

# Todo Spectrum

JULIO-AGOSTO 86 - 300 ptas.

AÑO II - Número 23

REVISTA EXCLUSIVA PARA USUARIOS

PROGRAMAMOS

## Turboprotector

OIMOS

Sonido por  
dos canales

DESCUBRIMOS

Guía del Hacker  
Cyberun

INVESTIGAMOS

Código máquina  
Canales y corrientes



**Suplemento al**  
Gráficos en tres  
dimensiones



ORDENADORES SOBRESALIENTES A PRECIOS QUE HACEN ESCUELA

# DOBLE REGALO FIN DE CURSO

Premie el esfuerzo  
de sus hijos por fin  
de curso.

Regádeles los mejores ordenadores  
personales a precios de auténtica  
oportunidad.

Investrónica, además, les hace otro gran regalo:  
joysticks, interfaces, cursos de Basic en vídeo,  
lápidas ópticas...

Spectrum Plus, Spectrum 128 y QL, tres ordenadores muy  
estudiados, a precios que son una lección magistral.

Dé un ejemplo. Haga un doble regalo fin de curso con Investrónica.

Y además, precios muy especiales para lotes de Interface I, Microdrives e impresoras.  
Infórmese en su concesionario Investrónica más cercano.



## SPECTRUM PLUS, SPECTRUM 128 Y QL

Regale un Spectrum Plus,  
que incluye un lote de 6 cintas  
de juegos. Su distribuidor In-  
vestrónica le regala, además:

Un joystick más un Inter-  
face II,  
o un Curso de Basic en vídeo,  
o un lápiz óptico.

Regale un Spectrum 128,  
que incluye dos cintas de  
juegos, un manual de utiliza-  
ción y una cinta de demostra-  
ción.

Su distribuidor Investrónica  
le regala, además:

Un joystick más un Inter-  
face II,  
o un Curso de Basic en vídeo.

Regale un ordenador QL  
desde 44.550 ptas. o, si lo  
prefiere, una configuración de  
ordenador y monitor desde  
65.300\* ptas.

Infórmese de nuestras  
grandes ofertas de QL con  
monitores monocromo y de  
color con media y alta resolu-  
ción e impresora.

(PROMOCION ESPECIAL POR TIEMPO LIMITADO)

\* Precio sin IVA.



**investronica**

Tomás Bretón, 62  
Tel. (91) 467 82 10  
Telex 23399 TYCO E  
28045 Madrid

Camp. 80  
Tel. (93) 211 26 58 - 211 27 54  
08022 Barcelona

etc



# SUMARIO

AÑO II - N.º 23 - JULIO 1986

## 6 TURBOPROTECTOR

Si quieres aumentar la velocidad de carga y al mismo tiempo proteger tus programas de miradas indiscretas, en este artículo te contamos cómo hacerlo.

## 22 SONIDO POR DOS CANALES

A partir de ahora podremos disfrutar de música a dos voces con un virtuosismo comparable al de cualquier otro ordenador y olvidar todas las críticas sobre el sonido de nuestro querido Spectrum.

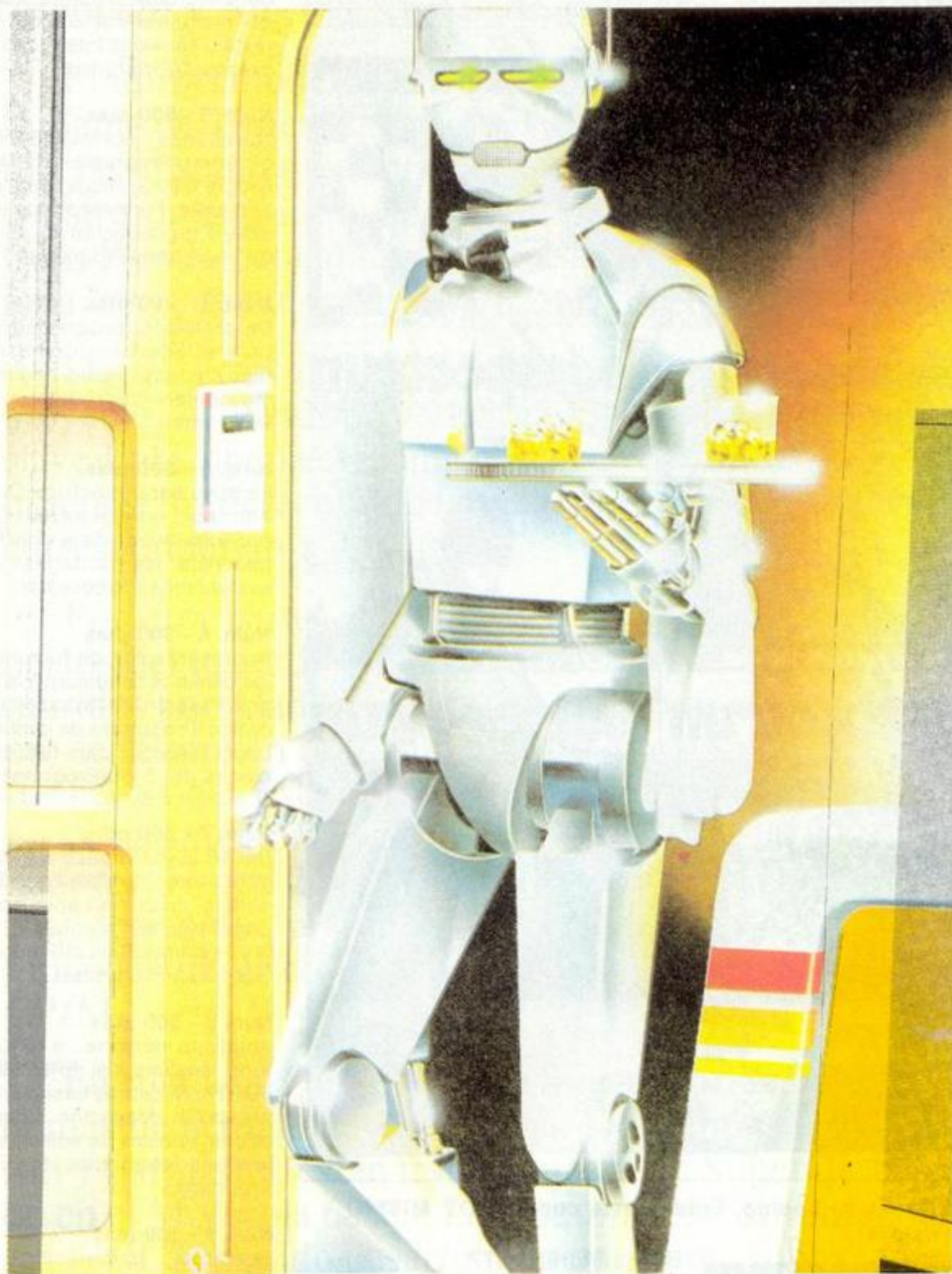
## 31 SUPLEMENTO QL

**Gráficos interactivos en tres dimensiones**

Ricardo García, nuestro experto en gráficos, comenta las técnicas básicas de los gráficos tridimensionales y nos propone un interesante ejemplo.

## 39 APRENDIENDO LENGUAJE MAQUINA

En nuestro recorrido por el sistema operativo encontramos la zona de información para canales y analizamos el tratamiento de los dispositivos de entrada y salida.



## 46 GUIA DEL HACKER: CYBERUN

Después de destripar el juego, conseguimos recoger todos los cristales de cybernita y escapar del campo de fuerza que nos atrapaba sin perder ninguna vida.

## 54 PROGRAMAS

Aunque muchos de vosotros ya estaréis de vacaciones, la sección de programas de este número está dedicada a las matemáticas, concretamente a las progresiones.



# SERVICIO DE EJEMPLARES ATRASADOS



Para hacer su pedido, rellene este cupón HOY MISMO y envíelo a:

**Todospectrum** Bravo Murillo, 377  
Tel. 733 96 62 - 28020 MADRID

Ruego me envíen los siguientes ejemplares atrasados de TODOSPECTRUM ..... al precio de 300 pts.

El importe lo abonaré  
☐ POR CHEQUE ☐ CONTRA REEMBOLSO ☐ CON MI TARJETA DE CREDITO ☐ AMERICAN EXPRESS ☐ VISA ☐ INTERBANK

Número de mi tarjeta:

Fecha de caducidad ..... Firma

NOMBRE .....

DIRECCION .....

CIUDAD ..... C. P. ....

PROVINCIA .....

Complete su colección de

## Todospectrum

A continuación le resumimos el contenido de los ejemplares aparecidos hasta ahora.

### Núm. 2 - 300 pts.

Gráficos profesionales/Desplazamiento pixel a pixel/Utilización de rutinas/Construcción del interface centronics/Programas de utilidad para microdrive/Rutina reset en código máquina/Análisis del editor de textos Tasword/Interfaces para impresoras/Programas.

### Núm. 3 - 300 pts.

Novedades sonimag'84/Ampliando el Basic/Programas para ordenar programas/Gráficos con el VU-3D/Lenguaje Forth/Archivos en microdrive/Programación de un interface de impresora/Programas.

### Núm. 4 - 300 pts.

De profesión: programador/Consola para el Spectrum/Comparación código máquina-Basic/Análisis programa contabilidad/Calendario/Pascal/Programas.

### Núm. 5 - 300 pts.

Floppys para Spectrum/Diseño asistido por ordenador/64 Caracteres por línea/Juego de la vida/Pascal/Así hacemos las portadas/Control de evaluaciones/Programas.

### Núm. 6 - 300 pts.

Representación de funciones/Todos los caminos conducen a la ROM/Juegos/Pascal/Construcción de un lápiz óptico/Programas de gestión. El SITI/Logo: tortugas para todos/ Interrupciones del Z-80/Programas.

### Núm. 7 - 300 pts.

Del 48 al PLUS paso a paso/¿Plotter para Spectrum?/Juegos/Libros de código máquina/Lápiz óptico. Programación del montaje/El LOGO en la escuela/Pascal/Floppys para Spectrum/Programas.

### Núm. 8 - 300 pts.

Amplia tu memoria... a 48 K/Arquitectura: análisis del PREYME/Juegos/FORTH. Nociones básicas/Una clave, please/QL Magazine. Últimas novedades, análisis de software, Lenguajes/Aula informática con Spectrum/Programas.

### Núm. 9 - 300 pts.

Spectrum parlanchin/Juegos/Aula informática con Spectrum/Análisis: Comercial 4/Pascal/Periféricos: Wafadrive/QL Magazine: EASEL lo mejor de PSION. Música con QL/Desplazamiento Pixel a Pixel, aportación de lectores/Programas/Programer II.

### Núm. 10 - 300 pts.

Discos: invetsdisc 200/Juegos/Dos programas simultáneos/Protección del software/Conozca extremadura, consulte a su ordenador/Desensamblador Z-80/Software educativo/QL Magazine: novedades Informat, Hoja de cálculo, Ajedrez/Construya su propio Joystick/Pascal/programas.

**DISPONEMOS  
DE TAPAS ESPECIALES  
PARA SUS EJEMPLARES DE ZX  
(sin necesidad de encuadernación)**

### Núm. 11 - 300 pts.

Actualidad/La otra cara del LOGO/Juegos/El Spectrum habla castellano/SOFTAID ayuda para Etiopía/S.O.S. aquí el Spectrum/Dibujar con lápiz óptico/QL Magazine: Procesador de textos. Teclas de función programables/Programas.

### Núm. 12 - 300 pts.

Actualidad/Inteligencia artificial/Lápiz óptico dk'TRONICS/Juegos/Análisis/Bingo/Z-80 PIO/Código máquina/Análisis: MASTERFILE/Programas.

### Núm. 13 - 300 pts.

Actualidad/Discos: Discovery 1/Juegos/Inteligencia artificial/Un nuevo sistema operativo/QL Magazine: Archive, Cartridge doctor. Aplicaciones comerciales/Código máquina/Programas.

### Núm. 14 - 300 pts.

Actualidad, Spectrum 128/Cálculo de estructuras para ingenieros y arquitectos/HELP utilidades en microdrive/Juegos/El microdrive ese desconocido/Código máquina/QL Magazine: GRAPHIC QL. Juegos. Discos de 720 K/Un nuevo operativo/Programas.

### Núm. 15 - 300 pts.

Actualidad/Spectrum 128/Un nuevo operativo/Círculos redondos/Juegos/Utilidades: BETA-BASIC/QL Magazine: Introducción al SUPER BASIC. Nuevas utilidades/Hardware: Puertas lógicas/Código máquina/Programas.

### Núm. 16 - 300 pts.

Actualidad/Cinco horas con SCREEN\$/Hardware práctico/Cálculos de infinita precisión/Juegos/Un nuevo operativo/QL Magazine: Gráficos en SUPER-BASIC. Dibujando con ratón. Archivos con Archive. Programa/La última batalla, Juego estratégico.

### Núm. 17 - 300 pts.

Actualidad/Gráficos interactivos/Juegos/Código máquina/Un nuevo operativo/Trucos de programación/QL Magazine: Radiografía del QL. Gráficos en SUPER-BASIC/Libros/Programas.

### Núm. 18 - 300 pts.

Actualidad/Introducción al C/Libros/Juegos/De cinta a microcinta/Visión panorámica de los microprocesadores más comunes/QL Magazine: Copy de grises. Microprocesadores 68000, una familia numerosa/Curióseando en la ROM/Programas.



## SPECTRUM PLUS II

### DIRECTOR:

Enrique F. Larreta

### REDACTOR JEFE:

Emiliano Juárez

### REDACCION:

Ignacio Borrell, Octavio López,

Antonio del Río

### DISEÑO:

Esteban Pérez

Editado por PUBLINFORMATICA, S. A.

Bravo Murillo, 377. 5.º A. Tel.:

733 74 13 - 28020 Madrid

### Presidente:

Fernando Bolin

Director Editorial Revistas de Usuarios:

Juan Arencibia

### Director de Ventas:

Antonio González

Producción: Miguel Onieva

### Servicio al cliente:

Julia González. Tel.: 733 79 69

### Administración:

PUBLINFORMATICA, S. A.

### Publicidad Madrid:

Emilio García

### Dirección, Publicidad y

### Administración:

Bravo Murillo, 377. 5.º A. Tel. 733 74 13.

Télex: 48877 OPZX e 28020 Madrid

### Publicidad Barcelona:

Lidia Cendrós. Pelayo, 12. Tels. (93)

318 02 89 - 301 47 00, ext. 27 y 28.

08001 Barcelona

Depósito legal: M-29041-1984

Distribuye S.G.E.L. Avda. Valdelaparra,

s/n. Alcobendas (Madrid).

Fotomecánica: Karmat, C/ Pantoja, 10.

Madrid.

Fotocomposición: Espacio y Punto

Imprime: Héroes, C/ Torrelara, 8. Madrid.

Distribuidor en VENEZUELA,

### SIPAM, S. A.

AVD. REPUBLICA DOMINICANA, EDIF.

FELTREC - OFICINA 4B BOLEITA SUR

CARACAS (VENEZUELA)

Esta publicación es miembro de la

Asociación de Revistas de

Información **ari** asociada a la

Federación Internacional de Prensa

Periódica, FIPP.

### SUSCRIPCIONES:

Rogamos dirijan toda la correspondencia

relacionada con suscripciones a:

TODOSPECTRUM EDISA: Tel. 415 97 12

C/ López de Hoyos, 141-5º

28002 MADRID

(Para todos los pagos reseñar solamente

TODOSPECTRUM)

Para la compra de ejemplares atrasados

dirijan a la propia editorial

TODOSPECTRUM

C/ Bravo Murillo, 377. 5.º A.

Tel. 733 74 13 - 28020 MADRID

Si deseas colaborar en TODOSPECTRUM

remite tus artículos o programas a Bravo

Murillo, 377. 5.º A. 28020 Madrid. Los

programas deberán estar grabados en

cassette y los artículos mecanografiados.

A efectos de remuneración, se analiza cada

colaboración aisladamente, estudiando su

complejidad y calidad.

En septiembre, coincidiendo con el Personal Computer World Show, asistiremos a la presentación de nuevos productos Sinclair. Amstrad lanzará el Spectrum Plus II, parecido al actual, pero con un cassette incorporado y con 128 Kbytes de memoria RAM.

El nuevo modelo de la gama Sinclair se montará en la planta de Timex en Dundee, y no en Corea del Sur como las restantes máquinas Amstrad. Timex también construirá una impresora matricial dedicada al Spectrum Plus II, probablemente basada en la que acompaña a los ordenadores PCW 8256 y PCW 8512.

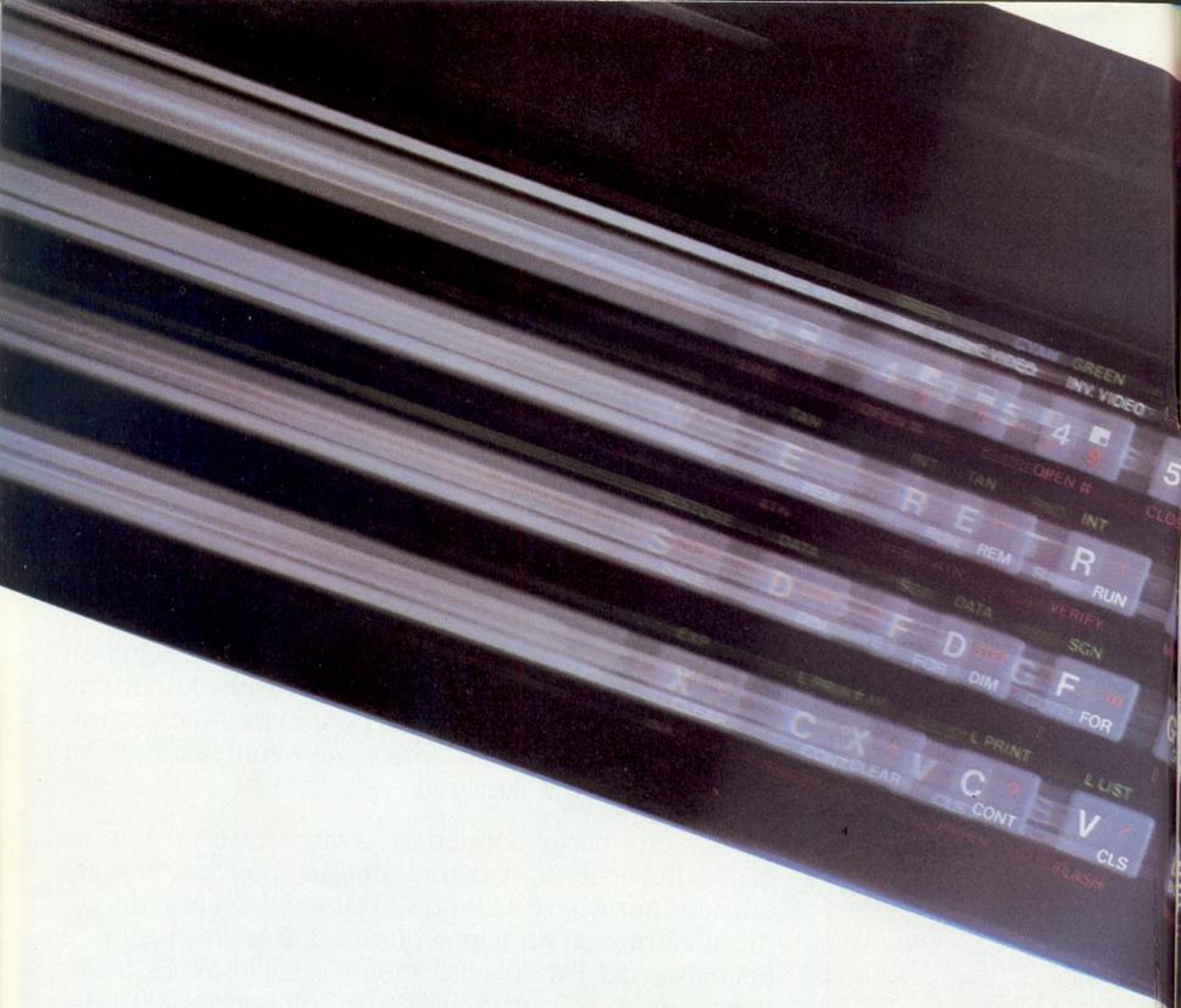
De las otras dos máquinas que Sir Clive tenía en proyecto, el Pandora y el SuperSpectrum de nombre clave Loki, apenas se sabe nada por el momento, ni siquiera si los derechos de fabricación y comercialización corresponderían a Amstrad.

