los programadores, ¿qué diferencia hay entre un programador y un analista de sistemas?

—*En principio, un programador es una persona que sabe programar, es decir, programa un número n de programas, pero el analista de sistemas es una persona que es capaz de diseñar, de resolver problemas prácticos, problemas que no están previstos, que, en su caso, podría realizar un intérprete o que podría manipular un determinado tipo de paquetes de software para aplicarlos a una determinada situación concreta.*

ta. O sea, el analista es lo que hoy en día es un ingeniero y el programador es una persona que conoce uno o varios lenguajes, pero no más.

—¿Cree usted que la formación informática que reciben los alumnos de otras carreras que no son de Informática, tales como Matemáticas, Física, etc., puede en algún modo, suponer una competencia con respecto a la formación que imparte la escuela?

—*A mí me parece que hay un trabajo específico que es el trabajo del informático y que luego en las distintas carreras se deben ir introduciendo asignaturas de informática, pero no sólo en las carreras científicas y técnicas, sino también en las carreras de humanidades, porque además de ser la informática una disciplina nueva, es también una manera de organizar la información. A mí me parece mal que se deje la informática sólo para los informáticos porque merma las posibilidades de análisis de una persona que se dedique a otra ciencia. No tiene que ser un especialista en informática, pero sí le puede decir a un especialista qué clase de aplicación necesita y para eso requiere unos conocimientos mínimos.*

—*De manera que yo creo que están, primero, los especialistas en algo, como los Sociólogos, que tienen un problema concreto y que saben hasta qué punto la informática puede resolver ese problema, y luego el especialista propiamente dicho que es el que resuelve el problema, que tiene que ser un informático, y ese informático debe ser capaz de hacer un diseño más o menos completo.*

—¿Qué diferencia existe entre un diplomado en in-

formática (el que ha estudiado en la Escuela Universitaria de Informática) y un licenciado en informática (el que ha estudiado en la Facultad de Informática)?

—*Yo creo que los licenciados tienen unos conocimientos más profundos en algunos temas que en la Escuela se tocan de un modo superficial, pero hay algunos conocimientos que no son estrictamente necesarios a un nivel profundo para crear un buen ingeniero de Software.*

—*Como ejemplo, se puede poner a las Matemáticas, ya que para ser un ingeniero no hay que tener los conocimientos de un especialista.*

—*El punto de equilibrio sería tener la formación adecuada básica y luego una serie de conocimientos que formarían esa ingeniería de Software que está distribuida a lo largo de cinco años, ahora seis, en la Facultad, y a lo largo de tres en la Escuela. Lo que se aprenda de más sobre un tema, como por ejemplo Sistemas Operativos, en la Facultad será la diferencia esencial, creo, entre una licenciatura y un diploma.*

## UNA ESCUELA SIN PARO

—¿Qué grado de ocupación laboral tienen los diplomados de esta escuela?

—*Esta es una escuela que no tiene paro, es más, aquí los alumnos rechazan ofertas de trabajo que en otros sitios serían aceptadas, la mayoría de las veces porque no se dan garantías de continuidad en el trabajo después de pasar un cierto tiempo en el mismo.*

—*Estas ofertas las rechazan no sólo los diplomados sino*

*los alumnos, ya que hay pocos diplomados. Al ser la escuela muy joven (se creó en 1978) en este momento el número de proyectos fin de carrera leídos, es decir, el número total de diplomados, es inferior a cien, pero sin embargo, el número de personas que han acabado la carrera son varios cientos.*

*Aquí el mercado de trabajo está esperando a la gente que sale y muchas veces los cogen antes de que salgan, de hecho, casi la mitad de los estudiantes de tercero trabajan.*

—¿Qué opinión le merece la programación de juegos?

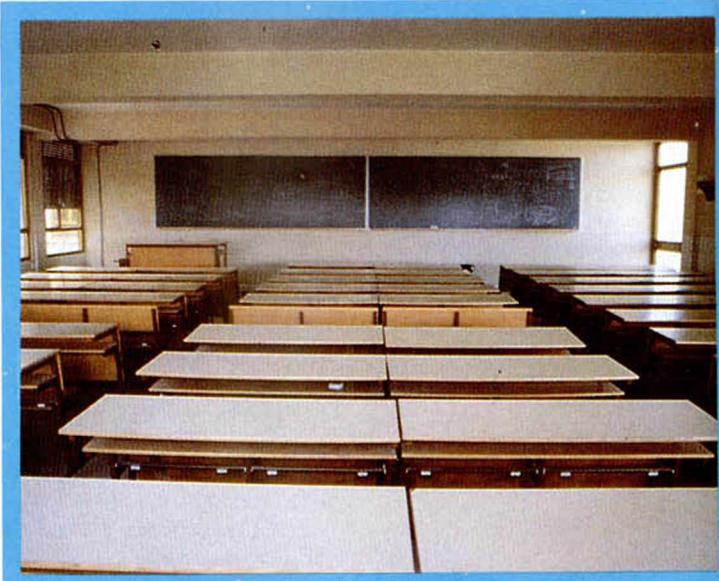
*—Yo creo que esta aplicación de la informática al juego va a desempeñar un papel cada vez más importante. El interés del microordenador reside entre otras cosas en sus posibilidades creativas. En este sentido, el micro ofrece la posibilidad de interacción con el usuario en contraposición con la televisión o el vídeo que sólo permiten una actividad pasiva y receptiva.*

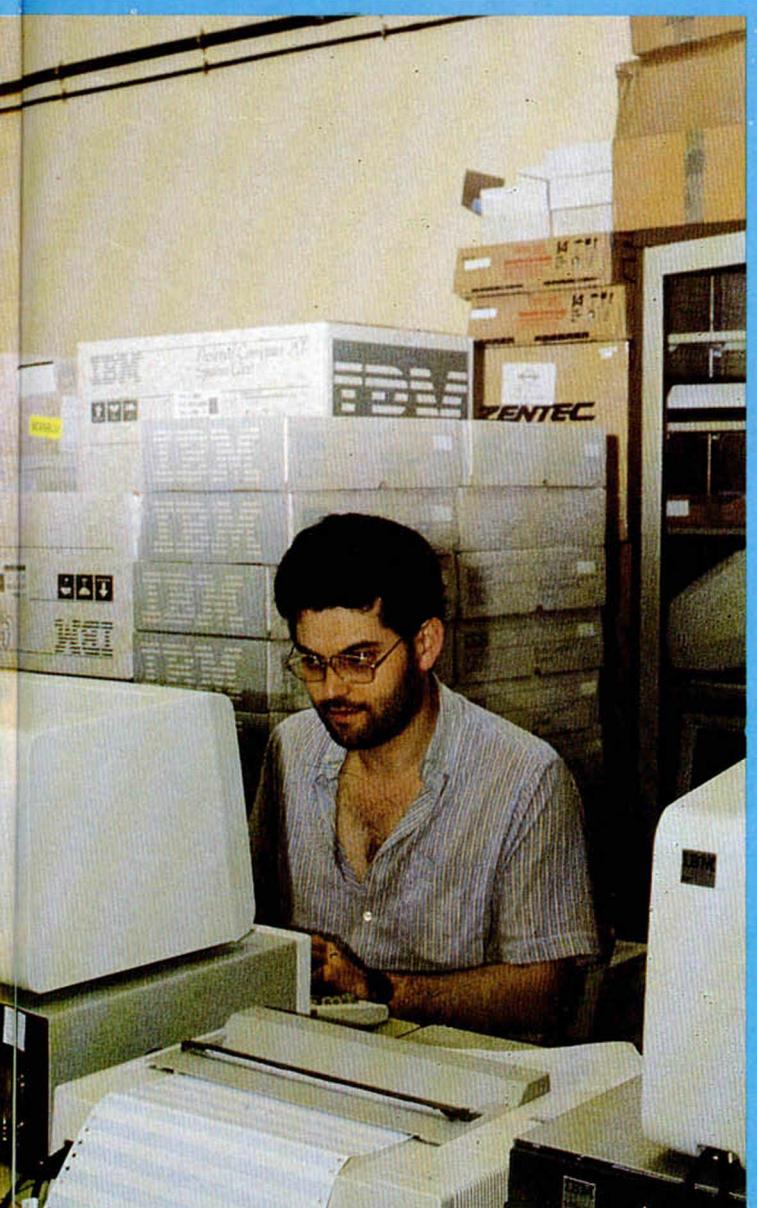
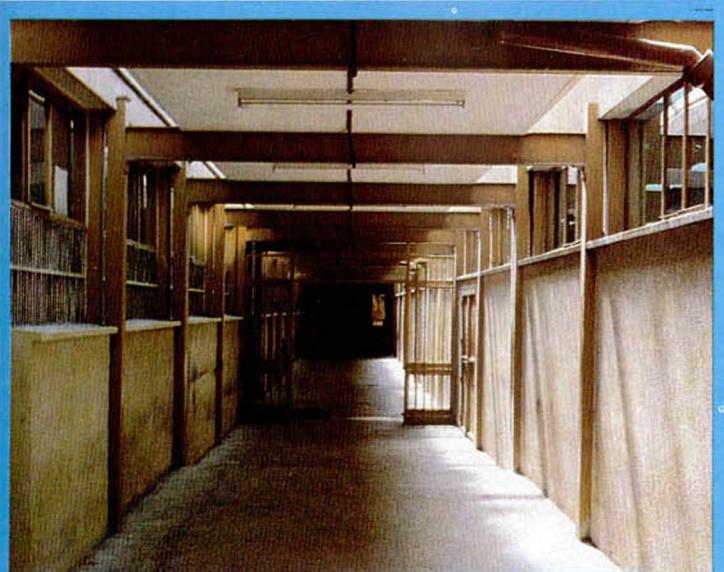
*Este es un campo con mucho futuro y enlaza, además, con el tema del entretenimiento y del aprender jugando.*

*Por otro lado habría que introducir el lenguaje máquina en la escuela, a nivel de EGB y BUP. Aquí impartimos una asignatura específica llamada O.E.O. (Organización Estructural de Ordenadores) que introduce en los lenguajes llamados de bajo nivel, y poder con ellos organizar las aplicaciones que se deseen.*

—¿Cuál es su opinión acerca de la introducción de la informática en el ambiente escolar?

*—En algunos colegios de EGB se ha introducido algo*





de BASIC, algunas técnicas de entretenimiento y poco más. No creo que exista un planteamiento serio de este tema, ni creo tampoco que se haya producido una introducción generalizada de la informática en las escuelas. Más bien creo que este fenómeno no pasa de ser una moda.

A mí, personalmente, no me cabe la menor duda de que los medios informáticos aplicados a la enseñanza van a ser de gran ayuda, tanto para los profesores como para los alumnos. Pero el tema tendrá que ser estudiado con detenimiento y su introducción deberá de ser de forma progresiva y planificada.

—¿A qué cree usted que se debe la diversidad de lenguajes y la enorme variedad de equipos que se encuentran en el mercado?

—Esta enorme variedad de equipos, y de una manera más general, la invasión de nuevas tecnologías a la que estamos asistiendo es fruto de la batalla feroz que algunas potencias industrializadas están librando por hacerse con el control del mercado, y ello, por una razón muy sencilla: el tema del valor añadido.

En efecto, las nuevas tecnologías brindan la posibilidad de fabricar, con muy pocos costes, productos muy caros, esto es, con un alto valor añadido.

Con este tipo de tecnologías, los costes reales de producción son mucho más bajos que los de los sistemas productivos tradicionales y se prevee que bajarán aún más.

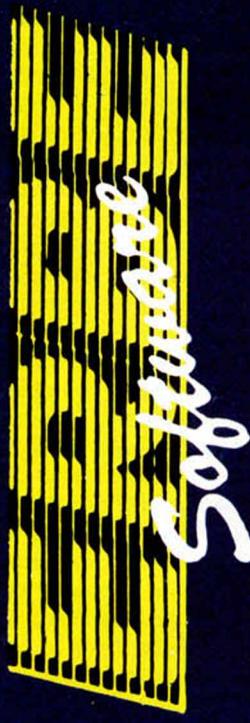
A la cabeza de la carrera está Japón, que goza de cierta ventaja sobre EE.UU., mientras Europa intenta abrirse camino, aunque está muy distanciada de los dos primeros.

Todo esto hace presagiar una división entre países ricos y países pobres muchísimo más profunda que la actual.

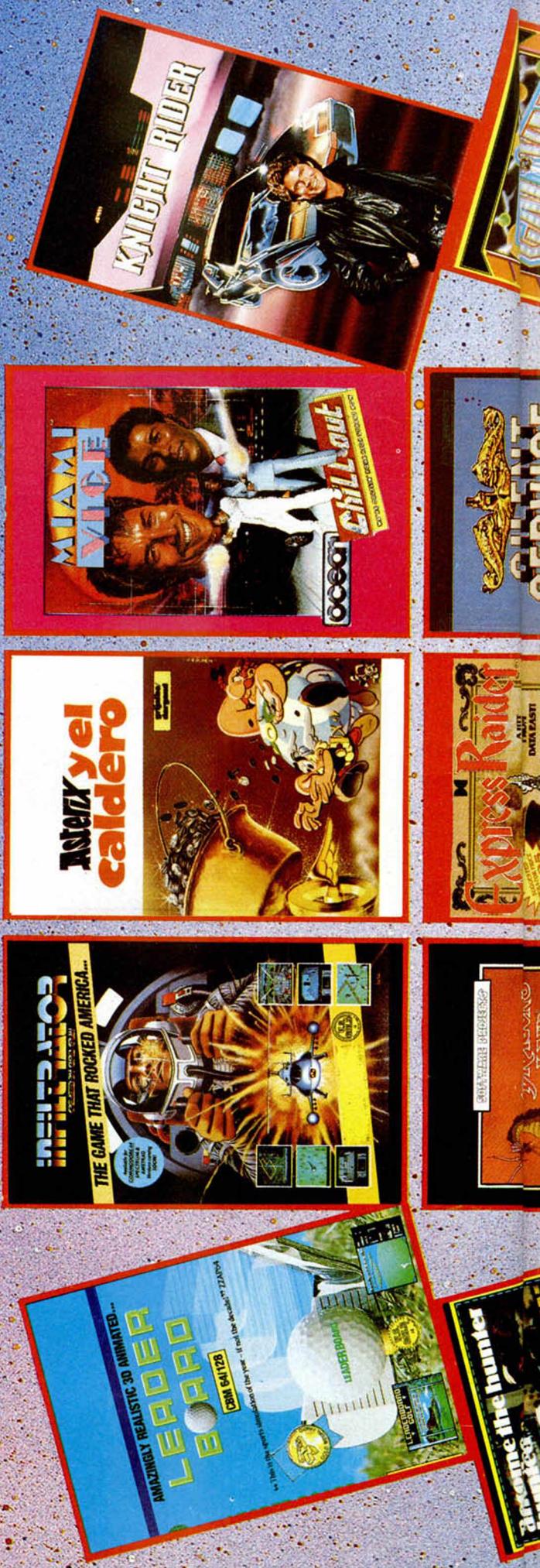
—¿En qué nivel de informática se encuentra España con respecto a Europa? ¿Cree usted que la entrada de nuestro país en la CEE tendrá una repercusión en este campo?

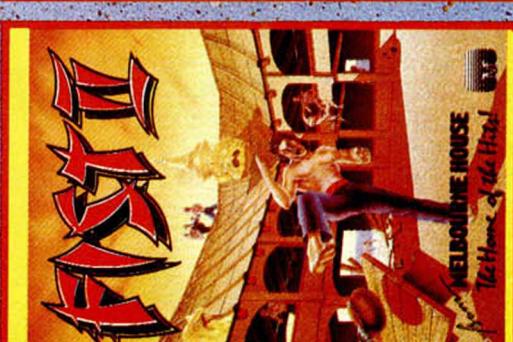
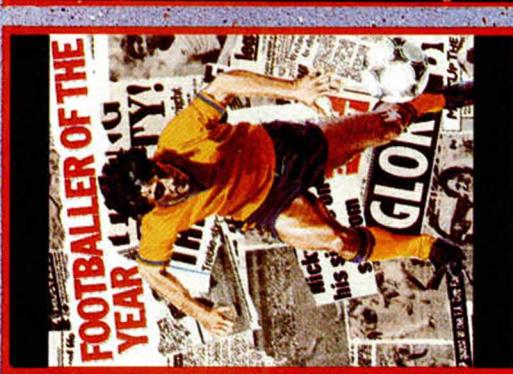
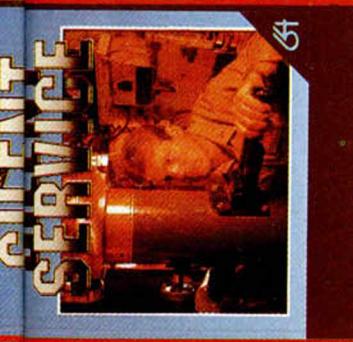
—Nos encontramos en un nivel muy inferior al resto de Europa. Mientras los países desarrollados dedican entre un 2 y un 3% de su PIB (Producto Interior Bruto) a la investigación, en la que están incluidas las nuevas tecnologías, España ha pasado en los últimos cuatro años, y según cifras del INI, del 0,47% al 0,58% del PIB. Y para colmo, dos leyes que podrían haber sido otros tantos instrumentos eficaces para corregir las enormes deficiencias de la investigación española, esto es, la Ley de la Ciencia y la Ley de Reforma Universitaria, no pasarán de ser declaraciones de buenas intenciones, ya que la segunda tiene unos presupuestos previstos insuficientes y la primera no incluye compromiso presupuestario alguno.

La falta de apoyo estatal a la investigación implica que en nuestro país ni se diseñan ni se comparten patentes sino que tal y como se venía haciendo desde los años del desarrollo industrial, España continúa importando tecnología extranjera.



# TELELOUNGE TOUCHDOWN TOUCHDOWN TOUCHDOWN





ERBE SOFTWARE  
C. STA. ENGRACIA 17  
DISTRIBUIDOR EXCLUSIVO  
PARA ESPAÑA  
28010 MADRID. TEL. (91) 447 34 10  
DELEGACION BARCELONA.  
AVDA. MISTRAL, N.º 10  
TEL. (93) 432 07 31

ERBE

ERBE

Marita CHACÓN

Tiene como objeto el poder listar, analizar y alterar aquellos programas en BASIC que se resisten a ser «escudriñados», bien por no poderse parar con «break», no ser posible el cargarlos con «MERGE» o tener ocultas las instrucciones como en el caso de estar escritas con tinta del mismo color del papel.

«MICROMIRON», que también permite listar las variables que acompañan al programa BASIC, puede ser salvado en cassette con o sin autoejecución después de ser modificado adecuadamente el programa en cuestión. Por otro lado, no puede manejar programas con cabecera falsa o sin ella, ni bloques de bytes o conjuntos de variables.

«MICROMIRON» es un programa totalmente realizado en código máquina de aproximadamente 3 Kb de longitud. Al cargarlo desde el cassette aparecen tres bloques:

BLOQUE 1.—Es un programa BASIC que se ocupa de cargar los otros dos bloques y de poner en marcha el programa en código máquina.

BLOQUE 2.—Es la pantalla de presentación, cuyo tercio superior se utiliza posteriormente como encabezamiento del menú del programa.

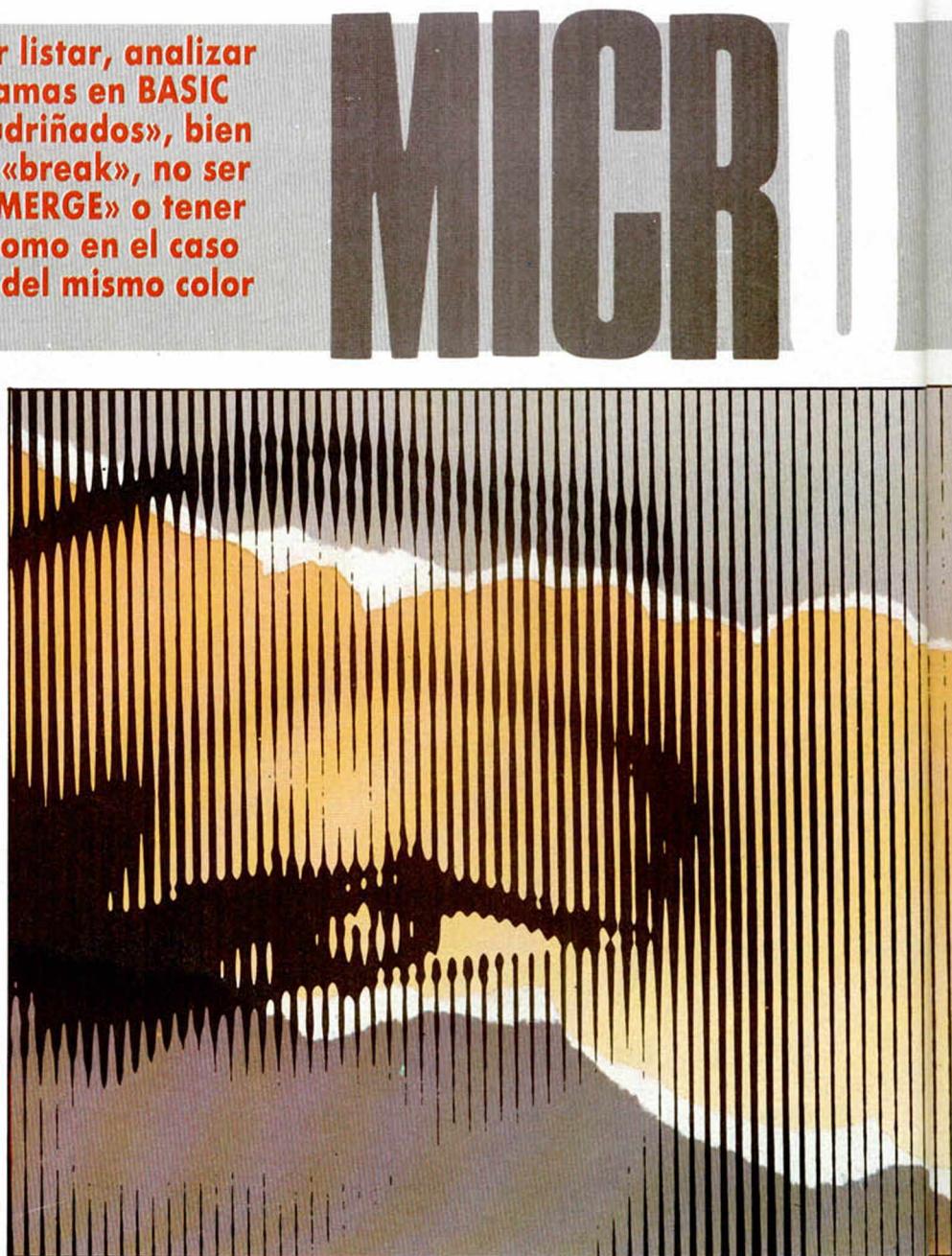
BLOQUE 3.—Está formado por el programa en código máquina, el cual se sitúa desde la dirección 37500, ocupando 2864 octetos.

## FUNCIONAMIENTO

El programa se pone en marcha por primera vez entrando en la dirección 37504 y en primer lugar guarda el tercio superior de la pantalla desde la dirección 35000, ocupando 2304 octetos, que se utiliza como encabezamiento del «MENU», al que se pasa posteriormente. Las sucesivas reentradas en «MICROMIRON» se deben hacer por la dirección 37500, con lo cual se pasa directamente al «MENU».

En el «MENU» se ofrecen las siguientes seis opciones:

- 1.—LOAD PROGRAMA
- 2.—LISTAR PROGRAMA
- 3.—SAVE PROGRAMA



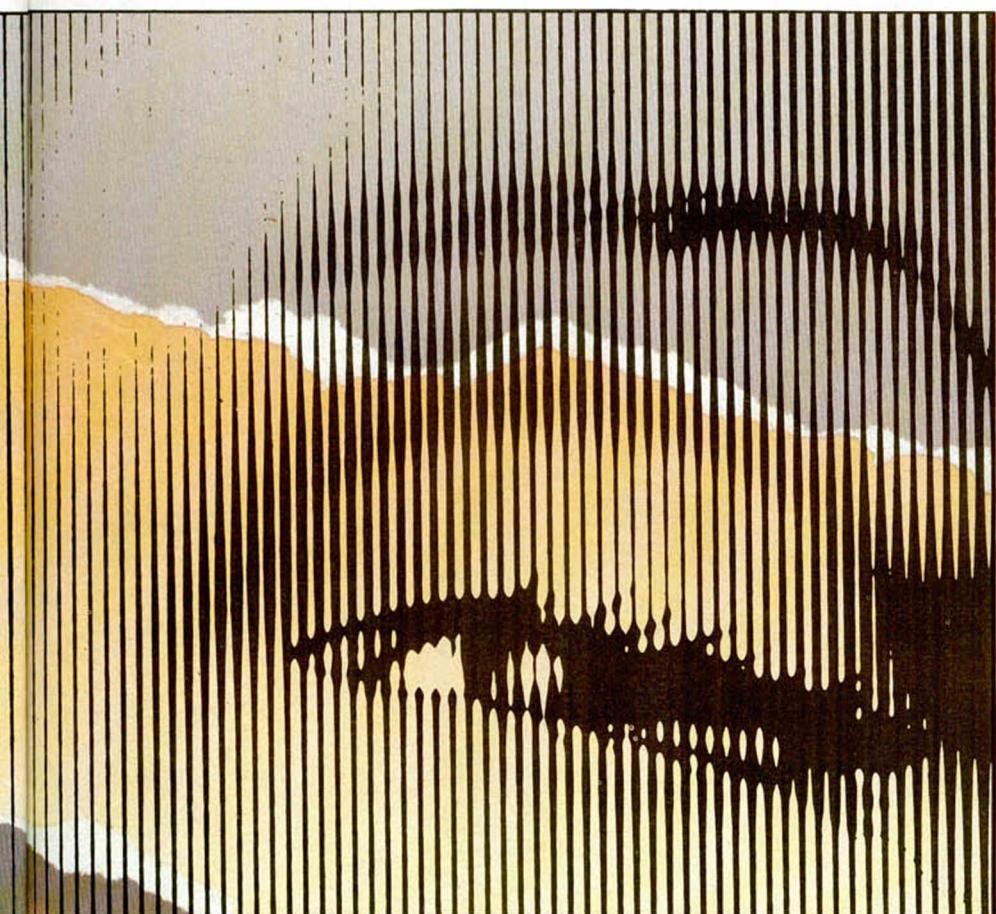
- 4.—LISTAR VARIABLES
- 5.—EDITAR MEMORIA
- 6.—RETORNO AL BASIC

1.—LOAD PROGRAMA. Sale el mensaje «PONER CASSETTE EN MARCHA» y se queda en espera de que aparezca una cabecera para su lectura. Con posterioridad a la carga de una cabecera, si es de un programa, aparece en la pantalla el nombre del mismo, su longitud, la longitud total del programa más variables y, si está

grabado con autoejecución, el número de línea de comienzo, pasándose posteriormente a la carga del programa comenzando en la dirección contenida en 40358 y 40359 que normalmente es 40366, pero puede cambiarse alterando el contenido de las direcciones anteriormente mencionadas.

Terminada correctamente la carga de un programa, se almacena un 1 en la dirección 40349, lo cual sirve como indicador, sin cuyo requisito no se puede tener acceso a las opciones de salvar, listar programa o listar variables.

# MICROMIRON



La cabecera del programa que inicialmente se guardó en un «buffer» transitorio, queda almacenada definitivamente en la zona del «print buffer» para su posterior utilización.

Por último, aparece el mensaje «PULSAR TECLA», lo cual permite el retorno al «MENÚ».

Al cargar un programa se borra el que se hubiese cargado anteriormente.

Si la cabecera leída no es de un programa BASIC, aparece en pantalla el nombre y tipo de información de que

se trate y se queda a la espera de otra nueva cabecera.

2.—LISTAR PROGRAMA. Si hay almacenado un programa BASIC, (dirección 40349 a 1), se solicita el número de línea desde el que se inicia el listado, a lo que se puede responder con cualquier número de cinco cifras o menos. Si sólo se pulsa «ENTER» se lista desde la primera línea.

La rutina de listado ignora los «octetos de color» y no considera terminada su tarea al encontrarse un carácter de código 13 seguido de un 128,

si no se ha agotado la longitud dada por la cabecera del programa cargado del cassette últimamente. Tampoco importa que los números de líneas estén desordenados o que tengan numeraciones superiores a 9999. Realmente podría listarse cualquier cosa sin ningún problema, aunque no se parezca en nada a un programa BASIC.

Para comenzar el listado se parte de la dirección contenida en 40358 y 40359 (normalmente 40366).

3.—SAVE PROGRAMA. Si anteriormente se ha cargado correctamente un programa BASIC, se pregunta si la grabación se hace con autoejecución, a lo que se puede contestar S ó N.

Posteriormente aparece el mensaje «PONER CASSETTE Y PULSAR TE-

«MICROMIRON» es un programa realizado totalmente en Código Máquina de 3KB de longitud.

CLA», tras lo cual se inicia la grabación. Para que ésta se efectúe con autoejecución es necesario que el programa ya la tenga al cargarlo.

4.—LISTAR VARIABLES. Además de cumplirse las condiciones de las opciones anteriores, lógicamente deben existir variables acompañando al programa BASIC que se cargó.

En el listado que se obtiene en pantalla, de las variables numéricas simples sólo aparece el nombre, de las matrices numéricas se dan también las dimensiones y de las cadenas y matri-

ces de caracteres se listan además los códigos de los caracteres que contienen en hexadecimal. Las variables aparecen relacionadas en el mismo orden en que están en la memoria.

5.—EDITAR MEMORIA. Para entrar en esta opción no debe de cumplirse ninguna condición especial.

En primer lugar se solicita la dirección de partida en decimal, a lo que puede contestarse con cualquier número de cinco cifras o menos. Si la dirección dada es mayor de 65535, se considera la diferencia con esa cantidad menos uno, y si no se da ninguna cifra y sólo se pulsa «ENTER» se toma cero como dirección de partida.

Posteriormente aparece el listado en cinco columnas, que de izquierda a derecha dan la dirección en decimal y en hexadecimal, el contenido de la posición en decimal y hexadecimal y por último, el carácter correspondiente para códigos mayores de 32.

En la pantalla hay 19 líneas para otras tantas posiciones de memoria y el cursor colocado en la primera de ellas.

?.—Aparece la pantalla de información y pulsando cualquier tecla se retorna al listado.

C.—Cuando el cursor está en la primera línea, continúa el listado desde la dirección siguiente a la última que aparece en pantalla. En caso contrario el listado continúa desde la dirección actual del cursor.

A.—Lista las 19 posiciones anteriores a la del cursor.

Tanto en esta opción como en la anterior, el cursor pasa a la posición correspondiente a la primera línea.

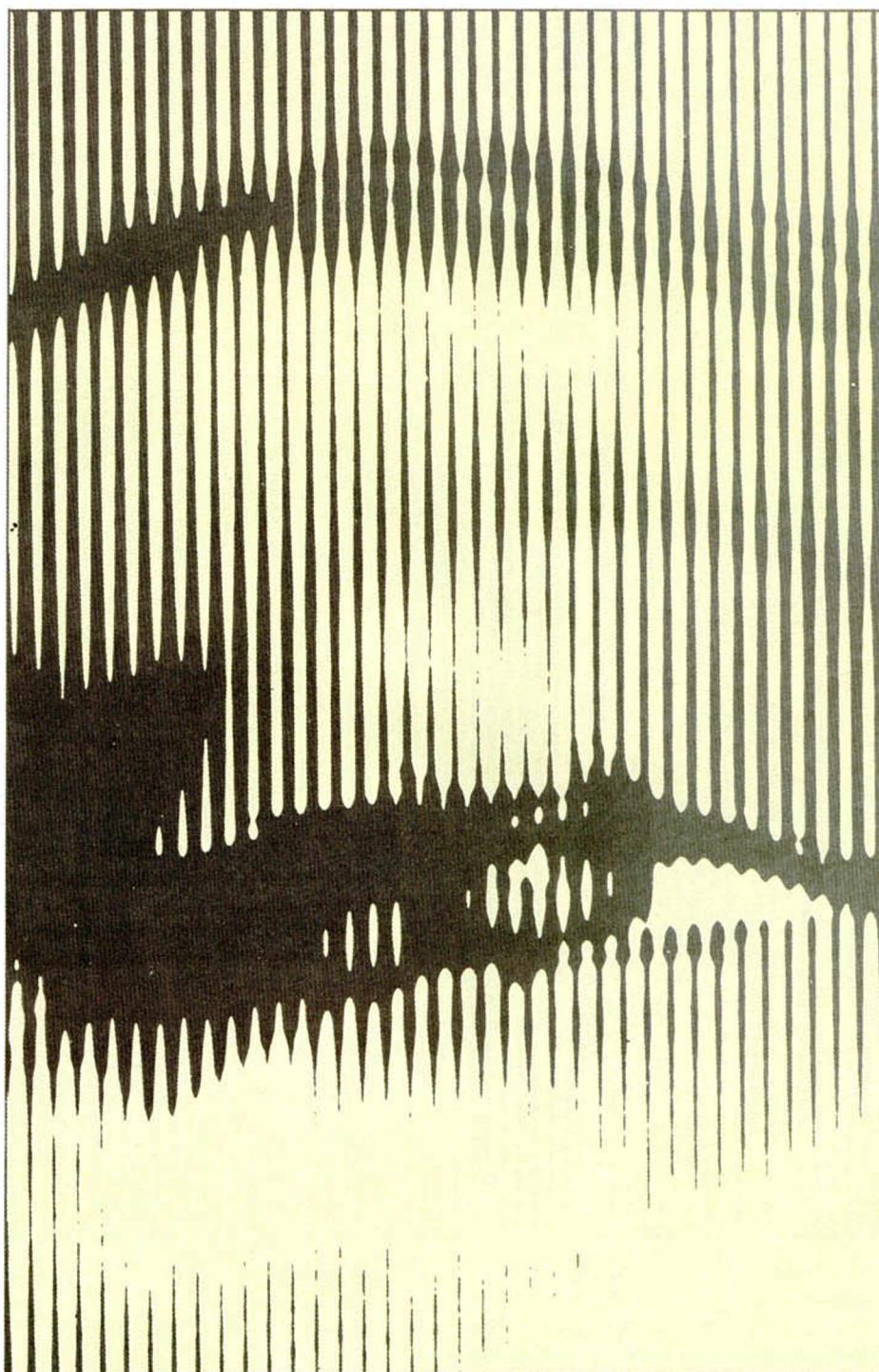
R.—Retorno al «MENU».

D.—Para cambiar la dirección del listado según se ha descrito anteriormente.

MOVER CURSOR.—Pulsando las teclas correspondientes se puede subir o bajar el cursor a cualquiera de las posiciones de memoria que aparezcan en la pantalla.

E.—Permite alterar el contenido de la posición actual del cursor, para lo cual se pregunta si el nuevo valor del octeto en cuestión se dará en decimal o en hexadecimal, a lo que se debe contestar D ó H, pero de pulsar otra tecla se considera anulada esta opción.

Posteriormente se solicita el valor del



octeto, que si es en decimal puede no tener ninguna cifra, pulsando solamente «ENTER», en cuyo caso se considera que el valor es cero, y hasta un máximo de cinco cifras, pero si el valor dado es superior a 255, se toma el del octeto menos significativo de los que fuesen necesarios para contener la cantidad dada.

Si el valor se da en hexadecimal, es necesario marcar dos cifras, no admitiéndose más ni menos.

Por último, se lista desde la posición actual del cursor.

6.—RETORNO AL BASIC. Se abandona «MICROMIRON». Para retornar se debe entrar por la dirección 37500.

**Este mes en tu kiosco  
un hobby de locura**

AÑO 1 - N.º 5

# hobby

PARA HACER Y CONOCER

250 Ptas.

LA LOCURA  
POR LOS VIEJOS AUTOS

Calculadoras  
programables,  
el futuro ordenador  
de bolsillo

Adiestra  
a tu perro  
en casa

GUÍA de VINOS

Monta tu propia bodega

CÓMO ENCUADERNAR UN LIBRO

HOBBY PRESS

Lamantina, Cuenca y Melilla, 240 pags.



CONCURSO

GANA UN  
EQUIPO DE MUSICA  
¡ES MUY FACIL!

## MICROMIRON

```

3 REM << MICROMIRON >>
4
5 BORDER 7: PAPER 7: INK 7: C
LEAR 34999
8 LOAD ""CODE 37500,2864
9 INK 0
10 RANDOMIZE USR 37504: REM PR
IMERA ENTRADA EN EL PROGRAMA
11 STOP
12 RANDOMIZE USR 37500: REM RE
ENTRADA EN EL PROGRAMA
1 18160000ED5BA49D2100 728
2 40010008EDB021005801 608
3 0001EDB02100003922A0 698
4 9DCD6B0D3E02CD01163E 836
5 08326A5C21B09CCDEB99 1214
6 2AA49D110040010008ED 690
7 B0110058010001EDB0AF 871
8 CD011621569DCDEB99FD 1350
9 36CE00FD7ECCEB728FAF5 1563
10 AFCD6E0DF1FE3138E2FE 1583
11 3730DEF5CD6B0DF1FE31 1439
12 2008CD5799CDD39618A7 1242
13 FE322008CDA198CDD396 1428
14 189BFE332008CD109CCD 1106
15 D396188FFE3620052AA0 1075
16 9DF9C9FE342009CDF996 1558
17 CDD396C39B92FE352006 1407
18 CD3393C39B92FD36CE00 1412
19 C39B92CD6E0DAFC00116 1227
20 216C95CDEB99CDB89BCD 1632
21 629BE5E5CD6B694E10613 1496
22 C5E5CDB9ACDCF94E1E5 2018
23 7CDC439BE1E57DCD439B 1557
24 E1E56E2600E5CDBB9ACD 1614
25 CF94E17DF5CD439BCDCF 1789
26 94F1FF92009D73E08D7 1433
27 3E0DD71808FE203801D7 880
28 3E0DD7E123C110BC0100 948
29 00CDD694E1CDEA94FE41 1698
30 2008111300B7ED52189E 760
31 FE43200AAFB820961113 940
32 00191890FE5222001C9FE 1017
33 44CA3343FF3F2022E5CD 1285
34 6B0D3E02CD0116218395 725
35 CDEB992AA69DCDB9A21 1569
36 8096CDEB99E1CDEA94C3 1878
37 4693FE0B200E78B728AB 1042
38 CDD694052BCDD69418A1 1367
39 FE0A200F78FE122898CD 1100
40 D6940423CDD694186FE 1388
41 45208AE5CD6E0DAFC01 1177
42 16219496CDEB99CDEA94 1533
43 F5CD6E0DF1E1FE442013 1412
44 E521AD96CDEB99CDB89B 1722
45 CD629B7DE177C34693FE 1593
46 48C29993E521BF96CDEB 1609
47 9906021D196CDEA94FE 1392
48 0D2844FE0C201578B728 783
49 F1052BE5C53E08D73E20 1094
50 D73E08D7C1E118E0F578 1531
51 FE022003F118D7F1FE30 1314
52 38D2FE4730CEFE41380A 1230
53 F5D7F1D63777042318C0 1344
54 FE3A30BCF5D7F1D63018 1535
55 F078FE0220B021D1967E 1342
56 CB27CB27CB27CB273B6 1185
57 E177E5CD6E0DE1C34693 1538
58 CD6B0DAFC011621F594 1154
59 CDEB993E02CD01162110 934
60 95CDEB99C93E20D73E20 1346
61 D7C9E5C53E16D7C1C578 1651
62 3C3CD7215395CDEB99C1 1386

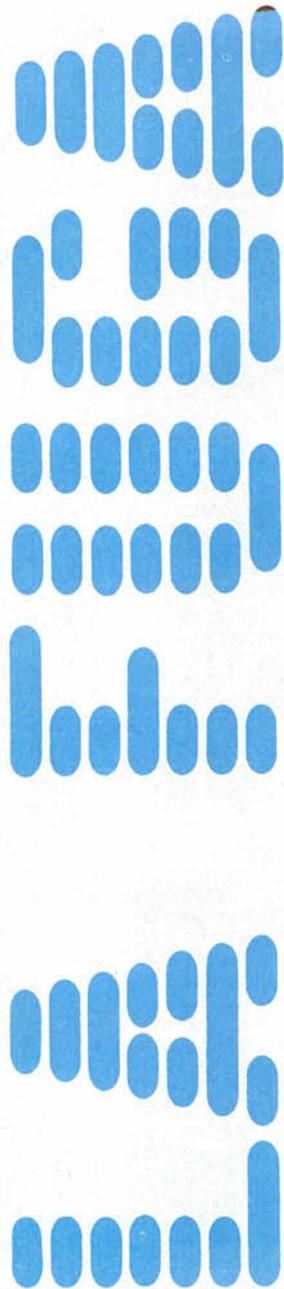
```

```

63 E1C9AF32085C3A085CB7 1092
64 28FAC95041524120494E 966
65 464F524D4143494F4E20 702
66 50554C534152203F0DFF 834
67 44495220442020445220 569
68 48202044454320204858 564
69 20202043415241435445 595
70 52200D3D3D3D3D2020 496
71 3D3D3D3D20203D3D3D3D 523
72 203D3D20203D3D3D3D 523
73 3D3D3D3D3D0DFF001501 595
74 20202020203C3E202020 378
75 203C3E2020203C3E1500 393
76 0DFF444952454343494F 846
77 4E20454E20444543494D 643
78 414C3F20FF1600041301 537
79 140120494E5354525543 605
80 43494F4E455320454449 691
81 544F5220140013000D0D 342
82 0D432E2D20434F4E5449 584
83 4E55415220204C49535441 723
84 E444F0D0D412E2D2052 521
85 4554524F434544455220 701
86 454E204C49535441444F 707
87 0D0D522E2D205245544F 545
88 524E4F20414C204D454E 668
89 550D0D442E2D2043414D 511
90 42494152204449524543 677
91 43494F4E204C49535441 710
92 444F0D0D452E2D204544 502
93 4954415220454C204F43 659
94 5445544F2044454C2043 660
95 5552534F520D0D202020 533
96 205345205054544454E 665
97 205554494C495A415220 692
98 5445434C415320444520 645
99 20202053554249522059 606
100 2042414A415220435552 650
101 534F520D0D2020202043 465
102 4F4D49454E5A4F205052 739
103 4F4752414D4120FF2028 798
104 454E2020202020444543 511
105 494D414C292E0DFF4445 783
106 43494D414C204F204845 642
107 5841443F2028442F4829 584
108 200DFF4F435445544F20 794
109 454E20444543494D3F20 628
110 FF4F435445544F20454E 896
111 2048455841443F20FF00 744
112 00AFCD0116CD6E0D21EB 999
113 96CDEB99FD36CE00FD7E 1635
114 CEB728FAC950554C5341 1269
115 52205445434C410DFF2A 785
116 A69DED5BD15B197EFE80 1484
117 C84F3A9D9DB7C879E5E6 1614
118 E0FE602012214198CD17 1102
119 983E0D73E0DD7E11106 980
120 0018DBFEE02012215498 1040
121 CD17983E0DD73E0DD7E1 1185
122 11130018C5FEA0201821 760
123 4198CD1798E1237ECB7F 1313
124 2005E5D7E116F5E5CBBF 1598
125 D718C2FE802014216798 1155
126 CD1798CDD5973E0DD7E1 1464
127 235E2356231891FE4020 804
128 33217A98CD17983E24D7 1051
129 CDFF97E1234E234623E5 1318
130 C5E5C57ECD439B3E20D7 1485
131 E320D7C1E1230B78B120 1102
132 EC3E0DD73E0DD7D1E1C3 1445
133 0097FEC02029218D98CD 1201
134 17983E24D7CDD597E100 1282
135 234E234623C5E55E1600 795
136 191923E5606987ED52ED 1254
137 52444D0BE118B4E1C93E 1155

```

138	28D73E29D73E0DD7212A	938	213	5A204341524143544552	703
139	98CDEB99D1E1E5D52323	1691	214	455320FF4F435445544F	901
140	2346235E2356E5C5EBCD	1221	215	5320FFDD212C9B0605DD	1055
141	DB9AC1E12310F23E0DD7	1374	216	5E01DD5602DD3600007C	803
142	C93E28D73E29D73E0DD7	1126	217	BA300ADD23DD23DD2310	1028
143	213798CDEB99D1E1E5D5	1709	218	EA18102808B7ED52DD34	1097
144	23060118D6C5E53E02CD	975	219	0018E87DBB38E818F23E	1184
145	0116E1CDEB99C179E61F	1416	220	02CD0116212C9B06050E	487
146	C640D7C944494D454E53	1126	221	207EB7200F05280B0481	577
147	494F4E45533AF4C4F4E	928	222	E5D7E123232310EFC904	1234
148	47495455443AF4F564152	927	223	0E3018F100102700E803	617
149	4941424C45204E554D45	690	224	00640000A0000001002A	153
150	5249434120FF4255434C	868	225	D65BF9C90000000F5E6F0	1470
151	4520464F522D4E455854	696	226	0604CB3F10FCCD569BF1	1231
152	20202020FF4D41545249	764	227	E60FC569BC9FE0A3004	1208
153	5A204E554D4552494341	718	228	C6301802C637D7C93E05	1008
154	202020FF434144454E41	763	229	32A89BDD21A99B210000	984
155	20434152414354455245	682	230	01AE9B0A035F0A57DD7E	882
156	5320FF4D415452495A20	873	231	00FE3A380E2025DD233A	765
157	4341524143544552453	733	232	A89B3D32A89B20ECC9FE	1480
158	20FF0210003922D65B	716	233	303815D6302804193D20	549
159	2AA69D7EFE80C83A9D9D	1445	234	FCDD23033AA89B3DC832	1203
160	B7C8E5ED5BD15B1922EA	1533	235	A89B18CD210000C9003A	844
161	5BAFCD0116CD6E0D2173	970	236	3A3A3A3A01000A006400	343
162	9DCDEB99CDB89ECD6E0D	1622	237	E8031027AFC0D11621A9	895
163	CD629B444DE17EB8380A	1204	238	9E0605C53E3A772310FC	905
164	2006237EB938042B1815	532	239	2BC1FD36CE00FD7ECEB7	1517
165	23235E23561923E5ED5B	902	240	28FAFD36CE00FE3A30F2	1405
166	EA5BB7ED52E1CA3B9B18	1492	241	FE0DC8FE0C20183E05B8	1040
167	DD7E23E56E67CDD89AE1	1627	242	28E60423363AC5E53E08	917
168	234E2346230B7EFE2030	724	243	D73E20D73E08D7E1C118	1251
169	41FE0D201878B120F13E	1020	244	D3FE3038CF32A89B6AFB8	1508
170	0DCD539923E5ED5B	1371	245	3AA89B28C5772B05C5E5	1211
171	B7ED52E1CA3B9B18D0FE	1629	246	18E62AA69D7EFE80C83A	1385
172	0E20093E050B233D20FB	512	247	9D9DB7C8AFC0D116217A	1255
173	18D0FE1620043E0218F1	873	248	9CCDEB99FD36CE00FD7E	1641
174	FE1728F8E1038C0FE16	1359	249	CEB728FAFE4E2804FE6E	1419
175	30BC230B188BCD539918	955	250	2008FD369500FD369680	1081
176	B3E5D7E1C9210003922	1173	251	AFCD0116CD6E0D21919C	1065
177	D65BAFCD011621D199CD	1308	252	CDEB99FD36CE00FD7ECE	1691
178	EB993E02CD0116D2184	1066	253	B728FAAFCD6E0D111100	1010
179	9D3E0011130037CD5605	606	254	DD21C25B9FCDC2040632	1173
180	3A849DB7C2129A21849D	1218	255	FB7610FD3EFFDD2AA69D	1541
181	11C25B011100EDB0216B	873	256	ED5BCD5BCDC204C94155	1378
182	9ACDEB99CDF999EB2175	1739	257	544F454454535543494F	746
183	9ACDBF99131ACB7F2803	1121	258	4E3F2028532F4E29200D	507
184	1318071B21889ACDBF99	949	259	FF504F4E455220434153	890
185	219B9ACDBF992AA69D3E	1318	260	53455454452059205055	707
186	8077AF329D9DC3519ACD	1421	261	4C534152205445434C41	699
187	EB99EB5E235623E8BD5CD	1526	262	0DFF16090C1401130120	384
188	DB9A3E0DD7D1C9504F4E	1310	263	4D454E5520140013000D	393
189	45522043415353455454	718	264	0D20202020312E2D204C	389
190	4520454E204D41524348	643	265	4F41442050524F475241	703
191	410DFF57EFEFF280E6	1456	266	4D410D0D202020322E	392
192	D7E12318F5D1C921849D	1476	267	D204C49535441522050	652
193	23060A7EE5E2030023E	804	268	524F4752414D410D0D20	579
194	3FD7E12310F3E53E0DD7	1316	269	202020332E2D20534156	504
195	E1C9FE01200B21AE9ACD	1290	270	452050524F4752414D41	702
196	EB99CDF999181BFE0220	1334	271	0D0D20202020342E2D20	329
197	0C21BF9ACDEB99CDF999	1590	272	4C495354415220564152	728
198	C33C9AFC0321D29ACDEB	1503	273	4941424C45530D0D2020	522
199	99CDF9990670DD21849D	1421	274	2020352E2D2045444954	534
200	3E0011000037CD56053E	492	275	4152204D454D524941	701
201	0DD7C36D99DD2AA69D3E	1333	276	0D0D20202020362E2D20	331
202	FFFDF5E93FD569437CD56	1582	277	5245544F524E4F20414C	726
203	05DD3600803E01329D9D	835	278	2042415349430DFF2020	718
204	C950524F4752414D4120	834	279	2020202020201201204D	320
205	FF544F54414C204F4354	905	280	4152434152204F504349	692
206	45544F5320202020FF	730	281	4F4E2012000DFF4C4953	707
207	434F4D49454E5A4F2045	713	282	5420515545204C494E45	679
208	4E204C494E454120FF4F	837	283	413F20FF434142454345	818
209	435445544F532050524F	739	284	5241205052494D455241	707
210	4752414D412020FF4D41	821	285	20202020202020000000	224
211	5452495A204E554D4552	752	286	0000000800005888AE9D	659
212	49434120FF4D41545249	873	287	000000000000000000000000	0



*Sobrevivir en aquel tiempo era una tarea casi imposible, el futuro no se presentaba nada esperanzador, la única posibilidad era marcharse a otro país.*

El objeto del juego es el ayudar a recoger a nuestro amigo todas las piezas que pueden hacer posible la hu-

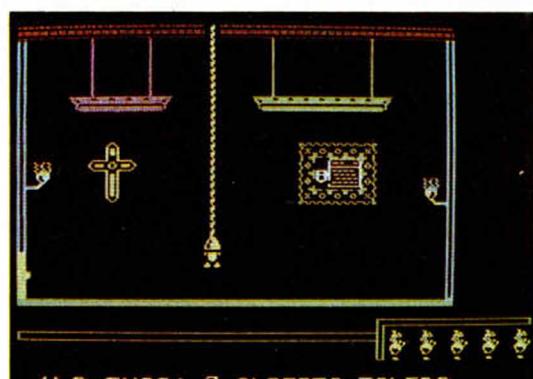
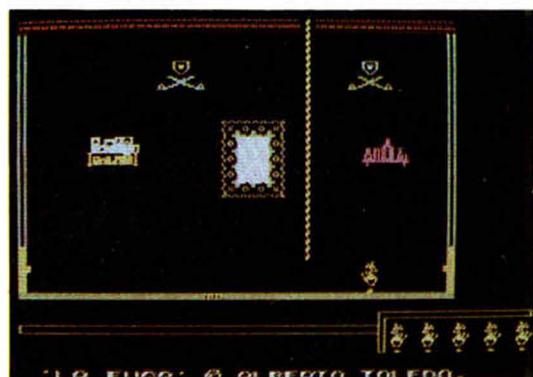
da, de las que algunas nos servirán y otras nos facilitarán el poder conseguir las necesarias.