Por otra parte, comienzan a conocerse nuevos aspectos del acuerdo Amstrad Sinclair. Los derechos adquiridos por Alan Sugar no son para todo el mundo, como se afirmó en principio, puesto que en Portugal y en los países del Este le corresponden a Timex. Esta empresa suministrará al gobierno de Polonia 800.000 ordenadores Timex 2068 (Spectrum con port de joystick, chip de sonido y algunas otras mejoras) y 200.000 dobles unidades de disco FDD 3000 con el sistema operativo CP/M.

En cuanto al QL, que según Sugar habría muerto, está más de actualidad que nunca. Dos empresas, Eider-soft y Farmintel, están trabajando en un SuperQL basado en el Motorola 68000. La máquina de Farmintel, llamada QLT, ha sido desarrollada por Tony Tebby, autor del QDOS. Ambos ordenadores serán plenamente compatibles con el QL, aunque le superarán en velocidad, memoria y otras prestaciones.

En los próximos números esperamos poder informaros más detalladamente de todos estos proyectos.

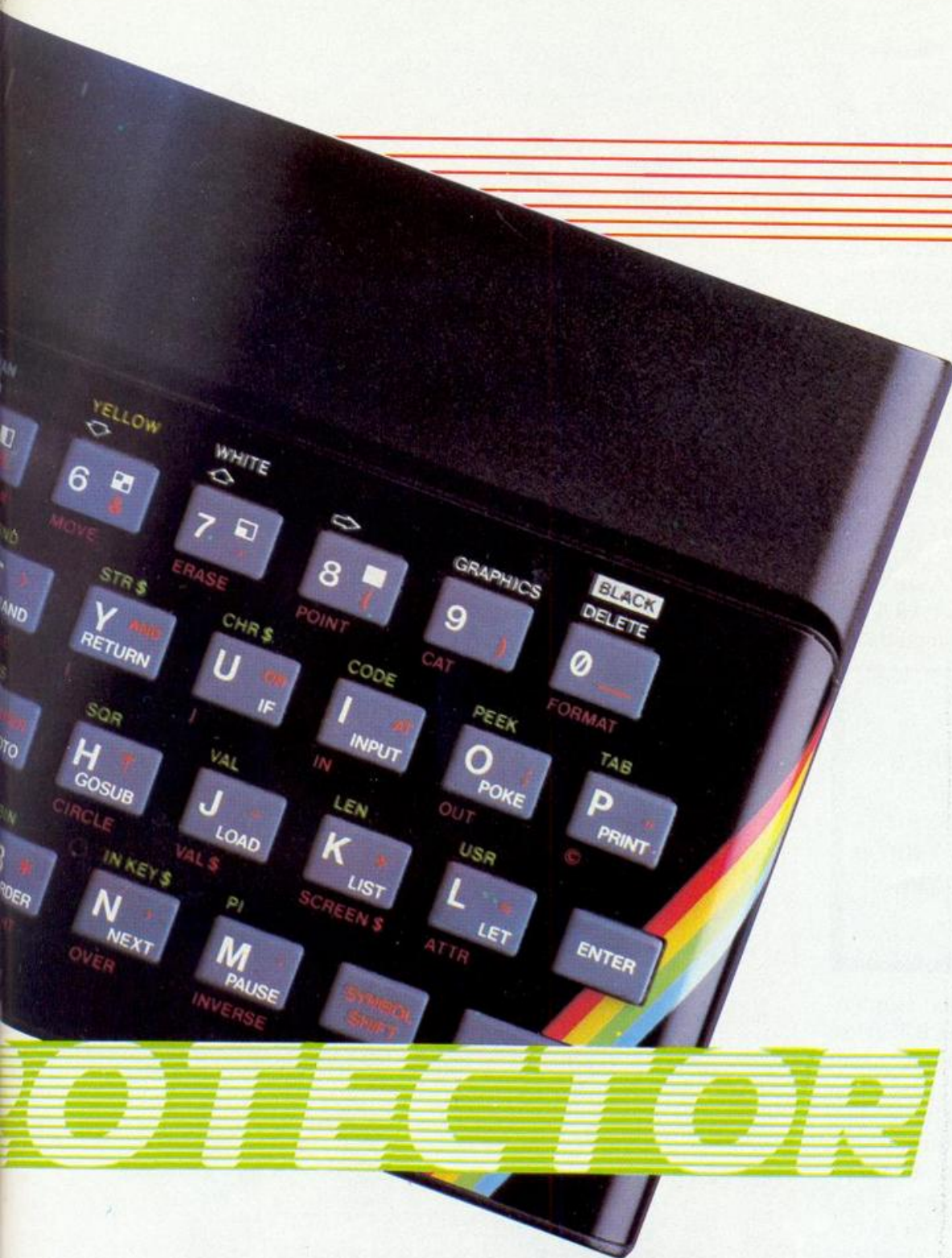




## TURBO

Ha llovido mucho desde que los primeros programas TURBO aparecieron en el mercado. Aquellos cargadores volvían locos a los piratas más hábiles y permitieron vender un gran número de cintas a las compañías que aprovecharon este nuevo sistema de protección. Sin embargo son pocos los usuarios que saben utilizar esta técnica para la protección de sus programas o para acortar los tiempos de carga de los mismos. He aquí un programa que lo hará posible sin necesidad de tener extensos conocimientos de programación. Basta con cargarlo, ejecutarlo y seguir los pasos que nos indique.





**E**ste programa es un híbrido BASIC-Código máquina que nos permitirá grabar una versión TURBO de cualquier programa que no utilice la parte superior de la memoria (aprox. 1 Kbyte). El programa a salvar podrá estar compuesto de una pantalla de carga, un programa BASIC y un bloque en código máquina o bytes, por lo que si usamos alguna matriz de datos deberemos unirla al BASIC antes de grabarlo, y también deberemos unir en un solo bloque todo el Código máquina que

utilicemos. Antes de ejecutar el programa deberemos, por lo tanto, tener preparados en una cinta cada uno de los bloques que vayamos a utilizar (si es posible, en el orden código máquina-pantalla-BASIC).

Quien disponga de ensamblador podrá teclear el código máquina en él como se explica más adelante, pero el que lo prefiera puede dejar a un lado el listado assembler y copiar simplemente el listado BASIC, que incluye un cargador hexadecimal.

Una vez tecleado el programa de-

be ser salvado a cinta o microdrive. Después puede ser ejecutado, con lo que comenzará a ser pokeado el código máquina de las líneas DATA mientras se comprueba, mediante las sumas de control, si hay algún error. Caso de haberlo, el programa indica en qué línea se ha producido para que pueda ser rectificado.

### Funcionamiento del programa

Cuando el proceso de carga de las



DATAs ha finalizado, aparece en pantalla el mensaje "PROGRAMA BASIC?: (S/N)", al que hay que contestar si lo que pretendemos convertir en TURBO es (o incluye) un programa BASIC. Sólo en el caso de que contestemos afirmativamente (lo que, evidentemente, se hace pulsando "S") nos pedirá la línea de autoejecución. Si queremos que, al terminar de cargarse, el programa ejecute a partir de una determinada línea (algo así como el SAVE con LINE) habrá que especificar aquí esa línea, y si no queremos autoejecución bastará con pulsar ENTER.

Para un programa BASIC también nos pedirá "LINEA PARA ON ERROR"; esta es una medida de

**Realizado en BASIC y código máquina, el turboprotector permite grabar una versión turbo de cualquier programa.**

protección que hace que, si mientras se corre el programa BASIC se pulsa BREAK (o se produce cualquier error) la ejecución pase a una línea determinada del programa, donde podría autodestruirse o simplemente imprimir "¡Ya cállate, pirata asqueroso!" y comenzar a ejecutar desde el principio (si queremos abstenernos de usar esta opción bastará con pulsar ENTER). Sólo hay una forma de salir de un programa que utilice esta protección, y es introduciendo "USR 8" (o algunos otros USRs) en un INPUT numérico; esto hace que se salte a la rutina de errores en busca del valor de USR, con lo que se acaba indefectiblemente en la rutina principal del editor. Esto puede evitarse suprimiendo a toda costa los INPUTs numéricos y utilizando alfanuméricos seguidos de un VAL que pase el valor a una variable numérica.

```

10 ; TURBOPROTECTOR(C) LGP
20 ORG 64460
30
40 BUFL10 DEFS 288
50 BUFCB DEFS 17
60
70 TRBPRT LD A,2
80 CALL CHN_OP
90 LD A,(BASIC)
95 OR A
100 CALL NZ,LD_BSC
120 TRBPRO CALL CLSL10
130 LD DE,#09A1
140 XOR A
150 CALL PO_MSG
160 CALL EP_TCL
170 CALL LD_L10
180 LD DE,17
190 LD IX,CAB_CB
200 XOR A
210 CALL SA_BYT
220 JR NC,ERR_R
230 LD HL,(FIN)
240 LD DE,(BASE)
250 OR A
260 SBC HL,DE
270 PUSH DE
280 LD (LONG),HL
290 PUSH HL
300 LD IX,CARG_B
310 LD DE,END-CARG_B
320 LD A,255
330 CALL SA_BYT
340 JR NC,ERR_R
350 POP DE
360 POP IX
370 CALL TRBSA
380 CALL CLSL10
390 TRBPR1 LD DE,MENSG
400 XOR A
410 CALL PO_MSG
420 CALL EP_TCL
430 CALL LD_L10
440 LD A,(IY-50)
450 CP "N"
460 JR Z,ERR_O
470 CP "S"
480 JR Z,TRBPRO
490 JR TRBPR1
500
502 ERR_O RST 8
504 DEFB 255
506
540 ERR_R RST 8
550 DEFB 26
560
570 LD_BSC LD DE,17
580 LD IX,BUFCB
590 XOR A
600 SCF
610 CALL LD_BYT
620 JR NC,LD_BSC
630 LD A,(BUFCB)
640 OR A
650 JR NZ,LD_BSC
660 CALL CLSL10
670 LD BC,#0F21
680 CALL CL_SET
690 LD DE,#09C0
700 XOR A
710 CALL PO_MSG
720 LD B,10
730 LD HL,BUFCB
740 P_NOMB INC HL
750 LD A,(HL)
760 RST 16
770 DJNZ P_NOMB
780 LD_BS1 LD BC,(PROG)
790 PUSH BC
800 POP IX
810 LD HL,(BUFCB+11)
820
830 PUSH HL
840 POP DE
850 ADD HL,BC
860 LD (HL),128
870 INC HL
880 PUSH HL
890 LD (E_LINE),HL
900 LD (CH_ADD),HL
910 LD HL,(BUFCB+15)
920 ADD HL,BC
930 LD (VARS),HL
940 CALL SET_MN
950 LD A,255
960 SCF
970 CALL LD_BYT
980 JR NC,ERR_R
990 POP HL
1000 LD A,(FIN+1)
1010 CP H
1020 RET NC
1030 LD (FIN),HL
1040 RET
1050 CLSL10 LD BC,#0E21
1060 CALL CL_SET
1070 LD B,32
1080 CLL10A LD A,32
1090 RST 16
1100 DJNZ CLL10A
1110 LD BC,#0E21
1120 JP CL_SET
1130
1150 SA_L10 LD A,(PANT)
1160 OR A
1170 RET Z
1180 LD HL,18496
1190 LD DE,BUFL10
1200 LD A,8
1210 S_BCL LD BC,32
1220 PUSH HL
1230 LDIR
1240 POP HL
1250 INC H
1260 DEC A
1270 RET M
1280 JR NZ,S_BCL
1290 LD HL,22848
1300 JR S_BCL
1310
1320 LD_L10 CALL CLSL10
1330 LD A,(PANT)
1340 OR A
1350 RET Z
1360 LD HL,BUFL10
1370 LD DE,18496
1380 LD A,8
1390 L_BCL LD BC,32
1400 PUSH DE
1410 LDIR
1420 POP DE
1430 INC D
1440 DEC A
1450 RET M
1460 JR NZ,L_BCL
1470 LD DE,22848
1480 JR L_BCL
1490
1500 TRBSA DI
1510 LD HL,250
1520 LD A,2
1530 LD B,A
1540 RTRD1 DJNZ RTRD1
1550 OUT (254),A
1560 XOR %00001010
1570 LD B,164
1580 DEC L
1590 JR NZ,RTRD1
1600 DEC B
1610 DEC H
1620 JP P,RTRD1
1630 LD B,47

```







Tanto si el programa incluye BASIC como si no, se nos pregunta después si utilizamos código máquina, y, en caso de que contestemos afirmativamente, se nos pide la dirección de inicio, la longitud y la dirección de ejecución. Esta última dirección, deberá ser el punto de entrada del código si es que lo hay (en caso contrario, o si no queremos

**El turboprotector crea un pequeño programa BASIC que contiene el cargador turbo y los datos del siguiente bloque.**

que se ejecute bastará con ENTER), con lo cual se saltará a esa dirección al terminar el proceso de carga. En el caso de que el programa incluya también una parte en BASIC con autoejecución comenzará a ejecutarse el código, pero cuando aparezca el RET que devuelva el control se seguirá en la línea correspondiente del BASIC.

El programa salvará toda la versión TURBO como un bloque en el que se incluye pantalla, BASIC y código máquina, por lo que es muy aconsejable que éste sea lo más compacto posible. Si incluye pantalla o BASIC, es importante que el código máquina ocupe la zona de memoria más baja posible; un sitio ideal para las rutinas cortas es el buffer de la impresora.

Después de consultarnos sobre el código máquina se nos pregunta si vamos a utilizar pantalla de carga, y tras confirmar la exactitud de los datos introducidos anteriormente (si la negamos el proceso comienza desde la primera pregunta) empieza la fase en la que se carga cada bloque.

### Carga y grabación

Una vez que hayamos confirmado los datos introducidos, el programa nos indicará cuándo está preparado para cargar cada uno de los bloques que hayamos escogido. Comenzará con el código máquina (si lo hubiera), seguido de la pantalla



```
10 REM TURBOPROTECTOR (C)LGP
20 POKE 23658,8
30 CLEAR 64459: GO SUB 500
40
50 LET ENTC=0: LET FIN=0: LET
ONERR=32768: LET BASE=65535: LET
LINE=10000
60 PRINT "PROGRAMA BASIC?: ";
70 GO SUB 310: LET B=A
80 IF NOT B THEN GO TO 110
90 LET BASE=23568: PRINT " L
INEA DE AUTOEJECUCION:
(ENTER SI NINGUNA) >";: INPUT
LINE A$: GO SUB 340
100 IF A$<>" THEN LET LINE=VA
L A$: PRINT " LINEA PARA ""ON
ERROR""? (ENTER SI NI
NGUNA) >";: INPUT LINE A$: GO S
UB 340: IF A$<>" THEN LET ONER
```

```
R=VAL A$
110 PRINT "CODIGO MAQUINA?: ";
120 GO SUB 310: LET C=A
130 IF NOT C THEN GO TO 170
140 PRINT " DIREC. DE INICIO?
: >";: INPUT ORGC: LET A$=STR$
ORGC: GO SUB 340: IF ORGC<BASE
THEN LET BASE=ORGC
150 PRINT " LONGITUD?:
>";: INPUT LNGC: LET A$=STR
$ LNGC: GO SUB 340: LET FIN=ORGC
+LNGC
160 PRINT " DIREC. DE EJECUCI
ON?: "" (ENTER PARA NO EJEC.)>
";: INPUT LINE A$: GO SUB 340:
IF A$<>" THEN LET ENTC=VAL A$
170 PRINT "PANT. DE CARGA?: ";
180 GO SUB 310: LET P=A: IF A T
HEN LET BASE=16384: IF FIN<2329
6 THEN LET FIN=23296
```





```

190 PRINT "QUE NOMBRE LE PONGO?
>";: DIM B$(1,10): INPUT LINE
B$(1): LET A$=B$(1): GO SUB 340:
FOR N=1 TO 10: POKE 65182+N,COD
E B$(1,N): NEXT N
200 IF NOT B+C+P THEN RUN
210 PRINT "DATOS CORRECTOS?:";
: GO SUB 310: CLS : IF NOT A THE
N RUN
220 POKE 65521,FN L(BASE): POKE
65522,FN H(BASE): POKE 65523,FN
L(FIN): POKE 65524,FN H(FIN)
230 POKE 65517,B: POKE 65519,FN
L(LINE): POKE 65520,FN H(LINE)
240 POKE 65525,FN L(ENTC): POKE
65526,FN H(ENTC): POKE 65518,P:
POKE 65527,FN L(ONERR): POKE 65
528,FN H(ONERR)
250 IF NOT C THEN GO TO 270
260 BEEP .2,20: PRINT AT 10,0;

```

```

FLASH 1;" PREPARADO PARA CARGAR
CODIGO M.": LOAD ""CODE ORGC,LNG
C: CLS
270 IF P THEN BEEP .2,20: PRIN
T FLASH 1;AT 10,0;" PREPARADO P
ARA CARGAR PANTALLA ": LOAD ""SC
REEN$ : RANDOMIZE USR 64990
280 IF B THEN BEEP .2,20: PRIN
T AT 10,0; FLASH 1;" PREP. PARA
CARGAR PROGR. BASIC "
290 RANDOMIZE USR 64765
300
310 PRINT " (S/N) >";
320 LET A$=INKEY$: IF A$="" THE
N GO TO 320
330 IF A$<>"S" AND A$<>"N" THEN
GO TO 320
340 PRINT A$: LET A=A$="S": BEE
P .1,10: RETURN
350

```







de carga (si la hubiera) y por último, ya desde la parte en máquina, nos pedirá el programa BASIC (también si lo hubiera). Habrá que conectar convenientemente el cassette, y pulsar PLAY al comienzo de cada una de las partes.

En el caso de que se produzca algún error durante el proceso de carga, vale con pulsar CONTINUE y buscar de nuevo el bloque donde se produjo el error, a excepción de si ocurre durante la carga del BASIC, en cuyo caso habrá que hacer un RANDOMIZE USR 64765 en lugar del CONTINUE. Es posible, sin embargo, que en ese caso el mensaje de error que aparece en la parte baja estropee la pantalla de carga (si la hubiera), en cuyo caso habría que comenzar de nuevo cargando y ejecutando "TURBOPROTECTOR". De lo que no hay que temer que estropee la pantalla de carga es del mensaje "PREPARADO PARA CARGAR PROG. BASIC" o "Program:...", ya que esa línea de pantalla es salvada a memoria antes de que aparezcan, y devuelta de ésta antes de hacer la grabación.

Una vez cargadas todas las partes del programa aparecerá el conocido

***El incremento de velocidad puede dar problemas si se utiliza en la carga un cassette diferente al que se utilizó en la grabación.***

mensaje "Start tape, then press any key" (precisamente en la línea que tenemos «reservada»), indicando que se haya dispuesto para grabar la versión TURBO del programa. Debemos, pues, poner el cassette en condiciones de grabar y colocar la cinta donde queramos tener la versión definitiva. Tras pulsar PLAY, REC y una tecla cualquiera, comenzará la grabación. Una vez realizada se nos ofrece la oportunidad de realizar nuevas copias; en caso de respuesta negativa por nuestra parte, se regresa al BASIC.

### Programa en ensamblador

Para los aficionados al lenguaje máquina se ofrece también el lista-

do en ensamblador de la parte correspondiente del programa. Quien lo desee, podrá ensamblar este trozo por su cuenta y ahorrarse teclear, en el programa principal, desde la línea 500 hasta el final. En su lugar debe incluir una línea como la siguiente: 500 LOAD "CODE: RETURN, y salvar en cinta el código objeto tras el programa BASIC (este último con autoejecución en la línea 20).

Para quien le guste saber cómo funcionan las cosas se incluye además unos breves comentarios del cometido de cada subrutina. Como visión general cabe decir que lo que hace este programa es crear un pequeño programa BASIC que incluye el cargador TURBO (en código máquina) y los datos del bloque que le sigue. Este programa se autoejecuta, y ello no puede ser evitado mediante MERGE al incluir una línea falsa con numeración superior a la «legal» que bloquea al sistema. Este cargador es salvado en la cinta con la rutina estándar de la ROM, seguido del bloque que abarca todas las partes de nuestro programa, grabado a velocidad rápida.

El TURBO es muy rápido, por lo que puede dar algún problema de

```
860 DATA "08012000E5EDB0E1243DF
820F421405918EFCDCBFD3AEEFFB7C82
1CCFB1140483E08012000D5ED",4767
870 DATA "B0D1143DF820F41140591
8EFF321FA003E024710FED3FEEEOA06A
42D20F50525F224FE062F10FE",4456
880 DATA "D3FE3E0D063710FED3FE0
E08062326007AB3280BDD6E007CAD673
E023718186C18F579CB7810FE",3778
890 DATA "3004062A10FED3FE06262
0EF05AFC602CB1520EA1BDD2306193E7
FDBFE1F30077A3C20C5C3C1FF",4136
900 DATA "CF14801708204F5452412
0434F5049413F2028532F4E29A000000
000000000000000051010000",1750
910 DATA "510100004901F9C028BE3
00E0000535C002B300E00000001002AB
E300E0000545C002B300E0000",1745
920 DATA "2B0000293AEA2AB25C110
4FF011800EDB813EBF92A535C1156001
91105FF01FB00EDB0CD3EFFED",3959
```

```
930 DATA "5BF9FFDD2AF1FFC305FFC
D59FF3AEDFFB7CC3EFFB7280A2AEFF2
2425CFD360A002104FF22B25C",5486
940 DATA "2B2B2B223D5C3AF8FF173
80601D9FF712370CD6B0D2AF5FF7CB5C
8E92A535C3680224B5C232259",4058
950 DATA "5CC3B016FDCB01AEFBFDC
B016E28FAC9F30E26BF206CCDE70530F
92E1810FE2D20FBCDE30530ED",5179
960 DATA "26C42E00069CCDE30530E
23EC6B830DF2420F106C9CDE70530D37
BFED430F4CDE705303779EE02",4867
970 DATA "4F260006B01808DD7500D
D231B06B22E01CDE305301E3EBDB8CB1
506B030F27CAD677AB320E17C",3911
980 DATA "B7200AFD7E0E1F1F1FD3F
EFBC9CDCBFD112415AFCD0A0CCD4DFFC
7ED7B3D5CCDB0162AF7FF2242",5061
990 DATA "5CFD360A00C39E1B00000
000000000000000000000000D7F0001"
,930
```



carga cuando se utiliza para ello un cassette distinto al que se usó para grabarlo; esto puede ser solucionado alineando correctamente la cabeza reproductora.

## Subrutinas

**TRBPRT:** Es el bucle principal, desde el que se llama a las demás subrutinas.

**LD\_BSC:** Carga el programa BASIC, actualiza las correspondientes variables del sistema e imprime el nombre del programa en la línea 10 (la que está almacenada en memoria).

**CLSL10:** Borra la línea 10. En realidad imprime 32 espacios en dicha línea.

**SA\_L10:** Copia, si es necesario, el contenido de la línea 10 (parte de la pantalla de presentación) a partir de BUFL10, incluidos los atributos.

**LD\_L10:** Lo contrario que la anterior; recupera de memoria la línea 10.

**TRBSA:** Junto con TRBLD es la más importante, ya que es la encargada de hacer la grabación TURBO de la zona de memoria que se le especifique. Funciona de forma similar a la rutina SA\_BYTES de la

**El turboprotector incluye una protección del tipo "ON ERROR", que en caso de que se intente detener el programa devuelve el control a una línea determinada.**

ROM, es decir, a la entrada el par de registros DE debe contener la longitud del bloque a salvar, y el par IX la dirección de la base del bloque; cuando es llamada comienza la grabación sin imprimir ningún mensaje.

**CAB\_CB:** Es la cabecera del cargador BASIC. Incluye el nombre que le dimos, que fue pokeado desde el BASIC a esta dirección.

**CARG\_B:** Es el cargador en sí. Incluye una línea cero, donde se ejecuta el código máquina relativo a PROG y a la que sigue una línea REM con el cargador TURBO y las variables; tras todo esto hay una línea 32525 que evita el MERGE.

**ENTCRG:** Aquí comienza a ejecutarse el código máquina cuando se autoejecuta el cargador BASIC. Cambia de sitio el stack de

ENTCR1-1 (si no ha fallado nada, esto es en la dirección 65284), y, después de copiar todo lo que le sigue a la parte alta de la memoria, salta allí.

**ENTCR1:** Llama a TRBLD y actualiza las variables del sistema; después salta a la dirección de ejecución que tenga marcada o vuelve al BASIC (donde autoejecutará si le corresponde).

**NEWBSC:** Borra el programa BASIC. Lo que hace es poner PROG, VARS y E\_LINE a sus mínimos valores.

**EP\_TCL:** Espera a que se pulse una tecla.

**TRBLD:** Se encarga de cargar el bloque en TURBO de forma similar a TRBSA. Caso de que se detecte algún error de carga se salta a ERROR.

**ERROR:** Imprime "Tape Loading error" e inicializa el ordenador en cuanto se pulsa una tecla.

**ON\_ERR:** Es la subrutina a la que se salta cuando se produce un error del BASIC (siempre que se hubiera seleccionado esta opción en su momento). Pone el número de línea y sentencia correspondientes y salta a LIN\_NW (en la ROM) para que la ejecución de BASIC pase a esa línea.

Luis Gala

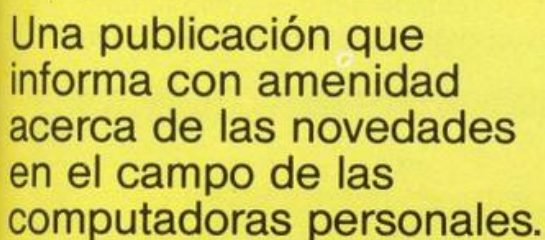
## GUSANEZ

por José C. Tomás





LA REVISTA QUE INTERESA TANTO AL AFICIONADO COMO AL PROFESIONAL



**ORDENADOR POPULAR,**  
la revista para el  
aficionado a la  
informática.

**Ya está a la venta**

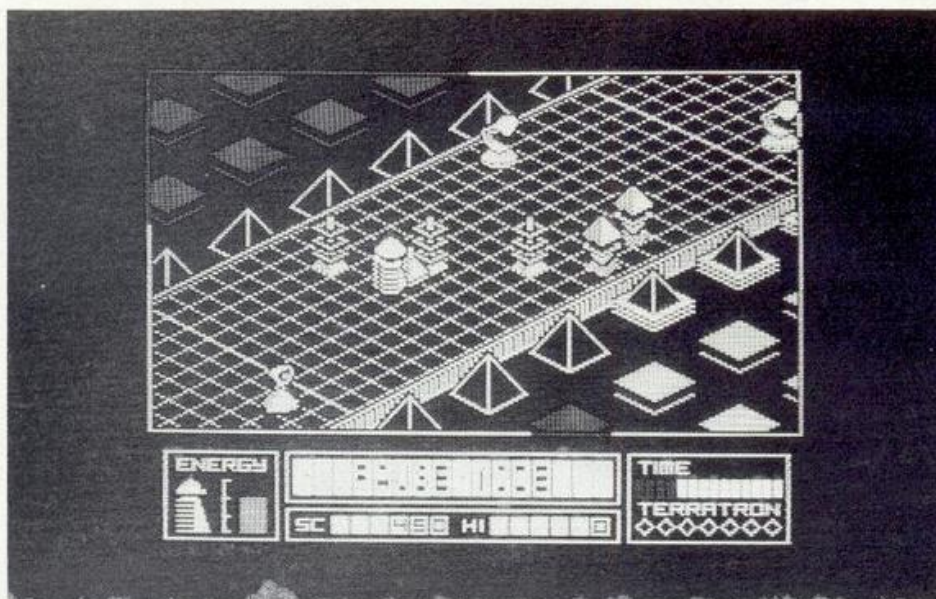
**Cómprela en su kiosco habitual o solicítela a:**

# ORDENADOR POPULAR

**Bravo Murillo, 377**  
**Tel. 7339662**  
**28020 - MADRID**



# JUEGOS



## ALIEN HIGHWAY

ABC SOFT  
SPECTRUM 48 K

De nuevo deberemos conducir un Lasertrón a lo largo de muchos tramos de una futurista autopista (valga lo rimbombante) que nos conducirá al cuartel general de Alien, el enemigo de la Galaxia. Sólo si maniobramos con mucha habilidad el Vortrón único de que disponemos, y logramos esquivar o destruir a los múltiples alienígenas que se cruzarán en nuestro camino, podremos dar por finalizada esta misión tan dificultosa como esencial para el mantenimiento de la paz en la Galaxia.

Si te ha sonado muy a chino todo el párrafo anterior a éste, será sin duda por que nunca pasó por tus manos un juego tan famoso como fue no hace

autopista que hay que cruzar para llegar allí no están siempre en el mismo orden, por lo que no vale aquí eso de aprenderse de memorieta pantalla a pantalla para finalizar.

Aunque las rutinas de movimiento usadas son las mismas que en Encounter, se ha sacado aún más partido al original estilo gráfico creado en esa ocasión, tanto en los personajes, objetos y paisajes de la aventura como en las pantallas de presentación, etc. Lo mismo hay que decir de la mayoría de esos detalles que hacen a un juego atractivo a primera vista y agradable de manejar.

En definitiva, se trata de una honorable segunda parte que será bienvenida por los Highwaymaniacos de nuestro país (de todo hay en la viña del Señor). Desearíamos que a partir de ahora los señores de Vortex hagan un esfuerzo para no atascarse aquí y volver a dedicarse a la creación de nuevos tipos de juegos que rompan moldes como lo hizo Highway Encounter en su día.

mucho el Highway Encounter. En esta segunda y (según reza la publicidad del juego) definitiva parte de la odisea las cosas han cambiado un tanto respecto a su predecesora. Como se ha señalado, en esta ocasión es un único Vortrón, el que debe empujar al Lasertrón hacia la base enemiga, y los tramos de





# RIDDLER'S DEN

PROEINSA  
SPECTRUM 48 K

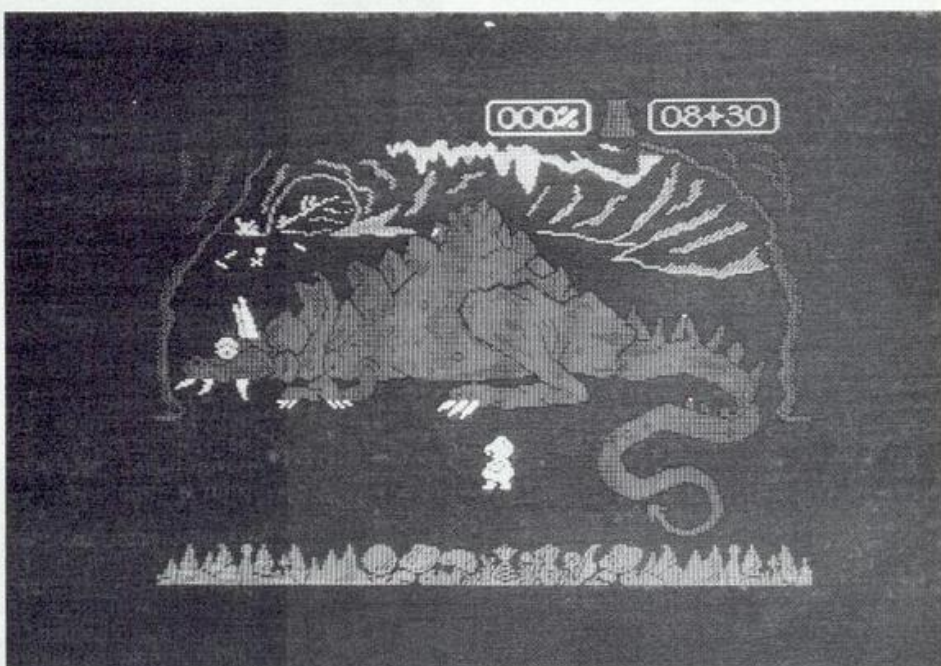
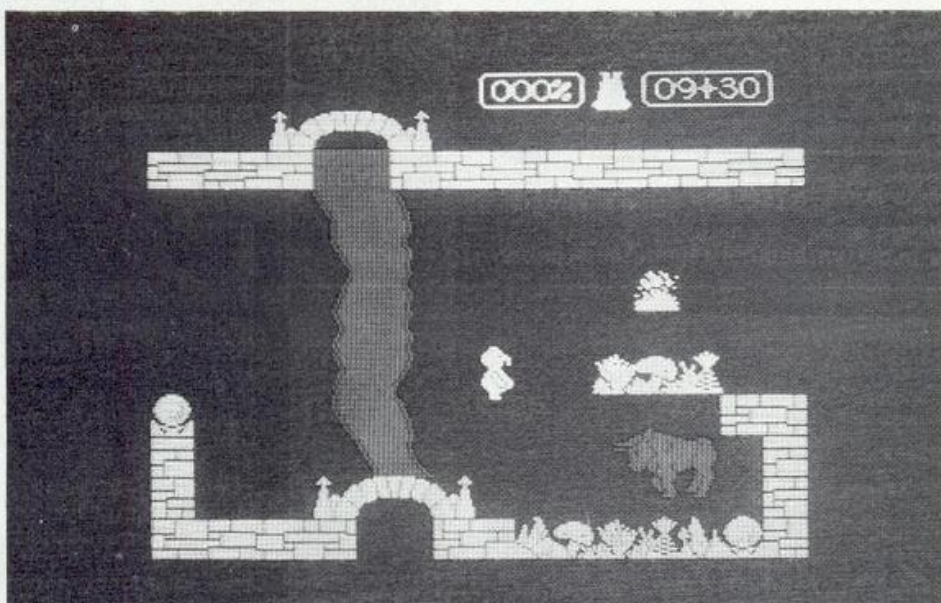
Del otro lado del Atlántico nos llega este juego, uno de tantos entre la oleada de arcade-aventuras que desde hace algún tiempo invade el mercado Spectrum. Sin buscar la sofisticación de los últimos grandes éxitos de esta clase, Riddler's Den destaca (si destaca en algo) por lo complicado de sus enigmas, que nos hace perder una vez tras otra la única vida que disfrutamos en cada partida. De hecho, es poca la acción que alcanza aún en sus momentos más álgidos, consistiendo todo el juego en un simple ir y venir en busca de objetos y un volverse tarumba averiguando las utilidades de los mismos.

El protagonista (nada menos que el hombre elefante) debe andarse la laberíntica Cueva de los Enigmas en busca de ciertos tesoros, especialmente del Colmillo Dorado. Habrá para ello de superar muchas pruebas y tratar de conocer los objetos que pueden resultarle de alguna utilidad en su aventura.

Para recuperarse del agotamiento que le producirá el acoso de ciertos enemigos, en especial los ogros y los duendes (otros acabarán con él fulminantemente), el protagonista puede optar por echar una cabezada. Para ello necesitará el objeto adecuado, evidentemente una almohada, difícil de reconocer por lo mal que ha sido dibujada. No es nada aconsejable el abusar de esto,

pues con ello se fuerza al paso de un día y la prueba se desarrolla durante un número determinado de estos, transcurridos los cuales nada podrá hacerse para triunfar. El estilo gráfico usado, sin ser malo, no es demasiado elogiable, tanto por el diseño en sí mismo como por la poca calidad de las secuencias de animación. Aún así la rutina utilizada para el movimiento de

los sprites es buena, con movimiento de un pixel por interrupciones, lo que le da un aspecto de conjunto bastante pasable. En general, puede hablarse de un programa no excesivamente volcado en los detalles, que cumple sin resaltar en nada, y más preocupado en lo estratégico de la aventura que en hacer que las partes más duras sean entretenidas y llevaderas para el jugador.





## PUD-PUD

ZAFIRO

SPECTRUM 48 K

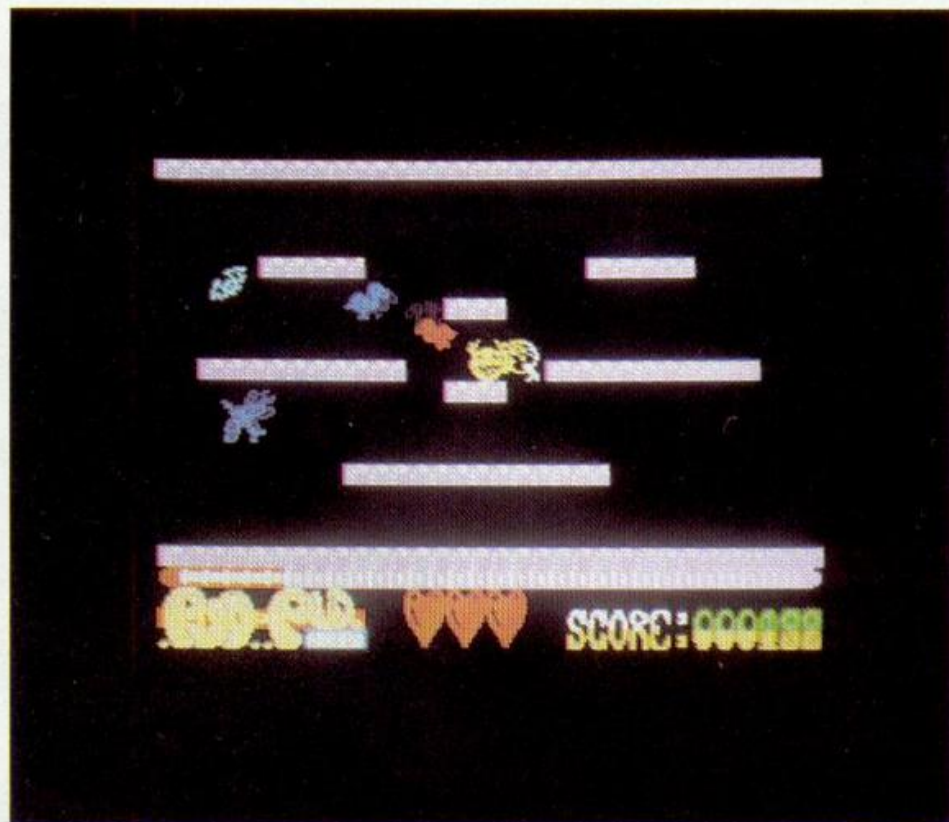
El objetivo que debemos perseguir en este juego es ayudar al pequeño Pud-Pud a escapar del loco mundo donde ha entrado en una pesadilla y del que no sabe cómo salir. Para encontrar la salida secreta del satánico laberinto debe comerse uno por uno todos los puddings (diez en total) que se encuentran diseminados por los más recónditos rincones del mapa.

Pero esto no es fácil, sobre todo si tenemos en cuenta que la energía de las tres vidas de que dispone se gasta con cierta rapidez, sobre todo cuando entra en contacto con determinados enemigos.

Debemos tener en cuenta también que Pud-Pud, a pesar de su extraña predilección por los puddings de todo tipo (por la que sus familiares y amigos le toman por loco), procede de una famosa estirpe de Pud-Pud insectívoros, por lo que es en este tipo de seres donde encontrará las energías que palién el desgaste de tanta aventura.

Pud-Pud es una criatura muy sensible, y en sus pesadillas suelen manifestarse sus paranoias en forma de la Sra. Pud-Pud, irrefrenable ser que constantemente suspira de amor por nuestro héroe. Deberemos tener cuidado, la Sra. Pud-Pud es muy absorbente y acabará con la poca energía con que suele contar su ídolo.

El nivel técnico del juego es decente aunque sin aportar



nada innovador. La rutina de impresión resulta algo molesta por los parpadeos en ciertas zonas de la pantalla, pero por lo demás los personajes se mueven con relativa rapidez y «resultan». Los gráficos son bastante buenos, con un estilo que se ve más resaltado en presentación o en algunas partes del laberinto que en los

propios sprites, a los que además les afea mucho el comentado parpadeo. Es, pues, un juego que puede resultar entretenido y ciertamente adictivo, aunque acaba cansando por su sencillez, que debería ser contrarestrada por algún planteamiento que lo hiciera más atractivo.