La mansión por la que tendrá lugar nuestra aventura tiene 72 habitaciones, repartidas en 6 plantas con 12 habitaciones cada una, en las que hallaremos lanzarrayos y espadas que surgen del suelo. Para pasar de una a otra, encontraremos puertas, cuerdas y trampillas.

Esta es la relación de piezas a encontrar, objetivo de nuestra misión:

**El cheque, el bolígrafo, el pasaporte, el mechero, la bombona, el barreño, el casco, el dinero, la tinta, el enchufe, la caja fuerte, la maleta, la llave, el pasaje y el dulce.**

Si tenemos el bolígrafo vacío y cogemos la tinta, conseguiremos llenarlo. Con el bolígrafo cargado, podre-





'LA FUGA' © ALBERTO TOLEDO.



'LA FUGA' © ALBERTO TOLEDO.



'LA FUGA' © ALBERTO TOLEDO.



'LA FUGA' © ALBERTO TOLEDO.



'LA FUGA' © ALBERTO TOLEDO.



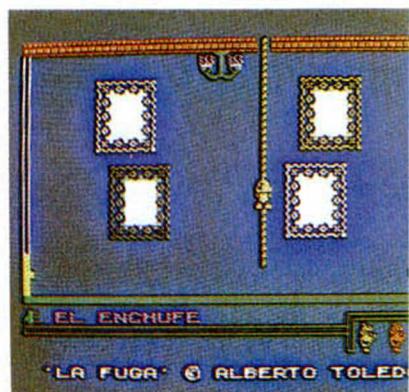
'LA FUGA' © ALBERTO TOLEDO.



'LA FUGA' © ALBERTO TOLEDO.



'LA FUGA' © ALBERTO TOLEDO.



'LA FUGA' © ALBERTO TOLEDO.

mos firmar el cheque. Con el cheque firmado y el pasaporte conseguiremos el pasaje. Con el mechero sin gas y la bombona, llenaremos el mechero. Con éste lleno, podremos encender el barreño, y ya con el barreño encendido y el casco, conseguiremos derribar el muro que nos impide coger la maleta. Con el pasaje y la maleta obtendremos la llave, con la que, junto con el dinero, finalizaremos la misión.

El enchufe y la caja fuerte no tienen ninguna utilidad.

El programa lleva algunas rutinas en código máquina muy cortitas y simples, como una rutina que genera un juego de caracteres simplemente rotando alternativamente los bits y comparándolos con los originales.

En cuanto a las teclas de control son:

- a - subir/bajar
- z - on/off interrup.

o - izquierda  
p - derecha

Si se hace BREAK posiblemente saldrán unos signos muy raros. Haciendo GO SUB 9998 aparecerán de nuevo los caracteres normales.

**Nota:** Todos los espacios, salvo los de las líneas 101 y 102, deben teclearse pulsando en modo gráfico «8». En los textos y DATAs no hace falta.

## 20 PROGRAMA

```

1210 PRINT INK 6; AT 17,x-1,0$(1+
(x/2=INT (x/2))), AT 16,x-1, "0"
1215 IF Z>0 THEN IF tr=1 THEN PR
INT AT 17,x-1; INK c(z); v$(z): L
ET s(z)=x-1: LET r(z)=hb: LET tr
=0
1217 RETURN
1300 RETURN
1310 RANDOMIZE USR 64245: LET sf
=1: RETURN
1510 IF POINT (8*x+4,.30)=0 THEN
LET hb=hb+12: GO SUB 2500: GO SU
B 5900: LET w$=SCREEN$ (17,x): F
OR h=1 TO 15: PRINT INK 6; AT h-1
,x,"b"; AT h,x,"c"; AT h+1,x,"d";
BEEP .03,-h: NEXT h: PRINT AT 15
,x,"b"; AT 16,x,"O"; AT 17,x,"P";
RETURN
1520 IF POINT (8*x+3,.48)=1 THEN
PRINT AT 14,x,"c"; AT 15,x,"d"; AT
17,x,"": FOR h=15 TO 1 STEP -1
: PRINT AT h+1,x,"b"; AT h,x,"d";
AT h-1,x,"c": BEEP .03,-h: NEXT h:
LET hb=hb-12: GO SUB 2500: GO
SUB 5900: LET w$=SCREEN$ (17,x):
PRINT AT 16,x,"O"; AT 17,x,"P";
RETURN
1530 RETURN
2002 IF CODE w$=115 THEN FOR a=-
20 TO 20 STEP 5: BEEP .005,a: NE
XT a: FOR a=20 TO -20 STEP -1: B
EEP .005,a: NEXT a: LET vi=vi+1:
PRINT AT 20,21+vi*2,"0"; AT 21,2
1+vi*2,"P": LET r(15)=0: RETURN
2005 LET z=di: LET di=m: LET m=c
ODE w$=100
2010 PRINT AT 19,0; INK c(m); v$(m);
: GO SUB 9998: PRINT "INK c(m); v$(
m)": GO SUB 9998: PRINT "INK c(m)-1;s$(m); x$": GO SUB 9999: I
F di=0 THEN PRINT AT 21,0; INK c
(di); v$(di); GO SUB 9998: PRINT "
",INK c(di)-1;s$(di); x$: GO
SUB 9999
2011 IF m=1 OR di=1 THEN GO SUB
9998: PRINT AT 19+2*(di=1),12; I
NK (RND*5)+3; u$(c4): GO SUB 9999
2012 IF m=2 OR di=2 THEN GO SUB
9998: PRINT AT 19+2*(di=2),15; I
NK (RND*5)+3; u$(c1) ( TO 5): GO S
UB 9999
2013 IF m=4 OR di=4 THEN GO SUB
9998: PRINT AT 19+2*(di=4),13; I
NK (RND*5)+3; u$(c2): GO SUB 9999
2014 IF m=6 OR di=6 THEN GO SUB
9998: PRINT AT 19+2*(di=6),13; I
NK (RND*5)+3; u$(c3): GO SUB 9999
2015 IF (m=2 OR di=2) AND (m=9
R di=9) AND (c1=1) THEN GO SUB 999
8: GO SUB 3000: LET c1=2: PRINT
AT 19+2*(di=2),15; u$(c1) ( TO 5):
GO SUB 9999
2016 IF (m=1 OR di=1) AND (m=2 O
R di=2) AND (c4=7 AND c1=2 THEN G
O SUB 9998: LET c4=8: GO SUB 300
0: PRINT AT 19+2*(di=1),12; u$(c4
): GO SUB 9999
2019 IF (m=4 OR di=4) AND (m=5 O
R di=5) AND (c2=3 THEN GO SUB 999
8: LET c2=4: GO SUB 3000: PRINT
AT 19+2*(di=4),13; INK (RND*4)+3
;u$(c2): GO SUB 9999
2020 IF (m=1 OR di=1) AND (m=3 O
R di=3) AND (c1=2 AND b2=0 THEN G
O SUB 9998: GO SUB 3000: GO SUB
9999: PRINT AT 17,s(14)+2*(ATTR
(17,s(14))=70); INK c(14); v$(14)
: LET b2=1
2021 IF (m=13 OR di=13) AND (m=1
4 OR di=14) AND b3=0 THEN GO SUB
9998: GO SUB 3000: GO SUB 9999:
PRINT AT 17,s(12)+2*(ATTR (17,s
(12))=70); INK c(12); v$(12): LET
b3=1
2022 IF (m=12 OR di=12) AND (m=8
OR di=8) THEN BEEP 1,50: GO SUB
5400: GO SUB 4900
2023 IF (m=4 OR di=4) AND (m=6 O
R di=6) AND (c2=4 AND c3=5 THEN G
O SUB 9998: LET c3=6: GO SUB 300
0: PRINT AT 19+2*(di=6),13; INK
(RND*4)+3; u$(c3): GO SUB 9999
2028 LET s(m)=0: LET r(m)=0: LET
w$="": LET tr=1: BEEP 1,30: RE
TURN
2500 PRINT AT 1,0,q$: LET e1=5:
RETURN
3000 FOR a=1 TO 4: FOR b=1 TO 7:
PRINT AT 19,2; OVER 1; INK b; "

```

```

+!"HASTA CONSEGUIDO FUGARTE!" +1$(6
1 TO )
5020 CLS : GO SUB 9995: PRINT AT
1,0; INK 0; L$: GO SUB 9996: RAN
DOMIZE USR 64539: RESTORE 5031: FOR a=1 TO 48
READ bp: BEEP .15, bp
5031 DATA 0,19,4,16,7,12,0,19,4,
16,7,12,-3,16,0,12,4,9,-3,16,0,1
2,4,9,-7,12,-3,9,0,5,-7,12,-3,9,
0,5,-5,7,-1,11,2,14,-5,7,-1,11,2
,14
5040 IF INKEY$="s" THEN GO TO 50
70
5050 IF INKEY$="n" THEN GO TO 51
00
5060 NEXT a: GO TO 5030
5070 GO SUB 9997: RANDOMIZE USR
64539: GO TO 4
5100 CLS : PRINT AT 6,9; FLASH 1
;"AUTODESTRUCCION";#0;"PULSA UNA
TECLA PARA DETENERLA": FOR a=49
9 TO 0 STEP -1: PRINT AT 10,14,a
;"": IF A/10=INT (A/10) THEN BE
EP,.005,20
5105 IF INKEY$<>"" THEN GO TO 50
00
5110 NEXT a: FOR a=1 TO 10: RAND
OMIZE USR 64575: NEXT a: STOP
5100 FOR a=1 TO 100: NEXT a: PAP
ER 5: INK 0: CLS : LET l=18: LET
fl=-1: LET fi=1: GO SUB 5600: G
O SUB 5750: GO SUB 5420
5405 FOR c=1 TO 3: CLS : LET l=9
: LET fl=0: LET fi=1: GO SUB 565
0: GO SUB 5420: NEXT c
5410 CLS : LET fl=1: LET fi=-1:
GO SUB 5600: GO SUB 5420
5415 CLS : GO SUB 5650: GO SUB 5
700: FOR b=29 TO 11 STEP -1: FOR
9=1 TO 3: BEEP .005,0: NEXT 9:
PRINT INK 1; AT 20,b;"xy": NEXT
b: GO SUB 5430: RETURN
5420 FOR a=29 TO 1 STEP -3: FOR
b=0 TO 2
5425 PRINT AT l,a-b; INK 1;"xy"
;AT l+fi,a-b;"": FOR 9=1 TO 3
: BEEP .005,0: NEXT 9: NEXT b:
LET l=l+fl: NEXT a: RETURN
5430 POKE 65304,0: POKE 65305,0:
POKE 65306,184: PRINT INK 1; AT
20,11;"xy": FOR c=10 TO 4 STEP -
2: FOR b=1 TO 2: FOR a=1 TO 10:
NEXT a: PRINT AT 19,c-b;"R": AT
20,c-b,o$(2+b): BEEP .005,0: NEX
T b: NEXT c: PRINT AT 19,2;"": A
T 20,2;"": RETURN
5500 PRINT INK 2; AT 21,0;m$+m$(T
O 5): FOR a=1 TO 23 STEP 11: PR
INT INK 7; AT 4,1+a;"": AT
5,1+a;"": AT 6,1+a;"": AT
"": NEXT a: RETURN
5500 PRINT INK 2; AT 21,0;m$+m$(T
O 5): FOR a=1 TO 23 STEP 11: PR
INT INK 7; AT 4,1+a;"": AT
5,1+a;"": AT 6,1+a;"": AT
"": NEXT a: RETURN
5700 PRINT AT 19,1; INK 1;"z": AT
20,0;"": RETURN
5750 PRINT AT 20,30; INK 1;"xy":
FOR c=0 TO 28 STEP 2: FOR b=1 T
0 2: FOR a=1 TO 10: NEXT a: PRIN
T AT 19,c+b;"0": AT 20,c+b,o$(b
): BEEP .005,0: NEXT b: NEXT c: P
OKE 65304,6: POKE 65305,5: POKE
65306,186: PRINT AT 20,30;"": RE
TURN
5910 PRINT AT 0,0; PAPER 2: INK
0,$: INK 4; AT 18,0; PAPER 0; -
5920 INK 5: PLOT 0,167: DRAW 0,-
135: DRAW 0,0: DRAW 0,135: PLOT
212,167: DRAW 0,-135: DRAW 3,0:
DRAW 0,0,135: INK 6: -
5940 LET q=p(hb): LET q1=p(hb+1)
: FOR r=1 TO VAL P$(q1-1)
5945 IF CODE P$(q)<57 AND CODE P
$(q)<55 THEN LET sy=VAL P$(q+1
TO q+2): LET sx=VAL P$(q+3 TO q+
4): LET t=VAL P$(q+5)
5950 GO SUB 6000+VAL P$(q)*100:
NEXT r
5960 IF hb=r(12) AND b3=1 THEN P
RINT AT 17,s(12); INK c(12); v$(1
2)
5961 GO SUB 5962: GO TO 5964: IF
hb=10 THEN PRINT AT 15,0;h$(1);
AT 16,0;h$(2): AT 17,0;h$(3)
5962 FOR j=1 TO 11: IF hb=r(j) T

```

```

HEN PRINT AT 17,s(j)+(POINT (8*s
(j)+3,48)=1)+(POINT (8*s(j)+4,30
)=0); INK c(j); v$(j)
5963 NEXT j: RETURN
5964 IF hb=r(15) AND vi<5 THEN P
RINT AT 17,s(15); INK c(15); v$(1
5)
5965 IF hb=r(13) AND b1=1 THEN P
RINT AT 17,s(13); INK c(13); v$(1
3)
5966 IF j(hb)=1 THEN INK 5: PRIN
T INK 3+RND*4; AT 14,25;"v": PLOT
211,63: DRAW -3,0: DRAW 0,-2: D
RAU 3,0: INK 6
5967 IF hb=r(14) AND b2=1 THEN P
RINT AT 17,s(14); INK c(14); v$(1
4)
5968 IF hb=42 AND b1=0 THEN FOR
a=1 TO 3: PRINT AT 14+a,5; INK 5
;"t": AT 14+a,9;"": NEXT a: PRIN
T INK c(13); AT 17,s(13); "q"
5969 IF hb=9 THEN PRINT AT 15,26
;i$(1); AT 16,26;i$(2); AT 17,26;i
$(3)
5970 RETURN
6005 INK 5: FOR v=2 TO 3*VAL P$(q
+1): STEP 3: LET t$=p$(q+v TO q+
v+1)
6030 IF VAL t$>70 THEN PRINT INK
VAL P$(q+v+2); AT VAL t$-70,25;""
: INK (VAL P$(q+v+2)-1); AT VAL
t$-71,25;"a": PLOT 208,(21-(VAL
t$-70))$: DRAW 4,0: DRAW 0,1:
DRAW -4,0: GO TO 6070
6040 IF VAL t$>41 THEN PRINT INK
VAL P$(q+v+2); AT 1,VAL t$-40;n$(1
): AT 2,VAL t$-41;n$(2 TO ): IN
K (VAL P$(q+v+2))-1; AT 1,VAL t$-
41;"a": AT 1,VAL t$-39;"a": GO TO
6070
6050 IF VAL t$<21 THEN PRINT INK
VAL P$(q+v+2); AT VAL t$;1;"E":
INK (VAL P$(q+v+2)-1); AT VAL t$-
1,1;"a": PLOT 4,(21-VAL t$)*8: D
RAU 3,0: DRAW 0,1: DRAW -3,0:
6070 NEXT v: INK 6: LET q=q+2+3*V
6120 PRINT AT sy,sx; INK t;a$(T
0 4); AT sy+4,sx;a$(9 TO ): FOR
l=1 TO 3: PRINT AT sy+l,sx; INK
t;a$(5): INK 5: PAPER 2:a$(6 TO 7
): PAPER 0: INK t;a$(8): NEXT l:
LET q=q+6: RETURN
6220 PRINT AT sy,sx; INK t;b$(T
0 5): AT sy+1,sx;b$(6): INK VAL P
$(q+6),b$(7 TO 9): INK t;b$(10):
AT sy+2,sx,b$(11): INK VAL P$(q+
5),b$(12 TO 14): INK t;b$(15): AT
sy+3,sx,b$(16 TO ): LET q=q+7:
RETURN
6320 PRINT AT sy,sx; INK t;g$(T
0 3): AT sy+1,sx;g$(4 TO ): LET q
=q+6: RETURN
6420 PRINT AT sy,sx; INK t;e$(T
0 3): AT sy+1,sx; INK t-1;e$(4 TO
): LET q=q+6: RETURN
6520 PRINT AT sy,sx+1; INK t;k$(1
): AT sy+1,sx; INK t-1;k$(2 TO 1
): LET q=q+6: RETURN
6620 PRINT AT sy,sx; INK t;f$(T
0 2): AT sy+1,sx;f$(3 TO 4): AT sy
+2,sx;f$(5 TO 6): LET q=q+6: RETU
RN
6720 PRINT AT sy,sx; INK t;j$(1
): FOR l=1 TO 10: PRINT INK t;j$(2
): NEXT l: PRINT INK t;j$(3)
6725 INK t: PLOT (sx+1)*8+2,167:
DRAW 0,-(sy*8)-9: PLOT (sx+lo
)*8+5,167: DRAW 0,-(sy*8)-9: I
NK 6: LET q=q+8: RETURN
6820 PRINT AT sy,sx+1; INK t;l$(1
): AT sy+1,sx;l$(2 TO 4): AT sy+2
,sx+1;l$(5): AT sy+3,sx+1;l$(6):
LET q=q+6: RETURN
6905 FOR v=2 TO 2*VAL P$(q+1) ST
EP 2: LET t$=P$(q+v TO q+v+1)
6915 IF VAL t$>31 THEN FOR e=0 T
O 15: PRINT AT e,VAL t$-31;"b":
NEXT e: GO TO 6960
6920 IF t$="00" THEN PRINT INK 6
: AT 15,0;h$(1): AT 16,0;h$(2): AT
17,0;h$(3): GO TO 6960
6930 IF t$="26" THEN PRINT INK 6
: AT 15,26;i$(1): AT 16,26;i$(2): A
T 17,26;i$(3): GO TO 6960
6940 IF VAL t$>31 THEN PRINT INK
6: AT 15,VAL t$-31;c$( TO 2): AT
16,VAL t$-31;c$(3 TO 4): AT 17,VA
L t$-31;c$(5 TO 6): GO TO 6960
6950 PRINT AT 18,VAL t$: INK 6;d
$:
6960 NEXT v: LET u=q+1: LET q=q+
2+2*VAL P$(u): RETURN
6998 RESTORE 8991: PRINT AT 10,9
: FLASH 1;"LEYENDO DATOS": FOR a
=64245 TO 64284: READ dt: POKE a
,dt: NEXT a
8991 DATA 237,86,201,62,250,237,
71,237,94,201
8992 DATA 1,251,243,245,197,213,
229,6,10,62,235,211,254,120,211
254,237,95,211,254,16,243,225,20
9,193,241,255,251,237,77
9000 RESTORE 9001: FOR g=64506 T
O 65343: READ da: POKE g,da: NEX
T 9
9002 DATA 33,0,61,17,24,246,1,0
3,126,203,47,182,18,19,35,126,18
,19,35,11,11,121,176,32,239,33,2
4,245,34,54,92,201
9004 DATA 14,10,33,0,64,17,0,72
,6,10,16,254,237,95,0,166,229,9
,8,107,165,18,225,35,19,62,72,188
40,2,24,232,13,32,223,201
9006 DATA 6,255,62,248,245,62,23
9,211,254,237,95,211,254,241,211
254,214,8,254,0,32,238,16,234,2
01
9010 DATA 51,205,2,42,17,169,130
226,226,130,145,41,42,146,129,2
25,226,130,169,17,42,2,203,51,24
3,129,153,36,24,0,51,204
9020 DATA 71,65,149,136,84,64,17
9,204,71,65,137,148,84,72,129,13
5,204,179,64,84,136,149,65,71,20
4,51,0,24,36,153,129,243
9030 DATA 0,1,1,1,1,1,1,125,71,127
,69,69,125,16,1,1,0,255,0,116,0
,255,0,109,0,54,0,27,0,109,0,255
,0,254,1,13,1,255,1,177,1,217,1,1
0,1,181,1,255,0
9040 DATA 0,60,66,66,66,66,60,0
,0,124,66,90,90,66,124,0,255,85,8
,5,255,0,0,0,0
9050 DATA 60,60,36,60,63,63,60,6
3,255,64,92,87,94,94,255,64,7,25
3,253,143,223,255,255,255,0
103,211,237,253,255,0,248,136,16
8,216,255,253,253,253
9050 DATA 255,114,218,250,250,25
0,255,2
9070 DATA 1,2,4,9,17,32,99,164,4
,0,47,40,36,35,32,63,17,17,17,17
,57,57,19,3,32,128,64,32,144,136,4
,198,101,148,148,20,36,196,4,252
,136,136,136,136,156,200,192
192
9080 DATA 0,0,0,0,0,0,51,50,50,1
22,74,122,255,255,32,32,0,12,12
12,30,63,179,191,179,179,179,191
,255,255,0,0,0,0,0,0,0,0,0,0,56,16,1
6,42,58,58,255,255,4,4
9090 DATA 240,240,240,240,240,240,24
0,240,240,240,240,252,252,252,25
2,240,240,15,15,15,15,15,15,15,15,1
5,15,15,63,63,63,63,15,15
9100 DATA 255,95,103,63,16,23,31
,7,255,199,255,255,0,255,255,255
,255,250,230,252,8,232,248,224
9110 DATA 255,129,189,153,153,66
,36,24,7,3,0,0,32,83,95,248,0,19
,7,255,60,255,197,0,0,224,192,0,0
,4,202,250,31,73,73,255,34,34,34
,255,68
9120 DATA 96,40,116,92,160,212,5
3,121,254,188,132,72,112,0,48,56
,121,158,141,77,60,219,198,108
9130 DATA 6,20,46,58,5,43,172,15
8,127,125,33,18,14,0,12,26,158,1
21,177,178,60,219,99,54
9150 DATA 24,36,66,90,90,90,90,6
,0,31,32,79,79,32,31,0,66,195,2
4,36,36,24,195,66,0,248,4,242,24
2,4,248,0,90,90,90,90,90,90,90,9
0,90,90,90,90,36,36,24,0
9170 DATA 0,16,36,64,17,32,68,8
,255,9,9,9,255,33,33,33
9180 DATA 255,126,24,24,24,24,24
,24,126,114,114,36,24,12,7,3,24
,24,24,24,24,60,255,195,126,78,78
,36,24,48,224,192,0,0,138,81,146
,73,149,126
9190 DATA 8,24,24,16,8,24,24,16
,24,60,78,94,159,159,255,0,189,18

```

# 22 PROGRAMA

```

9,60,60,24,66,231,231
9200 DATA 0,255,129,189,129,181,181
129,255,14,23,46,28,40,80,224,19
2,127,65,93,85,93,81,55,127,0,8,
8,28,62,115,121,62
9210 DATA 24,24,36,60,36,60,36,6
0,24,36,114,82,82,112,80,112,24
36,90,123,5,7,3,0,102,60,24,24,4
4,118,118,60
9230 DATA 62,8,62,65,93,73,73,12
7,56,63,56,120,248,248,191,55,25
5,129,141,161,213,165,129,255
9240 DATA 6,9,9,84,56,176,224,64
24,36,126,129,189,189,129,126,0
0,0,255,129,173,129,255,0,24,24
36,90,255,231,126,0,251,251,251
0,223,223,223
9250 DATA 224,232,144,168,134,7,
2,0,7,23,9,21,97,224,64,0
9260 DATA 8,20,20,20,20,20,20,20,12
7,0,0,184,191,255,184,191,15,0,0
129,243,253,127,254,4,8,28,34,7
3,85,73,65,255,63,65,255,129,181
181,192,255,1,13,109,97,13,13,1
3,255
9300 RESTORE 9301: LET P$$="": FOR
R=1 TO 14: READ y$: LET P$$=P$$+
y$: NEXT t
9310 DATA "025438229226171020342
100735607167305206564110570181380
5085109146705180549200266560020577
101509420805732031856920026552020
37401607106133311044610205920026
67080406401105803203510177920018
5
9320 DATA "102035011171021322100
87431120391266403023021225156090
85705160641091979200265302042302
21520904322091923505127920007560
30350181463021710312381007491136
40303310412550808431121203067776
62691265
9330 DATA "706030557061805410812
70213283693002620530203430220541
00354102042061167015339200207507
036403114717019057016229226485301
034608055605126210197301562492002
66103045311066207164702546732930
026185"
9340 DATA "303026609065506124402
20301807920011610304510320420911
73038161225369203495608043608204
709090778031250020367339126620404
72204173630908480814501107920025
64030345101274032130307279454693
0026385"
9350 DATA "104045310097706160540
17769200445803036803214509057310
19510712392260962080253410127703
10085610224940026075158020356100
57503153110144304226409225920051
73003542080456806127104183010639
226196"
9360 DATA "605043403137208105250
92040211673492002667060508531009
719520401045920049580803556040832
03147410819301037932642166711060
8340409650720428126079300263458
0603720210546081631052069200265
9370 DATA "405054504136307197209
1235011279200963020866080556041
44107200518079126620307462031437
2090552209166593026175105044105
12510519351116792004056030363041
03408125709170679226385"
9380 DATA "804042804127804204808
08680816592001767070610531004631
01134982240174792265065040475041
9450911620410250281410793026166
10303410320270709075809186609057
93002617630303431020541003740320
350512601536930026477"
9390 DATA "704030447041704580805
68081971081139200186604084504165
60804760812360821691266303044303
20520803352081974106126930026406
70607966509065803207409174018059
30026186604065604193309044309207
505126930026486"
9400 DATA "104047104185405126411
05441119592001067090513580308480
31460208781392001752040345204193
66090656091945051279200488705050
56705160577100320502484596912651
03055103196109064109187015349300
26476"
9410 DATA "804047305114804195209
10530210678793002648640404540419
64100474101944071130152792000975
03136507055507134507207511133932

```

```

32649630312430704530712730719631
11230445560410680592002678060568
06197203114320911629200265"
9420 DATA "105057105214504136509
13502036737940026064968050468052
07707080953100843101630153491007
20404542041273024195660906580916
40108692004175030865032054080471
07124308213940026114867050805580
80466091374092142031856017779200
167"
9430 DATA "103054103175209076720
91434011169126630706480613740719
6070450937447974755446092002655
0404658084410012500427509203015
4492026780405530512480418520911
73020858009302640670606192540905
4409177509116037557945179200546"
9440 DATA "105054105197604135310
13691265206054720617658061230250
65759200265507025706091164091376
091940103492003761030543051271903
17580907680915491266604045204117
36042044100865101659200266705030
43808046705140742081747021078169
20042630404550412730420360985610
811440920591477"
9500 DIM P(73): FOR o=1 TO 73: R
EAD dat: LET P(o)=dat: NEXT o
9510 DATA 1,41,79,114,151,183,21
8,259,298,332,372,411,454,486,52
3,559,595,632,671,709,747,779,81
6,854,897,934,974,1006,1045,1082
1114,1151,1185,1222,1253,1286,1
323,1351,1403,1444,1488
9520 DATA 1529,1564,1605,1645,16

```



'LA FUGA' © ALBERTO TOLEDO.



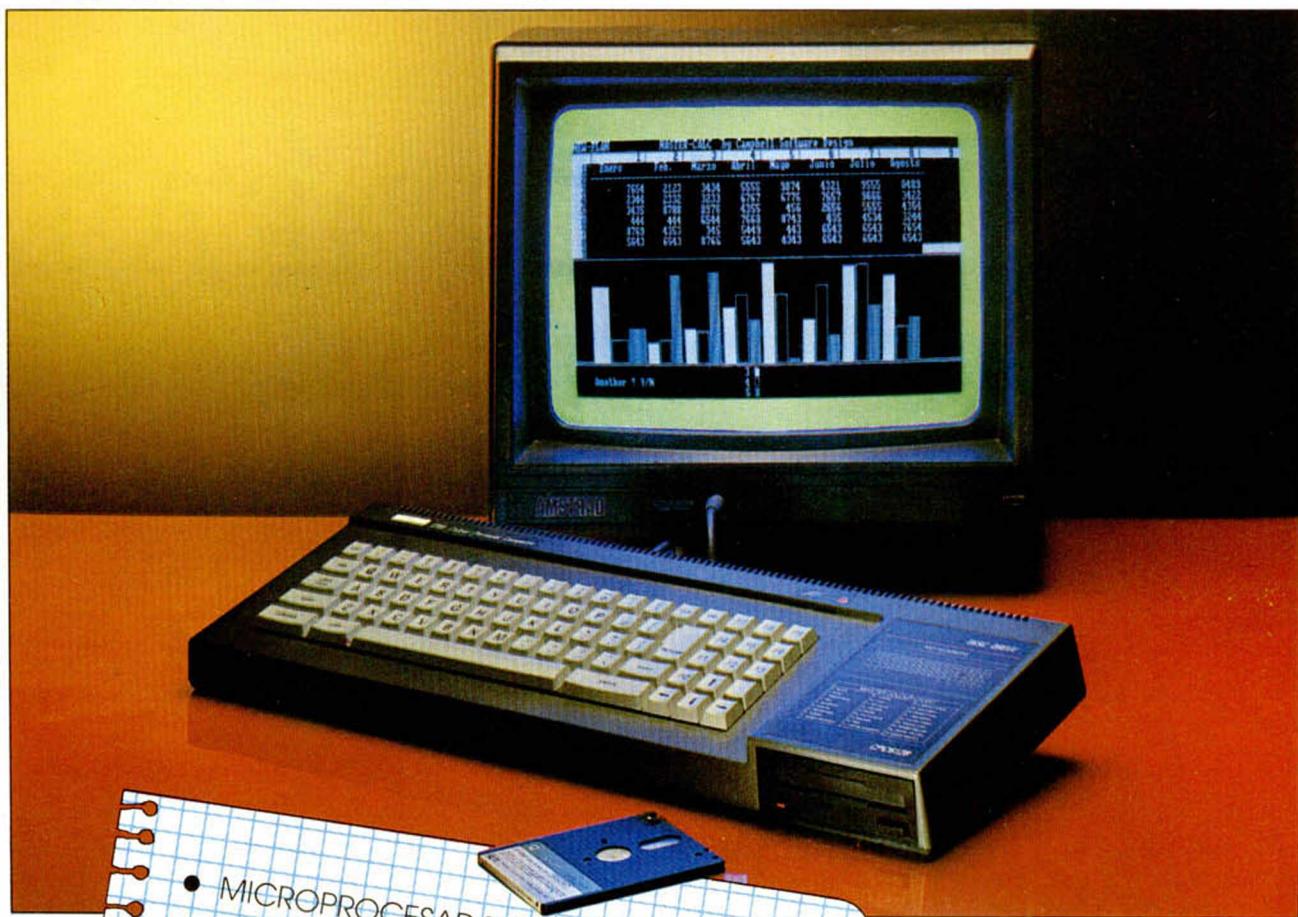
'LA FUGA' © ALBERTO TOLEDO.

```

84,1721,1756,1795,1832,1870,1912
1954,1993,2044,2077,2120,2162,2
207,2248,2293,2329,2377,2419,246
1,2505,2534,2569,2607,2642,2680,
2724,2765
9610 READ Z$: DIM S$(14,12): LET
Z$="": FOR z=1 TO 14: FOR k=0+z
TO LEN Z$: STEP 14: LET Z$=Z$+Z$(k)
: NEXT k: LET S$(Z$)=Z$: LET Z$=""
: NEXT z
9620 DATA "EEEELEEELELLLELLALL
LALARAL CBPHBBCDTEC
LMPHOREORAIIN.LRAELSCRSNNCFALSQ
IAHBRCETHUVERAUGPEOEORAEETJERORN
N O FR AE AROAO ET FT
E OE
9710 DIM U$(8,9): LET U$(1)="VAC
IO": LET U$(2)="LLEN0": LET U$(3
)= "SIN GAS": LET U$(4)="CON GAS"
: LET U$(5)="APAGADO": LET U$(6)-
="ENCENDIDO": LET U$(8)="FIRMADO"
9800 LET U$(7)="SIN FIRMA"
9800 LET Q$="": FOR L=1 TO 17: L
ET Q$=Q$+": NEXT L
9850 DIM J(72): FOR V=1 TO 72: L
ET J(V)=INT (RND*3)+1: NEXT V: L
ET J(42)=3
9900 PRINT AT 10,9; "
" RANDOMIZE USR 64506: LET L
$=" ALBERTO TOLEDO DIAZ
PRESENTA
" LA FUGA": GO SU
B 9994: PRINT AT 0,0,9$
9910 LET L$="INSTRUCCIONES?": PR
INT AT 3,9, INK 0,L$: GO SUB 99
5: GO SUB 9996: RANDOMIZE USR 64
539
9912 IF INKEY$="s" THEN PRINT AT
11,24,"SI": FOR a=1 TO 50: NEXT
a: GO TO 9920
9914 IF INKEY$="n" THEN PRINT AT
11,24,"NO": FOR a=1 TO 50: NEXT
a: GO SUB 9997: RANDOMIZE USR 6
4539: RETURN
9916 GO TO 9912
9922 GO SUB 9997: RANDOMIZE USR
64539: LET L$=" TE ACABAS RECOGER DE
LA CARCEL ERES BUSCADO PO
R TODA LA POLI-CIA.ESTE PAIS YA
NO ES SEGURO, POR LO QUE TENDRA
S QUE MARCHARTE AL EXTRANJERO PAR
A INICIAR ALLI UNA NUEVA MUEHAS
DIFICULTADES."
9924 GO SUB 9994: PRINT AT 0,0,9
$ 9926 LET L$=" DEBERAS RECOGER DE
TU CASA TODOLO INDISPENSABLE PA
RA PODER CO-GER EL AVION Y DIRI
GIRTE A TU PISO EN EL EXTRANJE
RO.PERO ESTO NO TE RESULTARA NAD
A FACIL PUES ENCONTRARAS MUCHAS
DIFICULTADES."
9928 GO SUB 9994: PRINT AT 0,0,9
$ 9938 LET L$=" UN DURO OBSTACUL
O LO REPRE- SENTA LOS OBJETOS.N
O TODOS TE SERAN UTILES.NECESI
TARAS DE UNOSPARA CONSEGUIR OTRO
S.AQUI ENTRA EN JUEGO LA LOGICA.
BUSCA LA UTI-LIDAD DE LOS OBJETO
S."
9940 GO SUB 9994: PRINT AT 0,0,9
$ 9946 LET L$=" "
!!!!SUERT
E!!!!: GO SUB 9994: PRINT AT 0,0
: Q$: RETURN
9949 GO SUB 9995: PRINT AT 1,0
: INK 0,L$: GO SUB 9996: RANDOMIZE
USR 64539: FOR A=1 TO 400: NEXT
A: GO SUB 9997: RANDOMIZE USR 6
4539: RETURN
9995 PRINT AT 0,0, INK 0; "
",RT 7
,0; "
": RETURN
9996 POKE 64548,20: POKE 64553,2
03: POKE 64554,255: POKE 64559,1
82: RETURN
9997 POKE 64548,10: POKE 64553,0
: POKE 64554,0: POKE 64559,166:
RETURN
9998 POKE 23606,24: POKE 23607,2
45: RETURN
9999 POKE 23606,88: POKE 23607,2
51: RETURN

```

# AMSTRAD CPC-6128



- MICROPROCESADOR Z80A.
- 128 K DE MEMORIA RAM (41 K DE USUARIO EN BASIC Y 61 K EN CP/M PLUS)
- 48 K DE MEMORIA ROM QUE INCLUYEN EL LOCOMOTIVE BASIC Y EL SISTEMA OPERATIVO.
- 76 TECLAS, TECLADO NUMERICO Y DE CURSOR INDEPENDIENTE.
- TEXTO EN MONITOR DE 20, 40 U 80 COLUMNAS Y GRAFICOS CON DEFINICION DE HASTA 640 X 200 PUNTOS. 27 COLORES DISPONIBLES.
- HASTA 8 VENTANAS EN PANTALLA.
- GENERACION DE SONIDOS EN 3 VOCES Y 8 OCTAVAS.
- UNIDAD DE DISCO DE 3" (169 K BYTES)
- SISTEMAS OPERATIVOS AMS-DOS Y CP/M/PLUS
- CONECTORES PARA IMPRESORA, JOYSTICKS, CASSETTE, SEGUNDA UNIDAD DE DISCO, ETC.

SISTEMA COMPLETO CON MONITOR EN FOSFORO VERDE, MANUAL EN CASTELLANO, GARANTIA OFICIAL AMSTRAD ESPAÑA, DISCO CON SISTEMA OPERATIVO CP/M 2.2 Y LENGUAJE DR. LOGO, DISCO CON SISTEMA OPERATIVO CP/M PLUS (CP/M 3.0) Y UTILIDADES, DISCO CON SIETE PROGRAMAS DE OBSEQUIO

**84.900 Pts. + I.V.A.**

SISTEMA COMPLETO IGUAL AL ANTERIOR PERO CON MONITOR EN COLOR.

**119.900 Pts. + I.V.A.**

**AMSTRAD**  
ESPAÑA

Avd. de Mediterráneo, 9, 28007 MADRID.  
Tels. 433 45 48 - 433 48 76

Delegación Cataluña: C/. Tarragona, 110,  
08015 BARCELONA - Tel. 325 10 58

# SPRITES PARA EL SPECTRUM

Actualmente todo ordenador personal que aparece en el mercado incluye la opción de sprites y la posibilidad de manejarlos desde el

Basic. Sinclair no los incorporó en su tiempo al Spectrum, probablemente por no disponer de memoria para ello, pero nada hay que impida crear una subrutina en código máquina y una serie de comandos especiales para usarla. Con esta idea hemos desarrollado una rutina con la que tú también podrás controlar sprites desde tus programas Basic.

Las subrutinas de sprites pueden ser más o menos completas o sofisticadas, pero todas ellas tienen un punto en común: sirven para mover fácilmente bloques gráficos en alta resolución y de dimensiones variables (sprites) por la pantalla, utilizando comandos sencillos en un lenguaje de alto nivel, Basic en nuestro caso.

Entre las opciones que suelen ofrecer están la ampliación (que permite hacer un sprite un número de veces más grande sin variar por ello su grado de definición) y la asignación de una prioridad a cada sprite. Así, cuando un sprite (A) se imprime sobre otro (B), de mayor prioridad, sólo se presentará en pantalla la zona de A que no esté sobre B, con lo que se consigue que parezca que A pasa por detrás de B. Esta prioridad suele venir determinada por la posición del sprite en un espacio tridimensional, produciendo efectos tan impresionantes como los conseguidos en algunos programas de casas de software como Ultimate.

## LA SUBRUTINA

Nuestra subrutina, si bien cumple con las condiciones básicas, es mucho

más modesta. Te permite definir sprites con unas dimensiones máximas de 40x40 pixels, asignarles un atributo, moverlos por la pantalla y detectar choques entre ellos.

No hay ningún límite para el número de sprites que quieras definir, excepto el impuesto por la cantidad de memoria libre de que dispongas.

El sistema de impresión utilizado es del tipo XOR (OVER 1). Esto elimina cualquier posibilidad de asignar prioridades (adiós a tus esperanzas de hacerle la competencia a Ultimate), pero también tiene sus compensaciones: hace la rutina mucho más corta y bastante más rápida, factor, este último, decisivo, sobre todo cuando va a ser usada desde el Basic, que no se distingue precisamente por su velocidad de ejecución (y menos aún el sistema de intérprete utilizado por el Spectrum).

Si dispones de un ensamblador podrás situarla en la dirección que prefieras, aunque es recomendable que sea por encima de la dirección 32767 (#7FFF). Aconsejamos, igualmente, situar el stack por encima de esta dirección con un Clear 32999 al menos (cuidado con no montarlo encima de la rutina).

Si no tienes ensamblador, pero sí el cargador de código máquina, utiliza-

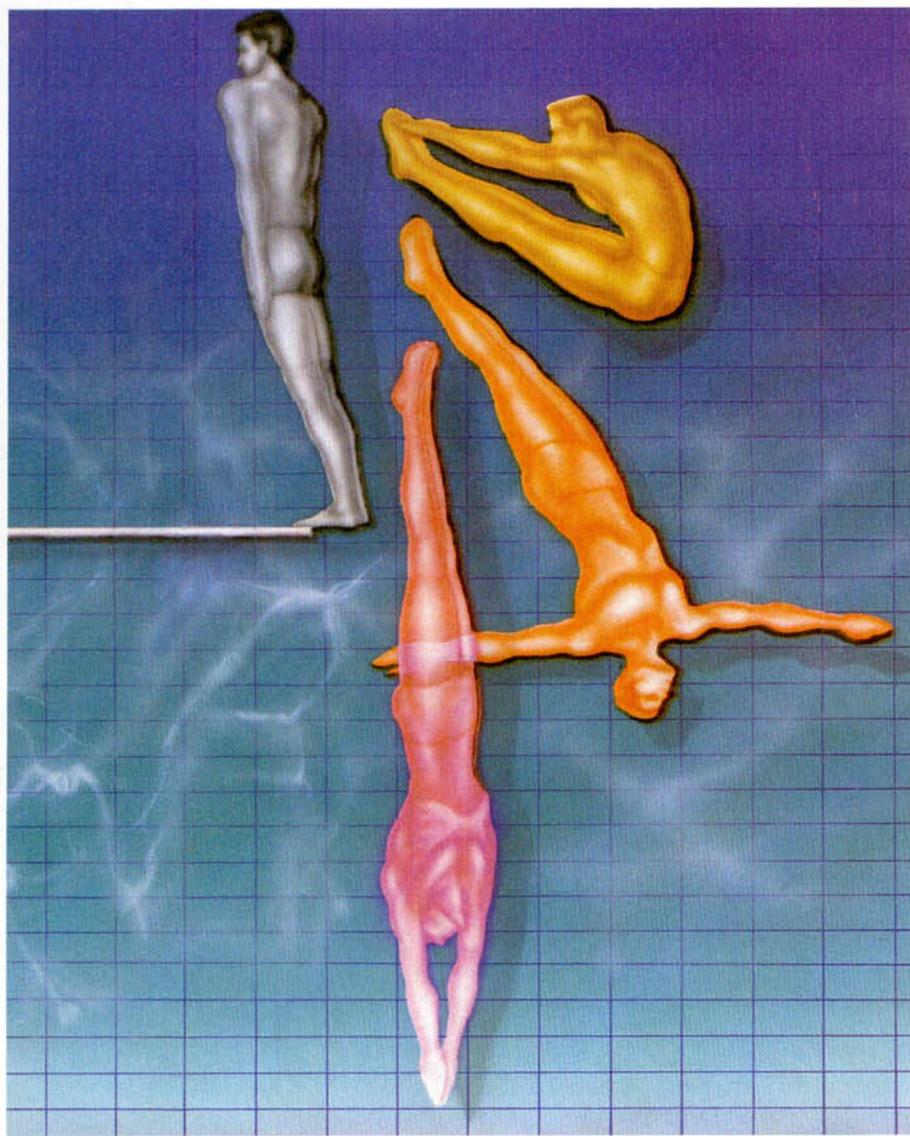
lo para situar el código objeto del listado 1 a partir de la dirección 60000 (en este caso no es reubicable) y salvarlo en cinta con longitud 1006.