# Todospectrum



**TODOSPECTRUM** es una publicación mensual que le ayudará a obtener el máximo partido a su **SPECTRUM** y al **ZX 81**.

CONOZCA LAS VENTAJAS DE SUSCRIBIRSE A

## Todospectrum

*Sensacional  
Oferta de Suscripción*

**GRATIS  
PARA USTED  
SI SE SUSCRIBE A  
TODOSPECTRUM**  
2 cintas cassettes  
cuyo valor real es de  
**1750 PTAS**



**ADEMAS**, le hacemos un **25 % DE DESCUENTO**  
sobre el precio real de suscripción (12 números)

VALOR REAL DE  
SUSCRIPCION

~~3.600~~ PTAS.

OFERTA ESPECIAL  
DE SUSCRIPCION

**2.700 PTAS.**

USTED AHORRA

**900 PTAS.**

**APROVECHE AHORA** esta oportunidad irrepetible para suscribirse a **TODOSPECTRUM**. Envíe **HOY MISMO** la tarjeta adjunta a la revista, que no necesita sobre ni franqueo. Deposítela en el buzón más cercano. Inmediatamente recibirá su primer ejemplar de **TODOSPECTRUM** más el **REGALO**.

## Todospectrum

Bravo Murillo, 377  
Tel. 733 79 69  
28020 MADRID



## TURBO ESPRIT

DURELL

SPECTRUM 48 K

Ciertas ciudades del Reino Unido se ven amenazadas por una importante banda de delincuentes especializados en el tráfico de drogas y armas; toda una mafia que no se detiene ante nada y tienen aterrorizados ya a los barrios bajos de la «city». Como agentes del orden, deberemos acabar con todo esto antes de que sea demasiado tarde. Dispondremos para ello de un potente Lotus que deberemos pasear por calles y avenidas esperando que un mensaje de la central nos indique la presencia de algún coche de esa organización por las cercanías.

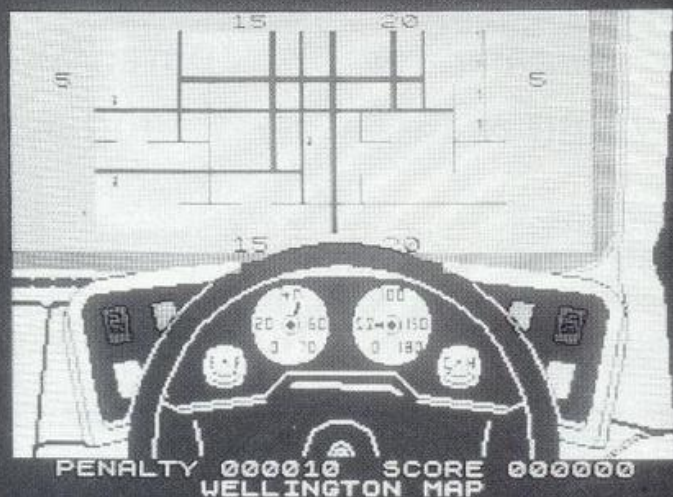
Al comenzar el juego se nos permite elegir entre varios el escenario donde se desarrollará la aventura, y, tras esto, un

surtido menú nos permite alterar el nivel de dificultad, redefinir teclas, ver las tablas de mejores puntuaciones y records de accidentes, salvar o cargar a/de cinta esas tablas, o comenzar el juego propiamente dicho.

Cuando empezamos a jugar nos encontramos con una representación tridimensional de las calles de bastante calidad, que, aunque empañada por unos gráficos medianos, no deja

de dar muy buena sensación de movimiento al vehículo. Al principio es bastante difícil controlar el coche; sobre todo en los cruces, en los que es frecuente acabar empotrados en las esquinas o contra otro coche. En este sentido no resulta de mucha ayuda que el juego sea inglés, pues trae como resultado el que se circule por la izquierda para mayor despiste del personal.

Cuando conseguimos acostumbrarnos a las reacciones de nuestro bólido, acaba resultando un juego muy atractivo, sobre todo por la cantidad de detalles que adornan las calles por donde se desarrolla la acción. Entre estos, hay que destacar la forma en que se mueven los abundantes coches que pueblan la ciudad, ya que lo hacen de forma bastante «inteligente», pudiéndose esperar de ellos reacciones totalmente lógicas. Otros detalles como los semáforos, etc., terminan de dar forma a un juego de los que pueden ser catalogados como buenos.





# RASPUTIN

FIREBIRD

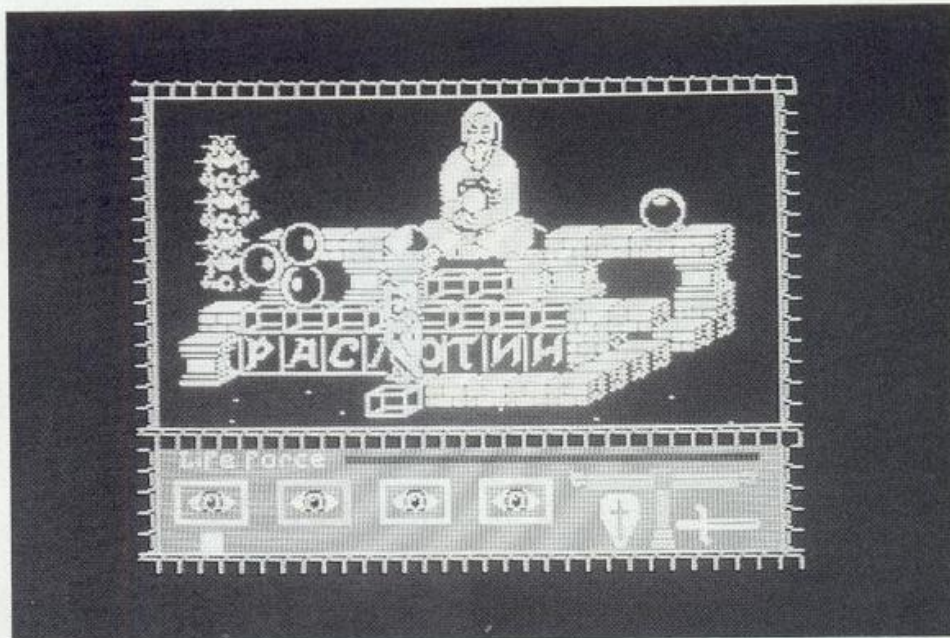
SPECTRUM 48 K

Ivan Kosmovichski, conocido por sus allegados como «el Cruzado» debe cumplir con la importante misión de destruir la Joya de los Siete Planetas —la fuente de energía del malvado espíritu de Rasputín, que está amenazando con volver a la vida para destruir el cosmos. Armado con un escudo y una espada mágica, Ivan viajará a través de la oscuridad y el vacío de los terroríficos Siete Planetas donde habrá de luchar con los cíclopes, las cyberatas, las esferas de poder psiónico y algunas obras pérfidas criaturas.

Con un estilo similar al que nos tiene acostumbrados Ultimate con su técnica Filmation, Rasputin se queda un poco por debajo en cuanto al nivel técnico de algunos detalles. Por ejemplo, el movimiento del protagonista es bastante peor que el de los mencionados «ultimates», y no ya por una inferior calidad de las rutinas empleadas, sino más bien por la secuencia de animación utilizada, que otorga al personaje el don del movimiento «ortopédico».

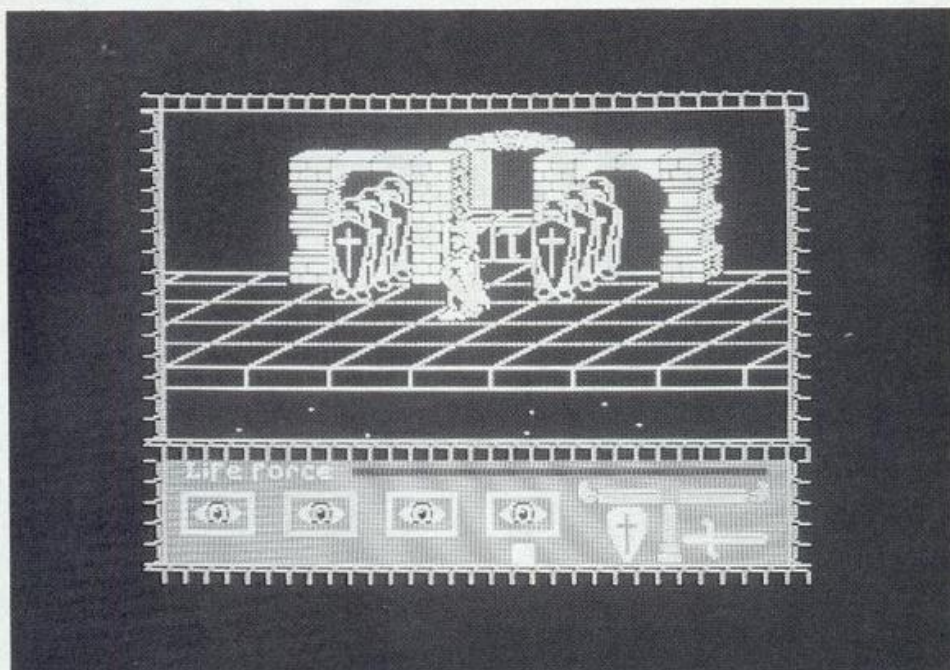
Sin embargo, el juego presenta grandes virtudes en otros campos, e incluso en el propio terreno gráfico, que puede elogiarse por la agradable línea seguida en la mayor parte del juego (que, desgraciadamente, se ve alterada algunos puntos).

El movimiento de los sprites es



suave pero no todo lo rápido que sería deseable en lo concerniente al protagonista, que anda, más o menos, tan poco como el viejo Sabreman. Los efectos especiales y detalles de presentación no son abundantes ni especialmente buenos, ni siquiera en lo relativo al sonido, que no es el fuerte de este juego.

Lo más resaltable, sin duda, es lo elevado que llega a ser su nivel de dificultad en algunas pantallas, en las que parece imposible imaginar siquiera la forma de pasar los interminables obstáculos que impiden el avance. Aconsejable para los grandes aventureros a los que no les gusta dejar sin batir a un juego de este tipo.

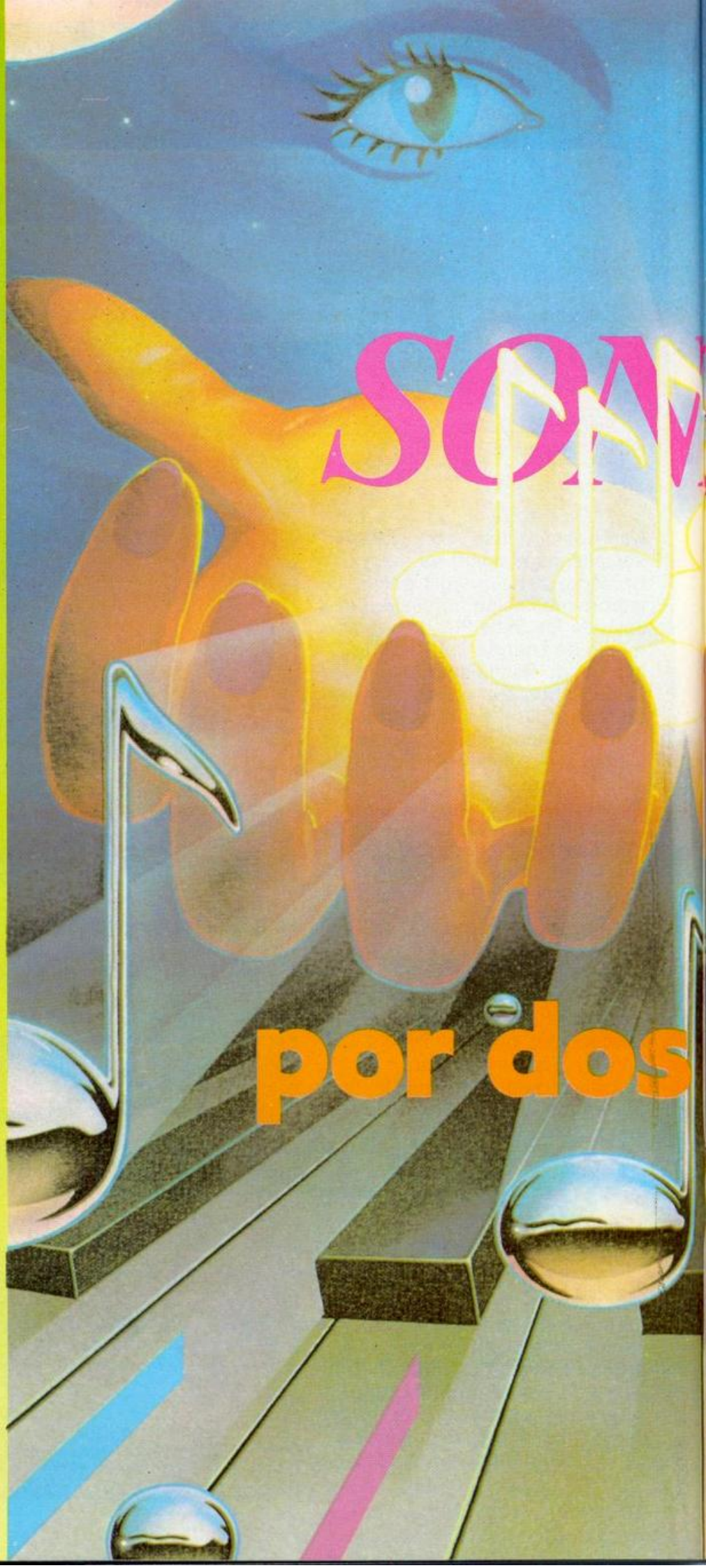




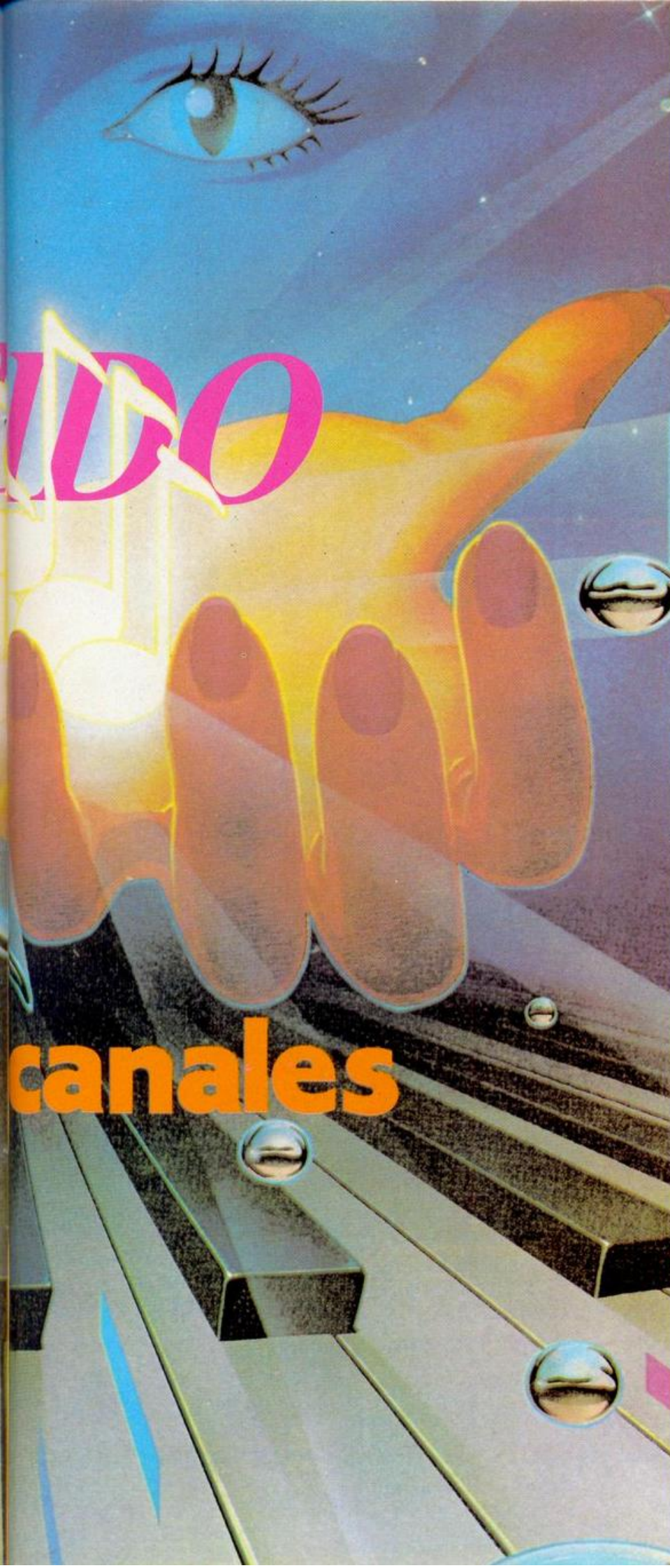
*Si hemos jugado alguna vez con otros ordenadores, nos habremos quedado sorprendidos al oír sonidos y música. Este es uno de los apartados más pobres del Spectrum. Sin embargo, a partir de ahora, también podremos disfrutar de música a dos voces.*

**E**l sonido en el Spectrum se genera mediante el altavoz interno, controlado por la ULA. Para producir un sonido durante un programa en código máquina, se hace uso de la instrucción OUT, en alguna de sus formas. El *port* que se ha de utilizar es el 254 (11111110b).

Como muchos lectores ya sabrán, al utilizar este *port* se accede tanto al altavoz como al color del borde. Ambos se controlan mediante el contenido del registro A en el momento de ejecutarse la instrucción OUT. Por ello, se hace una división por *bits*: El color del borde se indica con los *bits* 0 a 2 (los que valen 1, 2 y 4), mientras que para el altavoz se utiliza el *bit* 4, cuyo valor es 16. El resto de los *bits* tienen asimismo un significado pero esto no nos interesa para el tema de este artículo.







## Producción de sonido

Ahora concentrémonos en el *bit* de sonido. Este sólo puede estar a 0 o a 1, dos estados distintos, que coinciden con los dos posibles de una onda cuadrada, que es el único tipo de onda que se puede generar. El cero corresponde al estado alto y el uno al estado bajo. El sonido se produce al pasar de un estado a otro, ya sea de 0 a 1 o de 1 a 0. Con esto producimos un chasquido. Las distintas notas musicales se producen mediante una serie de estos chasquidos repetidos a intervalos de tiempo determinados. Cuanto menor sea este intervalo, mayor será el número de chasquidos producidos en un mismo tiempo y la frecuencia será mayor, o lo que es lo mismo, la nota será más aguda. Por tanto, para poder generar sonido puro, es necesario poder sincronizar el tiempo al máximo.

## T-Estados

Probablemente, la mayoría de los lectores habrán oído hablar poco o nada de los T-estados, pues es un tema que no se trata demasiado en libros, sobre todo en libros de iniciación. Por ello, definiremos brevemente que es un T-estado, ya que es



FIGURA 1. Listado en Ensamblador

10 ;		530 ;	
20 ;		540 NOTO	LD (CA),SP
30 ;	SIMULACION DOS CANALES	550	LD B,0
40 ;	DE SONIDO	560	LD E,D
50 ;		570	LD L,H
60 ;		580	LD A,(23624)
70	ORG 65368	590	RRA
80 ;		600	RRA
90 ;		610	RRA
100	LD HL,(23670)	620	AND 7
110 SONDO	DI	630	PUSH AF
120	LD A,I	640	EX AF,AF'
130	PUSH AF	650	POP AF
140	PUSH HL	660 LOC	LD I,A
150 LON	XOR A	670 LOSN	EX AF,AF'
160	IN A,(254)	680	EX DE,HL
170	OR 224	690	OUT (254),A
180	INC A	700	DEC L
190	JR NZ,FISO	710	JR NZ,NOCH
200	LD E,(HL)	720	XOR 16
210	INC E	730	LD L,H
220	JR NZ,SIG	740	DJNZ LOC
230	POP HL	750	DEC C
240	PUSH HL	760	JP NZ,LOSN
250	LD E,(HL)	770	LD SP,(CA)
260	INC E	780	RET
270 SIG	DEC E	790 NOCH	LD SP,HL
280	INC HL	800	DJNZ LOC
290	PUSH HL	810	DEC C
300	LD L,(HL)	820	JP NZ,LOSN
310	LD H,0	830	LD SP,(CA)
320	LD D,0	840	RET
330	LD BC,NORET	850 ;	
340	ADD HL,BC	860 ;	
350	LD H,(HL)	870 ;	TABLA DE NOTAS
360	EX DE,HL	880 ;	
370	ADD HL,BC	890 ;	
380	LD H,(HL)	900 NORET	DEFB 212,200
390 TEMPO	LD C,38;VELOCIDAD	910	DEFB 189,179,169,15
	DE EJECU-	9,150	
	CION	920	DEFB 142,134,126,11
400	CALL NOTO	9,112	
410	POP HL	930	DEFB 106,100,95,89,
420	INC HL	84,80	
430	JR LON	940	DEFB 75,71,67,63,60
440 FISO	POP HL	,56	
450	POP AF	950	DEFB 53,50,47,45,42
460	LD I,A	,40	
470	EI	960	DEFB 38,35,33,32,30
480	RET	,28	
490 ;		970	DEFB 27,25,24,22,21
500 ;		,20	
510 CA	DEFW 0	980	DEFB 19,18,17,16,15
520 ;		,14,13,1	



un concepto imprescindible a la hora de intentar producir sonidos.