El programa 1 está pensado para aquéllos que no tienen ni ensamblador ni cargador, sólo es necesario transformar las líneas del listado 1 de forma similar a como se ha hecho con la primera y darle al RUN.

## COMANDOS

Dispones de un total de seis comandos:

RESET (X). Si X es cero, los datos de los sprites se almacenarán a continuación de la subrutina (ten esto en cuenta a la hora de reubicarla). Si es distinta de cero, se tomará como dirección inicial de la zona de datos y todos los parámetros de los sprites se situarán a partir de ella. Este comando tienes que utilizarlo siempre al comenzar el programa, de otra forma el ordenador se quedará bloqueado cuando vayas a usar la subrutina. Basta con emplearlo una sola vez al inicio del programa, pero si lo vuelves a utilizar durante la ejecución de éste, borrarás los parámetros de todos los sprites, de manera que la situación será



la misma que si no hubieras definido nada.

DEFINE X: (Dx, Dy, F, Att, Dir). Define el sprite X según los parámetros que le siguen.

— Dx: dimensión horizontal en pixels.

— Dy: dimensión vertical en pixels.

— F: número de fases (gráficos) que componen un movimiento completo.

— Att: atributo (color de la tinta, flash y brillo) que quieras asociar al sprite. Notarás que no se puede definir el papel, esto es así porque el papel se considera transparente siempre.

El valor del atributo viene dado por la fórmula:

$$ATT = 128 * FLASH + 64 * BRIGHT + INK$$

Si  $ATT = 0$  el ordenador entenderá que quieras atributos transparentes, es decir, los atributos del sprite serán los que encuentre en cada momento en la pantalla.

— Dir: dirección de los datos, indica la primera dirección a partir de la cual está almacenada la información gráfica del sprite. Si el sprite tiene varias fases sólo tienes que dar la dirección del primer gráfico, el ordenador supone que el resto está a continuación de éste.

Los datos tienes que introducirlos en memoria de una forma específica: primero los de la primera línea del sprite, seguidos por los de la segunda y así sucesivamente hasta acabar con

las líneas que lo compongan (que coinciden con la dimensión de éste).

Como esto puede resultar bastante aburrido, sobre todo si defines sprites grandes, te ofrecemos un programa que lo hace por ti (programa 2). Para utilizarlo adecuadamente ten en cuenta los siguientes puntos:

1. Ajusta siempre los gráficos a la esquina superior izquierda, es decir, no dejes nunca líneas o columnas en blanco arriba o a la izquierda. Si, por ejemplo, tienes un gráfico de  $20 \times 13$  que tiene las tres primeras líneas y las dos primeras columnas en blanco, puedes redefinirlo quitando éstas y dando como dimensiones  $18 \times 10$ , esto te supondrá tanto un ahorro de memoria como de velocidad de ejecución.

2. Una vez ajustado, divide el gráfico en bloques de  $8 \times 8$  pixels (empezando siempre a contar por el extremo superior izquierdo) y utilizalos para definir los UDG del Spectrum. Si te salen más de 21, parte el gráfico en dos y trabaja con cada parte como si se tratara de un gráfico normal, sólo tienes que recordar poner la parte inferior a continuación de la superior al pasártela a la memoria.

3. Modifica la línea 20 del programa hasta que la figura completa (o partida si es demasiado grande) aparezca, tal como quieras definirla, en la esquina izquierda de la pantalla al ejecutar el programa.

4. Dale la dirección a partir de la cual quieras situar los datos. Si este gráfico es el primero del ciclo, ésa será la dirección que tendrás que especificar en el comando DEFINE. Si en cambio no es el primero, la dirección tiene que ser la siguiente a la última del gráfico anterior.

5. Especifica las dimensiones del sprite en pixels, procura no equivocarte porque si cometes algún error verás, cuando utilices la subrutina de sprites, que algunos gráficos aparecen cortados o tienen trozos de otros y tendrás que volver a introducirlos en memoria de nuevo.

Con estos datos el programa almacenará el gráfico en memoria de la forma señalada anteriormente. Cuando acabe te indicará la primera y última dirección de la zona utilizada. Apunta la dirección final más uno por

si la necesitas más adelante como dirección inicial de otro gráfico.

El número que define el sprite puede ser cualquiera entre 0 y 255, puedes definir, por lo tanto 256 sprites, pero lo más probable es que se te acabe antes la memoria.

Se generará un error 1 si intentas definir un sprite que ya haya sido definido.

**IMPRIME X:** (Cx, Cy, F). Imprime el sprite X en el punto de coordenadas Cx y Cy en la fase indicada.

La coordenada horizontal puede tomar valores entre 0 y 255 y la vertical entre -16 y 175. Los valores negativos indican que quieras imprimir por debajo de la línea 0. Tienes, en consecuencia, acceso a las dos últimas líneas.

Se producirá un error 3 si con las coordenadas dadas y la dimensión del sprite, éste no cabe completamente en la pantalla.

La fase puede oscilar entre cero y el número máximo de fases menos uno, siendo cero la primera fase, uno la segunda, ...

**BORRA X.** No necesita parámetros. Borra el sprite X y restaura los atributos originales.

Dará error 5 si se pretende borrar un sprite que no ha sido imprimido.

**MUEVE X:** (Cx, Cy). Borra el sprite X de su posición actual y lo imprime en las nuevas coordenadas Cx y Cy en la siguiente fase (la fase vuelve a cero cuando se alcanza el valor máximo).

**TEST X:** (SP1, SP2, ...). Comprueba si el sprite X está en contacto con los sprites SP1, SP2, ... Puedes colocar todos los sprites que quieras, siempre que hayan sido definidos y estén impresos en pantalla. Si en algún momento haces referencia a un sprite no definido, el programa se detendrá con un error 0.

El valor que devuelve es el número de sprites sobre los que está (sin especificar cuáles) o un cero si no está sobre ninguno, es suficiente con que tengan un pixel en común para que los considere en contacto.

Este valor es asignado a la variable del último LET. Para evitar problemas asegúrate de que la instrucción con la que llamas a la subrutina tenga, en este caso, la forma LET I=USR dirección,

donde I es la variable con la que quieres detectar el choque.

Si en algún momento introduces un número fuera del rango establecido en cada caso, se producirá un error 3.

ningún efecto, de forma que si haces un PRINTUSR no se imprimirá nada; si usas un LET, la variable no tomará ningún valor distinto del que tenía (excepto si uno de los comandos es TEST) y RANDOMIZE no alterará la secuencia de números aleatorios. De cualquier manera, las instrucciones más recomendables, debido únicamente a que son un poco más rápidas, son PRINT y RANDOMIZE. Utiliza éstas siempre que te sea posible.

Recuerda, por último, que la instrucción de llamada, junto con los comandos de control de los sprites, tiene que ser, bien la única instrucción de una línea o bien la última de ésta. Esto se debe a que el intérprete del Spectrum ignorará todas las instrucciones Basic que coloques después de ellos.

### SINTAXIS

**La forma base USR 60000: REM** tiene que aparecer siempre, seguida por el texto del comando que quieras ejecutar (en mayúsculas) y de los parámetros que necesite. Estos pueden venir expresados por números en decimal, en hexadecimal (precedidos por #) o bien a través de variables numéricas. No se aceptan ni expresiones ni elementos de matrices (no son válidas  $3^*X-2$  ó  $A(2,3)$ ). Si se utiliza alguna variable no definida, el programa se parará con un error 4.

Puedes encadenar comandos sin necesidad de volver a utilizar USR usando como separador el punto y coma. Por ejemplo:

**RESET (0); DEFINE 2: (...); IMPRIME 2: (...)**

es perfectamente válida y además es más rápida.

Si pones más parámetros de los necesarios (o menos), olvidas algún separador, das un comando erróneo o incompleto o, en general, no cumples algunas de las anteriores reglas de sintaxis, conseguirás un error 2.

Cuando se produce un error se detiene la ejecución del programa y en la parte inferior de la pantalla aparece un número, que indica el tipo de error, seguido por una coma y la línea e instrucción donde ha sido detectado. El sistema de numeración es el normal del Spectrum, pero aquí se sigue contando después de la instrucción USR. Así, si la instrucción de llamada era la sexta de la línea, el primer comando se considera la séptima instrucción, el segundo, la octava, etc. Esto te permitirá localizar con mayor precisión el error.

Para llamar a la subrutina puedes utilizar cualquier cosa que acepte la forma base anterior, pues no tendrá

### MODIFICACIONES Y MEJORAS

Evidentemente las posibles variaciones se te irán ocurriendo a medida que uses la rutina y necesites ajustarla a tus necesidades. De cualquier manera, aquí tienes algunas ideas.

Variar las dimensiones máximas. Puedes hacerlo siempre que el número total de bytes no exceda de 256 (el número real de bytes que ocupa cada gráfico del sprite es  $Dy^* (1+INT((Dx-1)/8))$ ).

Si quieras sprites más grandes, tendrás que variar las líneas 2650 a 2700 y utilizar instrucciones que trabajen con datos mayores de 255.

Recuerda ajustar la longitud del buffer cada vez que cambies las dimensiones máximas.

Otra posibilidad interesante puede ser introducir otros cuatro parámetros en el comando DEFINE. Dos de ellos determinarán un punto del sprite y los otros dos las dimensiones de un rectángulo interior a él, de tal forma que la subrutina de choque detecte cuándo están en contacto estas zonas del sprite y no todo el sprite.

Tanto en este caso como en el sistema utilizado normalmente, se detecta un choque cuando los rectángulos que definen el sprite están en contacto, in-

dependientemente de que los gráficos se toquen o no.

La rutina de impresión en sí ocupa poco más de 150 bytes (PRINT e INCH) y no hay problema para que la utilices en tus programas en código máquina, siempre que le proporciones los datos que necesite en cada momento.

El campo de cada sprite tiene los parámetros distribuidos de la siguiente forma:

IY+0 0 Longitud del campo  
 IY+01 Número del sprite  
 IY+02/03 Coordenadas x/y en pixels  
 IY+04/05 Dimensiones x/y en pixels  
 IY+06&07 Dirección del primer gráfico  
 IY+08&09 Longitud de cada gráfico  
 IY+10 Fase o gráfico actual  
 IY+11 Número máximo de fases  
 IY+12 Atributo asociado al sprite  
 IY+13 Flag de impresión

Seguida por la memoria suficiente para almacenar temporalmente los atributos de la zona de pantalla sobre la que se imprime el sprite. Si el sprite tiene atributos transparentes no se reserva ninguna memoria.

## UN EJEMPLO PARA ACABAR

¿Qué mejor que un ejemplo para ver de una forma práctica todo lo anterior? Para ello te ofrecemos el programa 3. Este divide la pantalla en tres partes y mueve simultáneamente distintos elementos en cada una de ellas. Encontrarás camiones, una pelota de tenis, autos y un típico comeculos persiguiendo a un típico fantasma. Fíjate sobre todo en el listado y en los pequeños trucos que se utilizan (el uso de la tinta y el papel negro en la parte central, por ejemplo) y empléalos en tus propios programas. No olvides asegurarte que tienes la subrutina de sprites en memoria antes de ejecutarlo.

Y Feliz subrutina.

## CUADRO RESUMEN

**RESET (X).** Inicia el área de datos.  
**DEFINE X:** (Dx, Dy, F, Att, Dir). Define el sprite X:  
 — Dx: dimensión x  
 — Dy: dimensión y  
 — F: número de fases  
 — Att: atributo del sprite  
 — Dir: dirección de los gráficos  
**IMPRIME X:** (Cx, Cy, Fase). Imprime el sprite X en las coordenadas Cx y Cy en la fase indicada.  
**BORRA X.** Borra el sprite X.  
**MUEVE X:** (Cx, Cy). Borra el sprite X y lo imprime en la posición Cx y Cy en la siguiente fase.  
**TEST X:** (SP1, SP2, ...). Comprueba si el sprite X está en contacto con alguno de los sprites que se enumeran.  
**ERROR 0.** Sprite no definido.  
**ERROR 1.** Sprite ya definido.  
**ERROR 2.** Error de sintaxis.  
**ERROR 3.** Número fuera de rango.  
**ERROR 4.** Variable no definida.  
**ERROR 5.** Sprite no impreso.

**DUMP: 6000**  
**DIR: 60000**  
**N.º BYTES: 1.010**

1	F311D9EDCD800RE7FEEA	1776
2	20353A475C3C32475C11	596
3	FBEDDSE7CD03ED302422	1495
4	5D5C06002142EE095E23	666
5	56D579FE04D8CD3DEDCD	1602
6	25ED8033E000779F08	993
7	C87FE28C83E02D7CD3D	1470
8	EDCD25ED30033E017FD	1298
9	7001CD9DEAFCB0DC6CD	1581
10	3DEDA72804FE2938033E	925
11	03D7FD7704CD3DEA728	1304
12	F4FE2930F0FD7705CD3D	1470
13	EDA728E7FD7708CD3DED	1561
14	E8C7FD770C2809CD03EC	1306
15	0C04F8110FDC60EFD77	1173
16	005F1600FDE5E11936FF	1158
17	62D7E04CDCEED6FFD5E	1587
18	05CD8LED7508FD7409	1380
19	CD4CEDFD7306FD7207C9	1457
20	C037EB2CDC8E8E80A3C	1586
21	FD8E801AFFF770A18	1092
22	292101E8E3FDCB0D4628	1119
23	033E05D7CD12ECAFCDAA	1294
24	EFCDCB0D86DC9CDCAEB	1969
25	CD3D0EFD8E0BD2C3EAFD	1849
26	770AFDCB0D86CD12E3C7	1246
27	C3AREC0E00FDE5DDE1CD	1748
28	3DEDCD25EDD296EAD2B	1782
29	01282D7FDCB0D462027FD	949
30	70E2D960230063DFD86	1003
31	01803DDBE043014FD7E	893
32	03DD9560330063DD8605	852
33	1803FDBE0530010CDFF	1013
34	2920C06000CD2B2DC3FF	1014
35	2ACD90EACD4CED7AB320	1489
36	031118EFFED534EEE3FF	1236
37	12C9CD3DDEFD8604D4C3	1526
38	ED5CD3DDEFD2807FE11D2	1478
39	C3FAED44C610FEC030F5	1687
40	3CFD8E0538E43DFD7703	1228
41	ED57D7302C9FD7E03CDCE	1573
42	ED57FD7E02CDCEED5FFD	1701
43	705CDCEED47FD7E04CD	1438
44	CEED4FC9FD6E06FD6607	1454
45	FD5E08FD5609FD7E0A47	1163
46	A728031910FDE5DDE1FD	1432
47	47E024FE607573E	919
48	BFCDAC22E5E52150EEFD	1664
49	4605FD7E04CDCEED5F4B	1276
50	DD7E000D2377230D20F6	1048
51	79A7280336002310EC28	713
52	154A1CFD4605AF8310FD	1027
53	2150EE47A7CB1E2310FB	1124
54	0D20F3D46054BFDE50F	1426
55	213A5CFB76F3FDE11150	1370
56	E0E1E5C51AEE772C13D0	1284
57	20E8C1E1CD9BEC10FEE1	1774
58	C9247CE507C07DC6206F	1256
59	D87CD60867C908FD7E0C	1265
60	17C7C0F0F0FF603F558	1103
61	67FDE5DDE110E000D19	1308
62	CDF5EBFD7E03E607FE07	1555
63	280104FD7E02E6072801	704
64	0C16003E20915FC508DC	793
65	F0ECD4FDEC23D230D20	1513
66	F3C11910FEC9087EDD77	1390
67	00E538FD860C7737C908	1116
68	DD7E0077C911BEE0E00	963
69	E51RA728158E20042313	763
70	18F500C1A13A720FB1A	814
71	FEFE1C818E6D137C9D0F	1906
72	284EEE471600FD7E00FE	1084
73	FFC85FFD7E01B837C9D0F	1622
74	1918EFFE5C5CD4CED7A97	1521
75	C2C3EA04C1E178C906FF	1630
76	E7F7E2D200204E70E0A0FE	1077
77	2320050F10E7182FCD1B	636
78	2D302ACD8D2CD2A1EFD	1383
79	E5FD213A5CCDB2283003	1139
80	3E04D7C9A1E23CD8433	1349
81	CDA22DDAC3E85059FDE1	1706
82	06FFC804C9C5CDB8EDDA	1707
83	A1EAE050606F50591808	809
84	4FC041ED09D4C3EAE5E7	1814
85	E1CDB8ED30F0EBC1C9D	1973
86	A930D0C3C3EACD1B2D38	1382
87	03D630C9F41D8FE473F	1389
88	D8C6137C8D637C9A728	1448
89	06301F1F1FE61F3C9CFD	935
90	213A5CF5CD6E0DF111F4	1258
91	09CD800A212021223B5C	635
92	C630D7114913ED7B3D5C	1083
93	FBD5C9DFF29C2A1EFD7	2003
94	FE3BCA6CEAFD213A5C11	1310
95	F409CD800R11B31BFED0	1086
96	28DCE718F9444546494E	1122
97	45005245534554005445	609
98	5354004D554556450092	619
99	4F525C24100494D505249	693
100	4D4500FFA4EAB5E87EB	1553
	1EEB33EB4DEB00000000	863

## PROGRAMA 2

```

1 REM CARGADOR DE GRAFICOS
10 BORDER 0: PAPER 0: INK 7: C
LEAR 29999: POKE 23658,8
20 PRINT AT 0,0;"ABCD";AT 1,0;
"EFGH";AT 2,0;"ijkl";AT 3,0;"MNO
P"
30 INPUT "DIRECCION ? ";K: LET
DIR=K
40 INPUT "DIMENSION X ? ";DX;"D
IMENSION Y ? ";DY: LET DX=1+INT
((DX-1)/8): IF DX>5 OR DY>40 TH
EN GO TO 40
50 FOR R=0 TO DY-1: LET D=1638
4+256*INT (R-8*INT (R/8))+32*INT
(R/8)
60 FOR S=0 TO D+DX-1: POKE DIR
,PEEK S: LET DIR=DIR+1: NEXT S
70 NEXT R: PRINT "";"ZONA OCUPA
DA: ";K;"-";DIR-1
80 INPUT "LO LISTO ? "; LINE A
$: IF A$<>"S" AND A$<>"N" THEN G
O TO 80
90 IF A$="N" THEN STOP
100 FOR R=1 TO DY: PRINT R;: FO
R S=1 TO DX: PRINT TAB 4*5;PEEK
K: LET K=K+1: NEXT S: PRINT : N
EXT R
110 STOP
9999 SAVE "CARGA-GRAF" LINE 10

```

# GENS SPRITE COMENTADO

18	ORG	68000		728	JR	NC,ERR3		1438	JP	ATTR			
20	SPRITE	DI	;Sin interrupc	738	LD	(IY+85),A	;Inicia dimension y	1448	;		;Pone el atributo		
38	LD	DE,ERROR	;Cambia la direccion de	748	CALL	COGEA	;Coge numero de fases	1458	CHOQUE	LD	C,##0	;Limpia el indicador	
40	CALL	CAMBIA	;impresion por la de error	758	AND	A	;no cero	1468	PUSH	IY		;Copia la direccion	
58	RST	##28	;Coge el siguiente caracte	768	JR	Z,ERR3		1478	POP	IX		;base en IX	
68	CP	HEA	;Es un REM ?	778	LD	(IY+11),A	;Inicia fases	1488	CHI	CALL	COGEA	;Coge sprite a comprob	
78	JR	N2,ERR2	;Error si no lo es	788	CALL	COGEA	;Coge el atributo	1498	CALL	BUSCA		;Ha de estar definido	
88	ENT2	LD	A,(SUBPPC)	798	AND	##C7	;No se acepta papel	1508	JP	NC,ERR3			
98	INC	A	;incrementa el	808	LD	(IY+12),A	;Inicia atributo	1518	CP	(IX+81)			
108	LD	(SUBPPC),A	;instrucciones	818	JR	Z,DFN	;Salta si es cero	1528	JR	Z,CH6			
118	LD	DE,RETSB	;Inicia el stack	828	CALL	PRM2	;Coge parametros	1538	BIT	8,(IY+13)			
128	PUSH	DE		838	INC	C	;Incrementa las	1548	JR	N2,CH6			
138	RST	##28	;Avanza un caracter	848	INC	B	;dos dimensiones	1558	LD	A,(IY+82)			
148	CALL	COMAND	;Tiene que ser uno de	858	XOR	A		1568	SUB	(IX+82)			
158	JR	NC,ERR2	;los comandos definidos	868	DF3	ADD	A,C	1578	JR	NC,CH2			
168	LD	(CHADD),HL	;Actualiza CHADD	878	DJNZ	DF3	;A=B=C	1588	DEC	A			
178	LD	B,##0		888	DFN	ADD	A,##8E	1598	ADD	A,(IY+84)			
188	LD	HL,DIRECC	;La direccion base	898	LD	(IY+88),A	;Suma la longitud minima	1608	JR	CH3			
198	ADD	HL,BC	;le suma el offset	908	LD	E,A	;Inicia longitud	1618	CH2	CP	(IX+84)	;Sigue adelante	
208	LD	E,(HL)	;y carga en DE	918	LD	D,##0		1628	CH3	JR	NC,CH6	;si no esta	
218	INC	HL	;la direccion	928	PUSH	IY		1638	LD	A,(IY+83)			
228	LD	D,(HL)	;de la subrutina	938	POP	HL		1648	SUB	(IX+83)			
238	PUSH	DE		948	ADD	HL,DE	;Pone el indicador de	1658	JR	NC,CH4			
248	LD	A,C	;Comprueba el offset	958	LD	(HL),##FF	;fin de datos	1668	DEC	A			
258	CP	##84	;Regresa directamente	968	LD	H,D	;Limpia H	1678	ADD	A,(IX+85)			
268	RET	C	;para DEFINE y RESET	978	LD	A,(IY+84)	;y calcula	1688	JR	CH5			
278	CALL	COGEA	;Coge el numero de sprite	988	CALL	INT8	;el numero de bytes	1698	CH4	CP	(IY+85)		
288	CALL	BUSCA	;Esta definido ?	998	LD	L,A	;que ocupa	1708	CH5	JR	NC,CH6	;Se incrementa C por	
298	JR	C,SP2	;sigue si lo esta	1008	LD	E,(IY+85)	;cada grafico	1718	INC	C			
308	ERR8	LD	A,##88	1018	CALL	MULT	;del sprite	1728	CH6	RST	##18	;Toma el ultimo caract	
318	RST	##18		1028	LD	(IY+88),L	;Pasa ese valor al	1738	CP	##29			
328	;			1038	LD	(IY+89),H	;area de parametros	1748	JR	N2,CH1			
338	* SPRITE NO DEFINIDO *			1048	CALL	TKDE	;Coge la direccion de	1758	LD	B,##0			
348	;			1058	LD	(IY+86),E	;los graficos y	1768	CALL	STBKC			
358	SP2	LD	A,C	1068	LD	(IY+87),D	;la guarda	1778	JP	LET			
368	CP	##88	;El offset	1078	RET	;Vuelve si es asi	B	1788	;				
378	RET	Z	;Es el de BORRA ?	1088	;			1798	RESET	CALL	SP3	;Comprueba sintaxis	
388	SP3	RST	##28	1098	MUEVE	CALL	BORRA2	1808	CALL	TKDE			
398	CP	##28	;Vuelve si es asi	1108	CALL	COORD	;Cogesus coordenadas	1818	LD	A,D			
408	RET	Z	;Regresa si	1118	LD	A,(IY+81)	;Incrementa el	1828	OR	E			
418	ERR2	LD	A,##82	1128	INC	A	;contador de fases	1838	JR	N2,REDF			
428	RST	##18	;le sigue '('	1138	CP	(IY+11)	;que vuelve a cero si	1848	LD	DE,DIRSP			
438	;			1148	JR	C,MOV2	;se alcanza el maximo	1858	REDF	LD	(DATSP),DE		
448	* ERROR DE SINTAXIS *			1158	XOR	A		1868	LD	A,##FF			
458	;			1168	MV2	LD	(IY+18),A	1878	LD	(DE),A			
468	DEFINE	CALL	COGEA	1178	JR	PONLO	;se imprime el sprite	1888	RET				
478	CALL	BUSCA	;Coge el numero de sprite	1188	;			1898	;				
488	JR	NC,DF1	;No tiene que	1198	BORRA	LD	HL,RTB2	1908	COORD	CALL	COGEA		
498	LD	A,##81	;estar definido	1208	EX	(SP),HL	;Cambia la direccion	1918	ADD	A,(IY+84)			
508	RST	##18		1218	BORRA2	BIT	8,(IY+13)	1928	CDR8	JP	C,ERR3		
518	;			1228	JR	Z,BRR2	;de retorno	1938	PUSH	DE			
528	* SPRITE YA DEFINIDO *			1238	LD	A,##05	;Esta el sprite imprimido	1948	CALL	COGEA			
538	;			1248	RST	##18	;Continua si lo esta	1958	JR	Z,CDR2			
548	DF1	LD	(IY+81),B	1258	;			1968	CP	##11			
558	CALL	SP3	;Inicia el numero de sprit	1268	* SPRITE NO IMPRESO *			1978	CDR1	JP	NC,ERR3		
568	SET	8,(IY+13)	;Comprueba sintaxis	1278	;			1988	NEG	;			
578	CALL	COGEA	;Sprite no imprimido	1288	BR2	CALL	PRINT	1998	CDR2	ADD	A,##18		
588	AND	A	;Coge dimension x	1298	XOR	A	;Borra el sprite	2008	CP	##C8			
598	JR	Z,ERR3	;no puede ser cero	1308	CALL	ATTR	;Carry a cero	2018	JR	NC,CDR1			
608	CP	##29		1318	SET	8,(IY+13)	;Restaura atributos	2028	INC	A			
618	JR	C,DF2		1328	RST	##18	;Sprite no imprimido	2038	CP	(IY+85)			
628	ERR3	LD	A,##83	1338	RET			2048	JR	C,CDR8			
638	RST	##18		1348	;			2058	DEC	A			
648	;			1358	IMPRIM	CALL	COORD	2068	LD	(IY+83),A			
658	* NUMERO FUERA RANGO *			1368	CALL	COGEA	;Coge las coordenadas	2078	POP	DE			
668	;			1378	CP	(IY+11)	;y el numero de fase	2088	LD	(IY+82),E			
678	DF2	LD	(IY+84),A	1388	JP	NC,ERR3	;esta en rango ?	2098	RET				
688	CALL	COGEA	;Inicia dimension x	1398	LD	(IY+18),A		2108	;				
698	AND	A	;Coge dimension y	1408	PONLO	RES	8,(IY+13)	2118	PARAM	LD	A,(IY+83)		
708	JR	Z,ERR3		1418	CALL	PRINT	;Sprite en pantalla	2128	CALL	INT8			
718	CP	##29		1428	SCF	;	;Lo imprime	2138	LD	D,A			

2148 LD A,(IY+82)	2988 PUSH BC	3668 SCF	3678 RET	
2150 CALL INT8	2918 PS9 LD A,(DE)	;Mezcla los datos del	3688 ;	
2168 LD E,A	2928 XOR (HL)	;buffer con lo	3690 REST EX AF,AF'	
2178 PRM2 LD A,(IY+85)	2938 LD (HL),A	;que hay en pantalla	3708 LD A,(IX+80)	;Restablece los
2188 CALL INT8	2948 INC L		3718 LD (HL),A	;atributos originales
2198 LD B,A	2958 INC DE		3728 RET	
2208 LD A,(IY+84)	2968 DEC C		3738 ;	
2218 CALL INT8	2978 JR NZ,PS9		3748 COMANDO LD DE,TEXT0	
2228 LD C,A	2988 POP BC		3758 LD C,##8	;Offset a cero
2238 RET	2998 POP HL		3768 CD8 PUSH HL	;Direccion inicial comando
2248 ;	3008 CALL INCH		3778 CD1 LD A,(DE)	
2258 PRINT LD L,(IY+86)	3018 DNZ PS8		3788 AND A	;Salta si el comando
2268 LD H,(IY+87)	3028 POP HL		3798 JR Z,CD3	;es correcto
2278 LD E,(IY+88)	3038 RET		3808 CP (HL)	
2288 LD D,(IY+89)	3048 ;		3818 JR NZ,NC0	
2298 LD A,(IY+10)	3058 INCH INC H		3828 INC HL	
2308 LD B,A	3068 LD A,H		3838 INC DE	
2318 AND A	3078 AND ##87		3848 JR CD1	
2328 JR Z,PR2	3088 RET NZ		3858 NC0 INC C	;Incrementa
2338 PR1 ADD HL,DE	3098 LD A,L		3868 INC C	;el offset
2348 DNZ PR1	3108 ADD A,##20		3878 CD2 LD A,(DE)	
2358 PR2 PUSH HL	3118 LD L,A		3888 INC DE	;Busca el final
2368 POP IX	3128 RET C		3898 AND A	;del comando
2378 LD B,(IY+83)	3138 LD A,H		3908 POP HL	
2388 LD A,(IY+82)	3148 SUB ##88		3918 JR NZ,CD2	
2398 LD C,A	3158 LD H,A		3928 LD A,(DE)	
2408 AND ##87	3168 RET		3938 CP ##FF	
2418 LD D,A	3178 ;		3948 RET Z	
2428 LD A,##BF	3188 ATTR EX AF,AF'		3958 JR CD8	
2438 CALL PIXEL	3198 LD A,(IY+12)		3968 CD3 POP DE	
2448 PUSH HL	3208 AND A		3978 SCF ;Carry a uno:	
2458 PUSH HL	3218 RET Z		3988 RET ;comando encon	trado
2468 LD HL,BUFFER	3228 LD A,H		3998 ;	
2478 LD B,(IY+85)	3238 RRCA ;del archivo d		4008 BUSCA LD IY,(DATSP)	
2488 LD A,(IY+84)	3248 RRCA		4018 LD B,A	
2498 CALL INT8	3258 RRCA		4028 LD D,##88	
2508 LD E,A	3268 AND ##83		4038 CHK LD A,(IY+88)	
2518 PS3 LD C,E	3278 OR ##58		4048 CP ##FF	
2528 PS4 LD A,(IX+88)	3288 LD H,A		4058 RET Z	
2538 INC IX	3298 PUSH IY		4068 LD E,A	
2548 LD (HL),A	3308 POP IX		4078 LD A,(IY+81)	
2558 INC HL	3318 LD DE,##80E		4088 CP B	
2568 DEC C	3328 ADD IX,DE		4098 SCF ;a uno si es e	
2578 JR NZ,PS4	3338 CALL PARAM		4108 RET Z	
2588 LD A,D	3348 LD A,(IY+83)		4118 ADD IY,DE	
2598 AND A	3358 AND ##87		4128 JR CHK	
2608 JR Z,PSN	3368 CP ##87		4138 ;	
2618 LD (HL),##88	3378 JR Z,ATT		4148 COGEA PUSH HL	
2628 INC HL	3388 INC B		4158 PUSH BC	
2638 PSN DJNZ PS3	3398 ATT LD A,(IY+82)		4168 CALL TKDE	
2648 JR Z,PS7	3408 AND ##87		4178 LD A,D	
2658 LD C,D	3418 JR Z,ATT8		4188 AND A	
2668 INC E	3428 INC C		4198 JP NZ,ERR3	
2678 LD B,(IY+85)	3438 ATT8 LD D,##88		4208 INC B	
2688 XOR A	3448 LD A,##20		4218 POP BC	
2698 PSS ADD A,E	3458 SUB C		4228 POP HL	
2708 DJNZ PSS	3468 LD E,A		4238 LD A,E	
2718 PS6 LD HL,BUFFER	3478 ATT2 PUSH BC		4248 RET	
2728 LD B,A	3488 ATT3 EX AF,AF'		4258 ;	
2738 AND A	3498 CALL C,PN		4268 TKDE LD B,##FF	
2748 PSLP RR (HL)	3508 CALL NC,REST		4278 RST ##28	
2758 INC HL	3518 INC HL		4288 CP ##20	
2768 DJNZ PSLP	3528 INC IX		4298 JR NZ,TK1	
2778 DEC C	3538 DEC C		4308 INC B	
2788 JR NZ,PS6	3548 JR NZ,ATT3		4318 RST ##28	
2798 PS7 LD B,(IY+85)	3558 POP BC		4328 TK1 LD C,##8A	
2808 LD C,E	3568 ADD HL,DE		4338 CP ##23	
2818 PUSH IY	3578 DJNZ ATT2		4348 JR NZ,TK2	
2828 LD IY,##5C3A	3588 RET		4358 LD C,##10	
2838 EI ;Sincroniza co	3598 ;		4368 RST ##28	
2848 HALT ;barrido de pa	3608 PON EX AF,AF'		4378 JR DHEX	
2850 DI	3618 LD A,(HL)		4388 TK2 CALL NUMBER	
2868 POP IY	3628 LD (IX+80),A		4398 JR NC,DHEX	
2878 LD DE,BUFFER	3638 AND ##38		4408 CALL ALPHA	
2888 POP HL	3648 OR (IY+12)		4418 JP NC,ERR2	
2898 PS8 PUSH HL	3658 LD (HL),A			

## **PROGRAMA 3**

```

1 REM PROGRAMA EJEMPLO
10 GO TO 5000
100 LET X1=X1+STX: IF X1<1 OR X1>241 THEN LET STX=-STX: OUT 254
110 LET Y1=Y1+STY: IF Y1<67 OR Y1>150 THEN LET STY=-STY: OUT 254
120 RANDOMIZE USR 60000: REM MUEVE
EVE 1: (X1,Y1): REH NO PONER AQUI
INSTRUCCIONES BASICAS
130 LET X2=X3+3: IF X2>241 THEN
LET X2=1
140 LET X3=X3+3: IF X3>243 THEN
LET X3=1
150 PRINT USR 60000: REM MUEVE
2: (X2,Y2); MUEVE 3: (X3,Y3)
160 LET X4=X4-2: IF X4<0 THEN LET
ET X4=215
170 LET X5=X5-2: IF X5<0 THEN LET
ET X5=215
180 PRINT USR 60000: REM MUEVE
4: (X4,Y4); MUEVE 5: (X5,Y5)
190 GO TO 100
5000 BORDER 0: PAPER 0: INK 7: C
LS : GO SUB 9000: REM GRAFICOS
5010 LET X1=127: LET Y1=100: LET
STX=4+3*(RND-.5): LET STY=4+3*(RND-.5)
5020 LET X2=3: LET Y2=23: LET X3=37: LET Y3=Y2
5030 LET X4=50: LET X5=135: LET
X6=200: LET Y4=47: LET Y5=Y4: LET
T Y6=Y5
5040 GO SUB 8000: REM DEFINE SPRITES
5050 GO SUB 7000: REM PANTALLA
5060 GO TO 100
7000 CLS : PRINT AT 1,0;""
PROGRAMA EJEMPLO
7001 FOR R=3 TO 14: PRINT AT R,1

```

```

6. PAPER 1;" ":" N
EXT R
7010 DRAW 255,0: DRAW 0,151: DRAW
U,-255,0: DRAW 0,-151: OVER 1: P
LOT 0,32: DRAW 255,0: DRAW 0,23:
DRAW -255,0: DRAW 0,-23: OVER 0
7020 PRINT INK 0: PAPER 0; AT 16,
0;" PAPER 2;" INK 7; PAPER 1;" PAPER
1;" PAPER 2;" INK 0; PAPER 0;"

7050 PRINT USR 60000: REM IMPRIM
E 1: (X1,Y1,0): IMPRIME 2: (X2,Y2,0)
: IMPRIME 3: (X3,Y3,0): IMPRIME 4:
(X4,Y4,0): IMPRIME 5: (X5,Y5,0)
7100 RETURN
8000 PRINT USR 60000: REM RESET
(9: DEFINE 1: (11,12,4,70,50191):
REM DEFINE PELOTAS
8010 PRINT USR 60000: REM DEFINE
2: (13,13,4,70,50000): DEFINE 3: (11,
13,13,55,50104): REM DEFINE CO
MECOCOS Y FANTASIA
8020 FOR R=4 TO 5: PRINT USR 600
0: REM DEFINE R: (40,7,1,0,50156
): REM DEFINE CAMION-COCHE
8030 NEXT R
8040 RETURN
9000 LET SUM=0: FOR R=50000 TO 5
0256: READ A: LET SUM=SUM+A: POK
E R,A: NEXT R: READ A: IF A<>SUM
THEN PRINT "ERROR EN DATOS": ST
OP
9010 RETURN
9100 REM DATOS COMECOCOS
9110 DATA 15,128,63,224,.115,.240,
.115,.240,.255,.248,.255,.248,.255,.25
5,.248,.255,.248,.127,.240,.127,.240,.63
,.224,.15,.128
9120 DATA 15,.128,.63,.224,.115,.240,

```

# ¡¡Gratis!!

Suscríbete a Microhobby o realiza ahora tu renovación y recibirás, totalmente gratis, este magnífico regalo.

Kit profesional de ajuste y mantenimiento.

Envíanos hoy mismo el cupón de suscripción que se encuentra cosido en las páginas de esta revista y te asegurarás todo un año de lectura estimulante y, además, evitarás todos tus problemas de carga.



¡PON A PUNTO TU CASSETTE Y OLVIDATE DE LOS PROBLEMAS DE CARGA!

(Oferta válida sólo para España, hasta el 31 de octubre de 1986).

- Destornillador especial para ajuste de azimut
- Spray limpiador de cabezas magnéticas «Computer Cleaner»
- Cassette con instrucciones de uso grabadas

Como ya sabéis, una de las «facultades» del Spectrum es la de poder imprimir gráficos en pantallas. De hecho, siempre trabaja en modo gráfico aunque lo que aparezca en pantalla tenga formato de texto. En este artículo vamos a verlo detalladamente.

**E**l problema empieza a la hora de realizar gráficos para juegos o dibujos complejos que requieran un gran detalle. Es en esos momentos cuando contemplamos los famosos UDG con lógica y desesperación. ¿Cuál es la solución REAL?: adquirir un programa especializado en gráficos. Pero como nunca llueve a gusto de todos resulta que al cabo de un tiempo (incluso el primer día) se echan en falta ciertas funciones que facilitarían

grama de archivo y volcado de gráficos, lo hemos adaptado especialmente de forma que los que ya lo poseen dispongan a partir de ahora de una herramienta muy potente y, aunque problemas de espacio y tiempo nos han impedido añadirle más opciones, el resultado final asombrará a la mayoría y dejará más que satisfechos a los exigentes. Aquéllos que no posean el programa SUPERGRAFICOS (Melbourne Draw) simplemente han de seguir las indicaciones que comen-

en detalle, y aparecerá en pantalla como sigue:

Para seleccionar cada opción basta pulsar la tecla que corresponde a cada letra. Las distintas opciones operan del siguiente modo:

**M.**—Al pulsar la tecla «M» aparecerá el menú de ayuda. Al final de la ejecución de cada una de las opciones seleccionadas (excepto MENU) se hace un volcado de la pantalla de trabajo quedando el programa a la espera de una nueva opción. Si se desea volver entonces al BASIC



el tedioso y complejo proceso de desarrollo de figuras, sobre todo en el caso de gráficos de animación. Una solución suele ser esperar a que salga una nueva versión que satisfaga nuestras necesidades, pero ante el precio unas veces y el hermetismo de manejo otras, resulta que al final solemos hacer aquello de «más vale viejo conocido...».

Esto ocurre, por ejemplo, con el excelente programa SUPERGRAFICOS (Melbourne Draw) el cual fue un programa de impacto en su día pero poco a poco está siendo desplazado por nuevos y más potentes programas.

Hemos de reconocer que como tenemos cierta debilidad por él, y con la excusa inicial de presentar un pro-

taremos después.

El programa está realizado íntegramente en código máquina salvo en las opciones de carga, verificación y almacenamiento que se manejan en parte desde BASIC con objeto de que cada usuario adapte el programa a los periféricos de que disponga (cinta, microdrive, wafadrive, disco...).

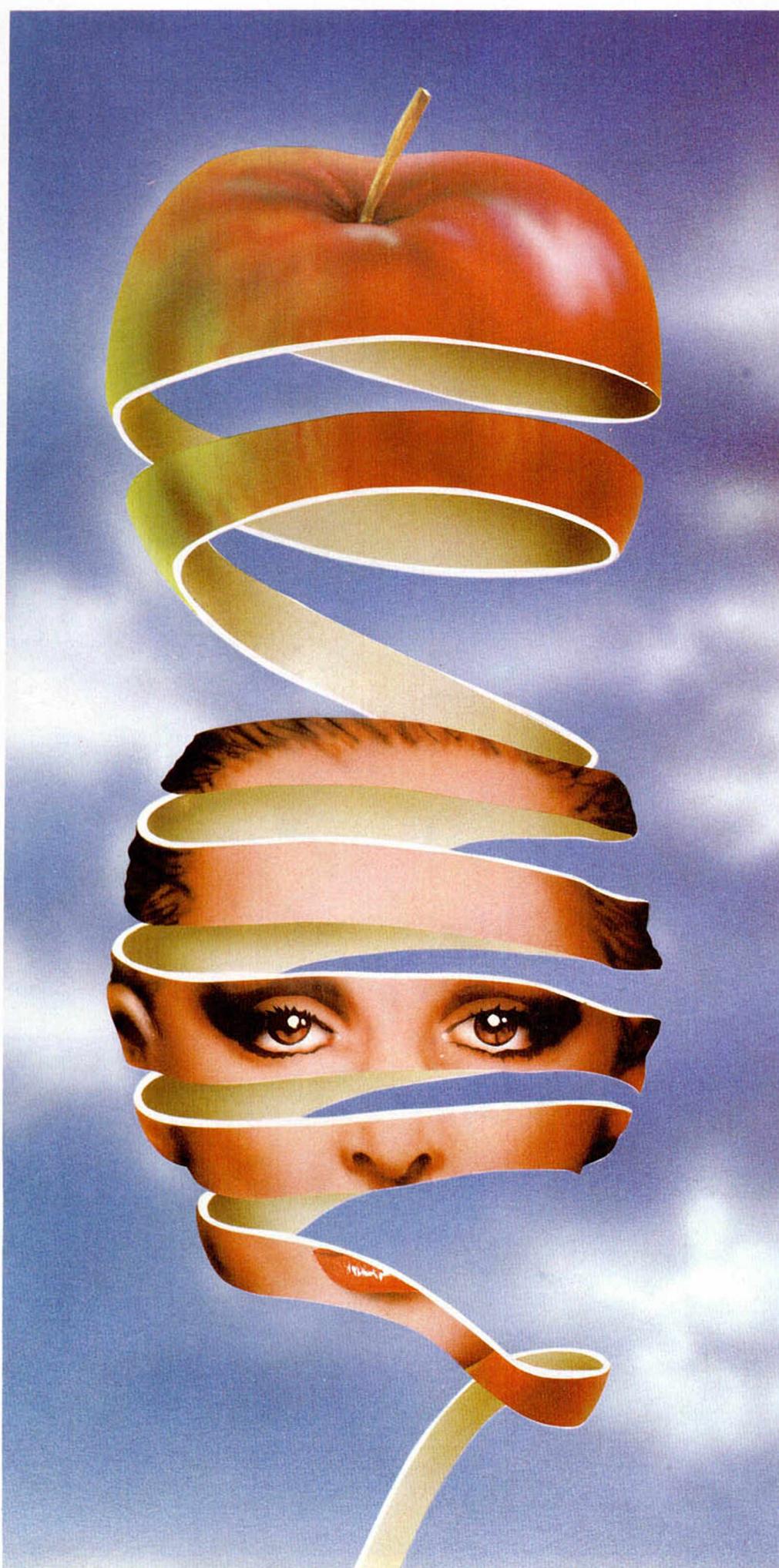
Dispone de un MENU de opciones, que explicaremos

\* \* MENU \* \*  
**M. MENU**  
**D. DIBUJAR**  
**A. ARCHIVAR GRAFICO**  
**C. COPIAR GRAFICO**  
**R. REVISAR GRAFICO**  
**B. BORRAR GRAFICO**  
**I. INSERTAR GRAFICO**  
**P. PINTAR ATRIBUTOS EN BLOQUE**  
**S. SAVE**  
**L. LOAD**  
**V. VERIFY**

bastará pulsar CAPS/BREAK.

**D.**—Esta opción salta a la rutina de dibujo del SUPERGRAFICOS (Melbourne Draw) por lo que aquéllos que no dispongan de este programa deberán efectuar los cambios que comentaremos después.

**A.**—Archiva en memoria bloques gráficos que pueden tener un tamaño comprendido entre un carácter y toda la pantalla. Al pulsar «A» se hace un volcado de pantalla y aparece un cursor formado por un atributo de 1x1 en la parte superior izquierda, el cual lo desplazaremos hasta una de las esquinas del gráfico a archivar por medio de las teclas Q-A (arriba-abajo) y O-P (izquierda-derecha). Entonces se pulsa ENTER



para definir esa esquina, se «enmarca» el gráfico a archivar y se pulsa de nuevo ENTER para finalizar.

El programa nos pregunta entonces: INCLUYE ATRIBUTOS (S/N) para luego informarnos del número de gráfico que le corresponde en el archivo hasta un máximo de 254 (255 gráficos en total) o hasta que la memoria se llene, en cuyo caso aparecerá el mensaje «\*ERROR\*».

La zona de archivo se encuentra justamente después de la rutina de dibujo disponiendo de un total de 17236 bytes lo cual es más que suficiente.

En cualquier momento se puede salir de esta opción pulsando la tecla SPACE.

**C.**—Al entrar se borra la pantalla y el programa nos pregunta: NUM. FIGURA?, debiendo introducir el número de figura que queramos copiar en la pantalla. Si no existe tal figura da un mensaje de error y retorna inmediatamente. En caso de que el gráfico hubiese sido almacenado con atributos nos preguntará si deseamos copiarlo o no con ellos. A continuación hace un volcado de la pantalla de trabajo y una copia en modo normal del gráfico en la parte superior izquierda. El modo de volcado se selecciona con las teclas 1 a 4 como sigue:

1 - normal; 2 - OR; 3 - AND; 4 - XOR.

Basta pulsar la tecla correspondiente para que el gráfico aparezca en el nuevo modo.

El siguiente paso consiste en mover el bloque gráfico a la zona de pantalla donde queremos copiarlo y finalmente se pulsa ENTER. Para abandonar la opción en cualquier momento basta pulsar SPACE.

**R.**—Al usar esta opción

el programa nos pregunta: REVISAR DIRECCIONES O GRAFICOS? La revisión de direcciones nos da varias informaciones: el número de figura (de 0 a 254), el tamaño vertical, el tamaño horizontal, la dirección del gráfico y lo que ocupa en bytes. Si el gráfico contiene atributos informa también de su dirección y longitud. Por último, informa del total de bytes ocupados.

Si elegimos revisión de gráficos éstos se irán mostrando uno a uno en la pantalla así como la posición que ocupa en el archivo.