Un T-estado es la unidad de tiempo del microprocesador. Todas las instrucciones del lenguaje máquina tardan un tiempo exacto en ser ejecutadas. Este tiempo no se mide en segundos ni en minutos, sino en T-estados. Esta es una medida mucho más cómoda de usar, sobre todo por que un T-estado *no se corresponde con una duración concreta de tiempo*. Entre los muchos componentes que tiene un ordenador hay uno muy importante y que se encuentra en todos ellos: el reloj. No es un reloj

normal de los que dan la hora. Es simplemente un dispositivo que emite una señal a intervalos regulares de tiempo. Su función es sincronizar el microprocesador con el resto de los componentes para que vayan todos a la misma velocidad. Sin el reloj, las distintas partes del ordenador no se entenderían. De todo esto se deduce que la velocidad del microprocesador depende de la velocidad de un reloj. Es por esto que un T-estado no es una magnitud absoluta de tiempo, si no que varía proporcionalmente con la velocidad del reloj. Por eso, para saber el

tiempo absoluto que tarda una instrucción en ejecutarse es necesario conocer tanto su duración en T-estados como la velocidad del reloj. En el Spectrum el reloj funciona a 3'5 Mhz, o 3500000 hz. Esto quiere decir que emite 3500000 pulsos en un segundo, por lo que en el Spectrum hay 3500000 T-estados en un segundo. Mediante una simple división comprobamos que un T-estado durará entonces  $1/3500000 = 0'00000085714$  segundos. Ahora ya estamos en condiciones de saber exactamente cuánto tarda un programa en ejecutarse, con sólo dispo-

**FIGURA 3. Programa para la composición musical**

```

10 POKE 23658,0: PAPER 0: BORD
ER 6: INK 2: BRIGHT 0: FLASH 0:
OVER 0: INVERSE 0: CLEAR 99999:
LOAD "CODE"
20 LET M=0: LET V=0: DIM A(12
,4): FOR X=1 TO 12: READ A(X,1):
NEXT
30 DATA "DO","DB","RE","RE","
"MI","FA","FA","SOL","SOL","LA
","LA","SI"
140 CLS: RESTORE 150: PRINT AT
1,14:"MENU" TAB 14:"*****": FOR
X=1 TO 8: READ B: PRINT AT 3+2*
X,4:X: "-":B: "-":NEXT X
150 DATA "GRABAR LA MUSICA","GR
ABAR EL CODIGO","CARGAR UNA MUSI
CA","VOLCAR LA MUSICA","TOCAR LA
MUSICA","CORREGIR LA MUSICA","C
OMENZAR A COMPONER","CONTINUAR C
OMPONENDO"
160 PRINT 10:"ELIGE LA OPCION D
ESEADA"
170 LET B=INKEY$: IF B<"1" OR
B>"8" THEN GO TO 170
180 CLS: IF M=0 AND B<"3" AN
D B<"7" THEN PRINT 10: FLASH
1:"NO HAS COMPUERTO NINGUNA MUSI
CA": PAUSE 0: GO TO 140
190 IF V=0 AND (B<"2" OR B<"5
") THEN PRINT 10: FLASH 1:"
NO HAS VOLCADO LA MUSICA": PA
USE 0: GO TO 140
200 GO SUB 1000*VAL B: GO TO 1
40
1000 LET A(1,600,1)=C(1): LET A
(2,600,1)=R(1): LET A(2,600,2)=R
(2)
1010 INPUT "NOMBRE? ": LINE B:
IF B="" OR LEN B>10 THEN BEEP
.5,-10: GO TO 1010
1020 SAVE B: DATA A(): RETURN
2000 INPUT "NOMBRE? ": LINE B:
IF B="" OR LEN B>10 THEN BEEP
.5,-10: GO TO 2000
2010 SAVE B:CODE 5E4,C(1)*2+1: R
ETURN
3000 DIM C(2): DIM R(2): LOAD "
DATA A(): LET C(1)=A(1,600,1):
LET C(2)=C(1): LET R(1)=A(2,600
,1): LET R(2)=A(2,600,2): LET M=1
: RETURN
4000 LET A=50000: FOR N=1 TO R(1
): FOR G=1 TO A(1,N,2): POKE A,A
(1,N,1): LET A=A+2: NEXT G: NEXT
N
4010 LET A=50001: FOR N=1 TO R(1
): FOR G=1 TO A(2,N,2): POKE A,A
(2,N,1): LET A=A+2: NEXT G: NEXT
N
5000 INPUT "VELOCIDAD DE EJECUCI
ON?"
(1=RAPIDO 256=LENTO)
":W: LET W=W-256*INT (W/256)
5010 POKE 45409,W: RANDOMIZE 5E4
5020 IF INKEY<" " THEN GO TO 5
020
5030 RANDOMIZE USR 65368: RETURN
6000 INPUT "QUE CANAL QUIERES CO
RREGIR? ":C: LET P=1: IF C>1 AN
D C<2 THEN BEEP .5,-10: GO TO
6000
6010 INVERSE 0: PRINT "NUM. NOT
A ESCALA DURACION"
6020 FOR H=P TO P+19: IF A(C,H,2
) THEN LET X=A(C,H,1): PRINT "
":X: " AND H<10:" " AND H<100:H:
":(A(1+X-12*INT (X/12))+
)+STR$ (INT (X/12)) AND X/12<4)
:("SILENCIO" AND X/12=4):
":A(C,H,2): NEXT H
6030 INVERSE 1: LET H=P
6040 PRINT AT H-P+2,0: OVER 1:"
": LE
T H=H-1: GO TO 6040
6050 IF INKEY<" " AND H=P THEN
PRINT AT H-P+2,0: OVER 1:"
": LE
T H=H-1: GO TO 6040
6060 IF INKEY<" " AND H=19: P AN
D A(2,H+1,2) THEN PRINT AT H-P+
2,0: OVER 1:"
": LET H=H+1: GO TO
6040
6070 IF INKEY<" " AND A(C,H,2)>
1 THEN LET A(C,H,2)=A(C,H,2)-1:
PRINT AT H-P+2,22:A(C,H,2): LET
C(C)=C(C)-1
6080 IF INKEY<" " AND A(C,H,2)<
9 THEN LET A(C,H,2)=A(C,H,2)+1:
PRINT AT H-P+2,22:A(C,H,2): LET
C(C)=C(C)+1
6090 IF INKEY<" " AND A(C,H,1)<
51 THEN LET A(C,H,1)=(A(C,H,1)-
49 AND A(C,H,1)<47)+49: LET X=A
(C,H,1): PRINT AT H-P+2,6:(A(1+X
-12*INT (X/12))+
)+STR$ (IN
T (X/12)) AND X/12<4) ("SILENCIO" AND X/12=4)
6100 IF INKEY<" " AND A(C,H,1)>
0 THEN LET A(C,H,1)=A(C,H,1)-1-
A(C,H,1)+49: LET X=A(C,H,1): P
RINT AT H-P+2,3:(A(1+X-12*INT (X
/12))+
)+STR$ (IN
T (X/12)) AND X/12<4) ("SILENCIO" AND X/12=4)
6102 IF INKEY<" " THEN LET C(C
)=C(C)-A(C,H,2): LET R(C)=R(C)-1
: FOR X=H TO R(C): FOR J=1 TO 2:
LET A(C,X,J)=A(C,X+1,J): NEXT J
: NEXT X: LET A(C,X+1,1)=0: LET A
(C,X,2)=0: CLS: GO TO 6010
6106 IF INKEY<" " THEN LET R(C
)=R(C)+1: FOR X=R(C) TO H+1 STEP
-1: FOR J=1 TO 2: LET A(C,X,J)=
A(C,X-1,J): NEXT J: NEXT X: LET
C(C)=C(C)+A(C,H,2): CLS: GO TO
6010
6110 IF INKEY<" " OR H+13 THEN GO
TO 6050
6120 INVERSE 0: IF A(C,P+20,2) T
HEN LET P=P+20: CLS: GO TO 601
0
6130 GO TO 7130
7000 DIM R(2): LET R(1)=1: LET R
(2)=1: DIM C(2): DIM A(2,600,2)
7040 LET DU=1: LET ESC=0: FOR C=
1 TO 2: FOR N=P(C) TO 599
7050 PRINT AT 11,0:"DURACION: ":
DU:"ESCALA: ":ESC:AT 0,0:"CANAL
":C:AT 3,1:"NOTA ":N:" ": INPUT
LINE B: IF B="" THEN LET R(C)=
N: LET N=600: GO TO 7120
7051 IF LEN B<2 THEN BEEP .5,-
10: GO TO 7050
7052 IF B<LEN B>"0" AND B<LE
N B>="9" THEN LET DU=VAL B<LE
N B>: LET B=B<LE N B>: TO LEN B<LE
N B>
7054 IF B<1>="0" AND B<1>="3
" THEN LET ESC=VAL B<1>: LET B
<1>=B<2 TO 1>
7055 IF B<1>="S" THEN LET A(C,N,1
)=49: LET A(C,N,2)=DU: GO TO 712
0
7070 LET B<1>=B<1> "1 TO 4)
7080 FOR H=1 TO 12: IF A(H)=B<1>
THEN GO TO 7100
7090 NEXT H: BEEP .5,-10: GO TO
7050
7110 LET A(C,N,1)=H-1+12*ESC: LE
T A(C,N,2)=DU: LET C(C)=C(C)+DU
7120 NEXT N: NEXT C
7130 IF C(1)=C(2) THEN PRINT 1
0: FLASH 1:" ERROR
": PAUSE 0: CLS: GO
TO 6000
7140 LET H=1: RETURN
8000 GO TO 7040

```



ner de una tabla con las duraciones de todas las instrucciones.

## Generando Música

Ya sabemos como se crea sonido mediante el cambio del *bit* 4 del registro A en una instrucción OUT. Ahora vamos a ver un ejercicio práctico para generar una nota. Supongamos, por ejemplo, que queremos tocar el DO central durante un segundo. La frecuencia de la nota DO tal y como se ve en la figura 1 es de 261'63 hz. Por tanto, debemos hacer cambios de 0 a 1 y de 1 a 0 261'63 veces por segundo. Es decir, que entre

cada OUT con el mismo valor en el *bit* cuatro debe haber una pausa de  $1/261'67$  segundos o  $3500000/261'63$  T-estados.

Como entre esos dos OUTS del mismo calor habrá otro con el valor contrario, la pausa entre cada OUT será justo la mitad de la anterior  $3500000/523'26=6688'8354$  T-estados.

Hay que tener en cuenta, sin embargo, que dentro de esa pausa hay que contar el tiempo que tarda en ejecutarse la propia instrucción OUT y algunas otras cosas.

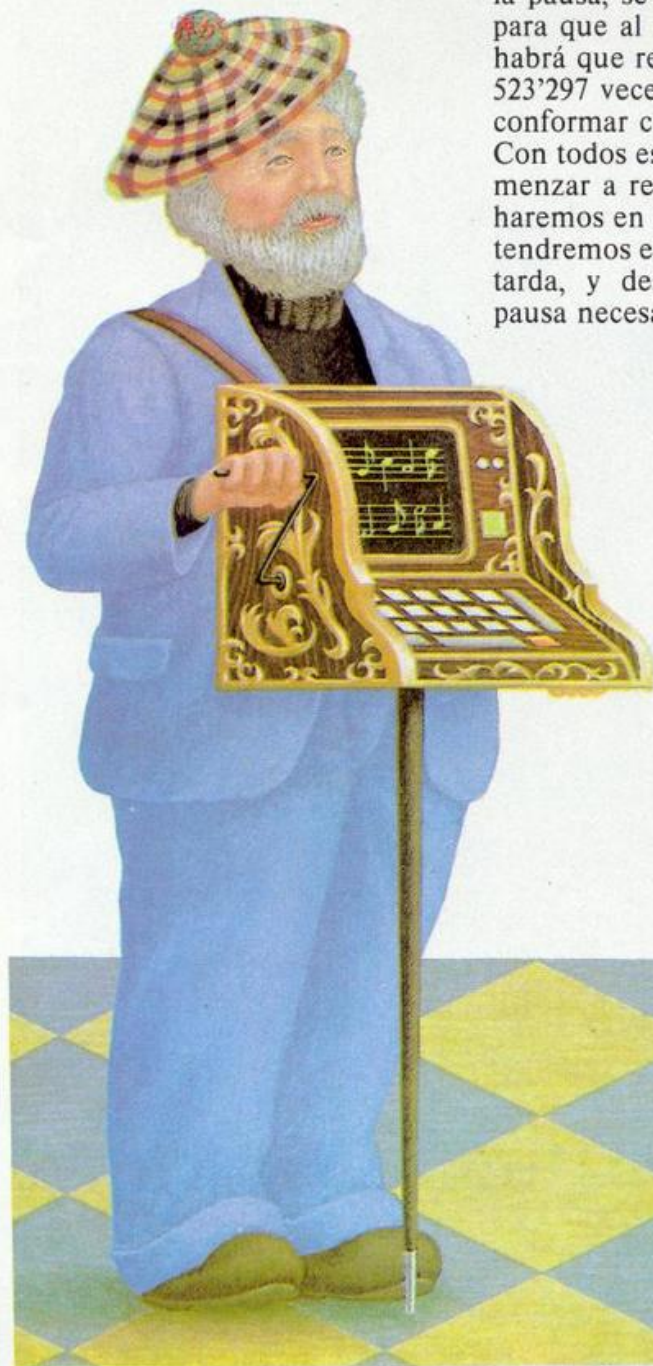
Por otra parte hemos dicho que queríamos que sonara durante un segundo. Si cada vez que se realice la pausa, se tardan 6689 T-estados para que al final dure un segundo habrá que repetirlo  $3500000/6689=523'297$  veces. Nos tendremos que conformar con repetirlo 523 veces. Con todos estos datos podemos comenzar a realizar el programa. Lo haremos en dos etapas. Primero no tendremos en cuenta el tiempo que tarda, y después colocaremos la pausa necesaria.

```
LDA,7;BORDE BLANCO
LD HL,523;REPETICIONES
LOOP.OUT (25 4),A
XOR 16;CAMBIA ESTADO
DEL ALTAVOZ
```

```
.....
.....
.....;INSTRUCCIONES
DE PAUSA
DEL HL
LD DDA
LD A,H
ORL
LDA,D
JR NZ,LOOP
RET
```

La instrucción JR NZ,LOOP la tomaremos como de 12 T-estados. Ahora sumemos todos los T-estados. Si lo hemos hecho bien dará 52. Esto quiere decir que de la pausa de 6689 T-estados ya hemos gastado 52, y ahora deberemos añadir en los puntos suspensivos algo que tarde en ejecutarse exactamente  $6689-52=6637$  T-estados.

```
LD BC,64513
INC BC
LOPA DJNZ LOPA
```





DEC C  
JRNZ,LOPA

Esto tarda en ejecutarse exactamente 6634 T-estados. Puede ser un buen ejercicio para el lector hacer los cálculos necesarios para comprobarlo.

Ahora el programa quedaría así:

DI  
LD A,7  
LD HL,523  
LOOP OUT (254),A  
XOR16;CAMBIA ESTADO  
DEL ALTA VOZ  
LD BC,64513  
INC BC  
LOPA,DJNZ LOPA  
DEC C  
JRNZ,LOPA  
DEC HL  
LD D,A  
LD ,H  
ORL  
LDA,D  
JRNZ,LOOP  
EI  
RET

**El altavoz interno y el color del borde son controlados por el acceso al port 254 mediante la instrucción OUT y el contenido del acumulador.**

Observe que es necesario inhabilitar las interrupciones, pues estas harían que no pudiéramos controlar exactamente el tiempo que tarda el programa en ejecutarse.

#### Dos notas

Hasta aquí hemos visto como se toca una nota, pero ahora surge la pregunta ¿cómo se puede simular que suenan dos notas a la vez?

En un principio se puede pensar en alternar varias veces las dos notas, tocando cada una durante muy poco tiempo. Con esto se consigue el efecto deseado, pero suena como si el ordenador estuviera «haciendo gárgaras». Tras haber hecho varias pruebas con distintos métodos, creemos estar en lo cierto al afirmar que sólo hay un método para conseguir dos notas simultáneas de una forma aceptable. En cierto modo es análogo al de la alternancia de notas, pero esta alternancia ha de hacerse a un nivel inferior, hay que mezclar las ondas de las dos notas. Es como si cogiéramos la gráfica de las ondas, la cortáramos en pequeños pedacitos y creáramos una nueva uniéndolos los fragmentos de las otras dos de forma alterna. Debemos distribuir el tiempo en dos mitades, una parte estará sonando una nota y la otra parte sonará la otra nota.

Por ejemplo, si en un momento determinado la onda de una de las notas está baja y la de la otra alta, se

## PROTEJA SU SPECTRUM PLUS CON ESTA PRACTICA FUNDA

### A UN PRECIO ESPECIAL

OFERTA LIMITADA  
Y EXCLUSIVA PARA  
NUESTROS LECTORES

AHORA  
PARA USTED  
975  
PTAS.



Aproveche la oportunidad de mantener como nuevo su Spectrum Plus con esta funda, y beneficiesse de un 30% de descuento sobre su precio normal.

¡APRESURESE! RECORTE Y ENVIE HOY MISMO ESTE CUPON A:  
PUBLINFORMATICA (Dto. FUNDAS), C/BRavo MURILLO, 377 5.º A 28020 MADRID

**CUPON DE PEDIDO**

Si, envíeme al precio de 975 Ptas. cada una, fundas para mi SPECTRUM PLUS.  
El importe lo abonaré: ☐ Con mi tarjeta de crédito ☐ American Express ☐  
Visa ☐ Interbank ☐  
Contra reembolso ☐ Adjunto cheque ☐  
Número de mi tarjeta \_\_\_\_\_  
Fecha de caducidad \_\_\_\_\_  
NOMBRE \_\_\_\_\_  
DIRECCION \_\_\_\_\_  
CIUDAD \_\_\_\_\_ C.P. \_\_\_\_\_  
PROVINCIA \_\_\_\_\_  
Sin gastos de envío



creará una «hibrida» que estará la mitad del tiempo baja y la mitad alta.

Para todo esto debemos cambiar un poco el esquema del programa de una sola nota.

En primer lugar, al ser dos notas cada una con su distinta duración de pausa entre OUTS, no se puede hacer esta pausa como se hizo en el programa anterior. Habrá que hacer un bucle que se ejecute continuamente pero en el que el XOR 16 no se ejecute nada más que cuando corresponda el cambio de estado a una determinada nota. Esto implica que debe haber bifurcaciones dentro del bucle, por lo que habremos de tener cuidado de que vaya por donde vaya la ejecución, siempre tarde el mismo tiempo en ejecutarse el bucle.

Además como tenemos dos notas, necesitaremos dos valores distintos para las pausas y otros dos para el estado de cada una de las ondas (alto o bajo). Para esto último utilizaremos los registros A y A'. Además del valor de las pausas necesitaremos un contador para cada pausa. Para estos cuatro valores (dos de pausa y dos de contador) utilizaremos los registros DE y HL.