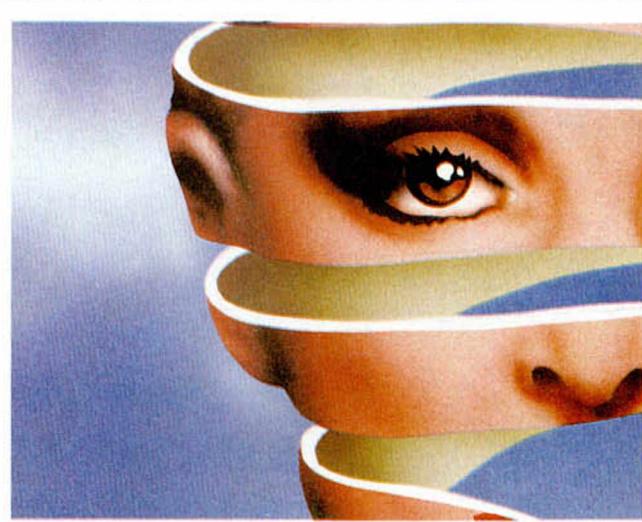
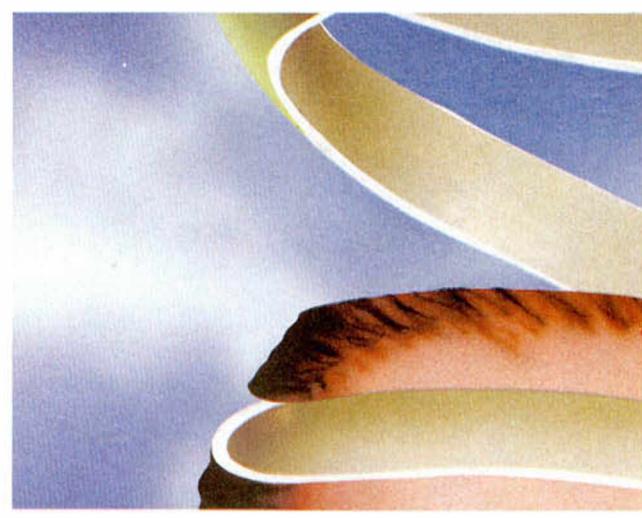
Como siempre basta pulsar SPACE para abandonar.

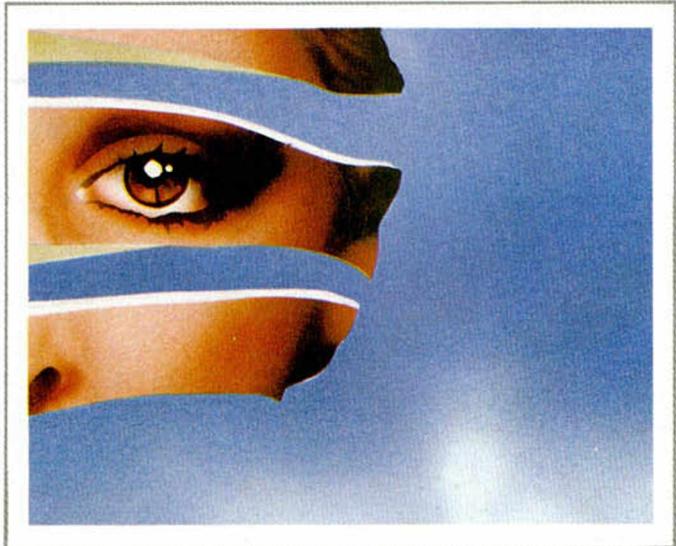
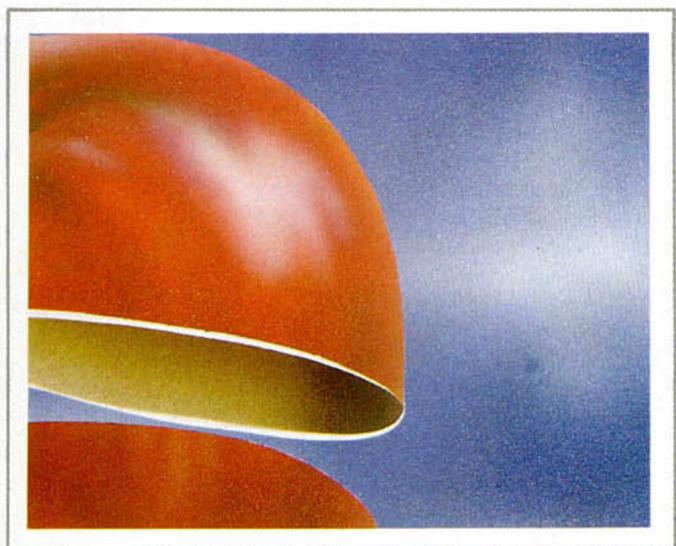
**B.**—El programa nos pedirá el código del gráfico a borrar. En caso de que éste no exista dará un mensaje de error.

**I.**—Esta opción nos permite mover un gráfico desde una posición dentro del archivo a otra, siendo la mayor posición donde se puede trasladar la del último gráfico archivado más uno. El programa nos pedirá la posición de origen y la de destino.

**P.**—Efectúa una copia de la pantalla de trabajo y coloca el cursor en su parte superior izquierda. El primer paso consiste en mover dicho cursor hasta una esquina de la zona a colorear y entonces se pulsa ENTER. A continuación se «enmarca» para luego, con las teclas del 0 al 7, seleccionar el color de TINTA del bloque o con CAPS-SHIFT más la tecla de color del PAPEL. EL BRILLO se obtiene pulsando CAPS-SHIFT+B y el FLASH con CAPS-SHIFT+V. Finalmente se pulsará ENTER.

Esta opción se puede abandonar en cualquier momento pulsando la tecla SPACE





**S.**—Al entrar en la opción el programa pregunta el tipo de datos a salvar: 1. PANTALLA, 2. UDG, 3. GRAFICOS. Pulsando el número seleccionado el programa retornará al BASIC, nos pedirá el nombre a asignar a los datos y el tipo de periférico a utilizar (cinta, microdrive o disco). Como cada usuario tendrá su propio periférico específico (wafadrive en lugar de microdrive, por ejemplo) hemos pensado que la mejor solución era que las opciones de carga, almacenamiento y verificación se hicieran desde BASIC con objeto de que pudierais alterar el programa de acuerdo a cada necesidad siendo las opciones que aparecen a título orientativo.

Los datos salvados tienen las siguientes características: la pantalla que se salva es la que se encuentra en la zona de trabajo (dirección 32768) y los UDG se encuentran en la dirección 47532 y los gráficos. Para diferenciar estos bytes, que son los primeros que se salvan, se añade el token DATA al nombre. A continuación, se salvan los gráficos propiamente dichos.

**L.**—Como para SAVE el programa nos pide el tipo de datos a cargar, luego el nombre y por último, el tipo de periférico que se va a utilizar.

Cuando se van a cargar gráficos el programa activa un banderín interno de forma que al regresar del BASIC éste sepa que debe efectuar la inserción de los nuevos gráficos (el programa debe asegurarse que el número total no exceda de 254 o que no se sobrepase

la memoria disponible). Cuando se produce un error de carga es necesario que dicho banderín se desactive. Para ello el programa recurre a un pequeño truco que detecta si se han cargado con error los datos: antes de efectuar la carga se asigna como línea para CONTINUE un número mayor que la permitida en BASIC (POKE 23662, 255) de forma que si se produce un error SIEMPRE cambia a un número de línea inferior (donde se ha producido el error). Una vez que se entra de nuevo en el programa éste mira si ha cambiado o no este valor aceptando o no los gráficos cargados.

**V.**—Esta opción verifica (salvo para el disco ya que no tiene esa opción) la pantalla de trabajo, los UDG almacenados en la dirección 47532 y los gráficos. Para estos últimos el programa pregunta la posición del primer y último gráfico a verificar (por ejemplo del 15 al 25).

Es preciso aclarar que la información referente a los gráficos que se van a salvar, cargar o verificar se traslada a un BUFFER (con dirección 39680) con objeto de evitar errores durante estos procesos.

El programa en código máquina se almacena en la dirección 29000 y tiene una longitud de 3155 bytes, por lo que debemos realizar el DUMP en la dirección 40000 e indicar 3155 como número de bytes. La tabla de información de los gráficos se encuentra a partir de la dirección 47788 y la dirección de los gráficos propiamente dichos desde la 48300. La dirección de los UDG se cambia automáticamente a la 47532 con objeto de que al crecer los gráficos no sean pisados por é-

tos. La razón de que entre la dirección de los UDG y la tabla de datos haya 256 bytes se ha tomado como medida preventiva para los usuarios que manejen disco debido a que éste (aunque no ocurre con todos) opera con sectores completos de 256 bytes.

Para aquéllos que no dispongan del programa SUPERGRAFICOS (Melbourne Draw) hay dos soluciones (lo sentimos en el alma):

1. Si disponéis de un programa gráfico éste deberá

cumplir los siguientes requisitos: su pantalla de trabajo debe encontrarse en la dirección 32768 y el programa deberá estar entre la dirección 40000 y la 47530 y NO DEBERÁ corromper el contenido de las direcciones superiores a la 29000 (con la excepción hecha).

Si estas condiciones se dieran bastará POKEar en la dirección 29144 la dirección de su programa (SUPERGRAFICOS se ejecuta en la 40960).

2. Ejecutar POKE 29143,24:

POKE 29144,252 con lo que el programa retornará al BASIC y desde allí le podrás mandar a vuestro propio programa de dibujo.

En alguna ocasión puede ser necesario hacer que la pantalla visual pase a ser pantalla de trabajo: para ello bastará efectuar un RANDOMIZE USR 20920. De igual modo, RANDOMIZE USR 31583 nos mostrará la pantalla de trabajo.

Estamos convencidos de que este programa será de gran utilidad y esperamos

encantados vuestras sugerencias.

## LISTADO 1

```

10 RANDOMIZE USR 29000
20 CLS : LET t$=CHR$ PEEK 2367
30 LET o$=CHR$ PEEK 23671
40 LET n$=("Pantalla" AND t$="1") + ("UDG" AND t$="2") + ("Grafico" AND t$="3")
50 GO SUB 100*(o$="S") + 1000*(o$="U") + 2000*(o$="L")
60 RUN
100 REM SAVE
110 GO SUB 5000: IF t$="2" THEN
GO TO 200
120 IF t$="3" THEN GO TO 300
130 IF p$="c" THEN SAVE n$CODE
32768,6912
140 IF p$="m" THEN SAVE *"M";1;
n$CODE 32768,6912
150 IF p$="d" THEN RANDOMIZE USR 15363: REM : SAVE n$CODE 32768,6912
160 RETURN
200 IF p$="c" THEN SAVE n$CODE
USR "a",168
210 IF p$="m" THEN SAVE *"m";1;
n$CODE USR "a",168
220 IF p$="d" THEN RANDOMIZE USR 15363: REM : SAVE n$CODE USR "a",168
230 RETURN
300 LET dg=PEEK 39680+256*PEEK
39681
310 LET lg=PEEK 39682+256*PEEK
39683
320 LET lt=PEEK 39684*2+1
330 IF p$="c" THEN SAVE "DATA
"+n$CODE 39684,lt: SAVE n$CODE d
9,lg
340 IF p$="m" THEN SAVE *"m";1;
"DATA "+n$CODE 39684,lt: SAVE *"m";1;n$CODE dg,lg
350 IF p$<>"d" THEN RETURN
360 RANDOMIZE USR 15363: REM :
SAVE "DATA "+n$CODE 39684,lt
370 RANDOMIZE USR 15363: REM :
SAVE n$CODE dg,lg
380 RETURN
1000 REM VERIFY
1010 GO SUB 5000: IF t$="2" THEN
GO TO 1100
1020 IF t$="3" THEN GO TO 1200

```

```

1030 IF p$="c" THEN VERIFY n$CODE
32768
1040 IF p$="m" THEN VERIFY *"m";
1;n$CODE 32768
1050 RETURN
1100 IF p$="c" THEN VERIFY n$CODE
47532
1110 IF p$="m" THEN VERIFY *"m";
1;n$CODE USR "a"
1120 RETURN
1200 LET dg=PEEK 39680+256*PEEK
39681
1210 LET lg=PEEK 39682+256*PEEK
39683
1220 LET lt=PEEK 39684*2+1
1230 IF p$="c" THEN VERIFY " DATA
"+n$CODE 39684,lt: VERIFY n$CODE dg,lg
1240 IF p$="m" THEN VERIFY *"m";
1;" DATA "+n$CODE 39684,lt: VERIFY *"m";1;n$CODE dg,lg
1250 RETURN
2000 REM LOAD
2010 GO SUB 5000: IF t$="2" THEN
GO TO 2100
2020 IF t$="3" THEN GO TO 2200
2030 IF p$="c" THEN LOAD n$CODE
32768
2040 IF p$="m" THEN LOAD *"m";1;
n$CODE 32768
2050 RETURN
2100 IF p$="c" THEN VERIFY n$CODE
47532
2110 IF p$="m" THEN VERIFY *"m";
1;n$CODE 47532
2120 RETURN
2200 LET dg=PEEK 39680+256*PEEK
39681
2210 IF p$="c" THEN POKE 23663,2
55: LOAD " DATA "+n$CODE : LOAD
n$CODE dg
2220 IF p$="m" THEN POKE 23663,2
55: LOAD *"m";1;n$CODE dg
2230 IF p$<>"d" THEN RETURN
2240 POKE 23663,255: LET r=USR 1
5363: REM : LOAD " DATA "+n$CODE
2250 LET r=r+USR 15363: REM : LO
AD n$CODE dg
2260 IF NOT r THEN RETURN
2270 PRINT FLASH 1; " *ERROR* "
STOP : RUN

```

```

5000 INPUT "Nombre de "+n$+" ";n
$5010 POKE 23658,0: PRINT #0;"Inta, Microdrive o Disco?"
5020 GO TO 5020+(INKEY$="c" OR INKEY$="m" OR INKEY$="d")
5030 LET P$=INKEY$: CLS : RETURN
8999 REM COPIA
9000 SAVE "EDIGRAF" LINE 9100
9010 SAVE "edigraf1" CODE 29000,3
155
9020 SAVE "edigraf2" CODE 40960,6
532
9030 RUN
9100 CLEAR 26999: LOAD "edigraf1"
"CODE"
9110 LOAD "edigraf2" CODE
9120 POKE 31123,0: POKE 31125,0:
CLS : RANDOMIZE USR 30920: RUN

```

## **LISTADO 2**

```

1 3A9379A7C40172ED738F 1299
2 793A9579A72016329679 991
3 3C3295792A7B5C11ACB9 1011
4 ED537B5C01A800EDB0ED 1354
5 7B8F79CD437B11837BCD 1258
6 7B7B1809CD437BCD5F7B 1097
7 CD6B7B217E71E5ED7391 1433
8 79FDCB016E28FAFDCB01 1435
9 AE3A045CFE50CA2673FE 1271
10 49CA3073FE42CA1674FE 1352
11 4D28C0FE52CRA774FE43 1451
12 CA6D74FE56CA8172FE41 1531
13 CA2376FE53CA6272FE44 1428
14 280BFE4C280ACD541F38 807
15 BAE1C9C300A0CD437B11 1379
16 507DCD7B7BCD0C73E106 1219
17 4CC6304FFE33C03E0132 1011
18 93793A9679CDC762200 1156
19 9B01334CC9AF329379FD 1230
20 7E35FFFC03A049B0600 1103
21 4F603A96796F097CA728 955
22 0779953DC832049BCD50 1032
23 72300A3A049B3DC83204 704
24 9B18F13A9679CD17764F 1174
25 3A049B813296792BEB3A 1003
26 049B26006F29444D2105 532
27 9BEDB0C93A049B2A009B 1183
28 E5DD21000021059B47C3 942
29 D875CD437B11467DCD7B 1269
30 7BCD0C73FE0328070653 848
31 C6304FE1C9CDA172E101 1457
32 3353C9CD437B115A7DCD 1167
33 7B7BCD0C73E1FE032807 1107
34 0656C6304FE1C9CDA172 1323
35 E1013356C9CD6B0D1185 1039
36 7DCD7B7BD5CD687ADAA0 1598
37 7632A579D1CD7B7BCD68 1423
38 7ADAA07632A6793E0DD7 1245
39 3AA579577BBADAA0767A 1358
40 CD17762B22A779CDC76 1236
41 22009BE53AA6793CCD17 1051
42 762B22A979CDC76D1A7 1386
43 ED5222029B2AA579247C 998
44 9532049BED5BA7792AA9 1185
45 79ED52444D21059BEBED 1250
46 B0C911647DCD7B7BCDBC 1463
47 763EF7DBFE2FE60728F4 1468
48 FE0530F0CB2F3CC9FD36 1365
49 4701CD1277C3C878CD6B 1241

```

```

50 0D11277DCD7B7BD5CD68 1167
51 7ADAA07632A979D1CD7B 1495
52 7BCD687AF53A96793CCA 1390
53 A076F1CADC73DAA07632 1602
54 AA796F3AA979BDF54FCD 1468
55 CA76E5E5793CCDC76D1 1693
56 A7ED52444DC53AAA79CD 1382
57 CA7622A579E53A9679CD 1403
58 CA76D1A7ED5222A77919 1362
59 545DEBC1C509DAA076EB 1542
60 ED4BA7791B2BEBDB8C1E1 1509
61 ED5BA579F1380109EDB0 1334
62 3AA979CD1776C53AAA79 1240
63 CD17762B5E53A9679CD 1381
64 1776D1A7ED52444D1954 1090
65 5D1B1BEB0BEDB8E1C171 1345
66 23703A96793C329679C9 1058
67 3AA979CD1776C54FCDC76D1A7 1377
68 76E5E5793CCDC76D1A7 1658
69 ED52444DC53A9679CDCA 1397
70 76C1D109DAA076ED42EB 1563
71 EDB03A96793C3296793D 1184
72 CD1776C1702B71C9CD6B 1320
73 0D11197DCD7B7BCD687A 1062
74 DAA0763A96793DBB2839 1170
75 7BF54FCDC76793CE5CD 1587
76 CA76E53A9679CDC76D1 1612
77 A7ED52444DEBD1EDB0F1 1729
78 CD17762B545D2323D5E5 1078
79 EB3A9679CD177623A7ED 1349
80 52444DE1D1EDB03A9679 1403
81 3D329679C9CD6B0D1119 950
82 7DCD7B7BCD687ADAA076 1503
83 CD1776C5CDC76C1AFCB 1639
84 79CBB9280C115D7CCD7B 1123
85 7BCD017730013C329479 876
86 22A579ED43A779CD567B 1326
87 C37778CD437B11A67CCD 1341
88 7B7BCD6B7B3EFDDBFECB 1672
89 67CAB875BC5B7C0CD437B 1486
90 11CE7CCD7B7BFDCB47C6 1523
91 3A967947AF040E141029 670
92 FDCB4786CD0A7511007D 1135
93 CD7B7BD53A9679CDC76 1518
94 11ACBCA7ED52444DCD2B 1256
95 2DCDE32DD1CD7B7BC36B 1484
96 7BCD0A75F5CDCB76F13C 1512
97 18C45F506004FCD1B1A 1011
98 0E06CD0E76F1F5CD1776 1189
99 0595FDCB474628130600 831
100 4ACD1B1A0E0BCD0E7606 700
101 004BCBB9CD1B1A0E10CD 956
102 0E76F1F5D5CDC7622A5 1555
103 79444DCD2B2DCDE32DD1 1245
104 FDCB474628500E18CD0E 974
105 7626006A54CB7BCBBF5 1307
106 CDA93022A779110800CD 974
107 A930E5444DCD2B2DCDE3 1316
108 2DE1F12829F1C1CD975 1517
109 C5F511F17CCD7B7BED5B 1603
110 A57919444DCD2B2DCDE3 1181
111 2D0E18CD0E76ED4BA779 1020
112 CD2B2DCDE32D3E0DD7F1 1301
113 C10DC0F5C5E5CD6B7BE1 1729
114 C1F10E16FD365217C93A 1141
115 9679A7C847AFC5F5CD6B 1638
116 0DF1F5CD1776C5CDC76 1567
117 C1110000CB79CBB9C5F5 1364
118 CDAB79F1C12806110000 994
119 CD167AF1F5CDF875CD6B 1717
120 7BCDDBC76F13CC110CBC9 1548
121 FDCB028626006F3E16D7 1040
122 AFD73E1BD7E5D51E20C3 1393

```

# 38 GRAFICOS

123	301A3E17D779D7AFC310	1096	196	11CB513E6280BCB593E	934
124	0021ACBA06004F09094E	572	197	AE2805CBB93AB57932B5	1198
125	2346C9FD364700CD1277	1026	198	79CB21C93A9479A72807	1099
126	FDCB02C6115D7CCD7B7B	1341	199	ED5B9879C3167ACD1578	1286
127	D5CD017730013C329479	966	200	C32D7ACD58797EE6FEC8	1586
128	CD5876CD420ED1CD7B7B	1356	201	2A98795FED4BA779050D	1026
129	3A967906004F3C329679	795	202	CB4B28077CA7280325CB	699
130	CD1B1AC36B7BCD837721	1171	203	FBCB5328097C80FE1728	1155
131	ACBAC5ED4B9679060009	1153	204	0324CBFBBCB5B28077DA7	1126
132	09C13A9479A72802CBF9	1190	205	28032DCBFBC6328097D	1018
133	712370CB89D5C5CDC776	1580	206	81FE1F28032CCFB2298	1141
134	C1E5E51600596268CDA9	1338	207	79CB23C921977936003E	981
135	30110800CDA930D11938	785	208	BFDBFECB472002CB063E	1435
136	11E1D1C5D5CDC779D1C1	1788	209	FBDBFECB472002CBCE3E	1503
137	3A9479A7C8C3FB79118A	1416	210	FDBBFECB472002CB63E	1513
138	7CED7B9179FDCB02C6D5	1619	211	DFDBFECB4F2002CBDE3E	1499
139	CD420ED1CD7B7B21901A	1148	212	DFDBFECB47002CBCE6C930	1844
140	114000C3B5033E7FDBFE	1122	213	712E71000000000000000000	272
141	1FD811977C18DC3A9679	1112	214	000000000000000000000000	0
142	21ACBCA7C8E5DD210000	1243	215	00000000C5D5CDC5D5CD	1230
143	21ACBA475EBCB7BCBBB23	1307	216	597A0608C5D51A7E1223	840
144	5623E5F526006A54CDA9	1197	217	1C0D20F8D1C11410F1D1	1209
145	30E5110800CDA930D1F1	1174	218	C11410E5C9C5D5CD597A	1485
146	280119EBDD19E110DDDD	1230	219	7AEEC0570608C5D51A77	1208
147	E5D1E119C93E7FDBFEE6	1781	220	231C0D20F9D1C11410F2	1037
148	08C83EFDDDBFEE60237C8	1483	221	D1C11410E2C905622E00	1222
149	18EFCDF5F7B2100002298	905	222	CB3CCB1DCB3CCB1DCB3C	1253
150	79229A79CD157877CD38	1156	223	CB1D165819D1C9E5CDE6	1441
151	78301BED5B9A79229A79	1107	224	797CEEC05706D1C5E57E12	1557
152	010101CD3F7ACD1578ED	976	225	2C130D20F9E10120009	624
153	5B9879CDE6793A9D7977	1375	226	C110EFC9E5CDE679D1C5	1840
154	CD8C76CD567B3A9779CB	1458	227	E51A772C130D20F9E101	957
155	4728D3CD2F782A987922	1043	228	200009C110EFC9CDE679	1246
156	9E79CD38783006CD8377	1169	229	51E5772C1520FBE11120	1051
157	CD3F7AED5B9E792A9879	1312	230	001910F2C9CDE67951E5	1350
158	229A79CD8A77CD9E77CD	1458	231	7CEEC0675EEE067732C	1443
159	BC76CD567B3A9779CB47	1324	232	1520F3E11120001910EA	845
160	28D8C32F78ED5B9E792A	1267	233	C97AE6070F0F0FB35F7A	1001
161	9A797CBA3001EB7DBB30	1229	234	E618F64057C9CD967A11	1346
162	026B5F7C923C477D933C	937	235	000021009B7EFE0D2814	641
163	4FC9FDCB47462803CDAD	1298	236	D630E5F5210A00CDA930	1201
164	773A9D79C32D7AD90E00	1048	237	F116005F19E8E12318E7	1133
165	3EEFDBFE2FE619280DCB	1332	238	7AA737C03A96794F7BB9	1252
166	472017CB5F0E0720110D	507	239	3FC9AF32A47932A279CD	1312
167	180E3EF7DBFE2FE61F28	1168	240	267BCD767B20FBFD36CA	1399
168	210CCB2F30FB3EFEDBFE	1383	241	FF21009B22A079CDC27A	1279
169	CB2F3A9D79380B8E6C7CB	1285	242	2AA079773E0232A479CD	1046
170	21CB21CB21B11803E6F8	1187	243	267BCD3A7BC9CD267BFD	1367
171	B1329D790E003EFEDBFE	1308	244	CB016E28F7DCB01AE3A	1290
172	CB4720192F4FCB213E7F	882	245	085CFE0DC8FE0C282CFE	1171
173	DBFE2FB1CB27CB27E6C0	1603	246	3038E5FE3A30E1083AA2	1146
174	4F3A9D79A9329D7976D9	1247	247	79FE0328D9082AA07977	1085
175	C9ED5B9879CDE679E57C	1711	248	2322A079D7CD3A7B3AA2	1171
176	EEC0677EE6203E382802	1081	249	793C32A2792806AF32A4	949
177	3E07329D79E1C9CD5879	1237	250	7918BD3E20D73E08D7CD	1133
178	7EE60120F8C9CD58797E	1378	251	3A7B3AA279A728EB3D32	1075
179	E6FEC82A98794FCB4928	1394	252	A2793E08D72AA0793620	977
180	077CA7280325CBF9CB51	1114	253	2B22A07918D92AA37923	960
181	28087CFE17280324CBF9	980	254	22A379CB4C3E5F28023E	858
182	CB5928077DA728032DCB	922	255	20D73E08D7C921C80011	983
183	F9CB6128087DFE1F2803	1050	256	1E00C3B503CD6B0DFDCB	1190
184	2CCBF9229879CB21C921	1273	257	02863A485CCB2FCB2FCB	1061
185	0002298793E7E32B579	847	258	2FD3FEC090188130B78B1	1177
186	180ACDD4783805CD1179	975	259	20FBC921008011004001	727
187	3016CD5F7BED5B98792A	1136	260	001BEDB0C9CD757B20FB	1370
188	A579ED4BA779C5CDAB79	1580	261	CD767B28FBC9CD8E021C	1315
189	C1CDFE78CD8C763EBFDB	1755	262	C91A13FEFFC8D718F817	1465
190	FE1F38D6CD5F7BED5B98	1458	263	08002A202A204D454E55	465
191	792AA579ED4BA779C5CD	1451	264	202A202A16044D2E20	333
192	AB79C13A9479A7C40479	1300	265	4D454E550D0D17040044	430
193	21004011008001001BED	507	266	2E20444942554A41520D	604
194	B0C93EF7DBFE4FCBF9CB	1893	267	0D170400412E20415243	397
195	413E7E2817CB493EB628	876	268	48495641522047524146	698

269	49434F0D170400432E20	404	293	4414004952454343494F	598
270	434F5049415220202020	574	294	4E4553204F2014014714	485
271	20207E0D170400522E20	390	295	0052414649434F533FFF	837
272	52455649534152202020	636	296	14014649472E20564552	550
273	20207E0D170400422E20	374	297	2E20484F522E20444952	612
274	424F5252415220202020	584	298	4543432E202054414D2E	585
275	20207E0D170400492E20	381	299	14000D0DFF0D41747269	714
276	494E5345525441522020	680	300	6275746F733A171000FF	909
277	20207E0D0D170400502E	369	301	0D0D544F54414C204F43	592
278	2050494E54415220204154	675	302	555041444F20FF204259	851
279	52494255544F530D1707	595	303	5445532EFF4E554D2E20	855
280	00454E20424C4F515545	635	304	4649475552413F20FF49	869
281	0D0D170400532E205341	362	305	4E53455254415220454C	720
282	56450D1704004C2E204C	425	306	20FF0D0D454E204C4120	665
283	4F414440170400562E20	416	307	504F534943494F4E20FF	899
284	564552494659FF160100	747	308	161100140120F81400FF	615
285	494E434C555945204154	718	309	161100140120EF1400FF	606
286	52494255544F533F2028	687	310	161100140120D61400FF	581
287	532F4E29FF0D47524146	805	311	0D0D312E2050414E5441	525
288	49434F204E554D45524F	721	312	4C4C410D322E20554447	582
289	20FF0D12012A4552524F	673	313	0D332E20475241464943	570
290	522A1200FF0D12012041	526	314	4F53FF44455344452045	875
291	4E554C41444F201200FF	756	315	4C20FF0D0D4841535441	758
292	52455649534152201401	593	316	20454C20FF03CDCF7DD1	1213



# 40 PROGRAMACION

por Agustín CONDE MARUGAN

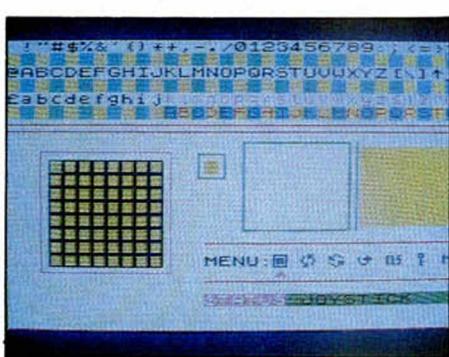
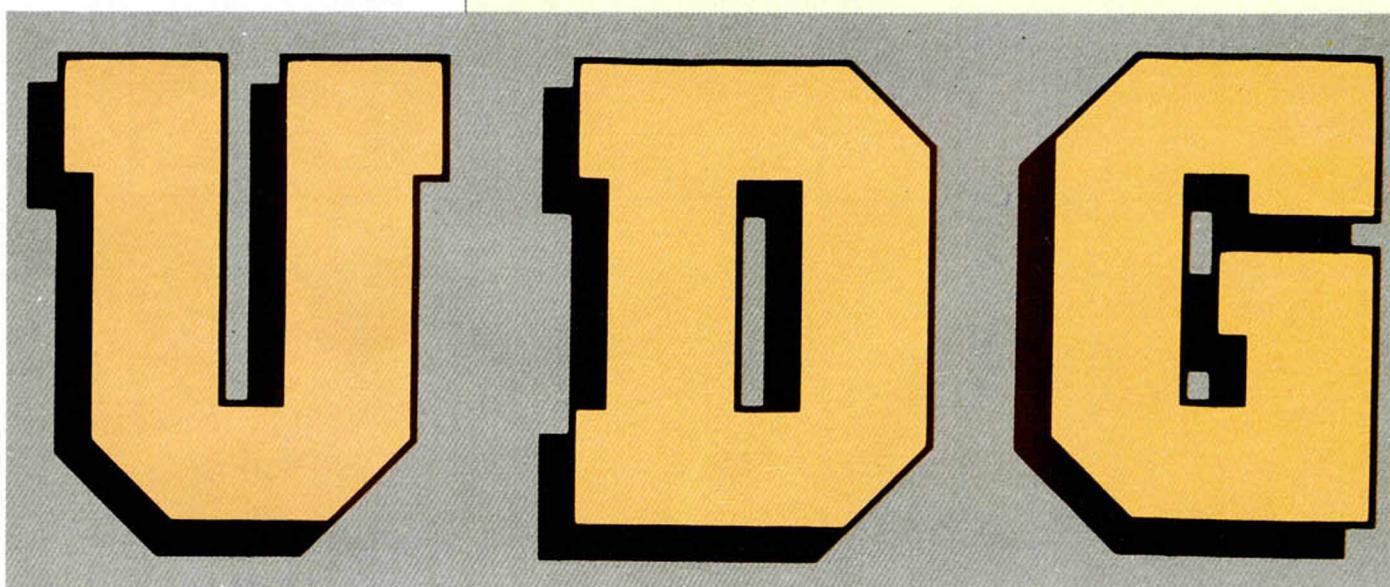
Los UDG son la base imprescindible de cualquier representación gráfica en la pantalla de un ordenador. Con ellos podemos crear multitud de imágenes que irán dando forma a las figuras que más tarde se convertirán en los protagonistas de nuestros juegos.

## A/ OBJETIVOS

### Generador de UDGs

Este programa puede generar directamente sobre la memoria un juego de UDGs, que es instalado además inicialmente a partir de la dirección 65368 en la que, naturalmente, están establecidos. Por supuesto, una vez almacenados en forma de Bytes, apar-

tanto como sustitutivo personal a los feos caracteres de la ROM como útil repertorio de UDGs de 96 caracteres frente a 21, con la posibilidad estando posibilitados de ser identificados por la función SCREEN\$ para los juegos, es algo de gran interés. El programa salvará el nuevo juego como Bytes instalados a partir de la dirección 64768, por lo que si no es reubicado para ser utilizado deberá pokearse en la variable CHARS (23607—23606) 252 y Ø respectiva-



te de poderse cargar con un simple LOAD "" CODE, pueden ser reubicados en nuevas direcciones para disponer de varios juegos de UDG una vez pokeado en la variable del sistema UDG (23676—23675) la nueva dirección. A la hora de generar un UDG, se debe tener en cuenta que los últimos 21 caracteres (desde la k minúscula al símbolo copyright) que aparecen en tinta magenta en la línea 5.º de la pantalla, equivalen a los 21 UDGs.

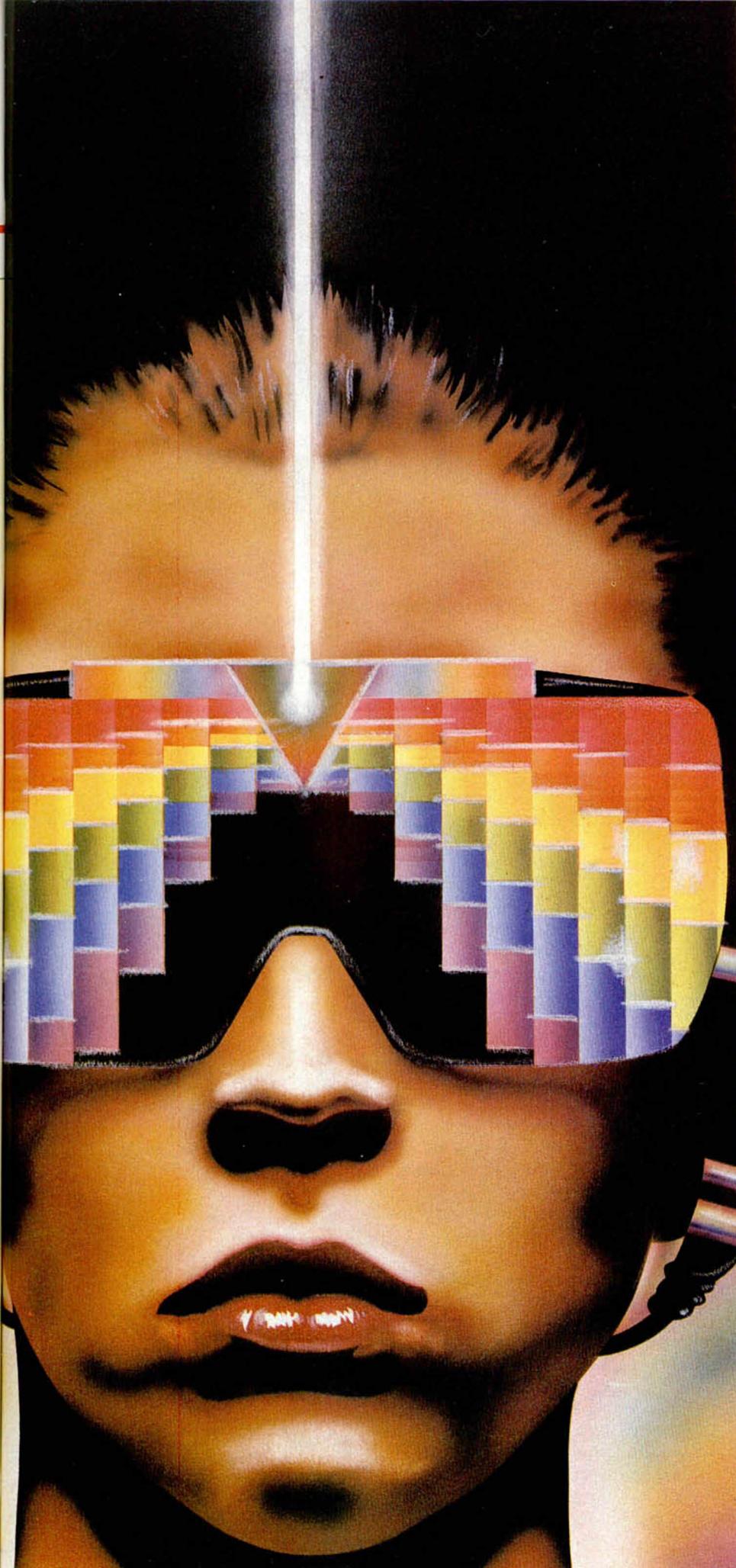
### Generador de un nuevo juego ROM

El generar un nuevo juego ROM,

mente, para reubicarlo basta recordar que dicha variable contiene la dirección de comienzo—256.

### Funcionalidad máxima para simplificar la tarea

La utilidad dispone de un amplio repertorio de posibilidades en el menú, reductoras en todo lo posible del esfuerzo que pueda suponer la tarea, permitiendo, sobre todo a los sufridos programadores de juegos propios, manipular y seguir los resultados de la tarea de una forma práctica y racional.



## B/ UTILIZACION DEL PROGRAMA

Existen dos menús de trabajo: menú 1 y menú 2.

### MENU I:

#### **Selección dentro del menú 1**

Las opciones del menú 1 están representadas por un conjunto de gráficos autoexplicativos que seleccionan, dispuestos a continuación de la palabra menú y entre dos rayas. Para seleccionar una opción dentro del menú habrá que pulsar SPACE y una vez elegida la que interesa, presionar ENTER.

#### **Control Joystick o teclado**

Es en esta situación en la que se comienza, accediéndose a ella normalmente desde el menú con el símbolo de un cuadrado enmarcado. Aquí aparecerá un cuadrado parpadeante a mover a través de la pizarra amarilla de la izquierda mediante el Joystick, cuando se presione el disparo, el cuadrado sobre el que estaba pasará a estar activado si no lo estaba, y a paper si estaba activado, configurándose así uno a uno los pixeles del gráfico. Una vez terminado el gráfico con SPACE saldrá al menú.

#### **Espejo horizontal**

Función representada por dos flechas horizontales, una contraria a la otra. Su finalidad es reordenar el gráfico de izquierda a derecha en sentido contrario al que se encontraba.

#### **Espejo vertical**

Las flechas en este caso son verticales, al igual que la reordenación.

# 42 PROGRAMACION

## Rotación a la derecha

Función representada por una única flecha. Gira al gráfico en el sentido de las manecillas del reloj.

## cls

Función representada por la palabra cls escrita en un mismo gráfico. Borra en tinta o en papel la pizarra del gráfico a conveniencia del nuevo gráfico a elaborar (requiere o no mucha tinta).

## Archivo en memoria

Es representada esta función por una llave. Su finalidad es introducir en la memoria el gráfico impreso en ese momento en la pizarra Iz. Para ello se pedirá que se indique a qué gráfico del juego de la ROM sustituirá en el que se está creando, en el caso de que lo que estemos haciendo sea un juego de UDGs, entonces se selecciona a qué UDG sustituirá, teniendo en cuenta que el UDG A es considerado como el primer carácter magenta de la fila 5.<sup>a</sup> (la k minúscula) y los demás sucesivamente detrás de éste. Al archivarse en memoria el gráfico se mostrará en la pizarra los números que deberíamos introducir en DATAS si, a pesar de todo, decidimos que no archivaremos en cinta el trabajo, y preferimos pokearlo en el programa que lo utilice.

## Acceso al menú 2

Es representado por una M y un 2. Conduce al menú 2, que a su vez dispone de 4 nuevas opciones a elegir presionando SPACE y después ENTER. A menos que aparezca el mensaje «REGRESO A MENU PRINCIPAL» debemos entender que permanecemos en él.

## MENU 2:

### Scrolls

Se elige el que necesitemos indicado por la flecha con SPACE, y presiona-

mos después ENTER, con lo que el gráfico impreso en la pantalla será scrollado, al igual que su copia a tamaño real. Si seleccionamos la M retornaremos al Menú principal, o sea, al 1.

## Sacar carácter a pantalla de trabajo

Esta opción sirve para retocar un gráfico que ya hayamos introducido en la memoria, o para hacer uno a partir de él. Se imprimirá el que seleccionemos en la pizarra de la izquierda desde la memoria, permitiéndonos actuar de nuevo sobre él con el Joystick u otra opción del menú. Una vez retocado deberemos recurrir de nuevo a la llave para introducirlo en memoria, encima del antiguo (que sigue archivado en la memoria tal como antes) o en cualquier otro sitio.

## Panel derecha

Sí varios de nuestros gráficos forman parte de una única forma, por ejemplo, si hacemos un hombre de 3 por 4 gráficos, será necesario realizar a menudo numerosos ajustes en los gráficos para que la pierna y el pie estén unidos por el sitio justo, y el brazo y la mano, al verlos juntos, en verdad parezcan lo que son. Estos retoques serán mucho más llevaderos al poder ir colocando gráficos en el panel de la derecha., ver cómo va el asunto y comprender al instante que, por ejemplo, al gráfico de la mano hay que scrollarle hacia abajo para que coincida con el del brazo.

Para sacar el máximo rendimiento al panel podemos realizar las siguientes operaciones:

**Situar.** Señala con el Joystick el lugar donde situar el gráfico ya programado en memoria. Entonces disparar y se nos preguntará qué cadena se debe situar allí. Para salir presionar SPACE.

**cls.** Borra el panel.

**Mem.** Permite archivar y sacar de memoria paneles. Ello puede ser útil por si se retocan o reelaboran varios gráficos del panel, pero al fin y al cabo para la memoria del panel siguen

siendo los mismos, por lo que en vez de tener que volver a situarlos encima de ellos mismos, bastará con mandar pintar el panel de memoria. Pero el interés de esta función está centrado aun así, en el caso de haber interrumpido de un día a otro nuestra labor, para que no tengamos que construir un rompecabezas cada que vez que continuemos.

**Menú.** Retorna al menú principal.

## Operaciones con cinta

**Salvar UDGs.** Graba en cinta los 21 UDGs que equivalen, como ya he explicado, a los 21 últimos caracteres en tinta magenta de la línea 5.<sup>a</sup>, desde la k minúscula hasta el copyright.

**Salvar juego paneles.** Graba la memoria de los 9 paneles en caso de que vayamos a continuar otro día.

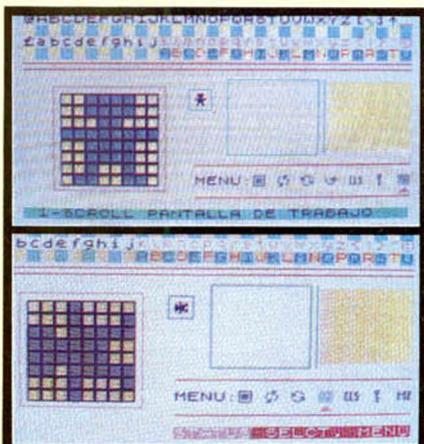
**Cargar.** Con las opciones opuestas a las anteriores en caso de reanudar la tarea iniciada días atrás.

```
10 CLEAR 64600: PRINT AT 10,0: FLASH 1: PAPER 6; INK 3;" SUPE R GENERADOR DE GRAFICOS ". PRI NT INK 1,AT 11,0;" @ AGUS par a MICROHOBBY ". PRINT AT 21,0; INVERSE 1;" " . esperar un se gundo : POKE 23675,88: POKE 23676,252: FOR f=64600 TO 64711: READ a: POKE f,a: NEXT f: FOR f=64768 TO 65367: POKE f,r: NEXT f
15 DATA 255,129,129,129,129,12 9,129,255,129,189,189,189,18 9,129,255,14,76,74,129,129,82,50 ,112
16 DATA 24,166,193,224,7,131,1 01,24,0,4,142,159,132,68,56,0,0, 211,146,147,145,145,219,0,56,40, 56,16,16,24,16,24,0,139,217,171, 138,138,139,0,0,0,8,28,62,127,0,0
17 DATA 0,4,6,255,6,4,0,0,0,0,32
96,255,96,32,0,0,16,56,124,16,1 6,16,16,16,16,16,16,16,124,16,16 ,16,0,137,219,169,137,137,137,0
18 REM [REMOVED]
19 CLS : BEEP ,2,0: BEEP ,5,20 : PRINT AT 19,3;" Selecciona TECL AS: ". PRINT AT 21,1; PAPER 5;" US A UN JOYSTICK SI LO TIENES ". INP UT "K?";q$: "J?";w$: "L?";e$: "M ?";r$: "B?";t$: LET q=CODE q$: LET w=CODE w$: LET e=CODE e$: LET L=CODE L$: LET r=CODE r$: LET t=CODE t$ 
20 REM [REMOVED]
21 CLS : INK 2: DRAW 255,0: PL OT 112,54: DRAW 143,0: PLOT 112,31: DRAW 143,0: PLOT 0,120: DRAW 255,0: PLOT 0,124: DRAW 255,0: BRIGHT 1: INK 3: PLOT 19,108: DR AW 73,0: DRAW 0,-72: DRAW -73,0: DRAW 0,72: BRIGHT 0: INK 4: PLOT 108,107: DRAW 15,0: DRAW 0,-14: DRAW -15,0: DRAW 0,14: PLOT 13
22 FOR f=8 TO 13: PRINT BRIGHT 1,AT f,17;" ". PAPER 6,AT f,25;" ". NEXT f: GO SUB 9200
25 LET a=0: GO SUB 1020
40 REM [REMOVED]
42 LET c=1: LET me=0: DIM x$(4 ,32): DIM a$(9,42)
```

```

45 LET X$(1) = " 1-SCROLL PANTAL
LA DE TRABAJO" : LET X$(2) = "2-L
LEVAR CARCT A PANT-DE TRABAJO"
LET X$(3) = " 3-UTILIZAR PANEL DC
ACIONES CON CINTA" : LET X$(4) = " 4-OPER
ACIONES CON CINTA
50 GO TO 5000
100 REM JOYSTICK
110 LET h=3: LET v=9: PRINT AT
19,20; PAPER 4; " JOYSTICK "
120 LET m=CODE INKEY$
175 IF m=q THEN LET h=h-(1 AND
h>3): GO TO 200
183 IF m=w THEN LET h=h+(1 AND
h<10): GO TO 200
187 IF m=r THEN LET v=v+(1 AND
v<16): GO TO 200
189 IF m=e THEN LET v=v-(1 AND
v>9): GO TO 200
190 IF m=t THEN LET c=ATTR (v,h
)/8: PLOT OVER 1;109+h,112-v: LE
T c=(5 AND NOT ((c-1))+1: PRINT A
T v,h; PAPER c; "B": BORDER c: BE
P 4,0: BORDER 7
192 IF m=32 THEN RETURN
195 PRINT PAPER 8; INK 4; AT v,h
;"B": BEEP .008,-9: GO TO 250
200 PRINT INK 4; AT v,h; PAPER 8
;"B": BEEP .07,-9: GO TO 250
250 PRINT AT v,h; PAPER 8; "B": GO TO 120
400 REM ESPEJO LATERAL
410 PRINT AT 19,20; PAPER 4; " E
SPEJ-VERTC": GO SUB 1025
420 FOR f=112 TO 119: FOR n=96
TO 103: IF POINT (f,n) THEN PRIN
T PAPER 1; AT n-87,f-109;"A"
430 NEXT n: NEXT f: GO SUB 9000
RETURN
600 REM ESPEJO LATERAL
610 PRINT AT 19,20; PAPER 4; " E
SPEJ-HORIZ": GO SUB 1025
620 FOR f=112 TO 119: FOR n=96
TO 103: IF POINT (f,n) THEN PRIN
T PAPER 1; AT 112-n,122-f;"A"
630 NEXT n: NEXT f: GO SUB 9000
RETURN
800 REM ROTACION
810 PRINT AT 19,20; PAPER 4; " R
OTACN-DCHA": GO SUB 1025
820 FOR f=112 TO 119: FOR n=96
TO 103: IF POINT (f,n) THEN PRIN
T PAPER 1; AT 128-f,106-n;"A"
830 NEXT n: NEXT f: GO SUB 9000
RETURN
1000 REM CLS
1010 PRINT AT 19,20; PAPER 4; " C
LS": BEEP .3,20: INPUT F
LASH 1;" F en "Paper=0": FLASH 0
;"F": FLASH 1;" F en tinta=1": FL
ASH 0;"F": a
1020 PRINT AT 9,14; INVERSE a; PA
PER 6; INK 0; "
1025 LET h=3: LET v=9: FOR n=9 T
O 16: PRINT ; PAPER 6-(5*a); AT n
,3;"AAAAAAA": NEXT n: LET a=0:
RETURN
1200 REM MEMORICHE
1210 PRINT AT 19,20; PAPER 4; " AL
MACEN GRAF": INPUT ; FLASH 1;"a
que caracter sustituye?": FLASH 0;
;"a": BEEP .03 3: LET m=CODE
a: IF m<32 OR m>127 THEN GO TO
1210
1215 LET d=m*8+64512
1220 FOR n=103 TO 96 STEP -1: LE
T P=0: FOR f=112 TO 119: IF POIN
T (f,n) THEN LET P=P+2^(119-f)
1230 NEXT f: POKE d,p: LET d=d+1
: PRINT AT 112-n,3; PAPER 3; "
"; AT 112-n,6,p: NEXT n
1240 POKE 23607,252: PRINT PAPER
5+(1 AND m/2=INT (m/2)): INK 1+
(1 AND m/2=INT (m/2)): AT INT (m/
32)*2-1,m-INT ((m/32)*32,a): POKE
23607,60: BEEP .4,20: PAUSE 0:
BEEP .2,8: GO SUB 9100: RETURN
1300 REM MENU 3
1310 PRINT AT 19,0; PAPER 1; INK
5; FLASH 1;"ABIERTO MENU-2. SELE
CCIONA OPCION": BEEP .01,1: BEEP
.02,-8: LET o=0: IF INKEY$()>" "
THEN GO TO 1310
1320 IF INKEY$() = " " THEN LET o=1+
(o AND o<4): FOR f=1 TO 32: PRIN
T PAPER 0+3; BRIGHT 1; AT 19,32-f
;x$(o)(1 TO f): BEEP .006,-3: NE
XT f
1330 IF CODE INKEY$=13 THEN BEEP
.2,3: GO SUB 2000+100+o: PRINT
AT 19,0; FLASH 1;" REGRESO A
MENU PRINCIPAL": BEEP 1,20:
PAUSE 50: BEEP .3,0: PRINT AT 19
,0;" "; INK 7; PAPE
R 3;"STATUS": RETURN
1340 GO TO 1320
2100 REM FIN

```



```

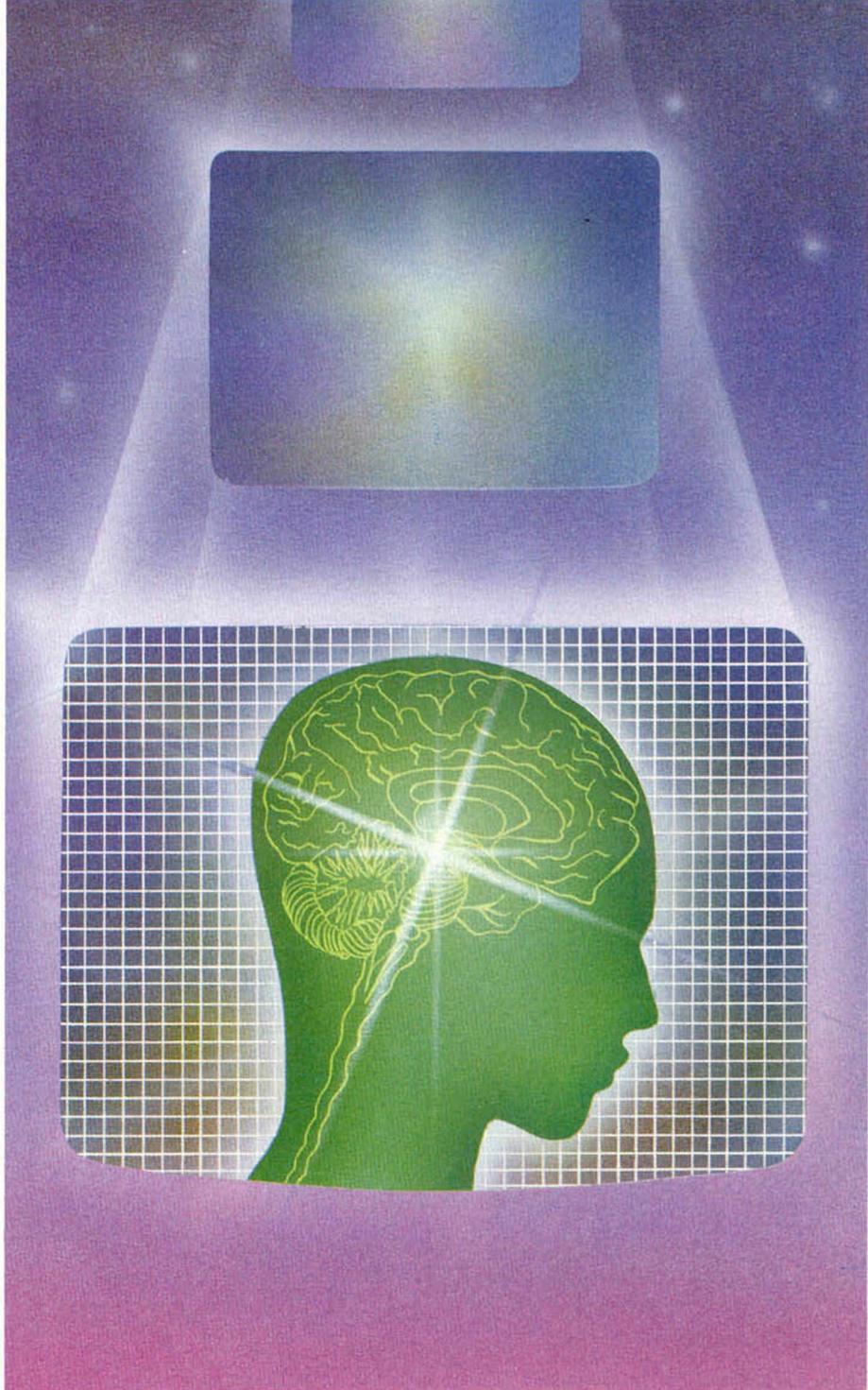
3735 GO TO 2310
3780 RETURN
2400 REM OPERACIONES CON CINTA
2410 INPUT " "; FLASH 1; "SAL
UAR=1"; FLASH 0; " "; FLASH 1; "C
ARGAR=2"; FLASH 0; " "; C: INPUT F
LASH 1; "UDGS=1 J.ENTERO 2 PANEL
S=3"; FLASH 0; " "; M: LET C=M+(3
AND (C-1))
2421 IF C=1 THEN INPUT PAPER 5;
NOMBRE JUEGO DE UDG'S?; C$: IF LEN
N C$<11 THEN SAVE C$CODE 6476
68: RETURN
2422 IF C=2 THEN INPUT ; PAPER 7
;"NOMBRE JUEGO DE CART."?; C$: IF LEN
C$<11 THEN SAVE C$CODE 6476
68: RETURN
2423 IF C=3 THEN INPUT PAPER 2;
NOMBRE J. DE PANELES?; C$: IF LEN
C$<11 THEN SAVE C$ DATA a(): RETURN
2432 IF C=4 OR C=5 THEN INPUT "N
OMBRE DEL J. DE CARCT?"; C$: PRIN
T AT 19,0; PAPER 6; INK 3; FLASH
1; "PON EN MARCHA EL CASSETT
E": LOAD C$CODE : PRINT AT 21
,0; "PLOT 0": PLOT 0
0: DRAW INK 2;255,0: GO SUB 922
6: RETURN
2434 IF C=6 THEN INPUT "NOMBRE D
EL J. DE PANELES"; C$: PRINT AT 1
9,0; PAPER 6; INK 3; FLASH 1; "
PON EN MARCHA EL CASSETTE"
: LOAD C$ DATA a(): PRINT AT 21,0
; "PLOT 0": PLOT 0,0: DRAW
INK 2;255,0: RETURN
2440 GO TO 2421
4999 REM MENU 1
5000 PRINT AT 16,14;"MENU": INK
1;"B C D E F G H"; AT 17,19; INK
3;"I"; INVERSE 1; AT 19,14;"STAT
US": GO SUB 100
5005 BEEP .2,13: BEEP .3,-3: PRI
NT AT 19,20; PAPER 2; INK 7; SE
LCT, MENU"
5010 IF INKEY$=" " THEN PRINT AT
17,19+ME: " "; AT 16,19+ME: OVER
1; INK 1; " "; BEEP 2,9: LET ME=
ME+2 AND ME<12: PRINT AT 17,19+ME
: INK 3;"I"; AT 16,19+ME: OVER 1
; FLASH 1; INK 1;
5020 IF CODE INKEY$=13 THEN GO S
UB (ME+2)*100-100: GO TO 5005
5025 GO TO 5010
9000 REM Print respectivo GRANDE
9010 PRINT AT 9,14; PAPER 6; "
FOR F=3 TO 10: FOR N=9 TO 16: I
F ATTR (N, F)=8 THEN PLOT F+109,1
12-N
9015 NEXT N: NEXT F: RETURN
9100 REM Print respectivo PEQUEÑO
9110 GO SUB 1025: FOR F=112 TO 1
19: FOR N=96 TO 103: IF POINT (
F, N) THEN PRINT AT 112-N, F-109; P
APER 1;"A"
9120 NEXT N: NEXT F: RETURN
9200 REM Print respectivo MEDIO
9210 BRIGHT 1: FOR F=32 TO 127:
PRINT INK 3 AND (F-106>0); AT INT
(F/32)*2-2, F-INT (F/32)*32; CHR$(
F: NEXT F
9220 BRIGHT 1: POKE 23607,252: F
OR F=32 TO 127: PRINT AT INT (F/
32)*2-1, F-INT (F/32)*32; PAPER 5
+(1 AND F/2=INT (F/2)); INK 1+(1
AND F/2=INT (F/2)); CHR$(F: NEXT
F: POKE 23606,0: POKE 23607,50:
BRIGHT 0: RETURN

```

Carlos BELLVER

# Ampliación del Basic

ESTE PROGRAMA, ESCRITO INTEGRAMENTE EN CODIGO MAQUINA, PERMITE USAR DOCE NUEVOS COMANDOS DENTRO DE UN PROGRAMA BASIC EN EL SPECTRUM 48 K.



**E**stos nuevos comandos deberán escribirse carácter a carácter (no importa si se hace en mayúsculas o en minúsculas) tras el símbolo &. Para usarlos en un programa, la primera línea de éste habrá de ser similar a la siguiente:

10 CLEAR 63999: LOAD "CODE: RANDOMIZE USR 64000

Si posteriormente se hace RUN, NEW o CLEAR, tendremos que ejecutar otro USR 64000 para poder volver a usar las nuevas instrucciones del EXTBASIC.

## LOS NUEVOS COMANDOS

Son los siguientes:

### &REPEAT

Se usa conjuntamente con &UNTIL para crear un bucle que se ejecutará hasta que se cumpla la condición que sigue a &UNTIL. Por ejemplo:

### EJEMPLO 1

```
10&REPEAT
20 PRINT AT RND*21,RND*31;"*"
30&UNTIL, INKEY$="K"
```

Dibujará asteriscos en la pantalla hasta que se pulse la tecla «K».

Se pueden anidar los bucles &REPEAT del mismo modo que se hace con los FOR-NEXT, hasta un límite de ocho anidaciones.

&REPEAT ha de ser la última instrucción en una línea, y sólo se puede usar dentro de un programa, nunca en modo directo, es decir, sin un número de líneas. Si se hiciera esto no se ejecutaría nada.

### &CLR

Cuando el usuario pulsa BREAK dentro de un bucle REPEAT, la pila de REPEAT (la zona de memoria en que se almacenan los números de línea a que ha de saltar UNTIL) no se borra, y si esta acción se efectúa varias veces, la pila acabará por llenarse y aparecerá un mensaje de error. Entonces habrá que usar &CLR, que borra la pila de REPEAT.

### &SCREEN, num

Esta instrucción pone el BORDER y el PAPER al color indicado por la ex-

presión «num» y el INK al color que mejor contraste con éste. También pone a cero el FLASH y el BRIGHT. Resulta más rápido que BORDER num:PAPER num:INK 9:BRIGHT 0:FLASH 0:CLS, cuyo efecto es equivalente al de &SCREEN, num.

### &RECOL, paper, ink

Cambia los atributos a los indicados por las expresiones «paper» e «ink», pero no altera lo que haya dibujado en la pantalla. Ejemplo:

### EJEMPLO 2

```
100&SCREEN,0
110 FOR I=0 TO 703: PRINT CHR$(32+INT(RND*96)): NEXT I
120 FOR I=0 TO 15
130 FOR J=0 TO 7
140&RECOL,0,J
150 NEXT J: NEXT I
160&SCREEN,0
```

### &SCROLL

Desplaza la pantalla una línea hacia arriba, lo cual es útil en juegos sencillos o en presentaciones como ésta:

### EJEMPLO 3

```
100 PRINT AT 21,0;"EXTBASIC V1
.9"
110 FOR I=0 TO 21
120&SCROLL: BEEP .1,I
130 NEXT I
```

### &CLSLOW

Borra la parte inferior de la pantalla, normalmente las dos últimas líneas, en las que se puede escribir mediante PRINT #0, o PRINT #1. Por ejemplo:

### EJEMPLO 4

```
100 FOR i=32 TO 255
110&CLSLOW: PRINT #0;"Pulsa 'c'
para ver el CHR$ ";i
120&REPEAT
130&UNTIL, INKEY$="c"
140 PRINT CHR$(i);
150 NEXT i
```

### &SOUND, f1, f2, step, dur

Produce un sonido de frecuencia «f1» (0-65535) y duración «dur» (0-65535). Suma «step» a «f1» y si el resultado es menor o igual a «f2» repite el proceso. Pueden obtener algunos efectos bastante buenos:

```
&SOUND,100,200,1,8
&SOUND,400,500,1,4
&SOUND,100,500,1,16
```

### &NOISE, dur

Produce ruido durante un tiempo «dur». Cuando «dur» vale más de 8000 los resultados no son muy aceptables. Ejemplo:

### EJEMPLO 5

```
100 BORDER 2:&NOISE,60
110 BORDER 1:&NOISE,100
120 BORDER 4:&NOISE,40
130 BORDER 7
```

### &WAIT, dur

Detiene la ejecución del programa, como PAUSE, durante «dur/50» segundos. Al contrario que PAUSE no sigue la ejecución cuando se pulsa una tecla. Por ello &WAIT,0 espera más de veinte minutos...

### &MOV, numbytes, org, dest

Copia un bloque de bytes de longitud «numbytes» en la dirección «org» a la dirección «dest». Su utilidad más inmediata es la de guardar pantallas en memoria y recuperarlas, pero se le pueden encontrar muchas otras.

El ejemplo muestra un caso de almacenamiento de pantallas sin atributos:

### EJEMPLO 6

```
100 CLEAR 26999: RANDOMIZE USR
64000
110&SCREEN,5
120 FOR i=0 TO 5: CLS
130 CIRCLE 128,88,10*i+10
140&MOV,6144,16384,27000+6144*i
150 NEXT i
160&SOUND,100,300,1,4
170 FOR i=0 TO 5
180&MOV,6144,27000+6144*i,16384
190&NOISE,20: NEXT i
```

### &DEL, line1, line2

Borra las líneas del programa Basic line1 y line2, ambas inclusive. Su utilidad es evidente.

## EL PROGRAMA EXTBASIC

Este programa se basa en el hecho de que se puede cambiar la dirección de la rutina de errores a la que se salta con RST 8. Esto se hace así:

```
LD DE, NEWADD
LD HL, (ERRSP)
LD (HL),E
```

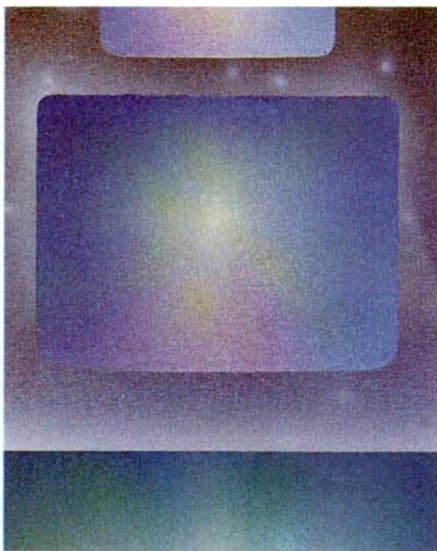
INC HL  
LD (HL),D  
RET

Siendo NEWADD la dirección de la nueva rutina de errores y ERRSP la variable del sistema en 23613d.

Pues bien, el programa EXT BASIC se sitúa en NEWADD (64010) y lo primero que hace es comprobar si el error es «Nonsense in Basic». Si no es así el error, no se puede deber a la introducción de una de las nuevas instrucciones. En tal caso hay que saltar a la rutina de la ROM que presenta un informe de error o a la que presenta un cursor parpadeante para señalar error de sintaxis, según estemos ejecutando una instrucción o comprobando su sintaxis (esto se sabe por el bit 7 de FLAGS, 23611d, puesto a 1 para indicar ejecución).

En caso de que sí fuera error «Nonsense in Basic», el programa EXT BASIC lee el carácter que lo ha provocado y si no es «&» pasa el control a las rutinas de la ROM antes comentadas. Si efectivamente es «&» el programa lee los caracteres que hemos escrito a continuación (usando RST 32) y si coinciden con alguno de los nuevos comandos, toma los parámetros que le siguen y, si estamos en tiempo de ejecución (bit 7 de FLAGS a 1), salta a la rutina correspondiente al comando de que se trate.

Tras ello hay que volver a la ROM para comprobar o ejecutar la siguiente instrucción. Esto se hace saltando a la dirección 7030d.



Para utilizar los dos bloques de código máquina debemos teclear el primero y realizar el DUMP. Sin borrarlo de la memoria procederemos a teclear el segundo listado y hacer el DUMP correspondiente, y, por último, grabarlos en conjunto indicando como dirección la 64000 y 860 como número de bytes.

## LISTADO 1

**DUMP: 64.000**  
**N.º BYTES: 330**

```

1  2A3D5C110AFA732372C9 937
2  110AFA053A3A5CFE0620 995
3  13CD2D0210313CD0325 887
4  200421CF12E5C3761BE1 1088
5  3A3A5CFD3600FCD3025 1060
6  28043C313132A5D5C22 598
7  5F5C225B5CC38D12FD36 1113
8  00FF2A3D5CE5E6D733D 1321
9  5CE1CD60F0E1223D5CFD 1533
10 CB007EC018C72A5D5C2B 1014
11 225D5C7EFE26C28A1C05 1003
12 021015BE5E7E16E6DFF 1533
13 413810FE5B300C77ED5E 989
14 5D5C122310EAC38A1C3E 911
15 109032005B21DFFA1101 824
16 5BCB7E20ED3A005B4623 943
17 B8200A1ABE2006231310 550
18 F818082310D23232318 713
19 DF46230405280FESDFF 1098
20 2C2001E7C5CD821CC110 1077
21 F3E1E5DFF1E0D2804F 1710
22 3A20B3CD3025C85E2355 974

23 EBE90653435245454E01 923
24 F4FB055245434F4C0200 875
25 FC0344454C0224FC0653 847
26 43524F4C4C00F0E0D0543 720
27 4C534C4F57006E0D054E 607
28 4F495345014EFC05534F 802
29 554E440472FC04574149 830
30 5401ABFC034D4F5603B 940
31 FC0652455045415400CE 913
32 FC05554E54494C01EDFC 1143
33 03434C520013FD800000 628

```

## LISTADO 2

**DUMP: 64.500**  
**N.º BYTES: 360**

```

1  CD94223A485C328D5CC3 1087
2  6B0DCD52DFE08301BF5 1165
3  CDD52DFE083013070707 813
4  47F1B021005811015801 715
5  BF0277EDB0C9CF13CDAS 1522
6  2D3823B02820606923CD 825
7  6E195CDAS2D3814B028 1071
8  116059CD5E19545DE1E5 1189
9  A7ED52E1D4E519C9CF19 1610
10 CDA52D381D50593A485C 891
11 0F0F04F21000007EE618 537
12 B1D3FE06010F231B79 1102
13 B320F0C9CF0ACDA52D38 1340
14 F9C5CDAS2D38F3ED4300 1464
15 SBCDA52D38EAE4D3025B 1193
16 CDA52D38E15069D1E5E5 1564
17 C12A025BA7ED42E1D8E5 1468
18 D5CDB503D1E1ED4B005B 1439
19 0918E7CDA52D38C0FB76 1296
20 0E78B120FAC9CDR52D38 1262
21 B3C5CDAS2D38ADC5CDAS 1587
22 2D38A7E1D1EDB0C93AS2 1456

23 FDFE08303F3C3252FD2A 1113
24 555C46234E00000002A53 485
25 FD712370232253FDC93A 1177
26 52FDA72827CDD52DAD728 1251
27 0D2152FD352A53FD2B82B 898
28 2253FD92A53FD2B462B 1105
29 4ECD2B2DC3671EAF3252 1006
30 FD2120FD2253FD9CF00 1349
31 0F00140000000000000000 35
32 0000000000000000000000 0
33 0000000000000000000000 0
34 0000000000000000000000 0
35 0000000000000000000000 0
36 0122FD0000000000000000 288

```

## LISTADO ENSAMBLADOR

```

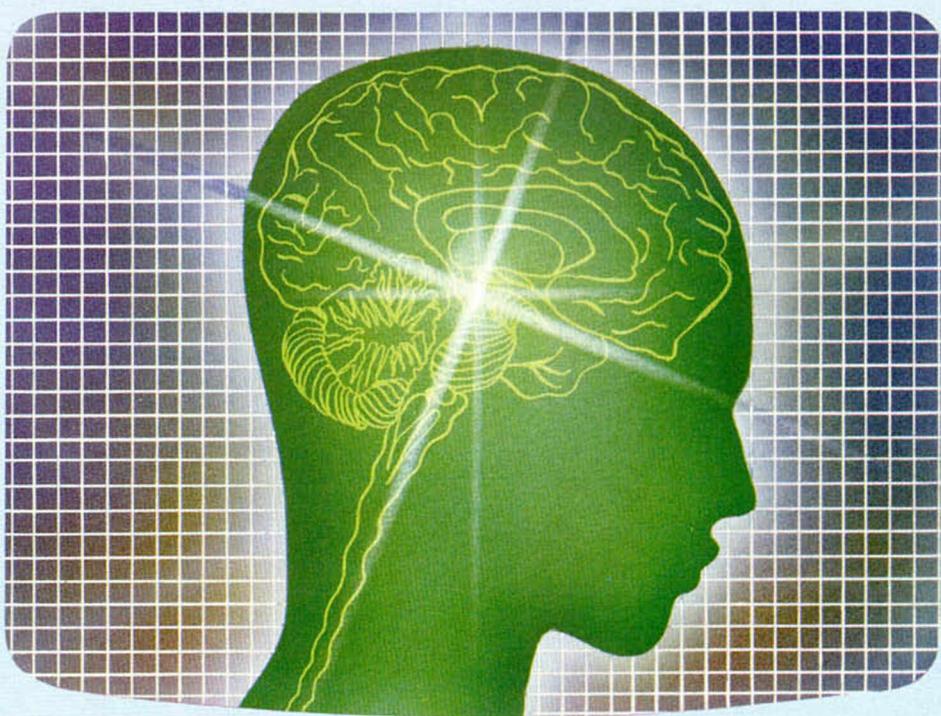
18  ORG 64000
20  LD HL,(23613)
30  LD DE,BUCLE1
40  LD (HL),E
50  INC HL
60  LD (HL),D
70  RET
80 BUCLE1 LD DE,BUCLE1
90 PUSH DE
100 LD A,(23610)
110 CP #0B
120 JR NZ,BUCLE2
130 CALL BUCLE3
140 LD HL,4867
150 CALL 9520
160 JR NZ,BUCLE4
170 LD HL,4815
180 PUSH HL
190 BUCLE4 JP 7030
200 BUCLE5 POP HL
210 BUCLE2 LD A,(23610)
220 LD (IY+0),#FF
230 CALL 9520
240 JR Z,BUCLE6
250 INC A
260 JP 4883
270 BUCLE6 LD HL,(23645)
280 LD (23647),HL
290 LD (23643),HL
300 JP 4797
310 BUCLE3 LD (IY+0),#FF
320 LD HL,(23613)
330 PUSH HL
340 PUSH HL
350 LD (23613),SP
360 POP HL
370 CALL BUCLE7
380 POP HL
390 LD (23613),HL
400 BIT 7,(IY+0)
410 RET NZ
420 JR BUCLE5
430 BUCLE7 LD HL,(23645)
440 DEC HL
450 LD (23645),HL

```

460	LD	A,(HL)	970	CP	#2C	1480	LD	C,H	1990	LD	C,H
470	CP	#26	980	JR	NZ,DATA8	1490	LD	D,E	2000	LD	D,D
480	JP	NZ,7306	990	RST	#28	1500	LD	C,H	2010		
490	LD	B,#18	1000	DATA8	PUSH BC	1510	LD	C,A	2020	INC	DE
500	LD	HL,23297	1010		CALL 7298	1520	LD	D,A	2030		NOP
510 BUCLE	PUSH	HL	1020		POP BC	1530	NOP		2040		ADD A,B
520	RST	#28	1030		DJNZ DATA7	1540	LD	L,(HL)	2050		ORG 64500
530	POP	HL	1040		POP HL	1550	DEC	C	2060	DATA8	CALL 8852
540	AND	#DF	1050	DATA6	PUSH HL	1560	DEC	B	2070	LD	A,(23624)
550	CP	#41	1060		RST #18	1570	LD	C,(HL)	2080		LD (23693),A
560	JR	C,BUCLE9	1070		POP HL	1580	LD	C,A	2090		JP 3435
570	CP	#5B	1080		CP #8D	1590	LD	C,C	2100		CALL 11733
580	JR	NC,BUCLE9	1090		JR Z,DATA9	1600	LD	D,E	2110	DATA12	CP #88
590	LD	(HL),A	1100		CP #3A	1610	LD	B,L	2120		JR NC,DATA13
600	LD	DE,(23645)	1110		JR NZ,BUCLE8	1620	LD	BC,DATA10	2130		PUSH AF
610	LD	(DE),A	1120	DATA9	CALL 9520	1630	DEC	B	2140		CALL 11733
620	INC	HL	1130		RET Z	1640	LD	D,E	2150		CP #88
630	DJNZ	BUCLE	1140		LD E,(HL)	1650	LD	C,A	2160		JR NC,DATA13
640 BUCLE8	JP	7306	1150		INC HL	1660	LD	D,L	2170		RLCA
650 BUCLE9	LD	A,#18	1160		LD D,(HL)	1670	LD	C,(HL)	2180		RLCA
660	SUB	B	1170		EX DE,HL	1680	LD	B,H	2190		RLCA
670	LD	(23296),A	1180		JP (HL)	1690	INC	B	2200		LD B,A
680	LD	HL,DATA1	1190	DATA1	LD B,#53	1700	LD	(HL),D	2210		POP AF
690 DATA2	LD	DE,23297	1200		LD B,E	1710		CALL M,22276	2220		OR B
700	BIT	7,(HL)	1210		LD D,D	1720	LD	B,C	2230		LD HL,22528
710	JR	NZ,BUCLE8	1220		LD B,L	1730	LD	C,C	2240		LD DE,22529
720	LD	A,(23296)	1230		LD B,L	1740	LD	D,H	2250		LD BC,703
730	LD	B,(HL)	1240		LD C,(HL)	1750	LD	BC,DATA11	2260		LD (HL),A
740	INC	HL	1250		LD BC,DATA8	1760	INC	BC	2270		LDIR
750	CP	B	1260		DEC B	1770	LD	C,L	2280		RET
760	JR	NZ,DATA3	1270		LD D,D	1780	LD	C,A	2290	DATA13	RST 8
770 DATA4	LD	A,(DE)	1280		LD B,L	1790	LD	D,(HL)	2300		INC DE
780	CP	(HL)	1290		LD B,E	1800	INC	BC	2310		CALL 11685
790	JR	NZ,DATA3	1300		LD C,A	1810	CP	B	2320		JR C,DATA14
800	INC	HL	1310		LD C,H	1820		CALL M,20998	2330		OR B
810	INC	DE	1320		LD (BC),A	1830	LD	B,L	2340		JR Z,DATA14
820	DJNZ	DATA4	1330		NOP	1840	LD	D,B	2350		LD H,B
830	JR	DATA5	1340		CALL M,17411	1850	LD	B,L	2360		LD L,C
840 DATA3	INC	HL	1350		LD B,L	1860	LD	B,C	2370		INC HL
850	DJNZ	DATA3	1360		LD C,H	1870	LD	D,H	2380		CALL 6510
860	INC	HL	1370		LD (BC),A	1880	NOP		2390		PUSH HL
870	INC	HL	1380		INC H	1890	ADC	A,#FC	2400		CALL 11685
880	INC	HL	1390		CALL M,21254	1900	DEC	B	2410		JR C,DATA14
890	JR	DATA2	1400		LD B,E	1910	LD	D,L	2420		OR B
900 DATA5	LD	B,(HL)	1410		LD D,D	1920	LD	C,(HL)	2430		JR Z,DATA14
910	INC	HL	1420		LD C,A	1930	LD	D,H	2440		LD H,B
920	INC	B	1430		LD C,H	1940	LD	C,C	2450		LD L,C
930	DEC	B	1440		LD C,H	1950	LD	C,H	2460		CALL 6510
940	JR	Z,DATA6	1450		NOP	1960	LD	BC,64749	2470		LD D,H
950	PUSH	HL	1460		CP #8D	1970	INC	BC	2480		LD E,L
960 DATA7	RST	#18	1470		LD B,#43	1980	LD	B,E	2490		POP HL

## 48 LENGUAJES

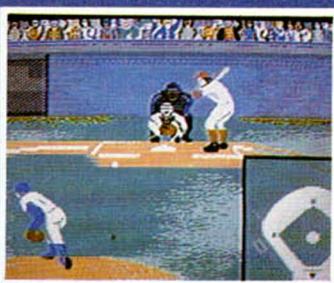
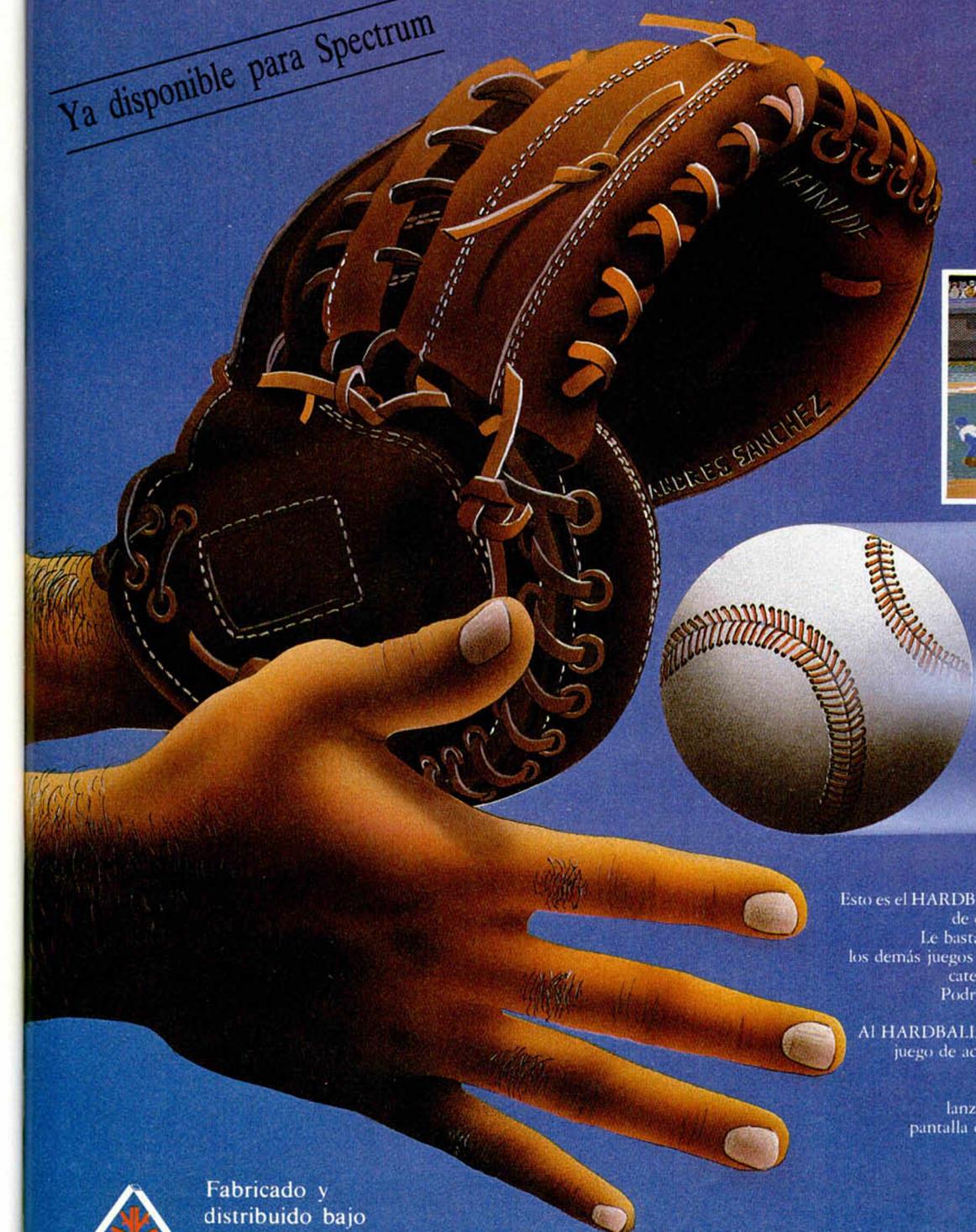
2500	PUSH HL	3010	— SBC HL,BC	3290	POP HL	3570	JR Z,DATA22
2510	AND A	3020	POP HL	3300	POP DE	3580	LD HL,DATA18
2520	SBC HL,DE	3030	RET C	3310	LDIR	3590	DEC (HL)
2530	POP HL	3040	PUSH HL	3320	RET	3600	LD HL,(DATA20)
2540	CALL NC,6629	3050	PUSH DE	3330	LD A,(DATA18)	3610	DEC HL
2550	RET	3060	CALL 949	3340	CP #08	3620	DEC HL
2560	DATA14 RST 8	3070	POP DE	3350	JR NC,DATA19	3630	LD (DATA20),HL
2570	ADD HL,DE	3080	POP HL	3360	INC A	3640	RET
2580	DATA10 CALL 11685	3090	LD BC,(23296)	3370	LD (DATA18),A	3650	DATA22 LD HL,(DATA20)
2590	JR C,DATA15	3100	ADD HL,BC	3380	LD HL,(23637)	3660	DEC HL
2600	LD D,B	3110	JR DATA16	3390	LD B,(HL)	3670	LD B,(HL)
2610	LD E,C	3120	DATA11 CALL 11685	3400	INC HL	3680	DEC HL
2620	LD A,(23624)	3130	JR C,DATA15	3410	LD C,(HL)	3690	LD C,(HL)
2630	RRCA	3140	EI	3420	NOP	3700	CALL 11563
2640	RRCA	3150	DATA17 HALT	3430	NOP	3710	JP 7783
2650	RRCA	3160	DEC BC	3440	NOP	3720	XOR A
2660	LD C,A	3170	LD A,B	3450	LD HL,(DATA20)	3730	DATA19 LD (DATA18),A
2670	LD HL,B	3180	OR C	3460	LD (HL),C	3740	LD HL,DATA23
2680	DATA24 LD A,(HL)	3190	JR NZ,DATA17	3470	INC HL	3750	DATA21 LD (DATA20),HL
2690	AND #18	3200	RET	3480	LD (HL),B	3760	RET
2700	OR C	3210	CALL 11685	3490	INC HL	3770	RST 8
2710	OUT (#FE),A	3220	JR C,DATA15	3500	LD (DATA20),HL	3780	NOP
2720	LD B,#08	3230	PUSH BC	3510	RET	3790	DATA23 CALL M,DATA12
2730	PAUSA DJNZ PAUSA	3240	CALL 11685	3520	LD A,(DATA18)	3800	INC BC
2740	INC HL	3250	JR C,DATA15	3530	AND A	3810	ORG 64851
2750	DEC DE	3260	PUSH BC	3540	JR Z,DATA21	3820	DATA18 NOP
2760	LD A,D	3270	CALL 11685	3550	CALL 11733	3830	DATA20 JR NZ,DATA18
2770	OR E	3280	JR C,DATA15	3560	AND A		
2780	JR NZ,DATA24						
2790	RET						
2800	DATA15 RST 8						
2810	LD A,(BC)						
2820	CALL 11685						
2830	JR C,DATA15						
2840	PUSH BC						
2850	CALL 11685						
2860	JR C,DATA15						
2870	LD (23296),BC						
2880	CALL 11685						
2890	JR C,DATA15						
2900	LD (23298),BC						
2910	CALL 11685						
2920	JR C,DATA15						
2930	LD H,B						
2940	LD L,C						
2950	POP DE						
2960	DATA16 PUSH HL						
2970	PUSH HL						
2980	POP BC						
2990	LD HL,(23298)						
3000	AND A						



# HardBall

## Nunca verá un juego de béisbol tan próximo a la realidad

Ya disponible para Spectrum



Esto es el HARDBALL, simple y a su vez el juego de simulación de deportes más realista de todos los tiempos.

Le bastarán sólo cinco minutos para ver que todos los demás juegos de béisbol para ordenadores son de menor categoría en comparación con el HARDBALL.

Podría jurar que está viendo un programa de la televisión un sábado por la tarde. Al HARDBALL, se puede jugar de dos maneras, una como juego de acción en el campo, y otra como un juego de estrategia de entrenador, o ambas a la vez.

Observe la curva descrita por la bola lanzada por encima de la rotunda o consulte la pantalla de entrenadores para una sustitución clave.

Puede incluso situarse dentro o fuera del terreno de juego para comprobar el estilo del bateador o la situación del juego.



Fabricado y  
distribuido bajo  
licencia por:

DISTRIBUIDO en Cataluña y Baleares por:

DISCLUB, S.A. - Balmes, 58 - BARCELONA - Tel: (93) 302 39 08 - P.V.P. 2.300 Ptas.

Santa Cruz de Marenado, 31 - 28015 Madrid - Teléf. 241 10 63

Para una mejor comprensión de los nemónicos utilizados por el Z80, os ofrecemos un diccionario que explica el significado y uso de las distintas instrucciones del lenguaje ensamblador, así como pequeños ejemplos aclaratorios.

Conviene que antes de adentrarnos en él, examinemos una serie de cuestiones que atañen al SPECTRUM en general y al código máquina en particular, y que contribuirán a un mejor empleo de las posibilidades del Z80 y, por tanto, del SPECTRUM.

Lo primero a considerar es el peculiar modo de trabajo del código máquina, así, mientras en el BASIC las variables tienen unos cometidos concretos, en éste se opera indirectamente con ellas, realizándose todas las operaciones a través de registros cuya misión es, precisamente, registrar temporalmente un valor o un resultado para finalmente almacenarlo en la memoria, que es la auténtica variable. Naturalmente, en subrutinas de pequeña extensión, se pueden considerar los registros como si de auténticas variables se tratara. De hecho, una de las facilidades que permite el BASIC del SPECTRUM es la de retornar un valor desde código máquina a través del registro doble BC lo cual hace que éste pueda ser considerado en ciertas ocasiones como una variable.

El número total de registros del Z80 es de veinticuatro, siete de ellos se denominan de uso general y el resto se utilizan para cometidos específicos. Un registro puede almacenar números y como consta de 8 bits codificados en binario natural, su valor puede variar entre 0 y 255. Estos bits se enumeran del 7 al 0 siendo el bit 7 el más significativo o de más valor («peso» en el argot). Cuando se quiere trabajar con signo se utiliza el bit 7 para diferenciar los números positivos de los negativos (si es cero el número es positivo y viceversa) lo que limita el rango de

# GUÍA DE



# COMANDOS



valores desde -128 a 127. Para no dejarnos desamparados el Z80 permite que estos registros, denominados simples, se unan a otros registros formando registros dobles, de esta forma el rango de valores aumenta de 0 a 65535 (o de -32768 a 32767 si operamos con signo).

Cada registro se designa por una letra conservando su nombre incluso cuando se une con otro para formar un registro doble. Helos aquí:

#### REGISTROS SIMPLES:

DE USO GENERAL	DE USO ESPECIFICO
A	F
B	
C	I
D	
E	R
H	
L	

#### REGISTROS DOBLES:

DE USO GENERAL	DE USO ESPECIFICO
AF	IX
BC	IY
DE	SP
HL	

El registro doble AF no existe como tal, pero ciertas instrucciones como PUSH, POP y EX los tratan conjuntamente. Observar también que los registros I y R no forman ningún registro doble y que NO EXISTEN los registros simples Ix, X, ly, Y, S o P sino que SIEMPRE operan en la forma IX, IY y SP (esto no es del todo cierto a nivel «extraoficial», pero esto ya excede de la misión de este diccionario).

El Z80 dispone a su vez de una serie de registros alternativos a los registros A, B, C, E, H y L y se denominan:

## REGISTROS ALTERNATIVOS

AF'  
BC'  
DE'  
HL'

Se denominan también como registros PRIMA. Su función es preservar los valores de estos registros y para acceder a ellos existen dos instrucciones que intercambian sus valores entre sí.

### Los banderines de aviso: los Flags

El Z80 dispone de un registro especializado denominado F (también llamado «registro de estado») cuya misión es la de almacenar varios bits de información de acuerdo a los resultados de los cálculos efectuados. Cada bit se usa como un banderín que toma un valor de 1 ó 0 según se active o no y su cometido, de izquierda a derecha, es como sigue:

**Flag de signo (S)** - almacena el signo, positivo o negativo, del último cálculo realizado. Un resultado positivo asignará este bit a 0, y el negativo a 1. Un valor de cero será tomado también como positivo. El valor del flag S es siempre igual al bit más significativo del resultado (el bit más a la izquierda) pudiendo ser testado por instrucciones como JP P (salta si positivo) y JP M (salta si negativo [menos]).

**Flag cero (Z)** - si el último resultado ha sido cero el flag se activa poniéndose a 1. No debe pasársenos desapercibido que las instrucciones DEC y ADD para registros pares NO AFECTAN a este flag.

**No usado** - este bit tiene un valor más o menos aleatorio.

**Half-carry (H)** - Este banderín se activa cuando en una operación se produce un acarreo del BIT 3 al BIT 4, o, en el caso de registros pares, del bit 11 al bit 12 y es usado internamente por el Z80 para instrucciones como DAA. No se puede testar directamente aunque es posible examinarlo usando la secuencia PUSH AF / POP BC / BIT 4, C y entonces mirar el flag de cero, pero esto raramente se hace.

### NO USADO

**Flag de paridad / sobrecarga (P/V)** - Cumple dos cometidos:

1— la paridad de un resultado es par o impar, dependiendo del número de unos, o de ceros, de dicho resultado. Si la paridad es par se asigna el flag a uno, y si es impar a cero.

Las instrucciones que asigna este flag de acuerdo con la paridad del resultado son:

AND r - OR r - XOR r - RL r - RLC r - RR r - RRC r - SLA r - SRA r - SRL r - RLD - RRD - DAA - IN r

2— una sobrecarga representa un cambio «accidental» del signo del resultado: un acarreo del bit 6 al bit 7. Las siguientes instrucciones asignan este flag según se produzca o no esta sobrecarga:

ADD A,r - ADC A,r - ADC HL,s - SUB A,r - SBC A,r - SBC HL,s - CP r - NEG - INC r - DEC r

**Flag de resta (N)** - mira simplemente si la última instrucción ejecutada es una suma o una resta. Esta instrucción se usa intermitentemente por el Z80 para instrucciones como DAA y no tiene apenas interés. Se puede testar con PUSH y POP como con HALF-HARRY.

**Flag de carry (C)** - detecta un acarreo del bit 7 al supuesto bit 8 en los registros simples o, en el caso de los registros pares, del bit 15 al supuesto 16. Una operación frecuente suele ser la de testar un bit de un registro moviéndolo al carry por medio de instrucciones de rotación o reinserir el bit «perdido» en el carry dentro de un registro. Este flag, junto con el de cero, son con toda probabilidad los más utilizados.

La forma en que un programa en código máquina posibilita la toma de decisiones y la correspondiente solución estriba precisamente en el empleo de instrucciones que tienen en cuenta el estado de los flags para operar y re-

ciben el nombre de instrucciones condicionales. Así, por ejemplo, durante la ejecución de un programa puede ser necesario que de acuerdo al resultado de una operación el programa se bifurque a otra dirección para lo cual utilizaremos instrucciones como JR Z o JP C, etc. También podremos efectuar llamadas a una subrutina (CALL Z, CALL P, etc.) al cumplirse ciertas condiciones, retornar de ella también de forma condicional que determina si la instrucción se ejecuta o no. Estos términos son los siguientes:

Z si el último resultado calculado es cero, la instrucción se ejecuta.

NZ la instrucción se ejecuta si el resultado no es cero.

C se ejecuta si se ha producido un acarreo.

NC se ejecuta si no se ha producido un acarreo.

PE este condicional chequea el flag P/V (Parity/Overflow) y realiza dos funciones:

— si nos referimos a la paridad entonces la instrucción se ejecuta si el último resultado calculado en formato binario tiene un número PAR de UNOS (o de ceros) afectándole instrucciones como AND, OR, IN A, (C), de rotación, etc.

— la denominación de sobrecarga (overflow) tiene a su vez dos formas de tratamiento: si en un cálculo (ADC, SBC, etc.) se produce un «acarreo» del bit 6 al bit 7 (el número excede el rango positivo-negativo) entonces la instrucción se ejecuta. Lo mismo ocurre para instrucciones de decremento e incremento con registros simples. La otra forma se refiere a instrucciones como LDI, LDD, CPI, CPR, etc., en las que mientras el resultado (BC en este caso) no sea cero la instrucción se ejecuta.