Nos queda libre el registro BC para controlar la duración, que habrá de ser una misma para las dos notas. Una sola pasada alrededor del bucle es un tiempo demasiado corto, así que haremos que el número mínimo de pasadas sea 256. Meteremos entonces en C un número que corresponda a la duración, y por cada

## FIGURA 2. Cargador Basic

```
1 DATA "2A765CF3ED57F5E5AFDBF
EF6E03C20215E1C2004E1E55E1C1D23E
56E2600160001C7FF0966EB09661225"
2 DATA "0E26CD91FFE12318D7E1F
1ED47FBC90000ED738FFF06005A6C3A4
B5C1F1F1FE607F50BF1ED4708EB1345"
3 DATA "D3FE2D200EEE106C10F20
DC2A6FFED7B8FFFC9F910E60DC2A6FFE
D7B8FFFC9D4C8BDB3A99F968E6618EB"
4 DATA "7E77706A645F5954504B4
7433F3C3835322F2D2A28262321201E1
C1B1918161514131211100F0E0D07EB"
5 DATA "010001"
200 RESTORE : LET a=10: LET b=1
1: LET c=12: LET d=13: LET e=14:
LET f=15: LET ad=65368
210 PRINT AT 0,0:"LEYENDO LINEA
": FOR z=1 TO 5: PRINT AT 0,16:
z
220 READ a#: LET b#=a#(LEN a#-3
```

```
TO ): LET a#=#( TO LEN a#-4):
IF LEN a#<2<>INT (LEN a#<2) THEN
PRINT FLASH 1:AT 0,0:"LINEA I
MPAR EN": STOP
230 LET w=0: FOR x=1 TO LEN a#
STEP 2: LET v=VAL a#(x)*16+VAL a
#(x+1): LET w=w+v
240 POKE ad,v: LET ad=ad+1: NEX
T x
250 LET v=0: FOR x=1 TO 4: LET
v=v*16+VAL b#(x): NEXT x: IF v<>
w THEN PRINT FLASH 1:AT 0,0:"E
RROR EN LINEA ": STOP
260 NEXT z
270 CLS
280 PRINT "PON LA CINTA PARA GR
ABAR"
290 SAVE "COPYCODE"CODE 65368,1
61
```

## FIGURA 4. Cargador Basic con música de demostración

```
1 DATA "240C240C2431260C27312
70C2911291129312B112C312C112B132
B312B0F2931270C27312414240F053A"
2 DATA "240C240F2414241424112
40C2908290C2711270C2613260E24082
40E2213220E220C220C220A220003E3"
3 DATA "22032207240C240F24132
40C24142403240C240C2431260C27312
70C2911291129312B112C312C11048D"
4 DATA "2E072E132C082C112B072
B16270C270C2731290C2B312B0C2C112
C112C312911273127112613310E04F9"
5 DATA "2413310C2213220A24002
4002400240024002400240C240E24102
410240E240C2B0C240C2B10241003AF"
6 DATA "291324132809240928102
410290024002B0C240C2D0C240C2B102
4102B132413291324132B102410041D"
7 DATA "290A220A290A220A28072
207260822082808220829082208280C2
80E28102810280E280C2B13281303DD"
8 DATA "2B112810280E280C240C1
F0C0C0C1F0C260E1F0E28101F10260E1
F0E240C1F0C260E210E260E210E03C3"
9 DATA "241021102611211126112
11126111F11240820082408200824082
00826081D0826081D0826081D0E039F"
10 DATA "24001C0024001C0024001
C00240C240C240C240C240C240C240E2
40C240C240C240C240C240C240C240E2
200 CLEAR 39999: RESTORE : LET
```

```
a=10: LET b=11: LET c=12: LET d=
13: LET e=14: LET f=15: LET ad=4
0000
210 PRINT AT 0,0:"LEYENDO LINEA
": FOR z=1 TO 10: PRINT AT 0,16:
z
220 READ a#: LET b#=a#(LEN a#-3
TO ): LET a#=#( TO LEN a#-4):
IF LEN a#<2<>INT (LEN a#<2) THEN
PRINT FLASH 1:AT 0,0:"LINEA I
MPAR EN": STOP
230 LET w=0: FOR x=1 TO LEN a#
STEP 2: LET v=VAL a#(x)*16+VAL a
#(x+1): LET w=w+v
240 POKE ad,v: LET ad=ad+1: NEX
T x
250 LET v=0: FOR x=1 TO 4: LET
v=v*16+VAL b#(x): NEXT x: IF v<>
w THEN PRINT FLASH 1:AT 0,0:"E
RROR EN LINEA ": STOP
260 NEXT z
270 CLS
280 PRINT "PON LA CINTA PARA GR
ABAR"
290 SAVE "COPYCODE"CODE 4E4,397
300 CLS : PRINT "PARA ESCUCHAR
ESTA MUSICA CARGAREL PROGRAMA EN
CODIGO MAQUINA Y HACER:
RANDOM1
10:PRIN
ZE 4E4:PAUSE
T USR 65368"
```

**Al intentar producir sonidos es imprescindible dominar el concepto del T-estado, unidad de tiempo del microprocesador que depende de la velocidad del reloj.**

unidad de este repetiremos el bucle 256 veces ayudados por el registro B. Con todos estos datos ya podemos escribir una subrutina que toque dos notas.

Podrá ver en el listado ensamblador de esta subrutina con el nombre NOTO.

A esta subrutina se la llama te-

niendo en el registro C la duración de la nota, y en los registros H y D los valores de las pausas de las dos notas.

Observará que la subrutina tiene algunas instrucciones extrañas, como LD, I,A y LD SP,HL. Esas se utilizan para temporizar y conseguir que el bucle dure siempre exactamente el mismo tiempo. Naturalmente, no podemos cambiar alegremente el registro SP, pues podría tener efectos catastróficos, así que su valor original es restaurado al final de la rutina.

El tiempo que pasa entre dos ejecuciones de la instrucción OUT es de 63 T-estados, pero cada vez será para una nota distinta, así que el tiempo entre dos OUTS de la misma nota es de 126 T-estados.

Esto nos servirá para calcular que la pausa ha de ser de unos 6689 T-estados. Para calcular entonces el valor que habría que colocar en H o en D para esta nota sería  $6689/126=53$ . Así se pueden calcular los valores de pausa de todas las notas. Al final del listado, en instrucciones DEFB se encuentra una tabla con todos estos valores para cuatro octavas completas, siendo la central la segunda de ellas.

Dijimos que el bucle se iba a ejecutar un mínimo de 256 veces. Como cada vez tarda 63 T-estados, el tiempo mínimo será  $256*63=16128$  T-estados o 0'004608 segundos. Así que para producir dos notas durante X segundos hay que meter en el registro C el resultado de dividir X por 0'004608.



Con esto creo que quedará bastante claro como se producen dos notas simultáneas.

### Almacenamiento en memoria

El listado en ensamblador, utilizando la subrutina NOTO, toca una música que hayamos colocado en la memoria. Va leyendo la memoria de dos en dos posiciones, tomando dos valores que corresponden a las dos notas. Estos valores pueden variar de 0 a 49, siendo el 0 el DO inmediatamente inferior al central y numerándose el resto al igual que con la instrucción BASIC «BEEP». El 49 no es una nota, sino que es el indicativo del silencio.

Con estos dos valores se busca en la tabla las duraciones de las pausas, y luego se carga el registro C con la duración de las notas. Variando este valor podemos tocar una música más deprisa o más despacio.

El programa considera que la música ha terminado al encontrar un

***Para controlar con exactitud el tiempo que tarda en ejecutarse un programa en código máquina es imprescindible inhabilitar las interrupciones.***

255, y comienza a repetirla desde el principio. Puede pararse en cualquier momento pulsando una tecla.

Como la duración de las notas es la misma para todas, si queremos, por ejemplo que una dure el doble de lo normal bastará con ponerla dos veces. Esto quiere decir que si ponemos seguidas dos notas iguales no se oirán dos notas separadas sino una nota de duración doble. Si queremos que suenen dos notas separadas habrá que intercalar un pequeño silencio.

Para los que no tengan ensamblador se publica también un programa

BASIC que carga el código en la memoria.

### Componiendo música

Para componer cómodamente para este programa, se incluye un sencillo programa BASIC que nos será fácil para introducir la música. Para su utilización habrá que hacer lo siguiente:

- Teclear el programa de la figura 3.
- Grabarlo en una cinta con LINE 10.
- Grabar a continuación el código obtenido con el listado ensamblador o el BASIC de la figura 2.
- Borrar la memoria y cargarlo con LOAD""

El programa nos presenta las siguientes opciones:

- Componer. Iremos introduciendo las notas, primero de un canal y luego de otro. Mientras no especifiquemos nada, la nota será de la octava central y su duración será 1



## SUSCRIBASE POR TELEFONO

- \* más fácil,
- \* más cómodo,
- \* más rápido

**Telf. (91) 733 79 69**

**7 días por semana, 24 horas a su servicio**

SUSCRIBASE A

**Todospectrum**



(1 es la duración de la nota más pequeña que utilicemos en la composición. Su duración real dependerá de la velocidad que le demos a la ejecución. (Por ejemplo, si a la corchea la llamamos 1, la negra 2, la blanca 3, etc., pero no podremos utilizar semicorcheas ni fusas ni semifusas, porque su duración sería 1,5 y sólo se pueden dar valores enteros. Por eso debemos tener cuidado de cuál es la duración a la que llamamos 2). Si queremos cambiar a otra octava, añadiremos antes de la nota y sin dejar espacio entremedias el valor de la octava de 0 a 3 (1=octava central).

**Sólo hay un método adecuado para conseguir dos notas simultáneas: mezclar sus ondas de forma alternada, dividiendo el tiempo que corresponde a cada una de ellas.**

vés de ellas con las teclas seis y siete. Cuando estemos sobre una determinada nota, tendremos las siguientes

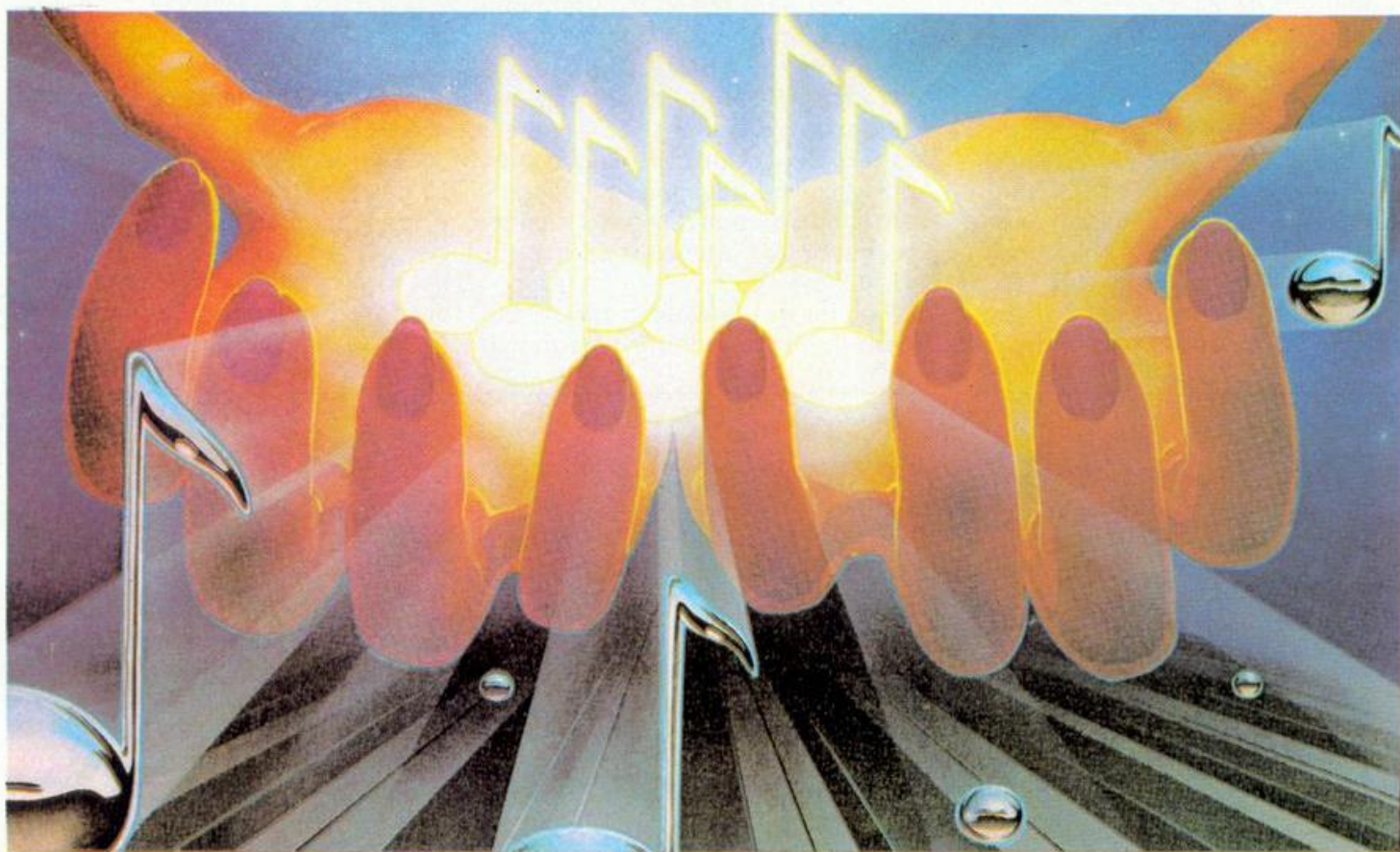
en una matriz numérica, ininteligible para el programa en código máquina. Esta opción vuelca en la memoria la composición dejándola lista para ser interpretada.

— Tocar: Interpreta la música tras habernos pedido la velocidad de ejecución.

— Grabar la música: Nos permite grabar una música a medio componer para continuar otro día.

— Cargar la música: Cargar lo grabado con la opción anterior.

— Grabar el código: Graba los datos de la memoria que utiliza el programa en código máquina. Para uti-



Si queremos cambiar las duraciones añadiremos un número del 1 al 9 detrás de la nota. Estos nuevos valores permanecerán vigentes hasta que los volvamos a cambiar. Para los silencios introducir «S».

Cuando hayamos terminado con un canal pulsaremos ENTER.

Al terminar los dos canales regresaremos al menú, a no ser que hayamos cometido algún error (la duración de un canal es distinta de la del otro) en cuyo caso pasaremos directamente a la opción de corregir.

— Corregir: Visualiza las 20 primeras notas del canal que hayamos elegido. Podemos movernos a tra-

tes posibilidades:

— 5 y 8: Varían la duración de la nota señalada por el cursor.

— 4 y 9: Varían la nota. Para obtener el silencio se pulsará el 9 hasta que aparezca la palabra «SILENCIO»

— Q: Elimina la nota señalada por el cursor.

— A: Duplica la nota señalada por el cursor.

En cualquier momento podemos pasar a las siguientes 20 notas pulsando ENTER.

— Continuar la composición.

— Volcar la composición: El programa BASIC almacena la música

lizar esto en nuestros propios programas habremos de hacer los siguientes:

— Cargar el programa en código máquina.

— Cargar los bytes grabados por esta opción en la dirección que queramos.

— Para tocarla hacer: RANDOMIZED:POKE65409,V:RANDOMIZEUSR 65368, siendo d la dirección de la música y v la velocidad de ejecución. Hay que tener cuidado no utilizar gráficos definidos por el usuario, pues estropearíamos la rutina.

PABLO ARIZA





# GRAFICOS INTERACTIVOS EN TRES DIMENSIONES







# GRAFICOS INTERACTIVOS EN

# TRES DIMENSIONES

**¿N**o se ha preguntado nunca cómo se producen esas imágenes que a menudo vemos en televisión donde las figuras rotan, vienen y van? Como, por ejemplo, en las imágenes de apertura diaria de televisión donde cuatro bloques forman un recinto en el que se asienta una esfera formando el logotipo de TVE. Seguro que siempre pensó que se hacían por ordenador pero nunca llegó a imaginar cómo podían seguir una trayectoria tan perfecta y un movimiento tan realista. En este artículo comentaremos la base de las técnicas que permiten producir tales efectos.

El programa que aquí presentamos es capaz de realizar gráficos interactivos en tres dimensiones, verlos en perspectiva desde diferentes puntos de visión y cambiarlos de tamaño. A diferencia de otros programas que permiten la construcción de figuras interactivamente, este es capaz de dibujar la figura en perspectiva, reflejando la realidad, ya que traza la figura a través de las coordenadas que se le proporcionan. Es decir, usted podría dibujar su coche y verlo en perspectiva desde el punto que desee simplemente tomando medidas de su automóvil y proporcionando las coordenadas al programa.

Existen una serie de conceptos

básicos que hemos de manejar antes de empezar a describir el programa y su funcionamiento. Estamos trabajando en el espacio tridimensional, que es el que existe en la realidad, y los objetos de este espacio queremos dibujarlos en el espacio bidimensional, que es el que existe en la televisión o en una hoja de papel. La idea es dibujar un objeto real, en tres dimensiones, en sólo dos dimensiones y que esta representa-

***Para evitar que el punto de visión esté dentro del objeto, y para ajustar la figura a la pantalla, necesitamos el volumen de dibujo.***

ción refleje la realidad. Para ello hemos de introducir el concepto de perspectiva. Para trabajar con la perspectiva hemos de saber desde qué punto estamos observando el objeto —no es lo mismo mirar a un coche de frente que desde abajo— y esto introduce el concepto de punto de visión.

Para tener una perspectiva adecuada desde el punto de visión del objeto que estamos dibujando es necesario que el punto esté fuera del objeto. Nosotros podemos ver nuestra casa en perspectiva desde

fuera de ella y no desde dentro. Para evitar que el punto de visión esté dentro del objeto y para ajustar la figura u objeto que dibujamos a la pantalla necesitamos lo que llamamos el volumen de dibujo. Se trata de una especie de caja en donde especificamos que vamos a dibujar. La definición de punto de visión y volumen de dibujo aparecen en la pantalla cuando se va a hacer uso de ellos.

Como nos estamos moviendo en el espacio tridimensional trabajaremos con tres ejes que llamaremos X, Y y Z. Colocándonos de frente a una pared de una habitación el eje X sería aquél que va desde la esquina inferior izquierda, donde está el origen, a la esquina inferior derecha. El eje Y sería la altura de la habitación, desde nuestra esquina origen al techo. Y el eje Z es aquel que vendría desde el origen hacia nosotros.

En las líneas 160 a 290 se definen las ventanas que se van a utilizar. Utilizaremos la ventana 0 para introducir datos y responder a preguntas; la ventana 1 la dedicaremos a dar instrucciones e información; y la ventana 2, la más grande, se destinará para dibujar.