PO la instrucción se ejecuta justamente si se cumple lo contrario de lo dicho en PE.

M si el último resultado calculado es negativo (menos) la instrucción se ejecuta.

P si el último resultado es positivo la instrucción se ejecuta.



## Instrucciones sin prefijo

ADC A,r	@@	@@	@@	@@	@@	0	@@	IND	?	x	?	-	?	1	-	OTIR	?	1	-	?	1	-	
ADC HL,s	@@	@@	-	@@	-	@@	0	@@	INIR	?	1	-	?	1	-	POP AF	x	x	x	x	x	x	
ADD A,r	@@	@@	-	@@	-	@@	0	@@	INDR	?	1	-	?	1	-	POP s	-	-	-	-	-	-	
ADD HL,s	-	-	-	@@	-	-	0	@@	JP pq	-	-	-	-	-	-	PUSH AF	-	-	-	-	-	-	
ADD IX,s	-	-	-	@@	-	-	0	@@	JP c,pq	-	-	-	-	-	-	PUSH s	-	-	-	-	-	-	
ADD IY,s	-	-	-	@@	-	-	0	@@	JP (HL)	-	-	-	-	-	-	RES b,r	-	-	-	-	-	-	
AND r	@@	%	-	1	-	@@	0	0	JP (IX)	-	-	-	-	-	-	RET	-	-	-	-	-	-	
BIT b,r	?	@@	-	1	-	@@	0	-	JP (IY)	-	-	-	-	-	-	RET c	-	-	-	-	-	-	
CALL pq	-	-	-	-	-	-	-	-	JR e	-	-	-	-	-	-	RETN	-	-	-	-	-	-	
CALL c,pq	-	-	-	-	-	-	-	-	JR c,e	-	-	-	-	-	-	RLCA	-	0	-	0	@@	@@	
CCF	-	-	-	x	-	-	0	@@	LD (BC),A	-	-	-	-	-	-	RRCA	-	0	-	0	@@	@@	
CP r	@@	@@	-	@@	-	@@	1	@@	LD A,(BC)	-	-	-	-	-	-	RLA	-	0	-	0	@@	@@	
CPI	@@	x	-	@@	-	x	1	-	LD (DE),A	-	-	-	-	-	-	RRA	-	0	-	0	@@	@@	
CPD	@@	x	-	@@	-	x	1	-	LD A,(DE)	-	-	-	-	-	-	RLC r	@@	@@	-	0	@@	@@	
CPIR	@@	x	-	@@	-	x	1	-	LD I,A	-	-	-	-	-	-	RRC r	@@	@@	-	0	@@	@@	
CPDR	@@	x	-	@@	-	x	1	-	LD R,A	@@	@@	0	-	x 0	-	RL r	@@	@@	-	0	@@	@@	
CPL	-	-	-	1	-	1	-	LD A,I	@@	@@	0	-	x 0	-	RR r	@@	@@	-	0	@@	@@		
DAA	@@	@@	-	@@	-	@@	-	@@	LD A,R	@@	@@	0	-	x 0	-	RRD	@@	@@	-	0	@@	@@	
DEC r	@@	@@	-	@@	-	@@	1	-	LD SP,HL	-	-	-	-	-	-	RLD	@@	@@	-	0	@@	@@	
DEC s	-	-	-	-	-	-	-	-	LD SP,IX	-	-	-	-	-	-	RST 00	@@	@@	-	0	@@	@@	
DI	-	-	-	-	-	-	-	-	LD SP,IY	-	-	-	-	-	-	RST 08	-	-	-	-	-	-	
DJNZ e	-	-	-	-	-	-	-	-	LD r,r	-	-	-	-	-	-	RST 10	-	-	-	-	-	-	
EI	-	-	-	-	-	-	-	-	LD s,mm	-	-	-	-	-	-	RST 18	-	-	-	-	-	-	
EX AF, AF'	-	-	-	-	-	-	-	-	LD A,(pq)	-	-	-	-	-	-	RST 20	-	-	-	-	-	-	
EX DE, HL	-	-	-	-	-	-	-	-	LD s,(pq)	-	-	-	-	-	-	RST 28	-	-	-	-	-	-	
EX (SP),HL	-	-	-	-	-	-	-	-	LD (pq),A	-	-	-	-	-	-	RST 30	-	-	-	-	-	-	
EX (SP),IX	-	-	-	-	-	-	-	-	LD (pq),s	-	-	-	-	-	-	RST 38	-	-	-	-	-	-	
EX (SP),IY	-	-	-	-	-	-	-	-	LDI	-	0	-	x 0	-	-	SBC A,r	@@	@@	-	@@	1	@@	
EXX	-	-	-	-	-	-	-	-	LDI	-	0	-	x 0	-	-	SBC HL,s	@@	@@	-	@@	1	@@	
HALT	-	-	-	-	-	-	-	-	LDI	-	0	-	0 0	-	-	SCF	-	-	0	-	0	1	
IM 0	-	-	-	-	-	-	-	-	NEG	@@	@@	-	@@	@@	1	@@	SET b,r	-	-	-	-	-	-
IM 1	-	-	-	-	-	-	-	-	NOP	-	-	-	-	-	-	SLA r	@@	@@	0	-	@@	@@	
IM 2	-	-	-	-	-	-	-	-	OR r	@@	@@	0	-	@@	0 0	SRA r	@@	@@	0	-	@@	@@	
INC r	@@	@@	-	@@	-	@@	0	-	OUT (n),A	-	-	-	-	-	-	SRL r	@@	@@	0	-	@@	@@	
INC s	-	-	-	-	-	-	-	-	OUT (C),r	?	x	-	?	-	1	SUB r	@@	@@	-	@@	1	@@	
IN A,(n)	-	-	-	-	-	-	-	-	OUTI	?	x	-	?	-	1	XOR r	@@	@@	0	-	@@	0	
IN r,(C)	@@	@@	-	@@	-	@@	0	-	OUTD	?	x	-	?	-	1	-	-	-	-	-	-		

0	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NOP	LD BC, NN	LD (BC), A	INC BC	INC B	DEC B	LD B, N	RLCA	EX AF, AF	ADD HL, BC	LD A, (BC)	DEC BC	INC C	DEC C	LD C, N	RRCA
1	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
1	DJNZ DIS	LD DE, NN	LD (DE), A	INC DE	INC D	DEC D	LD D, N	RLA	JR DIS	ADD HL, DE	LD A, (DE)	DEC DE	INC E	DEC E	LD E, N	RRA
2	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
2	JR NZ, DIS	LD HL, NN	LD (NN), HL	INC HL	INC H	DEC H	LD H, N	DAA	JR Z, DIS	ADD HL, HL	LD H, (NN)	DEC HL	INC L	DEC L	LD L, N	CPL
3	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
3	JR NC, DIS	LD SP, NN	LD (NN), A	INC SP	INC (HL)	DEC (HL)	LD (HL), N	SCF	JR C, DIS	ADD HL, SP	LD A, (NN)	DEC SP	INC A	DEC A	LD A, N	CCF
4	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
4	LD B, B	LD B, C	LD B, D	LD B, E	LD B, H	LD B, L	LD B, (HL)	LD B, A	LD C, B	LD C, C	LD C, D	LD C, E	LD C, H	LD C, L	LD C, (HL)	A
5	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95
5	LD D, B	LD D, C	LD D, D	LD D, E	LD D, H	LD D, L	LD D, (HL)	LD D, A	LD D, B	LD D, C	LD D, D	LD D, E	LD D, H	LD D, L	LD D, (HL)	A
6	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111
6	LD H, B	LD H, C	LD H, D	LD H, E	LD H, H	LD H, L	LD H, (HL)	LD H, A	LD L, B	LD L, C	LD L, D	LD L, E	LD L, H	LD L, L	LD L, (HL)	A
7	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127
7	LD (HL), B	LD (HL), C	LD (HL), D	LD (HL), E	LD (HL), H	LD (HL), L	LD (HL), (HL)	LD (HL), A	LD A, B	LD A, C	LD A, D	LD A, E	LD A, H	LD A, L	LD A, (HL)	A
8	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143
8	ADD A, B	ADD A, C	ADD A, D	ADD A, E	ADD A, H	ADD A, L	ADD A, (HL)	ADD A, A	ADC A, B	ADC A, C	ADC A, D	ADC A, E	ADC A, H	ADC A, L	ADC A, (HL)	A
9	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159
A	160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175
B	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191
C	192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207
D	208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223
E	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239
F	240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255

## ADC

SUMA con ACARREO. Se encuentra en dos formas: ADC A,r y ADC HL,s. La r debe entenderse como cualquiera de los registros A, B, C, D, E, H, L, un número, o el contenido de una dirección cuyo puntero es (HL), (IX+d) o (IY+d). La s se entiende como cualquiera de los registros dobles BC, DE, HL o SP. ADC A,r efectúa la operación A=A+r+CARRY y ADC HL,s opera en la forma HL=HL+s+CARRY. La instrucción ADC afecta a todos los FLAGS.

## ADD

SUMA. Esta operación se efectúa sin incluir el acarreo y se presenta en las formas ADD A,r, ADD HL,s, ADD IX,s y ADD IY,s. La r y la s toman idéntica forma que en ADC siendo diferente en el caso de IX e IY en las cuales HL se sustituye por IX e IY, respectivamente. La forma ADD A,r afecta a todos los flags mientras que las otras no afectan a los flags S, Z y P/V.

## AND

Toma la forma AND r (entendido como A AND r) siendo r cualquiera de los registros A, B, C, D, E, H, L, un número, o el contenido de una dirección cuyo puntero es (HL), (IX+d) o (IY+d). Conviene observar que esta instrucción sólo opera entre el registro A y r.

La operación lógica AND consiste en la MULTIPLICACION BINARIA, BIT a BIT, entre el valor del registro A con el correspondiente de r quedando el resultado en A. Su lógica es  $0 \text{ AND } 0 = 0$ ,  $0 \text{ AND } 1 = 0$ ,  $1 \text{ AND } 0 = 0$  y  $1 \text{ AND } 1 = 1$ . Si, por ejemplo, el valor de A es (en binario) de 01101100 y el de r 11100111 el resultado será 01100100 (ver figura 10).

AND altera todos los flags, especialmente el de CARRY que siempre se pone a CERO.

### AND

A	0	0	1	1	1	0	0	1
	↓	↓	↓	↓	↓	↓	↓	↓
B	0	1	0	1	0	1	0	1
<hr/>								
A	0	0	0	1	0	0	0	1

Fig. 10

## BIT

Esta instrucción examina el estado concreto de un bit de un registro o una indirección haciendo una copia de éste en el flag Z. Se escribe como BIT b,r donde b es el bit a examinar (del 0 al 7) y r puede ser uno cualquiera de los registros A, B, C, D, E, H, L, o un bit del contenido de una dirección cuyo puntero es (HL), (IX+d) o (IY+d). Esta instrucción altera todos los flags excepto el CARRY.

## CALL

Equivale al GOSUB del Basic. CALL efectúa una llamada a una SUBRUTINA especificada como una dirección. Si, por ejemplo, queremos llamar a una subrutina que se encuentra en la dirección 25000 se escribirá como CALL 25000. La instrucción se ejecuta como sigue: en primer lugar el microprocesador introduce en el STACK la dirección de retorno para efectuar a continuación un salto a la dirección especificada. La instrucción CALL puede, a su vez, operar de forma condicional: CALL Z, CALL NC, CALL PE, etc.

## CCF

COMPLEMENTA el CARRY FLAG. Si Carry es cero cambia su valor a uno, y viceversa.

## CP

COMPARA. Toma la forma CP r y efectúa una comparación, entendida como A-r, entre A y r afectando a TODOS los flags, pero sin alterar el valor del registro A o de r. La r puede ser cualquiera de los registros A, B, C, D, E, H, K, un número, o el contenido de una dirección cuyo puntero es (HL), (IX+d) o (IY+d).

## CPD

COMPARA CON DECREMENTO. Esta es una potente instrucción que permite comparar el registro A con una tabla de datos direccionada por HL. Opera como sigue: primero efectúa CP (HL), seguido de DEC HL, seguido de DEC BC. La instrucción CP (HL) efectuada afecta UNICAMENTE (en lo que interesa de dicha operación) al flag de CERO (Z). A su vez DEC BC afecta al flag P/V ya que mientras BC no sea cero este flag permanecerá activado, de esta manera, si queremos repetir el proceso BC veces bastará efectuar un JP PE a la instrucción CPD.

Todos los flags son afectados excepto el CARRY.

## CPDR

COMPARA con DECREMENTO y REPETICION. Efectúa la misma operación que CPD excepto en que el proceso se repite automáticamente mientras BC no sea CERO o hasta que el byte comparado sea idéntico al registro A.

## CPI

COMPARA con INCREMENTO. Igual que CPD excepto que HL se incrementa en lugar de decrementarse.

## CPIR

COMPARA con INCREMENTO y REPETICION. Como CPDR excepto que HL se incrementa.

**CPL**

COMPLEMENTA el registro A (complemento a uno). Efectúa el complemento bit a bit del registro A. Si un bit vale 0 lo pone a 1 y viceversa.

**DAA**

AJUSTE DECIMAL el registro A. Esta instrucción se utiliza cuando se trabaja con aritmética BCD en la cual un byte se parte en dos NIBBLE. Cada NIBBLE consta, respectivamente, del bit 7 al bit 4 y del bit 3 al bit 0. Se asume que cada nibble podrá tomar un valor del 0 al 9 y cualquier valor que lo exceda se tomará como un «acarreo». DAA se encarga, precisamente, de «ajustar» el valor de los dos nibbles al formato correcto. Si ese produjese un «acarreo» en el nibble más significativo (bits del 7 al 4) éste afectaría al CARRY. DAA afecta a todos los flags.

**DEC**

DECREMENTA. Toma dos formas posibles: DEC r y DEC s. Su cometido es simplemente reducir en uno el valor de r ó s. La r se entiende como cualquiera de los registros A, B, C, D, E, H, L, o el contenido de una dirección cuyo puntero es (HL), (IX+d) o (IY+d) y afecta a todos los flags excepto al carry. La s se entiende como cualquiera de los registros BC, DE, HL, SP, IX o IY y NO AFECTA a NINGUN FLAG.

**DI**

DESHABILITA INTERRUPCIONES. Impide que el microprocesador atienda a la señal INT de interrupción y, por lo tanto, la subrutina de interrupción.

**DJNZ**

DECREMENTA el registro B y SALTA SI NO es CERO. Esta instrucción es particularmente útil en bucles de corta longitud (128 bytes) y que no precisan una repetición superior a 256 (B=0). DJNZ NO AFECTA a los flags.

**EI**

HABILITA INTERRUPCIONES. Permite al microprocesador entender la señal de petición de interrupción y ejecutar, según el modelo de interrupción, la subrutina correspondiente.

**EX**

INTERCAMBIO. Hay cinco instrucciones diferentes para EX: EX, AF, AF', EX DE, HL, EX (SP), HL, EX (SP), IX y EX (SP), IY. La instrucción EX DE, HL intercambia el valor de DE por el de HL y viceversa. Ninguna de estas instrucciones afecta a los flags (la instrucción EX, AF, AF' toma los flags de F' pero no los altera).

**EXX**

INTERCAMBIO de REGISTROS PRIMA. Intercambia por sus correspondientes PRIMA los registros BC, DE y HL. Esta instrucción es muy interesante por la rapidez (y sencillez) de ejecución frente a instrucciones como PUSH y POP. En el SPECTRUM el registro prima HL contiene la dirección de salto a una subrutina de cálculo de la ROM a través de la cual se produce el retorno al sistema operativo BASIC. Si utilizamos una subrutina que corrompa su valor y queremos retornar al BASIC conviene ejecutar LD HL, 2758 hex/EXX/RET (mejor que EXX/PUSH HL/.../POP HL/EXX).

**HALT**

ALTO. Esta instrucción detiene el Z80 hasta que se produce una señal de interrupción y ésta se acepta. En el caso de que estén deshabilitadas las interrupciones el microprocesador permanecerá detenido indefinidamente (una forma de «ponerlo en marcha», aparte el consabido RESET, puede ser vía NMI).

**IM**

MODO de INTERRUPCION. Esta instrucción se aplica a las denominadas interrupciones enmascarables (o «habitables-deshabitables») y puede tomar una de las tres formas siguientes:

IM 0. El microprocesador, al producirse una señal de interrupción, asume que hay un periférico encargado de suministrársela la INSTRUCCION de INTERRUPCION a través del BUS de datos. Como en el SPECTRUM no existe este periférico específico, el BUS de datos (que trabaja con lógica inversa) contendrá 255 que es el código de operación de la instrucción RST 38 hex (CALL 38 hex) efectuándose accidentalmente una llamada a dicha subrutina. Como esta subrutina es precisamente la encargada de tratar la interrupción efectuando la lectura de teclado y el incremento de FRAMES, no tiene ninguna consecuencia apreciable.

IM 1. Es el modo utilizado por el SPECTRUM. Cada vez que se produce una interrupción el microprocesador ejecuta un RST 38 hex efectuándose la lectura de teclado y el incremento del valor de FRAMES.

IM 2. Podría decirse que éste es el modo de interrupción más potente. Cuando ésta se produce, el microprocesador debe tomar de la memoria dos bytes que van a formar precisamente la dirección de la subrutina de interrupción. ¿Cómo decide la DIRECCION donde se encuentran esos dos bytes?: en primer lugar asume que el byte MENOS significativo que forma esa dirección va a ser suministrado por un periférico encargado de introducirlo en el BUS de datos y, como no existe tal periférico, este byte SIEMPRE será igual a 255. Para generar el byte MAS significativo de la dirección PUNTERO se utiliza el registro especializado I (de Interrupción) de forma que la DIRECCION de la SUBRUTINA de INTERRUPCION podría expresarse así: DSI = PEEK (I\*256 + 255) + 256\*PEEK (I\*256 + 255 + 1).

## IN

**INPUT.** Se escribe en dos formas: IN A,(n) e IN r,(C). Se utiliza para leer datos suministrados por un periférico: el teclado, un Joystick, una impresora, etc. En la primera forma, la n es un número comprendido entre 0 y 255 y se refiere al byte menos significativo de la dirección del periférico. La instrucción se usa asignando primeramente al registro A el byte más significativo de dicha dirección y a continuación se efectúa la lectura. Por ejemplo: si queremos leer la semifila del teclado correspondiente a las teclas 1 a 5 escribiremos: LD A,F7 hex/IN A, (FE hex). Esta instrucción no altera ningún flag.

En la forma IN A,(C), la (C) se refiere a (BC), la r puede ser cualquiera de los registros A, B, C, D, E, H o L. Para efectuar una lectura como en el ejemplo anterior escribiríamos LD BC,F7FE hex/IN A,(C). Esta instrucción altera todos los flags excepto el carry.

Esta instrucción es también utilizada por la ROM en la subrutina de LOAD.

## JR

**SALTO RELATIVO.** Efectúa un salto, hacia delante o hacia atrás, un número especificado de bytes a partir de la posición del registro PC (el Contador de Programa cuando lee una instrucción se sitúa justamente al principio de la siguiente ANTES de ejecutar dicha instrucción). Esta instrucción puede ejecutarse de forma condicional, pero únicamente para los flags de CERO y CARRY.

## LD

**CARGA.** Es el equivalente al LET del BASIC permitiendo la asignación de valores a registros, la carga de un registro en otro, la carga del contenido de una dirección en un registro y viceversa, etc. Es, precisamente, la instrucción más utilizada.

## IND

**INPUT con DECREMENTO.** Opera en este orden: IN (HL), (C), seguido de DEC HL, seguido de DEC B. Altera todos los flags, excepto el carry, y especialmente el flag de cero que permanecerá activo mientras B sea diferente de cero.

## LDD

**CARGA con DECREMENTO.** Opera por orden como LD (DE), (HL) (entendido como POKE DE, PEEK HL) seguido de DEC HL, DEC DE y DEC BC. Esta instrucción altera el flag P/V que permanecerá activo mientras BC no sea cero.

## INDR

**INPUT con DECREMENTO y REPETICION.** Opera como IND repitiéndose el proceso mientras B no sea cero.

## LDDR

**CARGA con DECREMENTO y REPETICION.** Como LDD repitiéndose el proceso automáticamente mientras BC no sea cero. Esta instrucción, junto con LDIR, es probablemente la más potente del Z80 utilizándose para mover grandes bloques de datos de forma automática, efectuar un CLS o un SROLL.

## INI

**INPUT con INCREMENTO.** Como IND excepto que HL se incrementa.

## LDI

**CARGA con INCREMENTO.** Como LDD excepto que HL y DE se incrementan.

## INIR

**INPUT con INCREMENTO y REPETICION.** ComoINI, repitiéndose el proceso mientras B no sea cero.

## LDIR

**CARGA con INCREMENTO y REPETICION.** Como LDI repitiéndose el proceso mientras BC no sea cero.

## JP

**SALTO a una dirección.** Toma las formas JP n y JP s donde n es un número que especifica la dirección de salto pudiendo efectuarse éste de forma condicional.

Por s se puede utilizar uno de los registros HL, IX o IY efectuando el microprocesador un «salto» a la dirección que forma el valor de uno de estos registros. Esta forma de salto es incondicional.

## NEG

**NEGACION.** Invierte el signo del registro A (complemento a dos) y altera todos los flags. Si A vale uno, entonces NEG cambia su valor a menos uno (FF hex).

**NOP**

NO OPERACION. El registro PC (contador de programa) avanza hasta la próxima instrucción. Se suele usar como retardo o con objeto de ser reescrito más adelante.

**OR**

En la forma OR r (entendida como A OR r) donde r puede ser cualquiera de los registros A, B, C, D, E, H, L, un número o el contenido de una dirección cuyo puntero es (HL), (IX + d) o (IY + d). Efectúa, bit a bit, la función lógica OR en la que  $0 \text{ OR } 0 = 0$ ,  $1 \text{ OR } 0 = 1$ ,  $0 \text{ OR } 1 = 1$  y  $1 \text{ OR } 1 = 1$ . Así, si  $A = 10011011$  y  $r = 00011000$  el resultado será  $10011011$  (ver figura 11). Esta instrucción afecta a todos los flags especialmente al CARRY que se pone SIEMPRE A CERO.

**OUT**

OUTPUT. Efectúa la operación inversa a la instrucción IN tomando las formas OUT (n), A y OUT (C), r donde n y r son idénticas a las explicadas para IN. En la forma OUT (n), A no se especifica el byte más significativo. La ROM utiliza OUT en rutinas como SAVE y BEEP.

**OUTD**

OUTPUT con DECREMENTO. Efectúa OUT (C), (HL) seguido de DEC HL y DEC B. El flag de carry permanece inalterado siendo afectado el de cero de acuerdo al valor final de B.

**OTDR**

OUTPUT con DECREMENTO y REPETICION. Como OUTD, repitiéndose el proceso mientras B no sea cero.

**OUTI**

OUTPUT con INCREMENTO. Como OUTD excepto que HL se incrementa.

**OTIR**

OUTPUT con INCREMENTO y REPETICION. Como OUTI repitiéndose el proceso mientras B no sea cero.

**POP**

EXTRAER. En la forma POP AF y POP s donde s puede ser BC, DE, HL, IX o IY. Si efectuamos POP BC el efecto será como LD C,(SP)/INC SP/LD B,(SP)/INC SP. Excepto en el caso de POP AF (que recupera los flags) esta instrucción deja los flags inalterados.

**PUSH**

INTRODUCIR. Opera en modo inverso a la instrucción POP. Si efectuamos PUSH BC el efecto sería DEC SP/LD (SP), B/DEC SP/LD (SP), C. Esta instrucción no altera ningún flag.

**RES**

RESET bit. En la forma RES b, r donde b es un bit del 0 al 7 y r cualquiera de los registros A, B, C, D, E, H, L, un número, o el contenido de una dirección cuyo puntero es (HL), (IX + d) o (IY + d). Esta instrucción pone a CERO el bit especificado sin alterar ningún flag.

**RET**

RETORNA de la subrutina. Equivale al RETURN del BASIC pudiendo efectuarse de forma condicional. Esta instrucción toma del STACK la dirección de retorno de la subrutina y finalmente salta a dicha dirección.

**RETI**

RETORNA de la INTERRUPCIÓN enmascarable. Opera como sigue: antes de retornar espera hasta el siguiente pulso de interrupción con objeto de evitar la recursividad, luego habilita las interrupciones (el microprocesador las deshabilita automáticamente) y por último, toma la dirección de retorno del STACK efectuando un salto a dicha dirección.

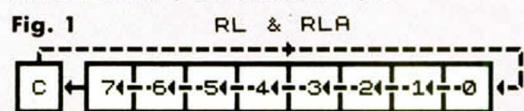
**RETN**

RETORNA de la SUBRUTINA de INTERRUPCIÓN NO ENMASCARABLE. Cuando llega un pulso de señal a la patilla NMI (petición de Interrupción No en Mascarable) del microprocesador éste efectúa un RST 66 hex inmediatamente ya que este modo de interrupción tiene PRIORIDAD ABSOLUTA y no existe para ella instrucción de deshabilitación. Antes de efectuar esta llamada el microprocesador almacena en un flip-flop interno el estado de las interrupciones (habilitadas o no), a continuación las deshabilita, almacena en el STACK la dirección de retorno y por último, salta a la dirección 66 hex. Cuando se ejecuta la instrucción RETN el microprocesador repone el estado de las interrupciones, toma del STACK la dirección de retorno y finalmente salta a esta dirección.

**RLA**

ROTACION a la IZQUIERDA del ACUMULADOR a través del carry. Como puede observarse en la figura 1 cada uno de los bits de A es movido una posición a la izquierda, pasando el bit 7 al CARRY y el valor inicial de éste al bit 0 de A. Esta instrucción afecta únicamente al CARRY y sólo precisa un byte.

Fig. 1



## RL

ROTACION a la IZQUIERDA a través del carry. Toma la forma  $RL\ r$  donde  $r$  puede ser cualquiera de los registros A, B, C, D, E, H, L, o el contenido de una dirección cuyo puntero es  $(HL)$ ,  $(IX+d)$  o  $(IY+d)$ . Efectúa la misma operación que RLA con la diferencia de que afecta a TODOS los flags y precisa de dos a tres bytes (según sea  $r$ ) empleando, mínimo, el doble de ciclos de operación. Puede observarse esquemáticamente su funcionamiento en la figura 1.

## RRA

Como RLA excepto que la rotación se produce a la derecha en lugar de a la izquierda (figura 4).

Fig. 4 RR & RRA



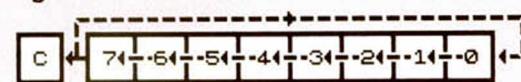
## RLCA

ROTACION CIRCULAR a la IZQUIERDA del ACUMULADOR. En la figura 2 se observa su funcionamiento. Cada uno de los bits de A se desplaza en forma circular hacia la izquierda pasando una copia del bit 7 al bit 0 y otra al carry. Esta instrucción afecta sólo al flag CARRY.

## RLC

ROTACION CIRCULAR a la IZQUIERDA. Toma la forma  $RLC\ r$  donde  $r$  es idéntica que para RLA. Opera de igual forma que RLCA referido a  $r$ .

Fig. 2 RLC & RLCA



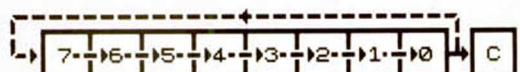
## RR

Como RL excepto que la rotación se produce a la derecha en lugar de a la izquierda (figura 4).

## RRCA

Como RLCA excepto que la rotación se produce hacia la derecha en lugar de a la izquierda (figura 5).

Fig. 5 RRC & RRCA

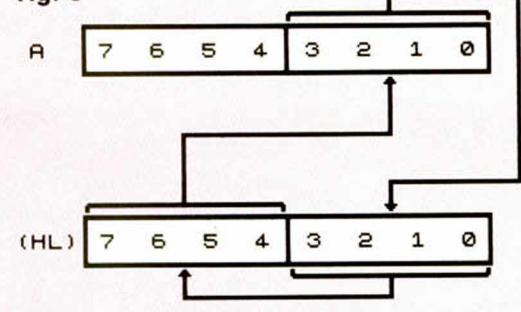


## RLD

ROTACION a la IZQUIERDA de DOS DIGITOS BCD. Esta instrucción se utiliza en aritmética BCD y opera de forma especial ya que no se produce una rotación propiamente dicha sino un desplazamiento. El contenido de la dirección cuyo puntero es  $(HL)$  se trata como dos NIBBLE (figura 3). El nibble MENOS significativo de  $(HL)$  pasa a la posición del nibble MAS significativo de  $(HL)$ . A su vez el nibble MAS significativo de  $(HL)$  pasa al nibble MENOS significativo del REGISTRO A. El nibble MENOS significativo de A pasa al nibble MENOS significativo de  $(HL)$ .

RLD

Fig. 3



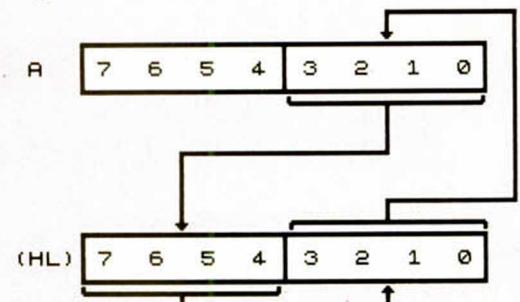
## RRC

Como RLC excepto que la rotación se produce hacia la derecha en lugar de a la izquierda (figura 5).

## RRD

ROTACION DERECHA de DOS DIGITOS BCD. El nibble MAS significativo de  $(HL)$  pasa al MENOS significativo de  $(HL)$  y éste a su vez al MENOS significativo del REGISTRO A. El nibble MENOS significativo de A pasa al MAS significativo de  $(HL)$  (ver figura 6).

Fig. 6 RRD



**RST**

RESTART. Produce el mismo efecto que la instrucción CALL con dos diferencias: sólo es posible ejecutar esta instrucción para OCHO posibles direcciones (en hex): 0, 8, 10, 18, 20, 28, 30 ó 38 siendo esta llamada INCONDICIONAL.

**SBC**

RESTA con ACARREO. Opera en dos formas: SBC A, r y SBC HL,s donde r y s toman la misma forma que para ADC. Esta instrucción afecta a todos los flags.

**SET**

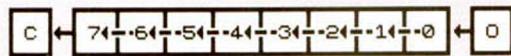
ASIGNA. Toma la forma SET b,r donde b y r son idénticos a la instrucción RES. SET pone a UNO el bit especificado de r.

**SLA**

DESPLAZAMIENTO ARITMETICO a la IZQUIERDA. Toma la forma SLA r donde r se escribe como en las instrucciones de rotación ya descritas. Los bits de r se desplazan una posición a la izquierda, el bit 7 pasa al CARRY y el BIT 0 se pone a CERO (figura 7). Altera todos los flags.

**Fig. 7**

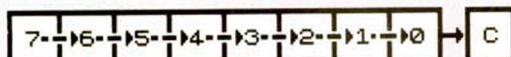
SLA

**SRA**

DESPLAZAMIENTO ARITMETICO a la DERECHA. Se escribe como SRA r donde r es la forma ya mencionada. Los bits de r se desplazan a la derecha, el bit 0 pasa al CARRY permaneciendo el bit 7 inalterado (figura 8). Altera todos los flags.

**Fig. 8**

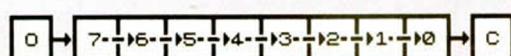
SRA

**SRL**

DESPLAZAMIENTO LOGICO a la DERECHA. Se escribe como SRL r donde r es la forma ya mencionada. Los bits de r se desplazan a la derecha, el bit 0 pasa al CARRY y el bit 7 se pone a CERO (figura 9). Altera todos los flags.

**Fig. 9**

SRL

**SUB**

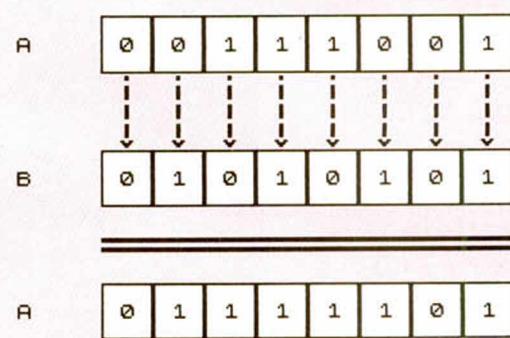
RESTA. Toma la única forma SUB r (se entiende como SUB A,r) ya que SOLO EXISTE PARA REGISTROS SIMPLES. La r se entiende como cualquiera de los registros A, B, C, D, E, H, L, un número, o como el contenido de una dirección cuyo puntero es (HL), (IX+d) o (IY+d). Opera como A=A-r y altera TODOS los flags. Si quisieramos efectuar una resta SIN acarreo para registros dobles deberíamos usar la secuencia AND A (u OR A)/SBC HL,s. La instrucción AND a (u OR A) no afecta al registro A pero pone el carry a cero.

**XOR**

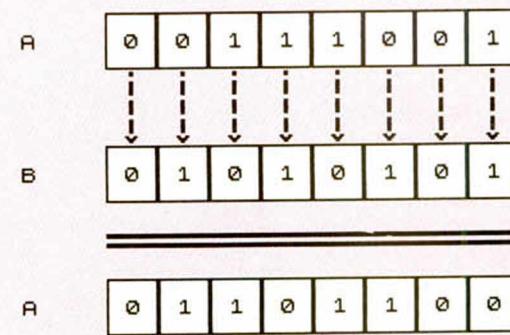
OR EXCLUSIVA. Toma la forma XOR r (entendida como A XOR r) donde r es idéntica a la explicada para AND y OR. Efectúa la función OR exclusiva, bit a bit, del registro A con r en la que 0 XOR 0=0, 1 XOR 0=1, 0 XOR 1=1 y 1 XOR 1=0. Si por ejemplo A=11000011 y r=00110011 el resultado será 11110000 (ver ejemplo fig. 12).

**Fig. 11**

OR

**Fig. 12**

XOR



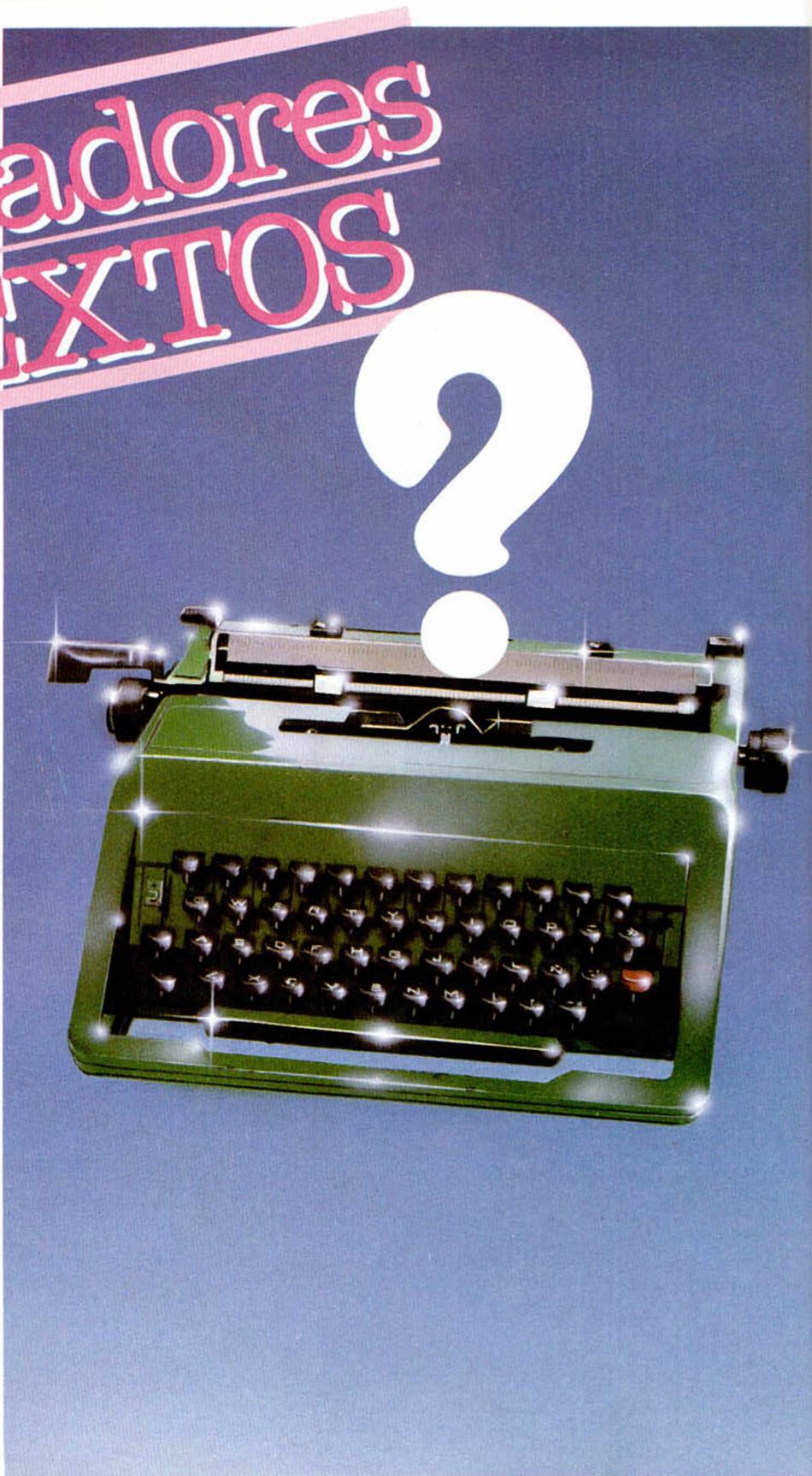
J. M. LAZO

# Procesadores de TEXTOS

**Un procesador de textos es, sin duda, la mejor herramienta de trabajo que un escritor puede usar. La prueba reside en que un 95 por ciento de los escritos que tenéis ocasión de leer están confeccionados con uno de estos maravillosos programas.**

**C**uando decidimos realizar un análisis de los procesadores de texto existentes para el Spectrum nos encontramos ante una caótica situación: por una parte existen varios (Context v2, Tasword two, Context v6 y Context v.8, por mencionar algunos) que son distintos, no cabe duda, pero distintos sólo en el listado Basic que incorporan y que se encarga de manejar las memorias externas y la impresora. La parte CM del programa y, por lo tanto, lo referente a las cualidades que tenga el mismo en lo que concierne a la escritura de un texto, formateado de pantalla, distintos comandos etc, son prácticamente iguales en éstos.

Por otra parte, recientemente ha sa-



lido al mercado un procesador de textos, *The last Word* (la última palabra), que es la nueva concepción por lo que es completamente distinto a los que hasta ahora disponíamos. Esto nos ha llevado a comentar nada más que dos **programas**: por una parte el **Context v.8** que incorpora las máximas novedades en los de «antigua» concepción. Y por otra parte el **Last Word**, que como hemos dicho, es totalmente distinto a éstos.

### The last word

Este procesador de textos se puede considerar que tiene dos puntos que lo destacan de los demás: por una parte una gran **sencillez de manejo**, ya que todas las órdenes que se le pueden dar se introducen con la sola pulsación de una o dos teclas. Por otra parte, el procesador entero está escrito en CM y en el extenso manual de instrucciones se da todo tipo de información detallada no sólo sobre la manera de manejarlo sino sobre él mismo a nivel variables de programa, rutinas y demás. Esto, unido a que es el único programa que conocemos que es capaz de manejar todo tipo de **interfaces de memorias de masa** habidos y por haber, hace pensar que *The last word* es un programa diseñado por una empresa **seria** preocupada por el usuario.

A nivel general trabaja con la pantalla dividida en dos trozos altamente diferenciados: tres líneas de la parte superior destinadas a contener la información del estado en que se encuentra el programa y las 20 líneas inferiores que contienen el texto propiamente dicho. En la parte superior de estas 20 líneas de texto se abrirá una ventana blanca, cuando sea oportuno para que podamos introducir la información que sea necesaria para alguno de los comandos de que dispone el programa.

### La zona de textos

El texto es muy cómodo de entrar, es decir, por muy *deprisa* que tecleemos el programa nos sigue sin perder

ni una sola tecla ya que una característica muy *aguda* del mismo es que dispone de un «buffer» de 21 letras en el que almacenará las pulsaciones que demos en el teclado si éstas no pueden ser atendidas en ese momento, esta cualidad hace las delicias de los mecanógrafos *habilidosos* que en un momento dado son capaces de alcanzar una alta velocidad en el teclado.

En esta zona de textos, aparte de presentarse él mismo, se nos mantendrá informado, en la última columna de la derecha, del estado de las distintas líneas. Es decir, podrán estar tres tipos de símbolos distintos:

— Una «C» invertida, que quiere decir que esta línea de texto está terminada con la pulsación de la tecla «Enter». Esto se indica así, ya que en la zona de memoria destinada al texto, que por cierto es bastante amplia (**26 K**), no se codifican los espacios que dejemos, y de igual forma sólo se pondrá el carácter de «Enter», o retorno de carro, cuando la misma tecla sea pulsada.

— Una flecha inclinada hacia abajo, que quiere decir que la línea es normal y corriente.

— La misma «C» invertida de antes pero con un *subrayado*, que nos quiere decir que en una línea en la que sólo está introducido el carácter «Enter» se han metido también *tokens* o códigos de control de la impresora, estos últimos no ocupan ningún espacio en el texto a la hora de presentarlo en la pantalla.

En esta zona de texto también va, como arriba hemos comentado, la *ventana de información* que necesitan algunos comandos. Cuando *invocemos* alguno de los mismos, el texto se *escrolará* hacia abajo tantas líneas como sea necesario y aparecerá esta ventana. En este momento dejaremos de tener control sobre el texto propiamente dicho y sólo podremos actuar sobre el comando en sí.

La zona de textos se puede presentar en la pantalla del ordenador en un formato variable de 40, 48, 60 u **80 columnas**, aunque esta última forma no será muy legible a no ser que dispongamos de un monitor de alta resolución.

Aunque tengamos un formato de pantalla determinado podemos tra-

jar con todas las columnas de texto que precisemos, **hasta 148**, con lo cual conseguimos hacer textos aptos para impresoras de *alto calibre*. Si tenemos unas columnas de texto mayores en número a las columnas de pantalla, el texto no se justificará ni se *enrollará* hasta que no lleguemos a la última columna de texto presentándose la información de la línea y columna en la parte superior de la pantalla acorde a esta circunstancia. Es decir, que si estamos escribiendo un texto de 100 columnas, por ejemplo, y el formateado de pantalla lo tenemos puesto sólo a 40, cada línea de texto ocupará 2 líneas y 20 caracteres de pantalla. Por supuesto, podemos escribir un escrito a un número determinado de columnas de *impresora* y luego modificar el mismo con una orden para otro número distinto, y ésta es una de las mayores ventajas de este procesador. Este artículo, por ejemplo, está siendo escrito a 40 columnas de pantalla y de impresora por comodidad, pero luego a la hora de imprimirlo se pondrá a 60 o las que hagan falta.

### La ventana informativa superior

Las tres líneas de arriba de la pantalla contienen, como arriba hemos comentado, información referente al estado en que se halla el programa y el texto que estamos escribiendo:

— Por una parte, la línea y columna donde se encuentra el cursor de texto y el espacio de memoria libre que nos queda en K's.

— Luego, ya en el centro, el número de columnas en que está la presentación así como un *switch* o bandera que indica si tenemos activado los marcadores de final de línea o no, el número de espacios del tabulador, que también tiene, y si estamos en mayúsculas o minúsculas, así como si el programa está esperando texto o un comando.

— En el centro derecho, se hallan los márgenes derecho e izquierdo del texto, éstos podrán ser un número cualquiera entre el 1 y el 148.

— Por último, a mano derecha se encuentran tres apartados: el prime-

ro nos dice si tenemos activada la detección de final de línea según escribamos o no. El segundo nos indica si nos hayamos escribiendo un texto o insertándolo en medio de otro más grande y el tercero y último nos dice si deseamos justificar la línea, o **enrollarla** como se dice en el **Context**, o por el contrario sólo separar la última palabra.

Hay que hacer una aclaración para los **neófitos**: justificar una línea significa separar todas las palabras de la misma proporcionalmente al espacio que ocupen para que el aspecto del texto sea más profesional.

## Los comandos

Hay dos formas de introducir los distintos comandos de que dispone el procesador de textos: por una parte con la pulsación conjunta de la tecla **CAPS** o **SIMBOL** junto con la del comando. Por otra parte activando el modo extendido con **CAPS + SYMBOL** y luego la tecla del comando o bien pulsada sola, o bien junto con **SYMBOL**. Toda la información, la de todos los comandos que hay, y las teclas necesarias para invocarlos se presenta en pantalla usando la orden **HELP** con extendido +H.

Las órdenes se pueden clasificar en 5 grandes grupos según lo que hagan:

— **Movimiento del cursor.** El cursor de texto se puede mover por el mismo de muy distintas formas. Con las **flechas del cursor** y se mueve, lógicamente, en las cuatro direcciones posibles, eso sí, no podremos desplazarlo por el sitio donde no haya texto ya que los espacios vacíos no están en la memoria.

Por otra parte, también se puede mover por palabras hacia un lado o el otro, y en pasos fijos según tengamos programado el **tabulador**. Una forma de moverlo muy interesante, es desplazarlo con una orden determinada hasta el próximo párrafo, lo que nos servirá para **reparar** el texto si hemos cambiado el número de columnas del mismo. Se puede ir, de igual forma, hasta una línea determinada con el siguiente comando. Por último, también se puede mover por páginas hacia arriba o hacia abajo así como

al principio o al final del escrito.

— **Manejo del texto.** Dentro de este párrafo están los comandos para borrar de textos. Esta operación se puede hacer de todas formas imaginables: borrar un carácter, una línea, hasta el final de un párrafo, entre dos líneas determinadas, y todo el texto completo, por supuesto. De igual forma, en este apartado nos encontramos con comandos que sirven para **justificar** o **desjustificar** la línea en la que se encuentre el cursor y una facilidad muy interesante para **reparar** el texto hasta el próximo párrafo. Con este comando y el que sirve para avanzar de párrafo en párrafo nos podemos cambiar las columnas de **impresora** del escrito completo en un **abrir y cerrar de ojos**.

Un par de facilidades más que se pueden encontrar en este apartado son: una orden para buscar y/o cambiar palabras dentro del texto y otra para centrar cabecera.

— **Comandos denominados de utilidades.** Aquí nos podemos encontrar con el **grueso de las fuerzas**, en este caso el bloque de comandos hacen operaciones muy generales sobre el texto. Por una parte están las órdenes para cambiar el estado de todos los **marcadores** de la parte superior de la pantalla (número de columnas en pantalla, en impresora, saltos del tabulador, justificación y **enrolle de líneas**, etc.). Por otra parte existen órdenes, dentro de este bloque, para cambiar los colores con los que queremos trabajar para adecuar el programa al gusto de cada uno. **The last word**, además de ser un procesador de textos, dispone de una calculadora totalmente completa con toda la potencia del Basic que sirve para efectuar cálculos complejos y almacenar los resultados en unas memorias así como usarlos en el texto que estemos escribiendo. Esto, que puede parecer de dudosa utilidad al principio, resulta luego muy interesante cuando precisemos introducir números en el texto productos de algún cálculo.

Por último, dentro de este bloque se puede destacar una orden para programar una alarma que nos avisará cada cierto tiempo al objeto de que **refresquemos** el backup del texto que tengamos en el disco u otra memoria de masa.

— **El cuarto bloque de comandos incorpora todo lo referente a la impresora.** Por cierto..., es de destacar el que el programa esté inicialmente preparado sólo para el interface de impresora Kempston E, y aunque no nos ha resultado difícil adaptarlo para el de Indescomp Centronics, sí puede resultar imposible para un usuario no avezado en conocimientos de CM. De todas formas, en el manual se da una información para adaptarlo a cualquier impresora pero que no resulta suficientemente clara.

Como decíamos, en este bloque se hallan tres órdenes para el **manejo de la impresora**, cosa ésta fundamental en un procesador de textos. El primero es, lógicamente, el que nos permite imprimir un texto en nuestra impresora, se pueden dar la primera línea, la última a imprimir, el espacio entre líneas así como el número de copias a sacar.

El segundo nos informa sobre cómo están programados los distintos códigos de control de la impresora, el programa dispone de 24 que inicialmente están previstos para una impresora **EPSON RX-80**, aunque una de las primeras cosas que deberemos de hacer con el procesador es programar estos **tokens** para nuestra impresora antes de sacar la copia de seguridad.

El tercer comando es una guía de la impresora que nos permite cambiar la dirección de la rutina de impresión así como los códigos de Enter y Retorno de carro.

— **Ya en el último bloque nos encontramos con todas las órdenes necesarias para manejar las memorias de masa.** Se puede grabar un texto, lógicamente, especificando de qué línea a qué línea se quiere grabar. Lo podemos cargar también, aunque no es una carga propiamente dicha, sino una **mezcla** al final del texto que tengamos en memoria. Si deseáramos cargarlo limpiamente tendríamos que borrar la memoria con la orden **ZAP**.

Tres órdenes quedan para los usuarios de memorias de masa distintas al cassette: sacar un catálogo del disco, erasear un fichero y formatear un disco.

## Resumiendo

Por muy **poco dinero** disponemos de un paquete de manejos de textos que resulta **innovador** en su técnica y sencillo en su manejo, aunque no hay que descontar que es nuevo en el mercado, por lo que pueda resultar que tenga algún *bug* importante. De todas formas, no hemos detectado nada raro.

## Context v.8

Si hay algún programa que se pueda considerar **veterano** en el Spectrum ése es el Context v.8 y toda su larga saga de predecesores. Esto nos da una enorme garantía de funcionamiento al haberse hecho multitud de versiones depuradas, optimizadas y ampliadas del mismo programa. Aparte de esto es, quizás, el programa más conocido, como utilidad, para este ordenador.

Context v.8 es un procesador de textos, último por ahora de su *estirpe*, que, uniendo una facilidad de manejo grande con una *performance* adecuada al Spectrum nos lleva al equilibrio entre lo **sofisticado** y lo sencillo en un procesador de textos.

Como al principio de este artículo se comentó, el programa se divide en dos partes: el CM, que asume todo el bloque de funciones de manejo del texto y demás rutinas que han permanecido prácticamente invariables y el Basic, que contiene todo lo referente al manejo de memorias externas e impresora y, en esta **última versión**, también un par de opciones de fichero que luego se comentarán.

El formateado en pantalla a la hora de escribir el texto está invariablemente en 64 columnas fruto del compromiso entre lo funcional y lo cómodo de usar. Sólo cambiará a las 32 normales del Spectrum cuando salgamos al menú principal por medio del uso de la función STOP. Hablando ya de este menú el mismo dispone de 9 opciones y es una buena idea el comenzar por ahí.

## El menú principal

La primera opción, **Texto**, es, como su nombre indica, para retornar al

editor de texto y poder hacer modificaciones en el mismo.

La segunda, **impresora**, sirve para imprimir el texto en una impresora grande. Primero nos preguntará la primera línea que queremos imprimir y luego, la última para, acto seguido, pasar a la impresión del texto. Después retornará automáticamente a este menú principal.

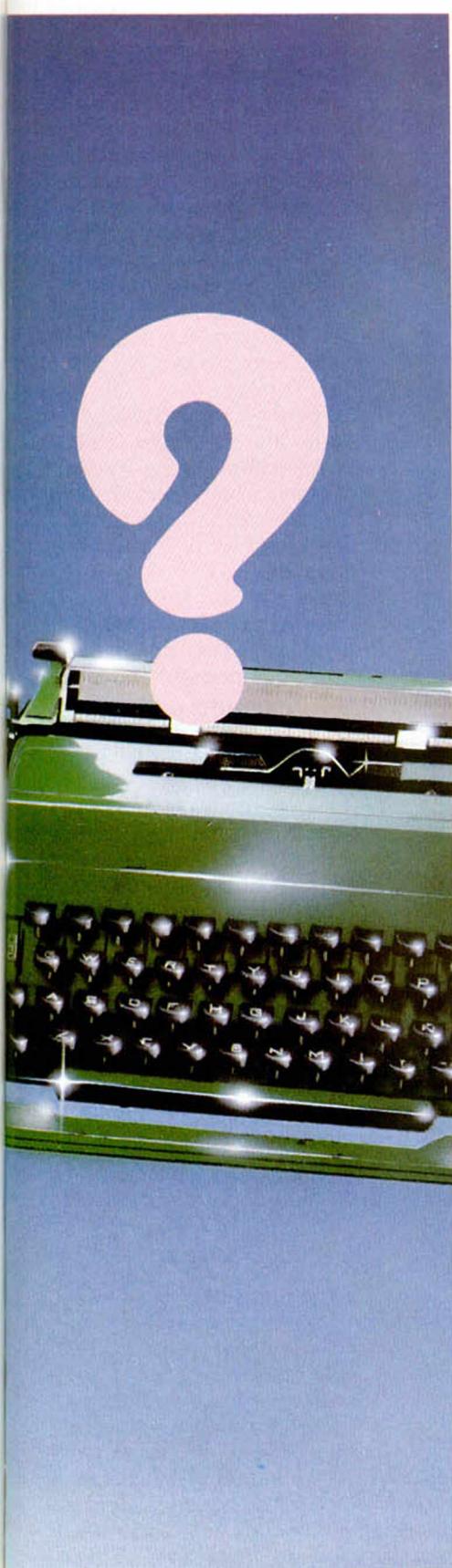
La tercera opción, **memorias externas**, nos lleva a un segundo menú con las opciones de salvado, cargado y eraseado del fichero de texto tanto en cassette como en microdrive.

La cuarta opción, **ficheros**, nos lleva a la impresión del texto con fichas. Esto es conveniente explicarlo más detalladamente:

Hay que tener en cuenta que el archivo de texto consta de 320 líneas de 64 caracteres y nosotros, con la opción de impresora, podemos imprimir un trozo cualquiera de un texto. Pero nos puede interesar tener, dentro del texto, un sitio en el cual, en el momento de la impresión introduzcamos esa ficha, pues bien, esta opción incluye la impresión con ficheros. Lo primero que tendremos que hacer será crear la o las fichas, para lo cual al final del texto que queramos imprimir escribiremos la misma entre dos corchetes ([ y ]). Luego, en los sitios donde queramos poner estas fichas, situaremos el carácter gráfico cuyo código ASCII es el 143 y por último, podemos imprimir el texto con la opción 4 del menú principal.

La siguiente opción, la 5, es la creación de **cartas personalizadas** y la usaremos cuando precisemos imprimir textos en los que haya nombres o direcciones que varíen. Para esto sólo tenemos que elegir esta opción del menú principal y escribir el texto, pero en los sitios donde vayan los nombres o direcciones situaremos un número entre corchetes ([ y ]). En el momento en que terminemos de escribir el texto el ordenador nos preguntará los párrafos que tiene que asignar a estos corchetes y procederá a imprimir el texto. Luego nos interrogará sobre si deseamos repetir la operación y lo volverá a hacer si así lo deseamos.

La sexta opción, **sustitución de CHR\$,** nos sirve para cambiar un ASCII de todo el texto por otro distinto. Esto lo utilizaremos cuando nuestra im-



presora tenga unos códigos para la ñ, por ejemplo, distintos a los que tiene el programa.

El procesador de textos tiene dos páginas de ayuda que se consiguen pulsando «Edit» estando en el modo editor, pues bien, una de estas páginas tiene la información referente a la impresora y si deseamos combinarla habremos de usar la opción 7 del menú principal, **modificación de la información**.

Las últimas dos opciones del menú principal son referentes a la impresora: la octava, **margen izquierdo**, sirve para fijar el margen izquierdo, valga la redundancia, que queramos cuando vayamos a imprimir el texto. Y la última, **cambio de interface**, se utiliza para seleccionar el interface de impresora que tengamos de entre un total de 4 que es capaz de manejar el programa.

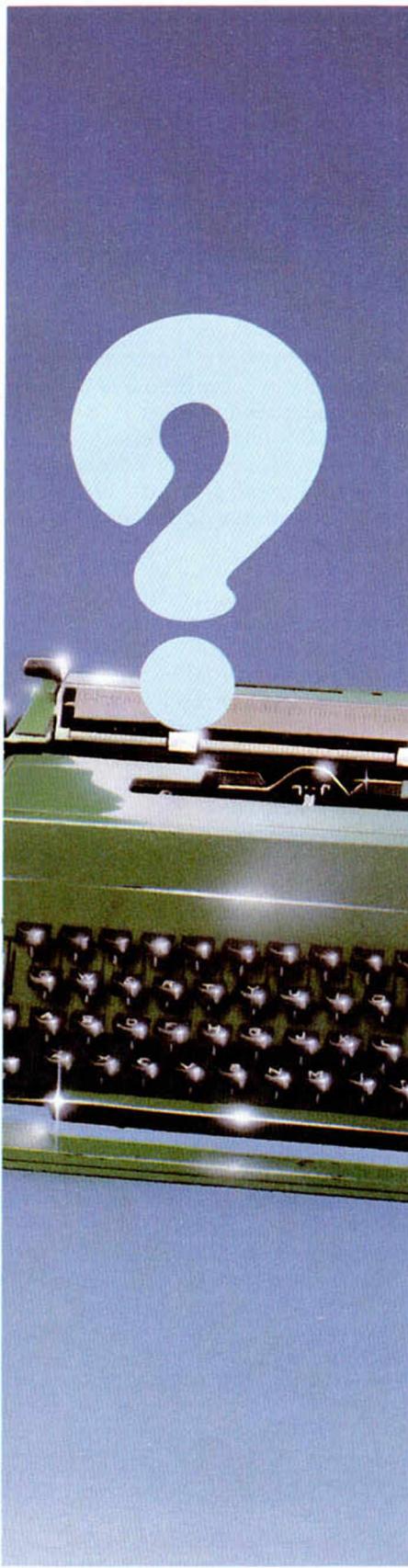
### El editor de texto

Una vez que entremos en el editor de texto con la opción 1 del menú principal, dispondremos del escrito que estuvieramos confeccionando en la pantalla y el cursor al comienzo del mismo. En este momento podemos seguir escribiendo texto o introducir algún comando del editor de un modo igual o parecido al que tiene *The Last Word*: algunas órdenes se dictarán con la pulsación de una tecla junto con **CAPS** o **SYMBOL** y en otras se habrá de poner el programa previamente en modo extendido con el uso de estas dos teclas.

Al igual que con *The Last Word* disponemos de un amplio muestrario de órdenes y comando de ayuda para confeccionar un texto:

— Por una parte el cursor se puede mover en las cuatro direcciones posibles con el uso de las flechas del cursor, aunque esta vez lo podemos mover también por debajo del final de texto ya que el Context v.8 codifica también los espacios en memoria. El cursor igualmente se puede mover de palabra en palabra, hacia delante o hacia atrás.

— También podemos movernos de pantalla en pantalla de texto para leerlo cómodamente así como llevar el



cursor al final o al principio del escrito. Si leyendo el texto viéramos que se nos ha olvidado una palabra la podemos insertar con la orden **AND** que nos abre una línea de texto a partir del cursor. Si fuera más de una palabra activaríamos el modo de inserción con lo cual según fuéramos escribiendo texto la ventana se iría ampliando. Para arreglar el desaguisado que hubiéramos ocasionado usaríamos la orden **STEP** que arregla un párrafo de texto. De igual manera también podemos borrar líneas completas con la orden **NOT**.

— Para poner cabeceras existe un comando que centra textos en la pantalla automáticamente aunque luego podemos correr la línea completa hacia la izquierda o la derecha usando otras dos órdenes.

— Al igual que *The Last Word* y como todo buen procesador de textos que se precie, dispone de una detección automática de final de línea según tecleamos que podemos inhibir o desconectar y de una justificación automática de la línea o no.

— Unos comandos muy interesantes que posee el Context y que le faltan al *Last Word*, son los referentes al manejo de bloques: se pueden marcar principio y final de bloque con sendas órdenes y luego mover o copiar este bloque en otra parte del texto.

— Por último, sólo destacar un par de cosas: que si nos resulta más cómodo trabajar en **32 columnas** lo podemos hacer con este procesador de texto sin perder la *profesionalidad* del texto a 64 ya que la pantalla ocupará nada más que una porción del texto que se irá *scrolando* horizontalmente, según escribimos. La otra es que también dispone de comandos para buscar y sustituir palabras por otras distintas.

### Resumiendo

Como veis las ventajas de ambos procesadores son grandes y parecidas por lo que puede resultar un poco difícil al principio decidirse por alguno en particular, aunque, prácticamente cualquiera de los dos puede valer para una pequeña aplicación que precise un usuario de Spectrum.

# Alistate a & Juegos ESTRATEGIA LA BATALLA DE INGLATERRA ha comenzado

Todas las unidades de la RAF están bajo tu mando, y la Luftwaffe —tu ordenador— intentará neutralizarlas. El destino del mundo libre depende de ti.



Oferta especial  
hasta el 31  
de noviembre:  
PIDE TRES NUMEROS  
Y PAGA  
SOLO DOS.

ENVIE HOY MISMO ESTE CUPON AL APARTADO 232 DE ALCOBENDAS (Madrid)



- Deseo recibir en mi domicilio tres ejemplares de **Juegos & Estrategia** al precio especial de 2.255 ptas., lo que me supone adquirir tres y **pagar sólo dos**. Marca los tres ejemplares que deseas con una cruz.
- Deseo recibir un solo ejemplar de **Juegos & Estrategia** al precio de 1.125 ptas. Marca con una cruz el ejemplar que deseas recibir.

**Spectrum**

- N.º 1  Arnhem  
N.º 2  Ratas del Desierto  
N.º 3  OTAN Alerta  
War Zone  
Especial 1  Elecciones Generales  
N.º 4  Su mejor hora (La batalla de Inglaterra)

**Amstrad**

- Arnhem  
 Ratas del Desierto  
 Teatro de Europa  
War Zone  
 La batalla de Inglaterra

**Commodore**

- Teatro de Europa  
 La batalla de Inglaterra

Fecha de  
nacimiento

NOMBRE \_\_\_\_\_

DIRECCION \_\_\_\_\_

LOCALIDAD \_\_\_\_\_

C. POSTAL \_\_\_\_\_

TELEFONO \_\_\_\_\_

PROVINCIA \_\_\_\_\_

PROFESION \_\_\_\_\_

Forma de pago:

Talón bancario a nombre de Hobby Press, S. A.  Giro Postal a nombre de Hobby Press, S. A., n.º de giro

Tarjeta de crédito: Visa n.º \_\_\_\_\_

Master Charge n.º \_\_\_\_\_

American Express n.º \_\_\_\_\_

Fecha de caducidad de la tarjeta \_\_\_\_\_

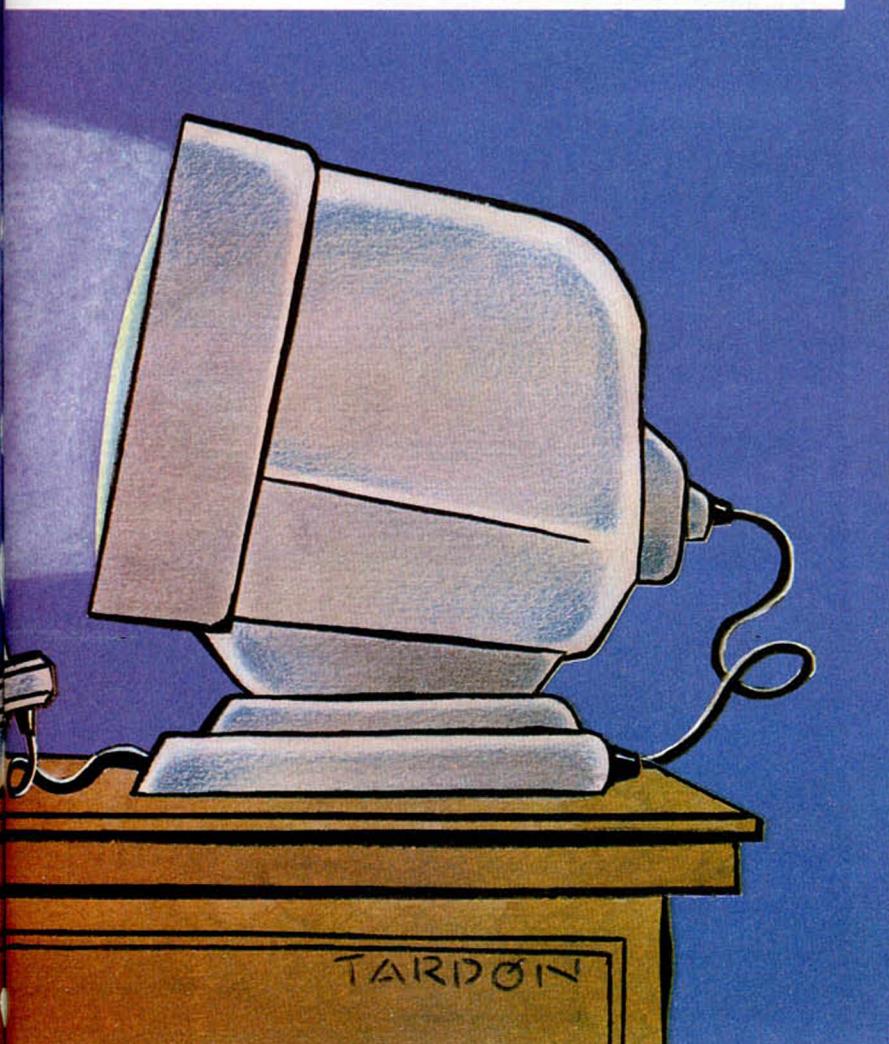
Fecha y firma \_\_\_\_\_

**Víctor PRIETO**

# TECNO **DELINCUENTES**

EL DELITO INFORMATICO ACOMPAÑA A LOS ORDENADORES DESDE SU NACIMIENTO. EN UN PRINCIPIO ERA UN COTO PRIVADO DE LOS PROGRAMADORES EN LA INDUSTRIA, PERO LA LLEGADA DE LOS ORDENADORES PERSONALES Y EL MODEM, HAN CREADO UNA NUEVA GENERACION DE TECNODELINCuentes.





**D**esde que los bits de datos informátizados reemplazan al papel como medio más importante para almacenar grandes cantidades de datos e información, nuestra sociedad está presenciando el advenimiento de una nueva raza de delincuentes.

Popularmente conocidos como los ladrones tecno, los protagonistas del delito informático desafían cualquier tipo de clasificación.

Su procedencia, localizada en cualquier profesión u ocupación, y los métodos utilizados en su nuevo campo de acción, son tan diversos como numerosos, y en la mayoría de los casos tan eficaces, que resulta difícil su detención.

Los requisitos mínimos para ser uno de esos tecnodelincuentes, se limitan a un conocimiento rudimentario del funcionamiento de los ordenadores, y un fuerte instinto delictivo.

Sorprendentemente, tener acceso a un ordenador no es vital en todos los casos, como demostró un cliente de un determinado banco de los Estados Unidos, reemplazando un error del banco en los ingresos, por otros con su propio número de cuenta, magnéticamente codificado sobre ellos.

Después de liquidar su cuenta al día siguiente, y retirar el balance en metálico, el distinguido cliente se hizo con una cifra de 75.000 libras (17 millones 250.000 pesetas).

### ELUDIR LAS MEDIDAS DE SEGURIDAD

La información sobre ordenadores es sorprendentemente fácil de conocer, exceptuando la referente a la entrada en sistemas de seguridad. El tema es enseñado en colegios y es objeto de numerosos artículos de prensa. Incluso documentación referente a métodos de operación para diferentes máquinas, se guarda raramente en secreto.

De hecho, aunque las medidas de seguridad internas de los ordenadores sean totalmente inexpugnables, a menudo es muy fácil pasarlas por alto.

Por ejemplo, donde están instaladas las llamadas líneas de comunicaciones

de seguridad, el número de teléfono puede no estar contenido en las guías, y no ser listado por caminos internos. Pero aún aparece en contratos de instalación, en facturas, y en ocasiones garabateado en las notas de los ingenieros de instalación.

Cuando se está en posesión de ese tipo de información, la diferencia entre cometer o no el acto delictivo, es puramente un asunto de poder o no resistir la tentación, y en el caso de los ordenadores, ésta es desmesuradamente grande.

## SIN VIOLENCIA

Hay una diferencia fundamental entre un asalto pistola en mano a una sucursal bancaria y el robo por ordenador, ya que éste se realiza sin el menor tipo de violencia, y tiene la ventaja de que no se delata por sí mismo.

De hecho, los más cualificados investigadores de este tipo de delitos, han abandonado la pretensión de ser capaces de descubrir el fraude hasta sus últimas consecuencias.

La cantidad de volumen de datos almacenados por las grandes compañías, hace imposible revisar cada trans-

sacción individualmente, incluso en el caso de que la sospecha de fraude sea completamente segura.

La política de las compañías, a menudo parece limitarse a ocultar los desfalcos, siempre que se encuentren dentro de unos límites admisibles.

Por ejemplo, en el caso del uso ilegal de las tarjetas de crédito, que desencadenan un gran número de transacciones electrónicas de fondos, unas perdidas que se encuentren dentro del 0,05 por 100 de los ingresos netos, se consideran como aceptables.

Incluso en el caso de que los límites sean sobrepasados, el costo efectivo de la acción tomada, es raramente dirigido hacia el criminal.

## EL TECNO-DELINCUENTE FRENTE A LA SOCIEDAD

La actitud social hacia el crimen computerizado, también hace aumentar su atractivo. Muchos robos hechos desde una máquina (especialmente cuanto éstos tienen como resultado pequeñas cantidades de dinero, o el he-

cho de eludir impuestos o gastos) es considerado como trivial.

El hecho de que esta actitud esté o no reflejada en el Código penal, no está claro todavía. Sin embargo, las sentencias por delitos cometidos con ordenador, son frecuentemente mucho menos severas que las de los casos dependientes de la brigada de investigación criminal.

Tomemos, por ejemplo, el caso de Jerry Neal Schneider, o el famoso desfalco de la Equity Funding.

En el primero, Schneider, un joven empresario (18 años) residente en Los Angeles, formó una compañía de distribución de material electrónico con un stock inexistente, introduciéndose telefónicamente en el ordenador de una compañía local de IBM, para desviar de su almacén las existencias necesarias para atender su cartera de pedidos.

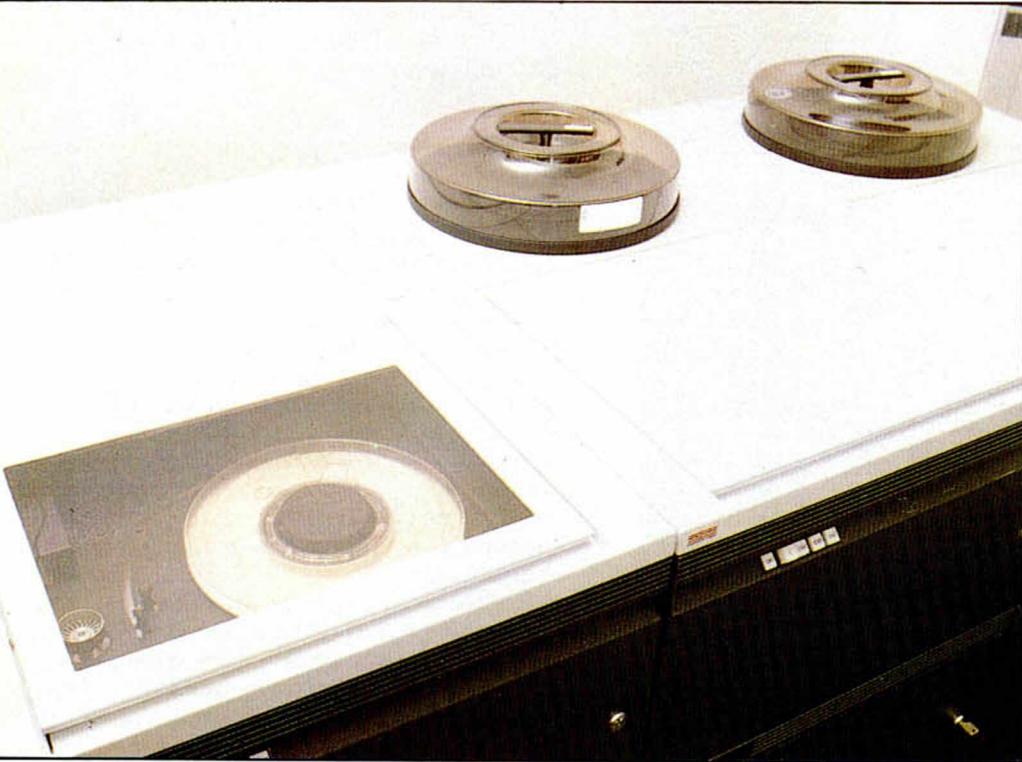
Después de varios meses de negocio, Jerry fue denunciado por un cómplice que se ocupaba de atender los pedidos nocturnos. Sin embargo, a pesar del hecho de haberse apropiado indebidamente del valor aproximado de un millón de dólares en material, el joven delincuente fue condenado a pagar una multa de unas 120.000 ptas. y a dos meses de condena, de los cuales solamente llegó a cumplir 40 días, antes de volver al mundo de los negocios como un consultor de seguridad de sistemas informáticos.

## UNA ESTAFA A LO GRANDE

El escándalo del Equity Funding (1972), considerado como el fraude del siglo, fue realizado por los altos ejecutivos de una empresa americana de rápido crecimiento durante el transcurso de ocho años.

Los ordenadores fueron usados, entre otras cosas, para la expedición de 64.000 pólizas de seguros de vida falsas, las cuales fueron vendidas también por medio del ordenador a sus coaseguradores.

Cuando fue descubierto, el fraude costó a los accionistas 600 millones de libras (138.000 millones de pesetas) y un valor perdido en las pólizas aseguradas de un billón de dólares.



Muchas de las víctimas quedaron arruinadas, pero los 24 responsables de la estafa solamente recibieron penas desde los ocho años de prisión, hasta multas y libertad provisional.

## EL SILENCIO DE LAS ENTIDADES AFECTADAS

Incluso si un delito informático es descubierto y el causante identificado, su persecución raramente se lleva a cabo.

Las víctimas son reacias generalmente a llevarlo ante los tribunales. Las entidades bancarias, un blanco muy popular entre los tecnodelincuentes, consideran mucho más dañina la publicidad que acompaña a un litigio de estas características, que el valor del fraude perpetrado.

En otros casos, son los mismos directivos de las compañías afectadas, los que ocultan el fraude, para evitar acusaciones de negligencia por parte de los accionistas.

Otro motivo de disuasión es el costo de los procedimientos legales, establecer la naturaleza exacta del delito y probar la existencia del mismo, especialmente en el caso de sofisticados robos por ordenador, requiere frecuentemente una desmesurada cantidad de tiempo, dinero y esfuerzo.

También hay que tener en cuenta que los jurados en un caso de estas características, no son expertos en el campo de los ordenadores, teniendo considerables dificultades para llegar a interpretar los hechos y pruebas aportados por los consultores informáticos.

A causa de la gran difusión de los ordenadores, las oportunidades de cometer delitos informáticos se han incrementado enormemente. Oficinas, secretarías, e incluso personal de limpieza de oficinas, tienen acceso a las terminales.

## EL PERSONAL DE LAS CONSOLAS

Precisamente por las características que debe reunir el personal a cargo



de los ordenadores, las empresas dedicadas a la selección, tienden a reclutar gente con una mente penetrante y un especial sentido de la precisión, características que les hacen erigirse en los candidatos más adecuados para intentar eludir las medidas de seguridad como un desafío a su intelecto.

La mayoría de los delitos por ordenador son de carácter oportunista, gente que no busca un beneficio financiero pero tiene la oportunidad de introducirse en el sistema de seguridad debajo de sus narices.

Muchos programadores podrían quebrantar un sistema como un acto de inofensiva malicia, sólo por el orgullo de demostrarse a sí mismos que pueden hacerlo, pero una vez dentro la tentación es demasiado grande como para no aprovechar la ocasión.

De nuevo el crimen informático es difícil de demostrar. El más famoso fraude de redondeo de la historia bancaria es un caso en esta línea.

El autor, que trabajaba para un gran banco, realizó un programa en el cual al ser calculado el interés en la cuenta de un cliente, las pequeñas cantidades sobrantes del redondeo no eran abonadas a las cuentas individuales, sino que eran transferidas a

una cuenta ficticia al final del archivo de clientes.

El fraude solamente pudo ser descubierto por accidente, cuando el presidente de la compañía, con objeto de demostrar las maravillas del sistema, sacó el saldo de la primera y la última cuenta.

## EL ERROR DE LA MAQUINA

La idea general de que los ordenadores por naturaleza son propensos a cometer errores, también trabaja en favor de los delincuentes.

Ahondando en esta técnica del error mecánico, tres empleados de New Securities, se las arreglaron para exprimir las cuentas de sus clientes, hasta el punto de conseguir al menos medio millón de dólares en varios años.

Si un cliente notaba algún error en el balance de su cuenta, el error en el sistema de ordenadores era el responsable.

También y no sorprendentemente, cuando los errores ocurren a favor de la cuenta de algún cliente, pocos son dados a informar de ello.

Llevado a casos extremos, nos encontramos con el de un contable que accidentalmente había cargado en su propia cuenta cerca de un millón de dólares, que se arregló para gastar antes de que el banco descubriese su error. Fue acusado con el cargo de robo.

## LA INFORMACION COMO OBJETO DE ROBO

El crimen informático toma millares de formas, y no solamente está limitado a casos claramente incluidos en el fraude y el desfalco.

El objetivo puede ser, por ejemplo, conseguir la propiedad de una compañía. En una ciudad de los Estados Unidos, el crimen organizado modificó los datos de pedidos de clientes de un ordenador, para eliminar 200 cajas de coches del inventario de una compañía de ferrocarriles.

Los causantes del delito, devolvieron con toda celeridad los coches a sus propietarios originales, consiguiendo desestimular a la empresa distribuidora.

Otra forma delictiva la constituye, no el robo electrónico de fondos, sino la apropiación indebida de información.

Los archivos de clientes son los favoritos. Uno de los casos récord en este campo, es el perpetrado por parte de operadores de ordenador que trabajando para la Encyclopædia Británica, vendieron la alarmante cantidad de dos millones de nombres y direcciones. El precio de tal información llegó a alcanzar más de un millón de libras. Incluso las grabaciones de los censos del gobierno no son inviolables.

El espionaje industrial es también muy común entre los ladrones tecno, la facilidad con que los datos pueden ser duplicados sin dejar rastro, hace que el robo de secretos comerciales, planes presupuestarios e información de negocios, tenga un mercado ávido de información entre las compañías competidoras.

Los programas de ordenador en sí mismos, también son objeto de la delincuencia informática, propietarios particulares de software, cuya obra es el fruto del intenso trabajo de varios años, caen dentro de las redes del ladrón tecno.

## IMPIDIENDO EL ACCESO FISICO

La variedad de métodos de protección de los ordenadores contra la entrada de intrusos, adquiere multitud de formas cada una de ellas basada en diferentes conceptos.

A medida que más y más microordenadores aparecen en las oficinas, así como unidades y terminales inteligentes, la posibilidad de encerrar el ordenador bajo llave, se hace cada vez más difícil.

Incluso cuando el departamento de ordenadores esté efectivamente aislado del mundo exterior, el ladrón tecno siempre puede recurrir a modificar el data antes de ser introducido.

En el fraude de Equity Funding tenemos un caso típico. El departamento de programación era alimentado con información enteramente ficticia por parte de los directivos, y los clientes de la corporación cometieron el error de tomar los resultados del ordenador como un lema de fe.

Otro ejemplo es el del consultor de seguridad de ordenadores, cuya estrategia favorita era entrar en el departamento de oficinas, llenar uno de los impresos en blanco que andaban desperdigados por allí, y dejarlo caer en el suelo.

Invariablemente el impresor que contenía una orden de pago del departamento económico de la empresa, dirigida a la dirección del consultor, era recogido y procesado.

El consultor podía entonces retirar su cheque y de esta forma justificar sus servicios.

## CLAVE SECRETA

A menudo para entrar en un sistema, el usuario necesita teclear una palabra clave, en algunos sistemas ésta es claramente visible al introducirla desde el teclado, en este caso el tecnodeinciente no tiene más que observar al usuario en el momento de teclearla para obtener la información deseada.

Cuando la palabra clave no es impresa en la pantalla, y no se encuen-



tra garabateada en notas de instalación o manuales de uso, es necesario la utilización de técnicas mucho más sofisticadas.

Una de ellas, demostrada por un estudiante escocés, es escribir un procedimiento que actuando vía telefónica pueda memorizar la palabra clave y simular un fallo en el sistema. Los sorprendidos usuarios introducirán por primera vez su clave en el sistema telefónico, descubriendo que hay un fallo en el sistema y luego volverán a intentarlo una vez más, esta vez en lugar correcto.

Las bases de datos de información basadas en líneas telefónicas han ele-

vado el refinado hecho de descubrir la clave de acceso a la categoría de un verdadero arte.

Existen listas en las cuales se dan los diez nombres clave más populares, habiéndose desarrollado complicados algoritmos capaces de calcular las permutaciones más probables de caracteres alfanuméricos.

## LLAVES ELECTRÓNICAS

Generalmente tienen la forma de una tarjeta plástica, con información

codificada en una cinta magnética contenida en una banda en cualquiera de sus caras.

Existen varios métodos de alterar dichas tarjetas: el más sofisticado consiste en usar un pantógrafo electrónico, para extraer la información almacenada. El menos complicado, usado con tarjetas empleadas en ciertos servicios públicos, como teléfonos o billetes de transportes, consiste en utilizar un imán para borrar la banda magnética.

## ENCRYPTION

Su uso se limita a la protección de información, especialmente la que ha de ser enviada a través de las redes de comunicaciones públicas, o como un método de protección de las líneas privadas.

Ello implica un proceso de codificación y decodificación de textos usando algoritmos específicos y una única clave.

Teniendo en cuenta que los resultados son solamente conocidos por el transmisor y el receptor de la información, incluso si los algoritmos son conocidos, el código permanece seguro.

La encryption, constituye una formidable barrera para el ladrón tecnó, debido a que descifrar el código requiere un ordenador de considerable poder. En cambio, las agencias del gobierno pueden descifrarlo como si se tratara de un mensaje en morse.

La Agencia Central de Seguridad, ha creado un sistema de codificación de siete dígitos, del cual se dice que no puede ser decodificado ni usando el ordenador más potente.

Lo cierto es que si la agencia está capacitada para codificar siete dígitos, no puede abordar las claves de ocho dígitos, con lo cual su campo solamente se reduce a la información puramente comercial.

Ha quedado claro que los métodos de protección van desde las medidas más elementales, hasta los sistemas más sofisticados en los que la tecnología empleada hace imposible la entrada de cualquier intruso, poniendo cada vez más difícil la tarea del tecnodelincuente, que tiene que suplir con ingenio la falta de medios.



Alejandro JULVEZ  
Marcos ORTIZ

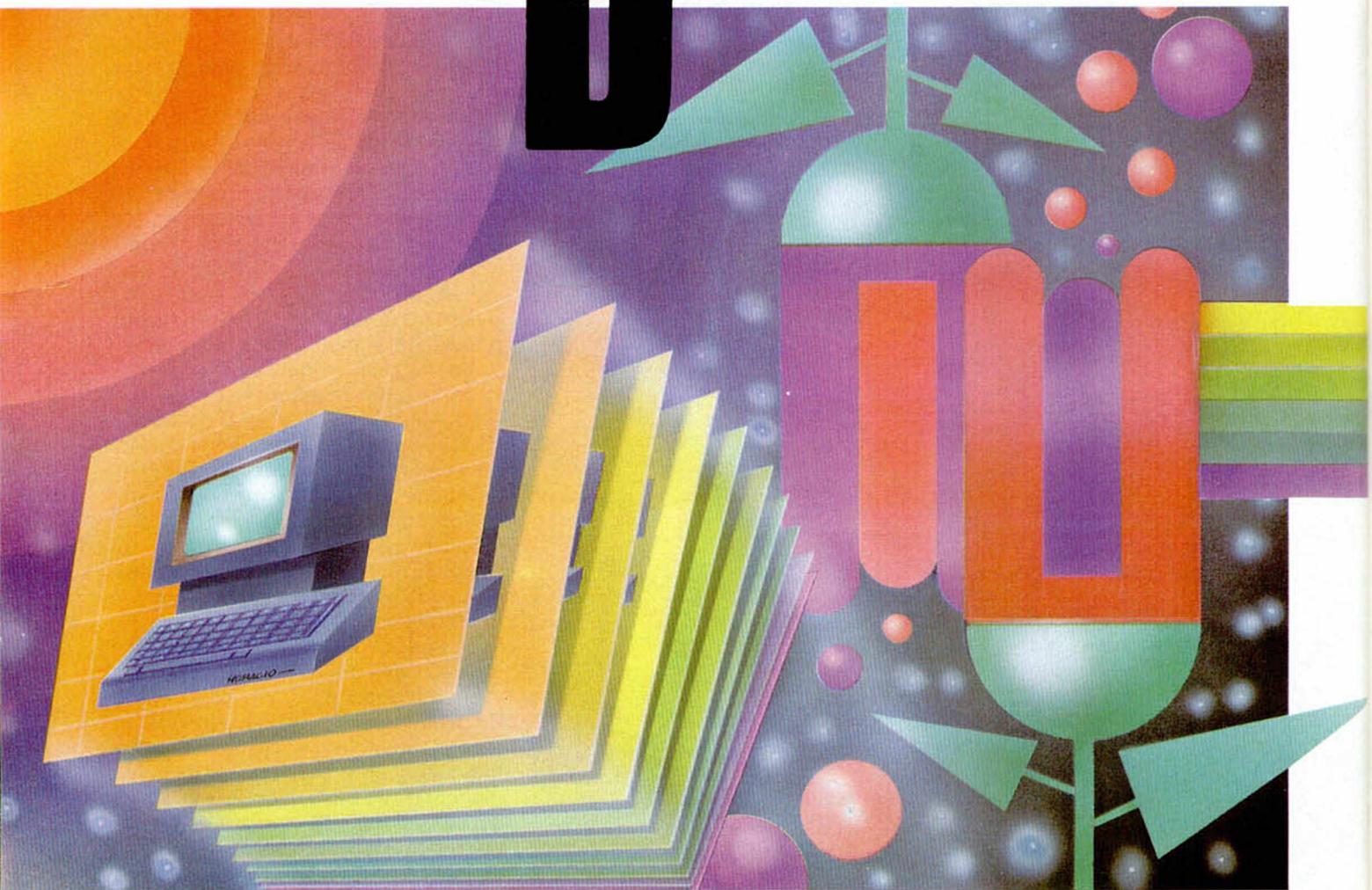
*Una estructura de datos no es otra cosa que un conjunto de datos con una organización determinada. Pues bien, para conocer el mecanismo de creación de estos tipos de estructura, os ofrecemos este amplio artículo en el que os explicamos cómo se representan en el ordenador.*

# ESTRUCTURA DE DATOS

Todos sabemos la gran cantidad de información que un ordenador moderno es capaz de almacenar y procesar.

En muchos casos esa información representa en cierta forma una abstracción de una parte del mundo real, y consiste en una selección de datos de la realidad, en concreto ese conjunto de datos que consideramos básico para la solución del problema y a partir del cual obtenemos los resultados deseados.

Es evidente la importancia del lenguaje de programación que se utiliza, de forma que nos permita el mayor grado de abstracción posible. En nuestro caso



contamos con un lenguaje que no ofrece muchas posibilidades en este sentido. Por supuesto, nos referimos al Basic.

Muchos de vosotros conoceréis el Pascal, un lenguaje que nos ofrece ciertos modos de definición de datos (en la mayoría de los casos se definen nuevos tipos de datos, en función de otros definidos previamente, y se dice que están estructurados).

Los tipos constituyentes definidos previamente, a su vez, pueden estar estructurados, con lo que podemos construir jerarquías de datos. De cualquier manera el componente último de una estructura, por muy compleja que sea, debe ser un componente atómico, es decir, elemental.

La mayoría de los ordenadores contienen lo que se llama tipos elementales normalizados. Comprenden los números reales, enteros, valores lógicos y un conjunto de caracteres de escritura, también números fraccionarios.

El tipo de valores entero es un subconjunto de los números enteros, cuyo tamaño depende mucho del ordenador en concreto. Las operaciones que se realizan entre valores de este tipo son exactas y se corresponden con las leyes de la aritmética.

El tipo real es un subconjunto de los números reales.

La aritmética real produce cierta imprecisión, dentro del error producido por el redondeo, al realizarse el cálculo sobre un número finito de dígitos.

El tipo lógico tiene valores verdadero o falso o bien TRUE y FALSE. En Basic no existe este tipo de datos y por consiguiente no existen variables de este tipo, aunque podemos asignar a una

variable numérica una expresión booleana, como por ejemplo, LET A=5=3, esta expresión es falsa luego la variable A tomaría el valor 0; si la expresión hubiese sido verdadera, la variable A tomaría el valor 1.

El tipo Char comprende el conjunto de caracteres imprimibles.

En este caso depende mucho del ordenador del que se trate, para saber qué conjunto de caracteres emplea. El más usado es el código (ISO) International Standard Organization y el ASCII (American Standard Code for Information Interchange). Sobre estos tipos elementales normalizados se construyen otros tipos más complejos, por ejemplo los arrays, que no es otra cosa que una estructura de datos cuyos componentes son homogéneos, son todos del mismo tipo elemental y se seleccionan por sus nombres fijos.

Un array es una estructura de tipo aleatorio, todos sus componentes pueden seleccionarse arbitrariamente y son igualmente accesibles. Para seleccionar un componente aislado, el nombre del array se amplía con un índice de selección del componente que indica a su vez la posición que ocupa un elemento dentro del array.

Existen más estructuras de datos, pero en esta ocasión vamos a tratar tres estructuras muy importantes: pilas, colas y listas.

## PILAS

Una pila es un conjunto de datos que únicamente pueden introducirse o extraerse por un extremo.

Es muy común a la hora de explicar el concepto de pila la analogía con la vida de cada día.

En una cafetería, los platos limpios para ser utilizados por los clientes se colocan en una pila en el mostrador. La forma más conveniente de utilizar un plato es coger el que está en lo alto de la pila. A medida que se van utilizando los platos se van sirviendo desde lo alto de la pila y cuando los platos utilizados se han lavado, se vuelven a colocar en lo alto de la pila.

Por tanto, el último plato que entró será el primero en salir.

Esta regla se llama LIFO en inglés «last in, first out», que es lo que caracteriza a una pila como estructura de datos.

En la pila se pueden realizar dos operaciones:

1) Extraer un elemento por la cima, en cuyo caso el elemento situado a continuación del extraído pasa a ocupar la cima.

2) Introducir un elemento por la cima, con lo que este elemento pasa a ocupar la cima.

A partir de ahora vamos a necesitar un elemento nuevo llamado puntero, que nos va a servir para denotar la posición de una variable en el ordenador.

Ante la imposibilidad de definir punteros en Basic, tendremos que crear todas las estructuras de datos que vamos a estudiar en este artículo, sobre matrices, de esta forma el índice de un elemento de la matriz indica su posición dentro de dicha matriz.

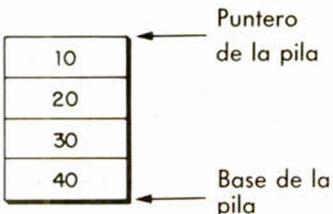
Cuando se almacena una pila en memoria, sus elementos ocupan posiciones consecutivas y el puntero señala la cima de la pila.

El puntero se modifica cada vez que se realiza una operación sobre la pila. El otro extremo de la pila está fijo y se llama base.

Para aquellos de vosotros que conozcáis el Código Máquina el concepto de pila debe seros familiar, el puntero de la pila es el registro SP y las instrucciones de introducir y extraer un elemento son respectivamente PUSH y POP.

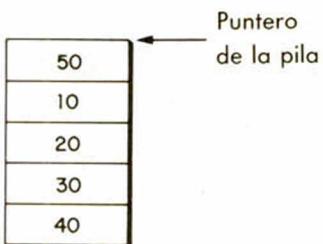
Veamos a continuación el efecto gráfico que tiene sobre una pila la ejecución de las dos operaciones.

## PILA INICIAL

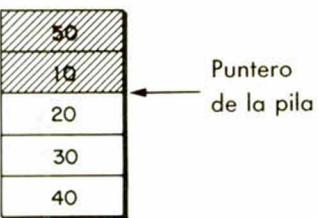


Realizamos una operación de introducción de un nuevo elemento por la cima:

### INTRODUCIR 50



Si realizamos dos extracciones consecutivas:



La parte rayada es información que ya no pertenece a la pila, aunque sigue permaneciendo en memoria, es pues el puntero quien indica el comienzo de información perteneciente a la pila.

El puntero se incrementa o decremente en una unidad dependiendo de la

operación concreta que realizamos sobre la pila. Es importante tener en cuenta, que la pila va creciendo a medida que se van introduciendo por la cima elementos nuevos. Esto en el caso del Código Máquina puede ser un problema si la pila se extiende sobre un programa concreto, pero no existe limitación en su crecimiento. Para nosotros el crecimiento de la pila sí supone un pequeño dato a tener en cuenta, porque al soportar la pila sobre una matriz, debemos controlar que la pila no se haga mayor que la matriz. De esta forma la pila va creciendo hacia arriba y está llena cuando el puntero apunte al primer elemento.

Estará vacía cuando apunte al último o base de la pila.

Hay que hacer notar que el puntero siempre apunta al índice del primer elemento libre de la matriz.

Al final, aparecen unos listados que realizan las operaciones sobre una pila y se usa una variable que conectará con el programa principal para indicarnos si la operación se realizó con éxito.

Correcto=0 operación incorrecta.

Correcto=1 operación correcta.