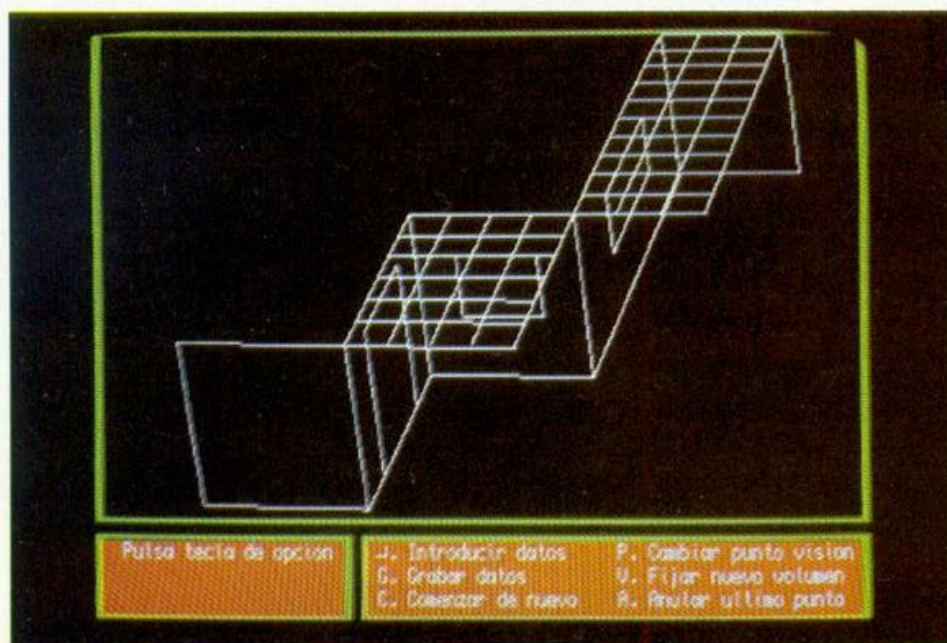
En la línea 300 se inicializan algunas variables que se utilizarán en una comparación inicial: n, el número de filas que tiene la matriz de puntos «original», y xv, yv y zv que definen el volumen de dibujo pues-





```
100 REMark =====
110 REMark  GRAFICOS INTERACTIVOS
120 REMark  EN TRES DIMENSIONES
130 REMark  RICARDO GARCIA Y GARCIA
140 REMark  =====
150 DIM tr(3,2),punto(2),vol3d(7,2),vol2d(7,1),original(500,3)
160 REMark Definicion de ventanas
170 MODE 4
180 WINDOW#0,155,36,26,220
190 PAPER#0,2
200 BORDER#0,2,4
210 CLS#0
220 WINDOW#1,302,36,186,220
230 PAPER#1,2
240 BORDER#1,2,4
250 WINDOW#2,462,210,26,5
260 PAPER#2,0
270 BORDER#2,2,4
280 CLS#2
290 CLS#1
300 n=0:xv=0:yv=0:zv=0
310 REMark Menu inicial
320 PRINT#1,"  PULSA LA TECLA APROPIADA PARA CADA OPCION"
330 PRINT#1,"      F. Lectura de datos desde fichero"
340 PRINT#1,"      T. Introduccion de datos por teclado"
350 slc$=INKEY$(#0,-1)
360 slc=CODE(slc$)
370 SElect ON slc
380 REMark Lectura de datos desde fichero (F)
390 ON slc=70,102
400 CLS#0
410 INPUT#0,"  Nombre del fichero:                ";nombre$
420 archivo$="mdv1_"&nombre$&".3d"
430 OPEN_IN#3,archivo$
440 INPUT#3,n
450 INPUT#3,a
460 INPUT#3,b
470 INPUT#3,c
480 INPUT#3,xv
490 INPUT#3,yv
500 INPUT#3,zv
510 FOR i=0 TO n-1
520 FOR j=0 TO 3
530 INPUT#3,original(i,j)
540 END FOR j
550 END FOR i
560 CLOSE#3
570 matriz_transformacion
580 ajustes_pantalla
590 redibuja
600 REMark Introduccion de datos por teclado (T)
610 ON slc=84,116
620 punto_vision
630 matriz_transformacion
640 volumen
650 ajustes_pantalla
660 REMark Ninguna de las anteriores
670 ON slc=REMAINDER
680 PRINT#0,"  Opcion no valida"
690 GO TO 350
700 END SElect
710 menu_principal
720 CLS#0
730 PRINT#0,"  Pulsa tecla de opcion"
740 slc$=INKEY$(#0,-1)
```





to que estas coordenadas son las de un punto oblicuamente opuesto al origen.

Una figura puede dibujarse a través de los datos almacenados en un fichero o introduciendo datos interactivamente. La opción se realiza en las líneas 310 a 350. Si se ha elegido leer los datos desde un fichero, se cargan en las líneas 380 a 560. Observe que los ficheros tienen la extensión 3d. a, b y c son las coordenadas que definen el punto de visión. Una vez cargados los datos del fichero se ejecutan tres procedimientos: matriz-transformación, ajustes-pantalla y redibuja.

El procedimiento matriz-transformación (líneas 5000 a 5200) realiza el grueso de cálculo del programa. Explicar aquí el porqué de este algoritmo nos ocuparía más que la propia revista, con lo que obviaremos la descripción detallada del mismo. En síntesis lo que hace este algoritmo es construir una matriz de transformación (trt), a la que se llega a través de operaciones trigonométricas y cálculo de rotaciones y traslaciones, que define la situación relativa entre el punto de visión y la figura.

El procedimiento ajustes-pantalla

(líneas 6000 a 6370) tiene como misión última el cálculo de un factor de escala (escala) y dos factores de traslación (muevex y muevey) que se encarga de ajustar a las coordenadas de la pantalla los puntos en el espacio bidimensional, dados en el vector «punto». Para hacer esto se carga la matriz de puntos que delimita el volumen (vol3d) y se pasa al espacio bidimensional (almacenán-

**Una figura puede dibujarse a través de los datos almacenados en un fichero o introduciendo directamente datos desde el teclado.**

dose en vol2d) a través de la transformación que realiza el procedimiento transforma. Este procedimiento (líneas 7000 a 7070) aplica la matriz tr a cada punto y después realiza la transformación de la perspectiva con una simple regla de tres. Una vez conocido el volumen transformado a dos dimensiones (almacenado en vol2d) no queda sino ejecutar algunas ecuaciones de primer grado para calcular los factores de

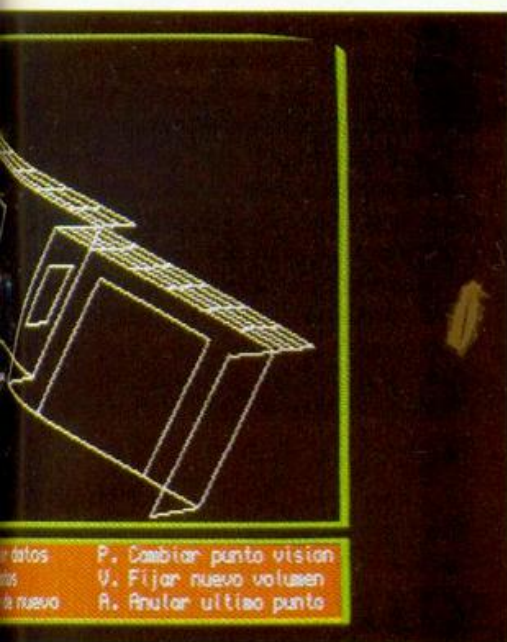
escala y traslación que harán que cada punto que se transforme de la matriz «original» puedan ser representados en la pantalla.

Una vez hecho el trabajo difícil, al procedimiento redibuja (líneas 9000 a 9110) no le queda sino dibujar las figuras con los datos de la matriz «original». La primera columna de esta matriz está compuesta de unos y ceros. Un uno significa dibujar y un cero mover. Las otras tres columnas definen el punto en el espacio tridimensional. El procedimiento redibuja transforma cada punto a dibujar a través del procedimiento transforma y después dibuja cada punto o se mueve a un punto, dependiendo del valor de la primera columna, aplicando los factores de escala y traslación para el ajuste en pantalla.

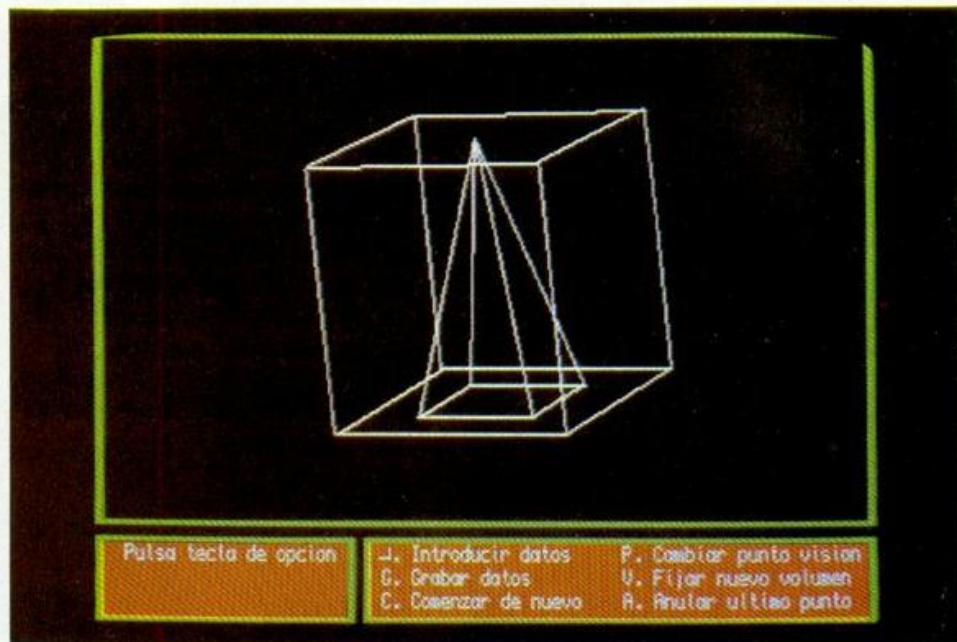
Una vez que tenemos nuestra figura en la pantalla podemos simplemente disfrutar de ella o realizar cualquiera de las operaciones que se nos ofrecen en el menú principal (líneas 4000 a 4080). Pero antes de examinar estas opciones veamos qué pasaría si al principio decidimos introducir datos desde el teclado en lugar de leerlos de un fichero.

En ese caso llegaríamos a la línea





datos  
P. Cambiar punto vision  
U. Fijar nuevo volumen  
A. Anular ultimo punto



Pulsa tecla de opción

J. Introducir datos  
G. Grabar datos  
C. Comenzar de nuevo

P. Cambiar punto vision  
U. Fijar nuevo volumen  
A. Anular ultimo punto

620 y se ejecutaría el procedimiento punto-visión (líneas 2000 a 2150). Este procedimiento nos define lo que es el punto de visión y pide que lo introduzcamos. Cuando lo hacemos se comprueba que no está dentro del volumen de dibujo y, si así es, da un aviso y pide que introduzcamos las coordenadas de nuevo. Si hubiéramos elegido la opción de introducir datos por teclado la primera vez ahora no tendríamos las coordenadas del volumen de dibujo para poder comparar, de aquí que se inicialicen en la línea 300.

Después de tener el punto de visión se ejecutaría el procedimiento matriz-transformación para ejecutarse después el procedimiento volumen (líneas 3000 a 3150). En este procedimiento se define el volumen o caja de dibujo, se piden las coordenadas y se comprueba que no contiene el punto de visión. Una vez que el programa conoce el volumen de dibujo ejecuta el procedimiento ajustes-pantalla.

Llegamos entonces a la línea 710 donde se ejecuta el procedimiento menú principal (líneas 4000 a 4070) donde lo único que se hace es mostrar las distintas opciones en la ventana informativa. Si pulsamos la te-

cla Intro accedemos a la parte de introducción de datos (líneas 790 a 880). Nada más introducir las coordenadas del punto se transforman (con el procedimiento transforma) y se dibujan con el procedimiento dibuja (líneas 8000 a 8090). Es uno de los procedimientos más sencillos. Aplicando los factores de escala y traslación para los ajustes de la pantalla no hace sino dibujar o mover

**El punto de visión y el volumen de dibujo se pueden alterar para obtener diferentes vistas del objeto y modificar su tamaño.**

de acuerdo al primer número del vector.

Se puede cambiar el punto de visión, para tener diferentes visiones de la figura, y el volumen de dibujo para ver la figura más o menos grande o extenderla. Para ambas operaciones la mecánica es similar: se introducen los nuevos datos, se hacen las transformaciones, se calculan los ajustes y se redibuja (líneas 890 a 1000).

Como lo que estamos haciendo es dibujar en tres dimensiones interactivamente existe la posibilidad de que nos equivoquemos y por eso existe la opción de anular el último punto (líneas 1020 a 1090). Para ello lo único que se hace es reducir  $n$  en uno y redibujar si se ha trazado una línea al último punto.

Las últimas opciones son las de grabar los datos (líneas 1100 a 1290) y de volver a comenzar. Observe que cuando decidimos grabar estamos grabando los datos (esto es, los puntos en el espacio tridimensional contenidos en la matriz «original» así como el punto de visión y el volumen de dibujo) y no la figura.

Por último, sólo una puntualización. El programa ajusta el volumen de dibujo a la pantalla. Da igual lo lejos que esté el punto de visión de la figura, seguirá siendo igual de grande. El punto de visión sirve como posición relativa para ver diferentes perspectivas del objeto pero no cambiará su tamaño en la pantalla. Para hacerlo habrá que cambiar el volumen de dibujo. Un volumen de dibujo más pequeño hará que la figura aparezca más grande en la pantalla y viceversa.

Ricardo García y García





```
750 slc=CODE(slc$)
760 Select ON slc
770 REMark Introducir datos (tecla intro)
780 ON slc=10
790 CLS#0
800 IF n=0 THEN original(n,0)=0:GO TO 840
810 PRINT#0," Mover (O)"
820 INPUT#0," o dibujar (1): ";original(n,0)
830 IF original(n,0)<>0 AND original(n,0)<>1 THEN CLS#0:GO TO 810
840 INPUT#0," coordenada x = ";original(n,1)
850 INPUT#0," coordenada y = ";original(n,2)
860 INPUT#0," coordenada z = ";original(n,3)
870 transforma original(n,1),original(n,2),original(n,3)
880 dibuja
890 REMark Cambiar punto de vision (P)
900 ON slc=112,80
910 punto_vision
920 matriz_transformacion
930 ajustes_pantalla
940 redibuja
950 menu_principal
960 REMark Fijar nuevo volumen (V)
970 ON slc=118,86
980 volumen
990 ajustes_pantalla
1000 redibuja
1010 menu_principal
1020 REMark Anular ultimo punto (A)
1030 ON slc=65,97
1040 IF original(n-1,0)=0 THEN
1050 n=n-1
1060 ELSE
1070 n=n-1
1080 redibuja
1090 END IF
1100 REMark Grabar datos (G)
1110 ON slc=71,103
1120 CLS#0
1130 INPUT#0," Nombre del archivo: ";nombre$
1140 archivo$="mdv1_"&nombre$&"_3d"
1150 DELETE archivo$
1160 OPEN_NEW#3,archivo$
1170 PRINT#3,n
1180 PRINT#3,a
1190 PRINT#3,b
1200 PRINT#3,c
1210 PRINT#3,xv
1220 PRINT#3,yv
1230 PRINT#3,zv
1240 FOR i=0 TO n-1
1250 FOR j=0 TO 3
1260 PRINT#3,original(i,j)
1270 END FOR j
1280 END FOR i
1290 CLOSE#3
1300 REMark Comenzar de nuevo (C)
1310 ON slc=67,99
1320 GO TO 280
1330 REMark Ninguna de las anteriores
1340 ON slc=REMAINDER
1350 CLS#0
1360 PRINT#0," Opcion no valida"
1370 GO TO 730
1380 END Select
1390 GO TO 720
2000 REMark ==== Toma de datos del punto de vision =====
```





```
2010 DEFine PROCedure punto_vision
2020 CLS#1
2030 PRINT#1,"          DEFINICION DE PUNTO DE VISION"
2040 PRINT#1," Aquel punto en el espacio tridimensional desde"
2050 PRINT#1,"          el que se va a mirar la figura dibujada"
2060 CLS#0
2070 INPUT#0," coordenada x = ";a
2080 INPUT#0," coordenada y = ";b
2090 INPUT#0," coordenada z = ";c
2100 IF a>=0 AND a<=xv AND b>=0 AND b<=yv AND c>=0 AND c<=zv THEN
2110 CLS#0
2120 PRINT#0," ERROR: Punto de vision dentro de caja de dibujo"
2130 GO TO 2070
2140 END IF
2150 END DEFine
3000 REMark ==== Toma de datos para delimitar el volumen de dibujo ====
3010 DEFine PROCedure volumen
3020 CLS#1
3030 PRINT#1," DEFINICION DE VOLUMEN DE DIBUJO: Punto en el "
3040 PRINT#1," espacio tridimensional, oblicuamente opuesto al"
3050 PRINT#1," origen, que define una caja en la que dibujar"
3060 CLS#0
3070 INPUT#0," coordenada x = ";xv
3080 INPUT#0," coordenada y = ";yv
3090 INPUT#0," coordenada z = ";zv
3100 IF a>=0 AND a<=xv AND b>=0 AND b<=yv AND c>=0 AND c<=zv THEN
3110 CLS#0
3120 PRINT#0," ERROR: Punto de vision dentro de caja de dibujo"
3130 GO TO 3070
3140 END IF
3150 END DEFine
4000 REMark ==== Definicion del menu principal ====
4010 DEFine PROCedure menu_principal
4020 CLS#1
4030 PRINT#1," . Introducir datos          P. Cambiar punto vision"
4040 PRINT#1," G. Grabar datos              V. Fijar nuevo volumen"
4050 PRINT#1," C. Comenzar de nuevo        A. Anular ultimo punto"
4060 LINE#1,27,95 TO 27,80 TO 13,80 TO 15,83
4070 LINE#1,13,80 TO 15,77
4080 END DEFine
5000 REMark ==== Calculo de la matriz de transformacion ====
5010 DEFine PROCedure matriz_transformacion
5020 LOCAL cg,sg,cb,sb,vuelca
5030 cg=SQRT(c^2+b^2)/SQRT(a^2+b^2+c^2)
5040 sg=a/SQRT(a^2+b^2+c^2)
5050 cb=c/SQRT(c^2+b^2)
5060 sb=-b/SQRT(c^2+b^2)
5070 vuelca=(-(c<0))+(c>=0)
5080 tr(0,0)=cg*vuelca
5090 tr(0,1)=0
5100 tr(0,2)=-sg
5110 tr(1,0)=sb*sg*vuelca
5120 tr(1,1)=cb*vuelca
5130 tr(1,2)=sb*cg
5140 tr(2,0)=-cb*sg*vuelca
5150 tr(2,1)=sb*vuelca
5160 tr(2,2)=-cb*cg
5170 tr(3,0)=((-a)*cg-sg*(b*sb-c*cb))*vuelca
5180 tr(3,1)=-(b*cb+c*sb)*vuelca
5190 tr(3,2)=a*sg-cg*(b*sb-c*cb)
5200 END DEFine
6000 REMark ==== Parametros para ajustar el dibujo a la pantalla ====
6010 DEFine PROCedure ajustes_pantalla
6020 REMark carga la matriz de puntos que delimita el volumen
6030 RESTORE
6040 FOR i=0 TO 7
```