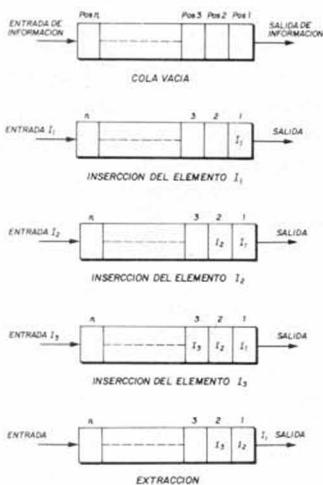
Debemos pasar a estas subrutinas una serie de parámetros desde el programa principal, como puede ser en el caso de introducir un elemento, el elemento en cuestión o en el caso de la extracción recibiremos el elemento extraído. La pila se soporta sobre una matriz T con DIM T (200). Crear una pila vacía es simplemente asignar al puntero de la pila el valor máximo, 200 en este caso.

Las subrutinas que tratan la pila son (6000-6270).

## COLAS

Una cola es una estructura de datos que se caracteriza porque sus elementos están ordenados y la inserción de ellos se realiza por la parte posterior y las extracciones por la parte anterior.

Tiene estructura FIFO (first in, first out), primero en entrar, primero en salir. Podemos realizar dos operaciones sobre la cola: inserción de un nuevo elemento por la parte posterior o extracción de un elemento por la anterior. Veamos un ejemplo gráfico del funcionamiento de una cola:



con el elemento  $I_3$ . Al realizar una extracción, la información contenida en la cola se desplaza una posición hacia la salida, es decir, el elemento  $I_1$  sale de la cola, la información  $I_2$  se desplaza a la posición ocupada por  $I_1$  y  $I_3$  a la ocupada por  $I_2$ .

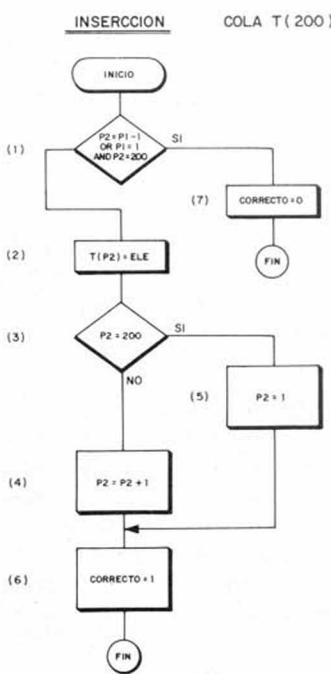
Una buena forma de tratar la cola sobre una matriz, es hacer que tome una estructura circular, es decir, que dé la vuelta.

Esto es debido a que a medida que se van sucediendo las operaciones de extracciones e inserción la cola va dejando espacios libres. Para evitar este problema y por supuesto el de tener que desplazar toda la información contenida en la cola cada vez que realizamos una operación sobre ella, vamos a utilizar dos punteros. Un primer puntero ( $P_1$ ) que indica la posición del elemento situado en el extremo anterior de la cola y el otro ( $P_2$ ) indica el espacio disponible en la parte posterior de la misma.

Por tanto, podemos decir

que la cola está llena cuando en la matriz sólo queda un elemento.

A continuación aparece un pequeño diagrama de flujo del proceso de inserción y extracción con los punteros  $P_1$  y  $P_2$  como punteros primero y segundo de la explicación anterior.

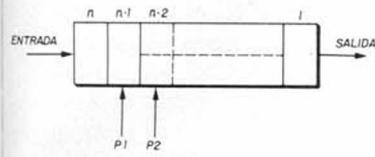
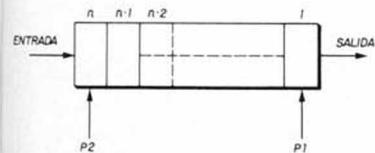


A continuación analizamos los pasos que hemos seguido en cada operación:

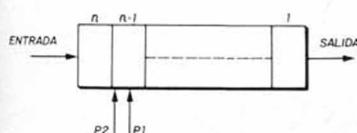
### INSTRUCCION.

#### PASO EXPLICACION

- ¿Cola llena? Son los casos expuestos en las figuras A y B.
- Inserción del elemento contenido en la variable ELE.  $T(P_2)$  es la posición de la cola indicada por el puntero  $P_2$ .
- ¿Está el puntero  $P_2$  al final de la cola?
- Si está al final hacemos  $P_2 = 1$ , da la vuelta.
- No está. Queda una posición menos.
- Operación correcta. El elemento ha sido insertado correctamente.
- Operación incorrecta. Cola llena.

Situación en la que  $P_2 = P_1 - 1$ Situación  $P_2 = n$  and  $P_1 = 1$ . En nuestro caso  $n = 200$  tamaño límite.**PASO EXPLICACION**

- (1) ¿Cola vacía?  $P_2 = P_1$ , o ¿cola no vacía?  $P_2 < > P_1$ .
- (2) La cola no está vacía. Extracción del elemento de la cola indicado por el puntero  $P_1$ .
- (3) ¿Está el puntero  $P_1$  al final de la cola (200)?
- (4) No está al final, incrementar el puntero.
- (5) Si está al final, dar la vuelta poner el puntero a 1.
- (6) Operación correcta.
- (7) Operación incorrecta, cola vacía.



- Cola vacía  $P_2 = P_1$   
Cola no vacía  $P_2 < > P_1$ .

Las subrutinas que realizan el tratamiento de la cola son (7000-7300).

Las colas son estructuras de datos que se utilizan normalmente en aplicaciones en tiempo real, en comunicaciones de datos y en programas de sistema.

Existe otra forma de representar la pila sobre la matriz, es la forma antes explicada en las figuras aclarativas sobre el funcionamiento de la cola:

— almacenar la cola en una matriz como hasta ahora;

— la salida de la cola se fija al elemento superior de la matriz;

— la inserción de un nuevo elemento se realiza por detrás, en la posición señalada por un indicador que nos dirá el espacio disponible tras el extremo posterior de la cola;

— cada vez que extraemos un elemento del extremo inicial todos los demás se desplazan una posición dentro de la cola, como se vio antes;

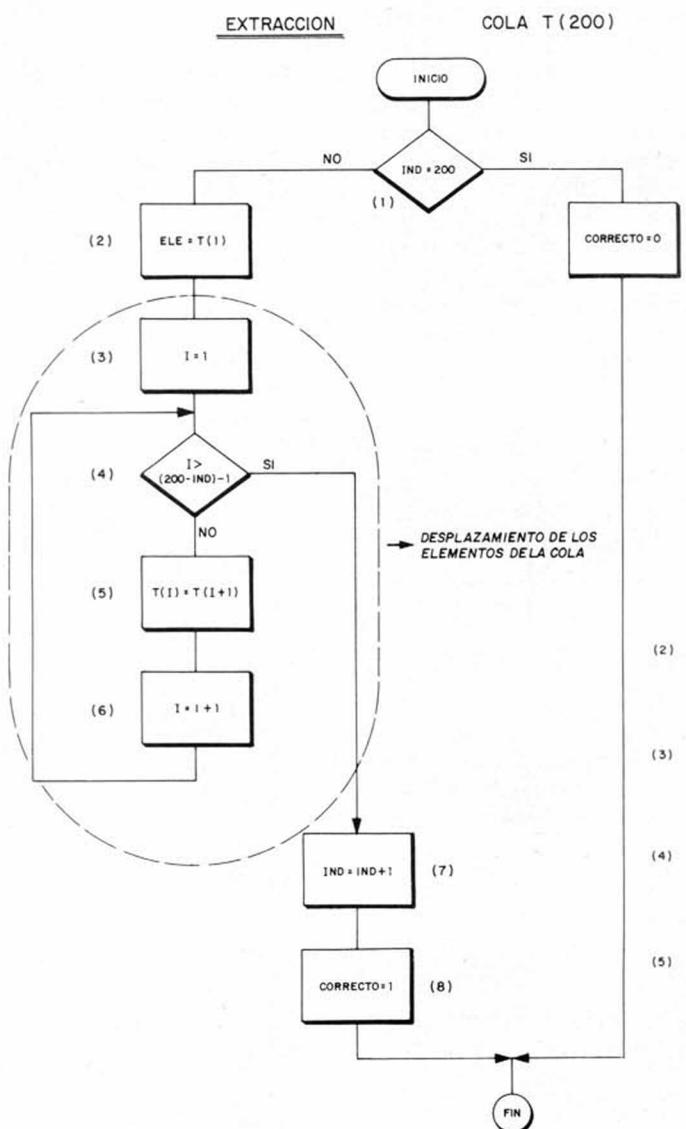
— vamos a realizar la

cola sobre una matriz de 200 elementos como la anterior.

A continuación os ofrecemos unos diagramas de flujo que esclarecen lo anteriormente explicado:

**LISTAS**

La forma más simple de relacionar una serie de elementos es alinearlos formando una única lista, en este caso se necesita únicamente un enlace por elemento para referenciar a su sucesor.

**PASO EXPLICACION**

- (1) Cola vacía?  
 $IND = 0$ . Indica que no hay espacio libre.
- (2) Cola no vacía. Cálculo de la posición donde vamos a insertar.
- (3) Inserción del elemento contenido en la variable ELE.
- (4) Hay una posición menos libre, tras la inserción.
- (5) Operación correcta.
- (6) Operación incorrecta. Cola llena.

**PASO EXPLICACION**

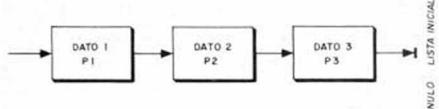
- (1) Cola vacía?
- (2) Cola no vacía. Extracción del primer elemento.
- (3) (4) (5) (6) Desplazamiento de  $I > (200-IND)-1$  indica que hemos desplazado todos los elementos hasta el último de los existentes.
- (7) Un espacio más.
- (8) Operación correcta.
- (9) Cola vacía. Operación incorrecta.

Las subrutinas que realizan estas operaciones son la (9000-9220).

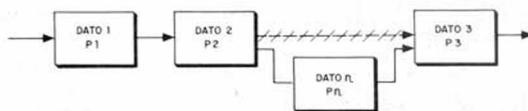
Luego, la lista es una estructura de datos que contiene un conjunto de ellos almacenados con cierto orden. Los elementos pueden insertarse o extraerse en cualquier punto de la lista. Un elemento en la lista consta de dos partes, el dato y su puntero que hará referencia al sucesor. En el caso de que un dato no tenga un elemento de lista sucesor, su puntero será nulo. De esta forma gráfica la lista tiene el siguiente aspecto:



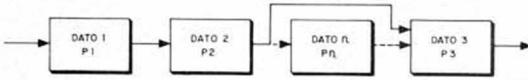
Sobre una lista pueden realizarse dos operaciones, insertar un elemento tras otro dado y extraer un elemento situado tras otro dado. Veamos el efecto gráfico de ambas operaciones:



- **Inserción de un elemento**



- Extracción del elemento insertado



Y ahora, veamos qué ocurre en cada caso. En la inserción lo que ha pasado es que el puntero del elemento tras el que ha de insertarse el nuevo, tiene necesariamente que modificar su contenido para apuntar a este nuevo elemento y, por consiguiente, el nuevo elemento debe tomar el valor del puntero del elemen-

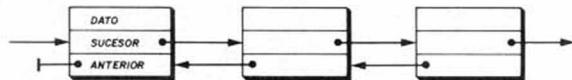
to antecesor para apuntar al sucesor.

El caso de la extracción es similar, el puntero del elemento anterior al extraído debe de apuntar ahora al elemento sucesor del elemento extraído, y debe ser ahora el puntero del elemento extraído, el que pase a ser el puntero de su antecesor.

Para representar una lista lo hacemos igual que en ocasiones anteriores, sobre una matriz.

En este caso habrá una matriz de datos, y dos matrices de punteros. Vamos a realizar la lista con una estructura como la que aparece a continuación:

Z punteros por elemento:  
uno referencia al sucesor y el  
otro al anterior.

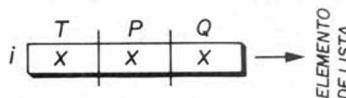


En este tipo de estructura, se necesitan dos punteros  $P_1$  y  $P_2$ , que nos indicarán el espacio libre y el comienzo de la lista en la matriz, respectivamente.

Datos: T(200)

Puntero sucesor: P(200)

Puntero antecesor: Q(200)



Observando que un elemento de lista en este caso estará compuesto por  $T(I)$ ,  $P(I)$ ,  $Q(I)$ , inicializaremos la lista con una profundidad de 200 elementos.

Las subrutinas que tratan las operaciones sobre la lista son las (9300-9980).

La subrutina de inserción necesita un parámetro puntero del elemento tras el que ha de insertarse, PUN.

## La subrutina de extracción

igualmente necesita el mismo parámetro, para conocer el elemento tras el que ha de extraerse. En el caso de que el puntero PUN sea igual a  $\emptyset$  no será posible la extracción puesto que no habrá ningún elemento sucesor a éste.

Para el caso de la inserción, si el puntero de espacio libre es igual a  $\emptyset$ , significa que no hay espacio en la lista para la inserción.

La subrutina de localización de un elemento en la lista sólo necesita como parámetro de entrada, el dato a buscar en la lista, devolviendo el puntero del elemento si es que lo encuentra. En el listado sólo devuelve el puntero de la matriz P, punteros a elementos sucesores.

## **CREAR PIJA VACÍA**

```

6000 REM CREAR PILA VACIA
6001 LET P=200
6020 RETURN
6100 REM INSERCCION
6110 IF P=1 THEN GO TO 6140
6120 LET CORRECTO=0
6130 RETURN
6140 LET T(P)=ELE
6150 LET CORRECTO=1
6170 RETURN
6200 REM EXTRACCION
6210 IF P<200 THEN GO TO 6240
6220 LET CORRECTO=0
6230 RETURN
6240 LET P=P+1
6250 LET ELE=T(P)
6260 LET CORRECTO=1
6270 LET P=UX
6300 REM COLA
6310 LET P2=P
6320 LET P1=1
6340 RETURN

7000 RETURN
7040 REM INSERCCION
7050 REM INTRODUCCION
7060 GO TO 7090
7070 LET CORRECTO=0
7080 RETURN
7090 IF T(P2)=ELE
7100 IF P2=200 THEN GO TO 7130
7110 LET P2=P
7120 GO TO 7140
7130 LET P2=1
7140 LET CORRECTO=1
7150 RETURN
7200 REM EXTRACCION
7210 IF P2=1 THEN GO TO 7240
7220 LET CORRECTO=0
7230 RETURN
7240 LET ELE=T(P1)
7250 IF ELE=UX THEN GO TO 7260
7260 LET P1=P+1
7270 GO TO 7290
7280 LET P1=1
7290 LET CORRECTO=1
7300 RETURN
7300 REM CREAR COLA VACIA
7310 LET IND=200
7320 LET P=UX
7330 REM INSERCCION
7340 IF IND<0 THEN GO TO 9070
7350 LET CORRECTO=0
7360 RETURN
7370 LET EXS=(200-IND)+1
7380 LET T(IND)=ELE
7390 LET CORRECTO=1
7400 RETURN
7400 REM EXTRACCION
7410 IF IND<0 THEN GO TO 9160
7420 LET CORRECTO=0
7430 LET ELE=T(1)
7440 LET IND=IND-1
7450 LET T(1)=T(IND+1)
7460 LET T(IND+1)=ELE
7470 LET CORRECTO=1
7480 RETURN
7480 REM EXTRACCION
7490 IF IND<0 THEN GO TO 9160
7500 LET CORRECTO=0
7510 LET ELE=T(1)
7520 LET IND=IND-1
7530 LET T(1)=T(IND+1)
7540 LET T(IND+1)=ELE
7550 LET CORRECTO=1
7560 RETURN
7560 REM EXTRACCION
7570 IF IND<0 THEN GO TO 9160
7580 LET CORRECTO=0
7590 LET ELE=T(1)
7600 LET IND=IND-1
7610 LET T(1)=T(IND+1)
7620 LET T(IND+1)=ELE
7630 LET CORRECTO=1
7640 RETURN
7640 REM EXTRACCION
7650 IF IND<0 THEN GO TO 9160
7660 LET CORRECTO=0
7670 LET ELE=T(1)
7680 LET IND=IND-1
7690 LET T(1)=T(IND+1)
7700 LET T(IND+1)=ELE
7710 LET CORRECTO=1
7720 RETURN
7720 REM EXTRACCION
7730 IF IND<0 THEN GO TO 9160
7740 LET CORRECTO=0
7750 LET ELE=T(1)
7760 LET IND=IND-1
7770 LET T(1)=T(IND+1)
7780 LET T(IND+1)=ELE
7790 LET CORRECTO=1
7800 RETURN
7800 REM EXTRACCION
7810 IF IND<0 THEN GO TO 9160
7820 LET CORRECTO=0
7830 LET ELE=T(1)
7840 LET IND=IND-1
7850 LET T(1)=T(IND+1)
7860 LET T(IND+1)=ELE
7870 LET CORRECTO=1
7880 RETURN
7880 REM EXTRACCION
7890 IF IND<0 THEN GO TO 9160
7900 LET CORRECTO=0
7910 LET ELE=T(1)
7920 LET IND=IND-1
7930 LET T(1)=T(IND+1)
7940 LET T(IND+1)=ELE
7950 LET CORRECTO=1
7960 RETURN
7960 REM EXTRACCION
7970 IF IND<0 THEN GO TO 9160
7980 LET CORRECTO=0
7990 LET ELE=T(1)
8000 LET IND=IND-1
8010 LET T(1)=T(IND+1)
8020 LET T(IND+1)=ELE
8030 LET CORRECTO=1
8040 RETURN
8040 REM EXTRACCION
8050 IF IND<0 THEN GO TO 9160
8060 LET CORRECTO=0
8070 LET ELE=T(1)
8080 LET IND=IND-1
8090 LET T(1)=T(IND+1)
8100 LET T(IND+1)=ELE
8110 LET CORRECTO=1
8120 RETURN
8120 REM EXTRACCION
8130 IF IND<0 THEN GO TO 9160
8140 LET CORRECTO=0
8150 LET ELE=T(1)
8160 LET IND=IND-1
8170 LET T(1)=T(IND+1)
8180 LET T(IND+1)=ELE
8190 LET CORRECTO=1
8200 RETURN
8200 REM EXTRACCION
8210 IF IND<0 THEN GO TO 9160
8220 LET CORRECTO=0
8230 LET ELE=T(1)
8240 LET IND=IND-1
8250 LET T(1)=T(IND+1)
8260 LET T(IND+1)=ELE
8270 LET CORRECTO=1
8280 RETURN
8280 REM EXTRACCION
8290 IF IND<0 THEN GO TO 9160
8300 LET CORRECTO=0
8310 LET ELE=T(1)
8320 LET IND=IND-1
8330 LET T(1)=T(IND+1)
8340 LET T(IND+1)=ELE
8350 LET CORRECTO=1
8360 RETURN
8360 REM EXTRACCION
8370 IF IND<0 THEN GO TO 9160
8380 LET CORRECTO=0
8390 LET ELE=T(1)
8400 LET IND=IND-1
8410 LET T(1)=T(IND+1)
8420 LET T(IND+1)=ELE
8430 LET CORRECTO=1
8440 RETURN
8440 REM EXTRACCION
8450 IF IND<0 THEN GO TO 9160
8460 LET CORRECTO=0
8470 LET ELE=T(1)
8480 LET IND=IND-1
8490 LET T(1)=T(IND+1)
8500 LET T(IND+1)=ELE
8510 LET CORRECTO=1
8520 RETURN
8520 REM EXTRACCION
8530 IF IND<0 THEN GO TO 9160
8540 LET CORRECTO=0
8550 LET ELE=T(1)
8560 LET IND=IND-1
8570 LET T(1)=T(IND+1)
8580 LET T(IND+1)=ELE
8590 LET CORRECTO=1
8600 RETURN
8600 REM EXTRACCION
8610 IF IND<0 THEN GO TO 9160
8620 LET CORRECTO=0
8630 LET ELE=T(1)
8640 LET IND=IND-1
8650 LET T(1)=T(IND+1)
8660 LET T(IND+1)=ELE
8670 LET CORRECTO=1
8680 RETURN
8680 REM EXTRACCION
8690 IF IND<0 THEN GO TO 9160
8700 LET CORRECTO=0
8710 LET ELE=T(1)
8720 LET IND=IND-1
8730 LET T(1)=T(IND+1)
8740 LET T(IND+1)=ELE
8750 LET CORRECTO=1
8760 RETURN
8760 REM EXTRACCION
8770 IF IND<0 THEN GO TO 9160
8780 LET CORRECTO=0
8790 LET ELE=T(1)
8800 LET IND=IND-1
8810 LET T(1)=T(IND+1)
8820 LET T(IND+1)=ELE
8830 LET CORRECTO=1
8840 RETURN
8840 REM EXTRACCION
8850 IF IND<0 THEN GO TO 9160
8860 LET CORRECTO=0
8870 LET ELE=T(1)
8880 LET IND=IND-1
8890 LET T(1)=T(IND+1)
8900 LET T(IND+1)=ELE
8910 LET CORRECTO=1
8920 RETURN
8920 REM EXTRACCION
8930 IF IND<0 THEN GO TO 9160
8940 LET CORRECTO=0
8950 LET ELE=T(1)
8960 LET IND=IND-1
8970 LET T(1)=T(IND+1)
8980 LET T(IND+1)=ELE
8990 LET CORRECTO=1
9000 RETURN
9000 REM EXTRACCION
9010 IF IND<0 THEN GO TO 9160
9020 LET CORRECTO=0
9030 LET ELE=T(1)
9040 LET IND=IND-1
9050 LET T(1)=T(IND+1)
9060 LET T(IND+1)=ELE
9070 LET CORRECTO=1
9080 RETURN
9080 REM EXTRACCION
9090 IF IND<0 THEN GO TO 9160
9100 LET CORRECTO=0
9110 LET ELE=T(1)
9120 LET IND=IND-1
9130 LET T(1)=T(IND+1)
9140 LET T(IND+1)=ELE
9150 LET CORRECTO=1
9160 RETURN
9160 REM EXTRACCION
9170 IF IND<0 THEN GO TO 9160
9180 LET CORRECTO=0
9190 LET ELE=T(1)
9200 LET IND=IND-1
9210 LET T(1)=T(IND+1)
9220 LET T(IND+1)=ELE
9230 LET CORRECTO=1
9240 RETURN
9240 REM EXTRACCION
9250 IF IND<0 THEN GO TO 9160
9260 LET CORRECTO=0
9270 LET ELE=T(1)
9280 LET IND=IND-1
9290 LET T(1)=T(IND+1)
9300 LET T(IND+1)=ELE
9310 LET CORRECTO=1
9320 RETURN
9320 REM EXTRACCION
9330 IF IND<0 THEN GO TO 9160
9340 LET CORRECTO=0
9350 LET ELE=T(1)
9360 LET IND=IND-1
9370 LET T(1)=T(IND+1)
9380 LET T(IND+1)=ELE
9390 LET CORRECTO=1
9400 RETURN
9400 REM EXTRACCION
9410 LET CORRECTO=0
9420 LET ELE=T(1)
9430 LET IND=IND-1
9440 LET T(1)=T(IND+1)
9450 LET T(IND+1)=ELE
9460 LET CORRECTO=1
9470 RETURN
9470 REM EXTRACCION
9480 RETURN
9490 LET P=UX
9500 RETURN
9510 LET P2=P
9520 LET P1=UX
9530 LET T(P2)=ELE
9540 LET CORRECTO=1
9550 RETURN
9560 REM INSERCCION
9570 IF P1>0 THEN GO TO 9540
9580 LET CORRECTO=0
9590 RETURN
9600 LET P1=UX
9610 LET T(P1)=ELE
9620 LET CORRECTO=0
9630 RETURN
9640 LET SU=0
9650 LET P1=P
9660 LET AUX1=PUN
9670 IF PUN<0 THEN LET PUN=P1: L
9680 LET P=P(PUN): P1=P1
9690 LET P(P1)=UX
9700 LET P1=P
9710 LET P=P(PUN): P1=P1
9720 LET P=P(PUN): P1=P1
9730 LET T(P1)=ELE: LET P1=RUX
9740 LET CORRECTO=1
9750 RETURN
9760 REM EXTRACCION
9770 IF PUN<0 THEN GO TO 9840
9780 RETURN
9780 REM EXTRACCION
9790 IF PUN<0 THEN GO TO 9840
9800 RETURN
9800 REM EXTRACCION
9810 IF PUN<0 THEN GO TO 9840
9820 RETURN
9820 REM EXTRACCION
9830 IF PUN<0 THEN GO TO 9840
9840 RETURN
9840 REM EXTRACCION
9850 IF PUN<0 THEN GO TO 9840
9860 RETURN
9860 REM EXTRACCION
9870 IF PUN<0 THEN GO TO 9840
9880 RETURN
9880 REM EXTRACCION
9890 IF PUN<0 THEN GO TO 9840
9900 RETURN
9900 REM EXTRACCION
9910 IF PUN<0 THEN GO TO 9840
9920 RETURN
9920 REM EXTRACCION
9930 IF PUN<0 THEN GO TO 9840
9940 RETURN
9940 REM EXTRACCION
9950 IF PUN<0 THEN GO TO 9840
9960 RETURN
9960 REM EXTRACCION
9970 IF PUN<0 THEN GO TO 9840
9980 RETURN
9980 REM EXTRACCION
9990 IF PUN<0 THEN GO TO 9840
10000 RETURN

```

La representación en la matriz es la siguiente:

DATOS	P	Q
1	1	1
2	2	2
...		
$n$	$n$	$n$

# MICRO-1

C/ Duque de Sesto, 50. 28009 Madrid  
Tel.: (91) 275 96 16/274 53 80  
(Metro O'Donell o Goya)

el IVA lo paga  
MICRO-1



QUICK SHOT I+INTERFACE  
2.695 PTAS.



QUICK SHOT II+INTERFACE  
2.995 PTAS.



1.395  
ptas.

1.695  
ptas.



QUICK SHOT V+INTERFACE  
2.995 PTAS.

1.695  
ptas.

NECESITAMOS DISTRIBUIDORES ¡¡GRANDES DESCUENTOS!!

**DIPROIMSA**  
**DISTR. de PRODUCTOS**  
**INFORMATICOS M., s.a.**

C/ GALATEA, 25. 28042 MADRID  
TF. 742 20 19 - 274 53 80

Recorta o copia este cupón y envíalo a:  
MICRO-1, C/ Duque de Sesto, 50. 28009 MADRID. Tf.: 275 96 16.

NOMBRE \_\_\_\_\_  
APELLIDOS \_\_\_\_\_  
CALLE \_\_\_\_\_  
C. POSTAL \_\_\_\_\_  
CANTIDAD \_\_\_\_\_  
DESCRIPCION \_\_\_\_\_  
PTAS. \_\_\_\_\_

PROVINCIA \_\_\_\_\_



¡SIN GASTOS  
DE ENVIO!

**L**os elementos principales del joystick son: la base o carcasa, la empuñadura, el sistema de articulación y los elementos eléctricos. Cada uno de ellos juega un importante papel y solamente un buen resulta-

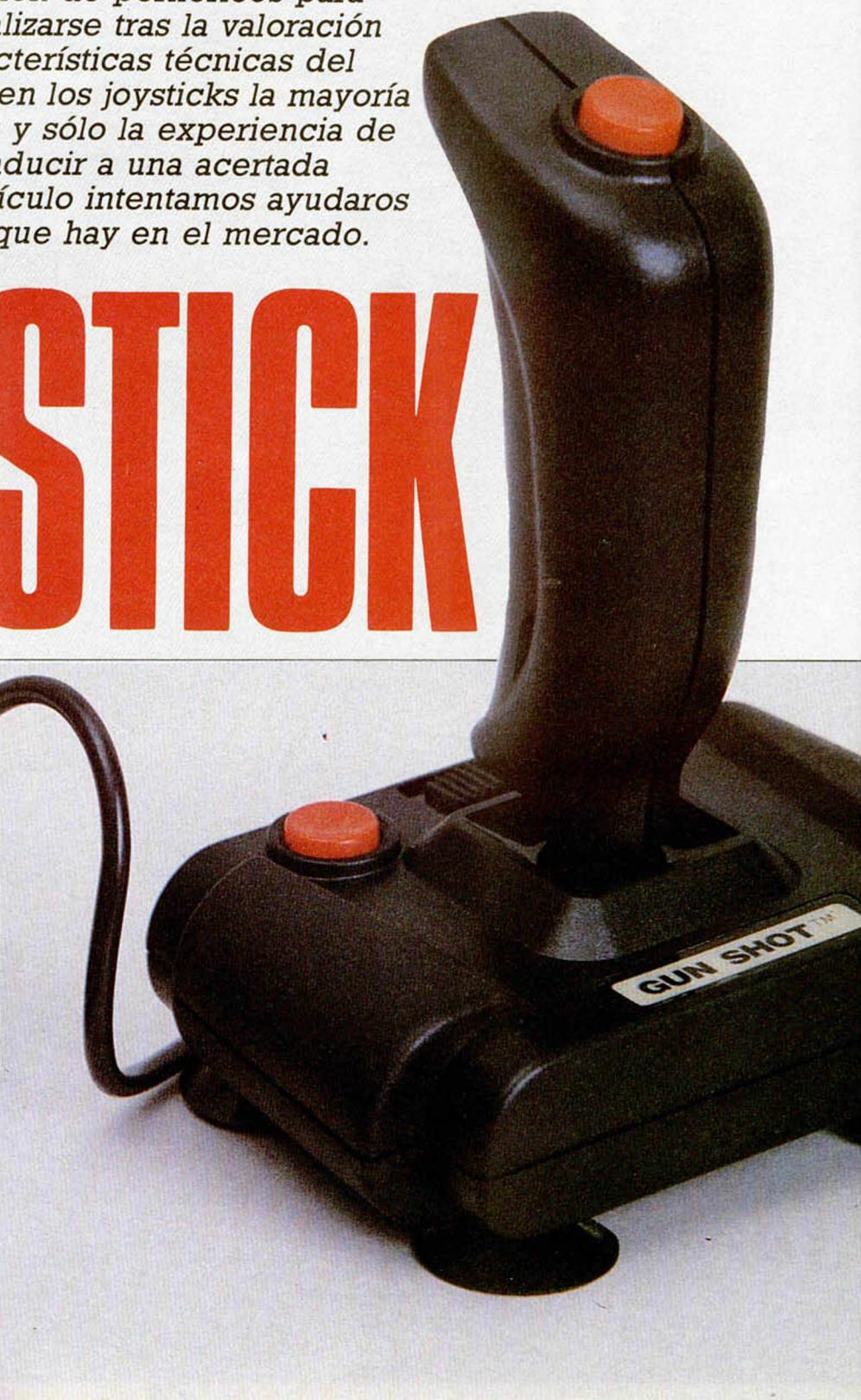
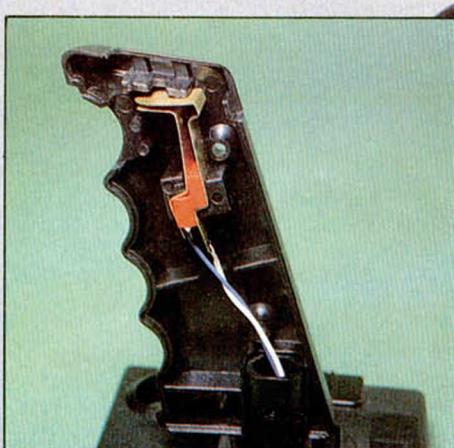
do conjunto podrá ofrecer la garantía de transferir eficaz y rápidamente al ordenador vuestos movimientos.

La empuñadura (stick), es el primer eslabón de la comunicación, las características particulares de este elemento son las que confieren al periférico

una buena parte de su comodidad de uso, por ello, su diseño se realiza en base a conceptos anatómicos, si bien sólo son unos pocos los modelos del mercado que logran una buena adaptabilidad a la mano del jugador y así permitir un uso prolongado consiguen-

*De ordinario la elección de periféricos para ordenadores suele realizarse tras la valoración personal de las características técnicas del producto; sin embargo, en los joysticks la mayoría de éstos son subjetivos y sólo la experiencia de uso nos puede conducir a una acertada valoración. Con este artículo intentamos ayudaros informándoos de los que hay en el mercado.*

# JOYSTICK



do con ello la mínima fatiga muscular.

La operatividad de la empuñadura es poder efectuar los desplazamientos correspondientes a las cuatro posiciones (N, S, E y O) que posteriormente analizará el ordenador y sus combinaciones de movimientos diagonales (NO, NE, SO y SE).

Una rótula esférica sirve de elemento de unión entre el stick y la carcasa, unión que debe permitir un suave desplazamiento del stick, esta suavidad de desplazamiento y la ausencia de holguras en el mecanismo determinará la precisión del joystick. El otro extremo de la empuñadura queda situado entre cuatro microrruptores que son activados o desactivados al alcanzar el stick la situación correspondiente. La construcción de estos contactos varía desde simples láminas flexibles hasta microrruptores mecánicos (reconocibles por su «clic» característico) e incluso simples circuitos impresos superpuestos que generalmente están integrados en una pieza de plástico que a veces llega a efectuar las funciones de la rótula.

A fin de que la empuñadura retorne a su posición central (neutra), un sistema de resortes o masas elásticas (gomas) unidas a la parte inferior de la rótula realizan tal operación.

La robustez de todo este conjunto mecánico es un factor determinante

para aquellos que día a día se enfrentan a duras batallas.

## MECANISMOS DE DISPARO

Para la ejecución de disparos (o saltos) hay gran variedad de versiones, para ello se dota al joystick de varias posibilidades, tantas como hábitos puedan tener los jugadores.

Generalmente éstos se producen al pulsar algún botón de diferentes formas y tamaños que se encuentran distribuidos en la empuñadura y/o la base.

Lo más habitual es que en la empuñadura haya al menos un pulsador al alcance del dedo pulgar y a veces se complementa con otro a modo de gatillo, accionable con el índice. También, la carcasa puede tener uno o varios pulsadores de efecto semejante a los del stick, todos estos pulsadores están concentrados eléctricamente en paralelo pudiendo efectuarse el disparo desde cualquiera de ellos. Algunos modelos están complementados con un interruptor de disparo permanente que sirve de gran ayuda en los juegos de trepidante acción (salvo en aquellos en los que la energía es en función inversa a los disparos).

En este capítulo es destacable la importancia de la recuperación de todos y cada uno de los ruptores a fin de que ésta no ralentice la sucesión de disparos en ráfaga.

## LA SUJECCIÓN

La carcasa o base del joystick cumple una doble misión, una, la de albergar en su interior todos los mecanismos descritos anteriormente, la otra, la de ofrecer una gran sujeción del conjunto a la superficie de la mesa en el caso de estar prevista para ello o la de acomodarse a la mano en aquellos tipos de joystick diseñados para este modo de utilización.

En el primer caso la sujeción del joystick viene realizándose a base de unas ventosas que, dispuestas en su parte inferior, los inmovilizan suficientemente. La mayoría de los modelos analizados poseen cuatro ventosas con la única excepción del Quick Shot V que utiliza cinco, por otra parte parecen haberse puesto de acuerdo todos los fabricantes en el tamaño de éstas...

tas..., todas ellas son de 30 mm de diámetro. Sin embargo, el grado de sujeción no sólo está en función del número de ventosas sino que también tiene su importancia la base de sustentación que éstas proporcionan y la longitud del stick, puesto que a mayor longitud de éste mayor empuje habrá de soportar la base.

## CONEXIÓN

La descripción del joystick queda completada con una pequeña alusión al cable de conexión en el que cabe destacar la importancia de una longitud que permita su manipulación a una distancia apropiada que casi todos poseen.

Por otro lado, la calidad del cable de conexión aunque no es influyente en la manipulación del periférico sí puede ser indicativa de la calidad general.

## DESARROLLO

En los últimos años la evolución de este periférico ha alcanzado cotas muy altas pero sigue siendo el diseño clásico el de mayor difusión, desarrollándose multitud de nuevos modelos más ergonómicos, cómodos y duraderos, incorporando a ellos ingeniosos complementos al efecto como bases más amplias (Quick Shot III y V, Cobra), mini teclados para introducción de niveles de dificultad y número de jugadores (QSV), utilización de mecanismos de alta calidad (baza que gana el Cobra), etc.

En cuanto a los modelos más avanzados éstos presentan innovaciones realmente ingeniosas y de conceptos absolutamente distintos de lo habitual, bien que su aplicación debe ser enjuiciada según cada particular.

Modelos como el Cheetach de mando a distancia evitan el a veces engorroso cable de conexión, si bien puede «jugarosla» durante una partida si en un momento de exaltación lo desvíamos de la dirección del interface receptor. Sistema muy similar utiliza el Quick Shot VII, pero sin dejar a un lado el cable conector.

El Joycard, es un reducido teclado que incorpora un joystick y un par de pulsadores en simulación a las «maquinitas» de los bares.

Quizás el modelo de más impacto



# 80 PERIFERICOS

visual sea el último de la extensa saga de Quick Shot, la versión nueve, una enorme bola de 10 cm de diámetro móvil en cualquier dirección, dotada de una gran precisión y que incorpora dos teclas de gran dimensión para disparo y complementado con un par de interruptores que permiten las opciones de fuego automático e inversión de sentido de desplazamiento, haciendo posible distintas situaciones del aparato.

## EL PRECIO DEL PODER

Realizar un análisis de precios correspondientes a cada modelo de joystick de una forma fehaciente no es tarea fácil dado que generalmente se encuentran formando parte de atractivas ofertas, cuando no se incluyen en la compra del ordenador, pero orientativamente oscilan alrededor de las 2.000/3.000 ptas. los modelos más convencionales, alcanzando 10.000 y 12.000 los modelos más precisos y/o sofisticados.



Commando



Joystick



Commodore



Joycard



Capitán Grant



Gun Shot



Investick



Cheetah



Cobra



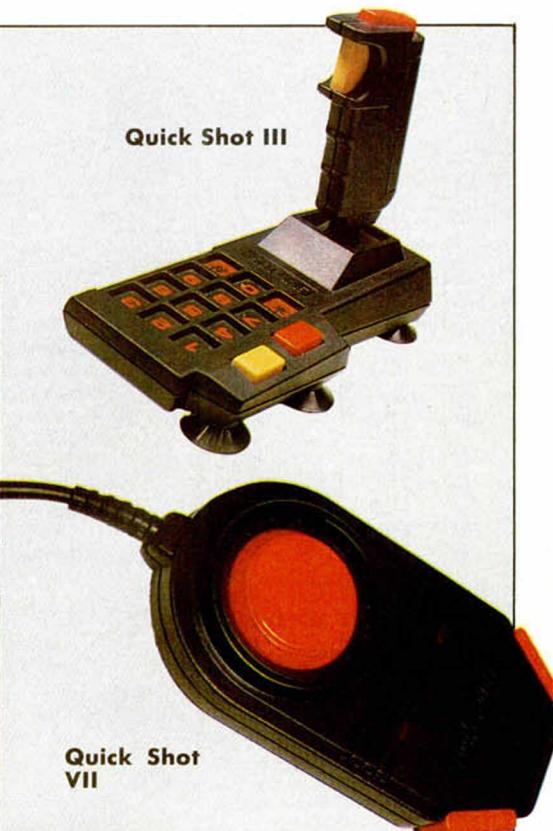
Quick  
Shot IX



Kempston



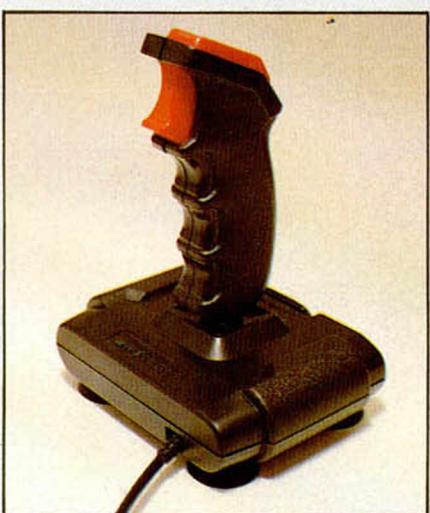
Quick Shot I



Quick Shot III



Kempston 3000



Quick Shot II



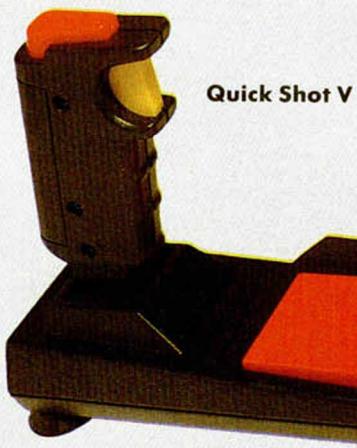
Quick Shot VII



Konix



Proto



Quick Shot V



Toshiba

## CARACTERISTICAS TECNICAS

MODELO	Tipo	EMPUNADURA		Sujección	BASE	DIMENSIONES (cm)	
		Disparo	Altura			Disparo	Totales
<b>CAPITAN GRANT</b>	Anatom.	1 pulsador	12	Manual	1 puls.	13x10x16	125
<b>CHEETAH</b>	Anatom.	1 pulsador		Manual		16x6.5x2.5	
<b>COBRA</b>	Anatom.	2 pulsad. 1 gatillo	16	4 ventosas		14x12.5x24	Mando distancia 135
<b>COMMANDO</b>	Anatom.	1 pulsador	3			4.5x3x17	170
<b>COMMODORE</b>	Lisa	1 pulsador	7,5	Manual	1 tecla	10x7x11	127
<b>GRAN CAP. II</b>	Anatom.	1 pulsador 1 gatillo	12	Manual	2 pulsad. +AUTO	12x12x16.5	125
<b>GUN SHOT</b>	Anatom.	1 pulsador	13,5	4 ventosas	1 pulsad.	13x11x16	125
<b>INVESTICK</b>	Anatom.	1 pulsador 1 gatillo	13,5	4 ventosas	2 pulsad.	13x10x18	104
<b>JOYCARD</b>	de bola		4,5	sobre mesa	2 pulsad.	18x10x6.5	27
<b>JOYSTICK</b>	Anatom.	1 pulsador 1 gatillo	13	4 ventosas	2 pulsad.	13.5x10x17	124
<b>KEMPSTON</b>	de bola		7	Manual	2 pulsad.	11x5x9x11	120
<b>KEMPSTON 3000</b>	Anatom.	1 pulsador 1 gatillo	12	Manual	1 tecla	13x7.5x16	160
<b>KONIX</b>	de bola		6			13x8x10	135
<b>PROTO</b>	Anatom.	1 pulsador 1 gatillo	12	4 ventosas	1 pulsad.	12x12x15.5	126
<b>QUICK SHOT I</b>	Anatom.	1 pulsador	12	4 ventosas	1 pulsad.	11x9x16	126
<b>QUICK SHOT II</b>	Anatom.	1 pulsador 1 gatillo	13.5	4 ventosas	1 pulsad. +AUTO	13x9.5x17.5	126
<b>QUICK SHOT III</b>	Anatom.	1 pulsador 1 gatillo	12	5 ventosas	2 pulsad. TECLADO	19x10x17	126
<b>QUICK SHOT V</b>	Anatom.	1 pulsador 1 gatillo	12	4 ventosas	1 tecla	19x9.5x17	126
<b>QUICK SHOT VII</b>	Disco					12x9x2.5	126
<b>QUICK SHOT IX</b>	Esfera 10 cm		10	4 ventosas	2 teclas +AUTO	22x14.5x12	121
<b>SUPERSTICK</b>	Cilindro	1 pulsador	10	Manual		9x9x13	157
<b>TOSHIBA</b>	Anatom.	1 pulsador	11	Manual	1 tecla	13x10x18	104

## VALORACION

MODELO	GRADO DE SUJECION	SUAVIDAD DE MOVIM.	SUAVIDAD DE DISP.	ADAPTACION A LA MANO	ROBUSTEZ MECANIS.	COMODIDAD USO PROLONG.
<b>CAPITAN GRANT</b>		***	***	****	***	**
<b>CHEETAH</b>		**	***	****	****	***
<b>COBRA</b>	*****	****	*****	****	****	***
<b>COMMANDO</b>		***	**	****	***	***
<b>COMMODORE</b>		*	**	*	***	*
<b>GRAN CAP. II</b>	***	****	***	****	**	***
<b>GUN SHOT</b>	***	****	***	****	**	***
<b>INVESTICK</b>	***	***	***	**	***	***
<b>JOYCARD</b>		****	****	***	***	***
<b>JOYSTICK</b>	***	*****	***	***	***	***
<b>KEMPSTON</b>		****	****	****	***	***
<b>KEMPSTON 3000</b>		*	**	**	***	***
<b>KONIX</b>		*****	***	****	***	***
<b>PROTO</b>	***	***	***	***	***	***
<b>QUICK SHOT I</b>	***	***	***	***	***	***
<b>QUICK SHOT II</b>	***	***	***	***	***	***
<b>QUICK SHOT III</b>	****	***	***	***	***	***
<b>QUICK SHOT V</b>	****	***	***	***	***	***
<b>QUICK SHOT VII</b>	***	***	***	***	***	***
<b>QUICK SHOT IX</b>	****	***	***	***	***	***
<b>SUPERSTICK</b>		***	***	***	***	***
<b>TOSHIBA</b>	*****	****	****	****	***	***

# Sound-on-Sound

La cinta virgen para ordenador

C15 y C20

¡NUEVA!



*Fabulosos  
REGALOS*



*Cintas de alta resolución*

La cinta más avanzada, producida exclusivamente para ordenadores. Precio: 12.000 ptas.

Comprando una cinta Sound-on-Sound, usted puede obtener uno de estos regalos:

- Un ordenador PCW 8256 AMSTRAD.
- Un ordenador CPC 6128 AMSTRAD.
- Un ordenador CPC 464 AMSTRAD.
- Una IMPRESORA para AMSTRAD.
- Una impresora electrónica INDESCOMP.
- Un cassette software INDESCOMP.

# SINCLAIR STORE

## EL CENTRO DE LAS NOVEDADES

SPECTRUM 128 K+2



INVES PC 640 X

Venga a Sinclair Store.

Le presentamos las más recientes **PC** totalmente compatibles por menos del Convertidor TV para tu Amstrad, hasta las cadenas de sonido con un precio inferior a 30.000 ptas., que van a revolucionar el mercado. **¡VA A SER UN ESCANDALO!**

### OFERTAS

Convertidor TV Amstrad  
Ampliación memoria Amstrad 464, 64 K  
Ampliación memoria Amstrad 464, 256 K  
Disco de silicio 256 K  
Lápiz óptico Amstrad  
Sintetizador de voz  
Fundas teclado, desde  
Opus Discovery  
Software Amstrad, Commodore, desde  
Joystick Quick Shot II + Interface Kempston

ABRIMOS SABADOS TARDE

**sinclair store**

**SOMOS PROFESIONALES**

BRAVO MURILLO, 2  
(Glorieta de Quevedo)  
Tel. 446 62 31 - 28015 MADRID  
Aparcamiento GRATUITO Magallanes, 1

DIEGO DE LEON, 25  
(Esq. Núñez de Balboa)  
Tel. 261 88 01 - 28006 MADRID  
Aparcamiento GRATUITO Núñez de Balboa, 114

AV. FELIPE II, 12  
(Metro Goya)  
Tel. 431 32 33 - 28009 MADRID  
Aparcamiento GRATUITO Av. Felipe II

INVES 100 HF

Los primeros en tener lo último.

novedades. Desde los ordenadores 90.000 ptas., lo último en Spectrum. 44.000 ptas., 500 ptas., 3.000 ptas., que van a revolucionar el mercado. **¡VA A SER UN ESCANDALO!**

	Pesetas
Lanzamiento	8.500
	21.500
	20.600
	5.600
	9.450
	800
	44.000
	500
	3.000