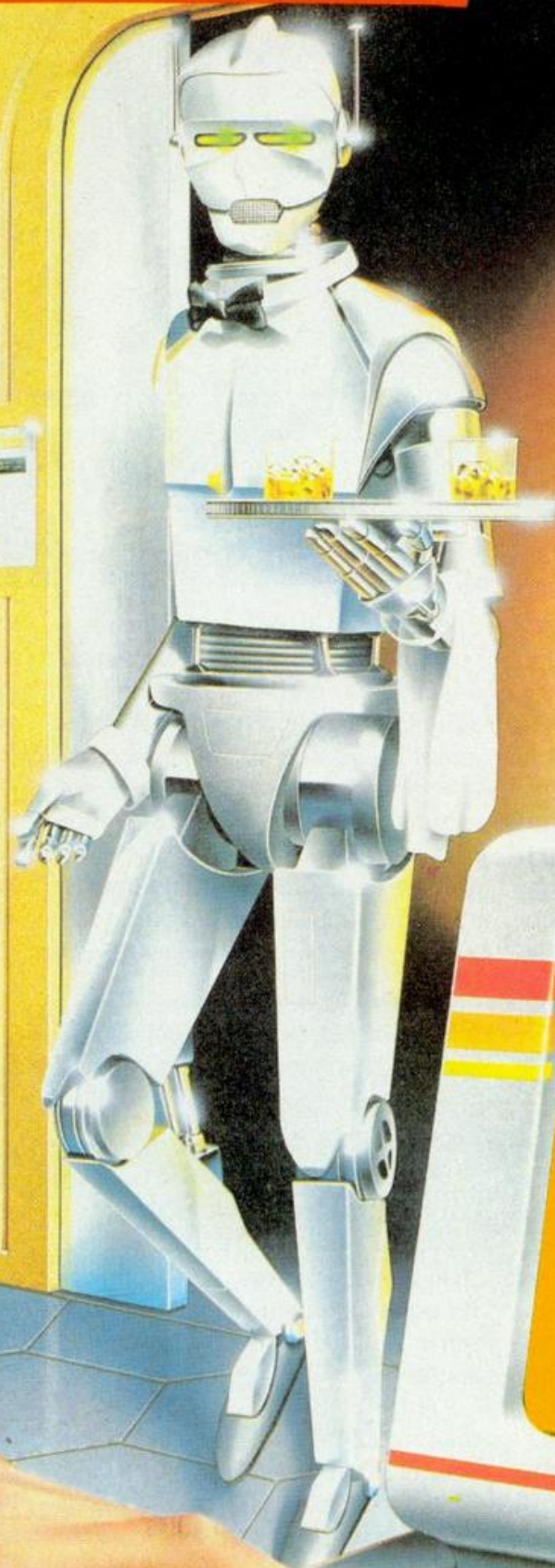


```
6050 FOR j=0 TO 2
6060 READ vol3d(i,j)
6070 NEXT j
6080 NEXT i
6090 DATA 0,0,0,xv,0,0,0,yv,0,xv,yv,0,0,0,zv,xv,0,zv,0,yv,zv,xv,yv,zv
6100 REMark transformacion y proyeccion de los puntos del volumen
6110 FOR i=0 TO 7
6120 transforma vol3d(i,0),vol3d(i,1),vol3d(i,2)
6130 vol2d(i,0)=punto(0)
6140 vol2d(i,1)=punto(1)
6150 NEXT i
6160 REMark maximo y minimo del volumen proyectado
6170 maxx=vol2d(0,0)
6180 minx=vol2d(0,0)
6190 maxy=vol2d(0,1)
6200 miny=vol2d(0,1)
6210 FOR i=1 TO 7
6220 IF vol2d(i,0)>maxx THEN maxx=vol2d(i,0)
6230 IF vol2d(i,0)<minx THEN minx=vol2d(i,0)
6240 IF vol2d(i,1)>maxy THEN maxy=vol2d(i,1)
6250 IF vol2d(i,1)<miny THEN miny=vol2d(i,1)
6260 NEXT i
6270 REMark determinacion del factor de escala y de traslacion
6280 IF 164/100<=(maxx-minx)/(maxy-miny) THEN
6290 escala=164/(maxx-minx)
6300 muevex=-minx*escala
6310 muevey=(100-(maxy-miny)*escala)/2-miny*escala
6320 ELSE
6330 escala=100/(maxy-miny)
6340 muevex=(164-(maxx-minx)*escala)/2-minx*escala
6350 muevey=-miny*escala
6360 END IF
6370 END DEFine
7000 REMark ==== Realiza la transformacion y proyeccion ====
7010 DEFine PROCedure transforma (x,y,z)
7020 punto(0)=x*tr(0,0)+y*tr(1,0)+z*tr(2,0)+tr(3,0)
7030 punto(1)=x*tr(0,1)+y*tr(1,1)+z*tr(2,1)+tr(3,1)
7040 punto(2)=x*tr(0,2)+y*tr(1,2)+z*tr(2,2)+tr(3,2)
7050 punto(0)=punto(0)/punto(2)
7060 punto(1)=punto(1)/punto(2)
7070 END DEFine
8000 REMark ==== Traza lineas segun valores de matriz original ====
8010 DEFine PROCedure dibuja
8020 IF original(n,0)=0 THEN
8030 LINE#2,punto(0)*escala+muevex,punto(1)*escala+muevey
8040 n=n+1
8050 ELSE
8060 LINE#2 TO punto(0)*escala+muevex,punto(1)*escala+muevey
8070 n=n+1
8080 END IF
8090 END DEFine
9000 REMark ==== Redibuja la matriz original completa ====
9010 DEFine PROCedure redibuja
9020 CLS#2
9030 FOR i=0 TO n-1
9040 transforma original(i,1),original(i,2),original(i,3)
9050 IF original(i,0)=0 THEN
9060 LINE#2,punto(0)*escala+muevex,punto(1)*escala+muevey
9070 ELSE
9080 LINE#2 TO punto(0)*escala+muevex,punto(1)*escala+muevey
9090 END IF
9100 NEXT i
9110 END DEFine
```



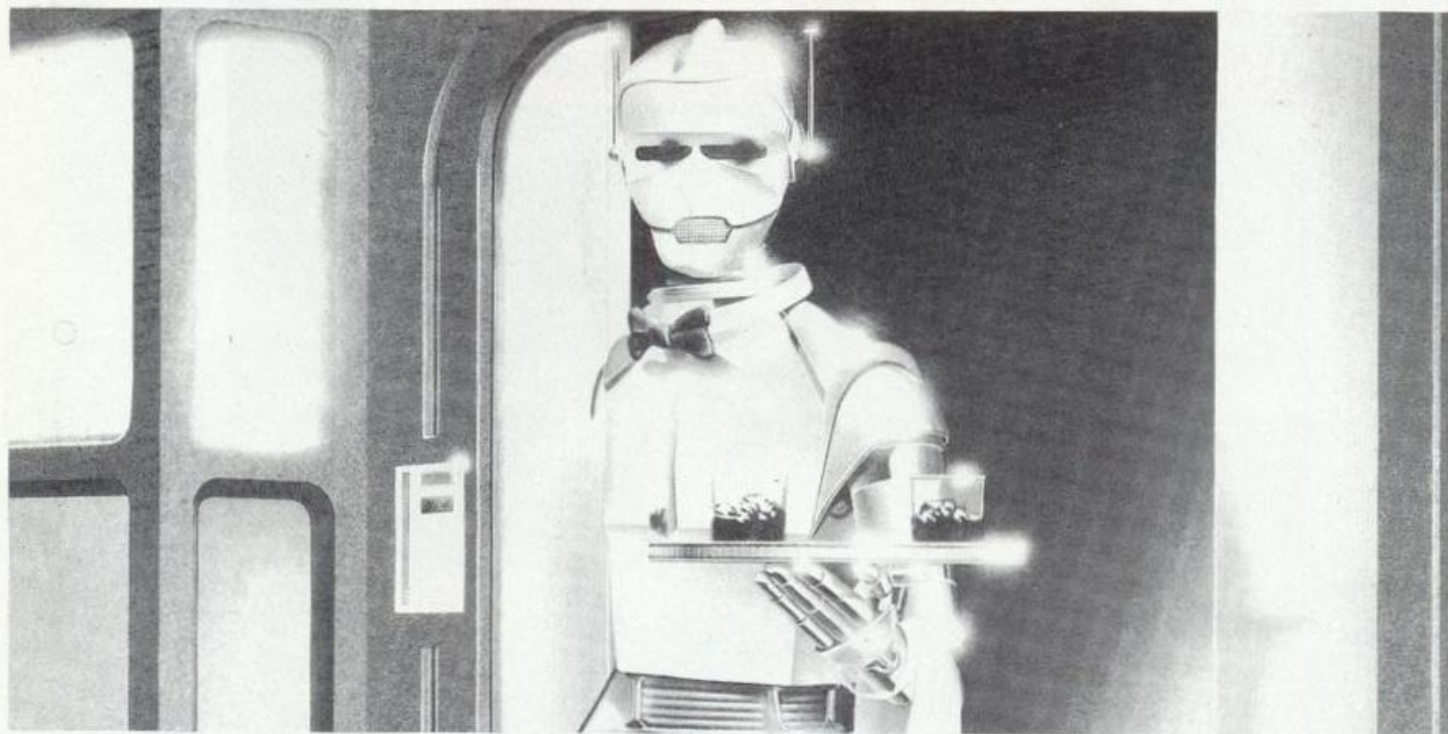
# APRENDIENDO

## CÓDIGO MÁQUINA



Continuamos escudriñando en la memoria del Spectrum para ver cómo marchan las cosas cuando el operativo está funcionando. Descubrimos una zona denominada «Información para canales» que desde el código máquina puede resultarnos muy útil para ciertos propósitos.





Si observamos el mapa de memoria que ofrecimos en el pasado capítulo veremos que, tras las variables del sistema y a partir de la dirección 23734, existe una zona denominada «Mapas de Microdrive». Es una zona que sólo aparecerá si estamos usando un microdrive con el Interface 1, por lo que la dejaremos de lado para meternos a fondo en el estudio de estos periféricos en otra ocasión.

Si no tenemos conectado el Interface 1 es la siguiente zona, «Información para canales», la que ocupa estos bytes. Esta zona, poco conocida por lo general, puede resultarnos de gran utilidad a la hora de abrir canales propios para el control de cualquier cosa que «pinchemos» al Spectrum o para hacer un uso de los canales existentes distinto del habitual, aunque quizá sea necesario explicar antes lo que entendemos por canal.

## Canales y corrientes

El tratamiento de los dispositivos de E/S (incluida la pantalla) se consigue en el sistema operativo del

Spectrum de una forma potente y versátil, gracias al uso de los llamados canales y corrientes. Un canal es aquella parte del sistema del ordenador a la que se puede enviar o recibir datos, mientras que corriente (*stream*) es la vía que usamos para transportar esos datos. En realidad, en castellano se utiliza muy a menudo la palabra canal para refe-

***Un canal es una parte del sistema a la cual se puede enviar datos, mientras que una corriente es la vía utilizada para transportar esos datos.***

rirse a ambas cosas, de forma que los puristas tomen un lápiz y prepárense para corregir cada vez que les apetezca.

Desde el BASIC hay varios comandos en los que podemos especificar el canal a utilizar, con PRINT, LIST y CAT para salidas, e INKEY\$ e INPUT para entradas (en verdad este último también puede actuar como salida). De esta forma si hace-

mos PRINT # 1 imprimiremos en la parte inferior de la pantalla, con PRINT # 2 lo haremos en la parte principal, mientras que con PRINT # 3 el texto irá a impresora (si estuviera conectada la ZX-Printer). Si tuviéramos conectado algún microdrive podríamos abrir un canal, por ejemplo el 4 para imprimir con PRINT # 4 directamente en un fichero del cartucho, mientras que si somos lo suficientemente hábiles, podremos abrir un canal propio para, por ejemplo, poder imprimir dentro de una determinada variable alfanumérica, como veremos después.

Cuando se inicializa el Spectrum, cuatro canales quedan especificados en la zona de información para canales. El canal «K» se usa, como salida, para escribir en la parte inferior de la pantalla, y, como entrada, para explorar el teclado. El canal «S» se usa como salida para escribir en la parte principal de la pantalla, mientras que dará error si intentamos usarlo como entrada. El canal «P» se usa para imprimir con la ZX-Printer (o compatible), no se usa como entrada. El canal «R» no puede ser



usado desde el BASIC, pero desde código máquina podemos utilizarlo para escribir en el área de trabajo (en el mapa, «Entrada de datos»). Las corrientes que se asignan a estos canales son: 0 y 1 para el canal «K», 2 para el «S», 3 para el «P», y, desde C/M, -3 (FDh) para el «K», -2 (FEh) para el «S» y -1 (FFh) para el «R».

## Formatos en memoria

Para cada canal, podemos encontrar 5 bytes en la zona de informa-

ción para canales que marcarán el uso que vaya a hacerse de ellos. Los dos primeros bytes de cada uno forman la dirección donde se encuentra la rutina que debe ser usada para las operaciones de salida. Los dos bytes siguientes son la dirección de la rutina de entrada, mientras que el quinto byte es el código de la letra usada como identificador de canal. Si un canal no admite ser usado, por ejemplo, como salida, deberá apuntar en los dos bytes correspondientes a la salida hacia una rutina de error específica, o bien a una simple su-

rutina que podría constar solamente de un RST 8 seguido del código de error que interese.

Lo explicado sólo es totalmente válido cuando no tenemos en Interface 1 conectado, pues este utiliza nuevos canales para manejar los microdrives que no responden al mismo formato.

Las rutinas de salida deben admitir, al ser llamadas, que sea en el acumulador donde se les pase el código al que deben dar salida. Lo hay que hacer con ese código depende de la imaginación del programador

10	ORG	60000	390	RST	8
20			400	DEFB	1
30	OP_t4		410	LOC_A1	
40	LD	HL, (PROG)	420	CP	"Z"
50	DEC	HL	430	JR	Z, A_Z\$
60	PUSH	HL	440	CALL	NEXT_O
70	LD	BC, 5	450	EX	DE, HL
80	CALL	MAKE_R	460	JR	LOC_Z\$
90	POP	DE	470		
100	LD	HL, TBLCHN	480	A_Z\$	
110	LD	BC, 5	490	INC	HL
120	LDIR		500	LD	E, (HL)
130	LD	HL, STRMS+14	510	INC	HL
140	LD	(HL), 21	520	PUSH	HL
150	RET		530	LD	D, (HL)
160			540	INC	DE
170	TBLCHN		550	PUSH	DE
180	DEFW	SALIDA	560	ADD	HL, DE
190	DEFW	ERROR	570	CALL	ONE_SP
200	DEFB	"Z"	580	PUSH	HL
210			590	POP	BC
220	;	"Invalid I/O device"	600	POP	DE
230			610	POP	HL
240	ERROR		620	FINVRS	
250	RST	8	630	LD	(HL), D
260	DEFB	18	640	DEC	HL
270			650	LD	(HL), E
280	SALIDA		660	EX	AF, AF'
290	OR	A	670	INC	BC
300	EX	AF, AF'	680	LD	(BC), A
310	LD	HL, (VARS)	690	RET	
320	LOC_Z\$		700		
330	LD	A, (HL)	710		
340	CP	128	720	PROG	EQU 23635
350	JR	NZ, LOC_A1	730	MAKE_R	EQU t1655
360			740	STRMS	EQU 23568
370	;	"Variable not found"	750	VARS	EQU 23627
380			760	NEXT_O	EQU t19B8
			770	ONE_SP	EQU t1652



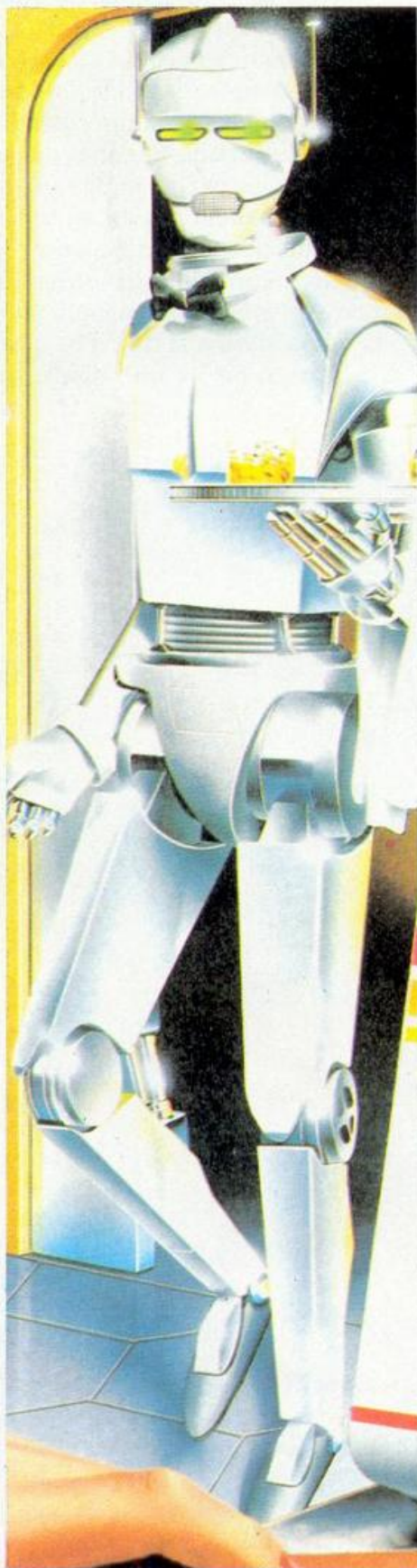
y de los fines que persiga. Por ejemplo, la rutina de salida usada para el canal «S» es nuestra conocida de la dirección 0010h (esa que usamos con RST 16), a la que hay que pasarle en A el código del carácter a escribir. Lo mismo hay que decir de las rutinas de entrada, que, al ser llamadas devuelven en el acumulador el valor conseguido del dispositivo de entrada (o cualquier cosa que usemos como tal).

Pero no sólo basta tener especificados en RAM los diferentes canales posibles, sino que tiene que haber un sitio en el que aparezca a qué canal corresponde cada una de las 19 corrientes (entre la -3 y la 15, incluidas las tres «fantasmas» ya comentadas y no accesibles desde el BASIC) que pueden llegar a coexistir en un Spectrum. Este lugar está situado entre las variables del sistema que vimos el pasado capítulo, concretamente a partir de la posición de memoria 23568, denominada STRMS. Allí podremos encontrar, con dos *bytes* para cada corriente, el desplazamiento relativo a (CHANS)-1 de cada una de las 19 corrientes empezando por la -3.

Por ejemplo, para el canal -3, podemos buscar en las direcciones 23568-9, para el 0 a partir de la 23568+6, o sea, en 23574-5; para la 1 en 23576-7, y así con todas. En el dos encontraremos, si no hemos enredado mucho aún, un 6, que sumado a (CHANS)-2 nos dará la dirección de comienzo de los 5 *bytes* correspondientes al canal «S» en memoria. Si el número que conseguimos en STRMS es un cero quiere decir que esa corriente no ha sido abierta (o que ha sido cerrada).

## Abriendo corrientes propias

Sabiendo todo esto, no resulta difícil abrir canales para que apunten a las rutinas que nos interesen, y nuestras propias corrientes para que



apunten a ellos. Lo primero que hay que hacer es abrir un canal en la zona de memoria correspondiente, es decir, al final de CHANS (otra opción menos elegante es situarlo en el *buffer* de la impresora), pero como aquí es donde normalmente se encuentra almacenado el programa BASIC, primeramente habrá que abrir un hueco de 5 *bytes* en esa dirección de forma que no alteremos dicho programa. Esto es bastante sencillo si utilizamos la rutina de la ROM llamada MAKE ROOM, a la que hay que pasarle en BC el número de espacios que queramos necesitamos, y en HL la dirección donde queremos insertar algo.

En el listado adjunto os ofrecemos una útil rutina cuyo cometido va a ser abrir un canal llamado «Z» que sólo funcione como dispositivo de salida e imprima cualquier cosa en la variable x\$; después dirigiremos hacia allí a la corriente 4 (o al «canal» 4), de modo que podamos, por ejemplo, ejecutar PRINT 4; 2\*2 y que un precioso «4» se añada al final del contenido de z\$.

Al comienzo se hace lo que comentamos dos párrafos más arriba para abrir un canal «Z» en la zona de información para canales. Esto lo efectúan las líneas 40-200, abriendo el hueco (líneas 40-80) y copiando los cinco *bytes* especificados en TBLCHN al final de esta zona (líneas 90-150). La subrutina ERROR es la de entrada, que hace que si, por ejemplo, intentamos ejecutar un INKEY\$ 4 aparezca el error correspondiente.

En la rutina de salida localizamos primero a Z\$ en el área de las variables (LOC\_Z\$), dando, si esta variable no existe o si está dimensionada, el error «Variable not found», o pasando a A\_Z\$ cuando está localizada. Allí se hace hueco para un nuevo elemento, se ajustan los punteros y se introduce el valor del acumulador al final de la cadena. El formato



**infodis, s.a.** LE OFRECE LOS MEJORES LIBROS  
PARA SU ORDENADOR

**(IVA INCLUIDO)**  
Muestra una visión más completa del correcto funcionamiento del juego de instrucciones del C-64.  
(108 páginas, tamaño 13,5 x 21,5).

C. P.



que tienen las variables en memoria dentro de su área correspondiente será estudiado extensamente, junto con la zona del BASIC, en el siguiente capítulo.

Hay que resaltar las limitaciones que tiene esta rutina: En primer lugar, la variable Z\$ debe haber sido creada (aunque sea como cadena vacía, es decir, con LET Z\$="") antes de intentar imprimir por el canal 4; si no ha sido así aparecerá el error «Variable not found», lo mismo que si la variable Z\$ había sido dimensionada anteriormente. La rutina no funcionará tampoco (lo hará incorrectamente) cuando se trate de imprimir cadenas de más de un carácter en forma directa, debiendo, en ese caso usar de intermediaria a cualquier otra variable. O sea, en lugar de hacer PRINT #4; "periquito", habría que hacer:

LET a\$="periquito": PRINT #4; a\$. Esto no será necesario cuando las sentencias son parte de un programa.

## Otros usos

Como hemos visto, no es difícil crear nuevos canales que respondan

a nuestras necesidades específicas, y que puedan ser usados en forma cómoda desde el BASIC para dar salida de cualquier cadena o valor numérico. Pero la cosa no se queda ahí, ya que podemos usar las posibilidades que nos brinda el BASIC del Spectrum para sacar por nuestro propio canal el catálogo de un cartucho o un listado BASIC. Por ejemplo, volviendo a la rutina anteriormente vista, podemos hacer LIST #4 para listar directamente en la variable Z\$, de esta forma tendremos en Z\$ todo el listado del programa BASIC que haya en ese momento, y allí, mediante un sencillo programa, podríamos localizar una determinada cadena o algo por el estilo.

Aunque el crear un nuevo canal no suele dar problemas, también puede usarse un canal ya existente y modificarlo para que sus rutinas apunten a donde nos interese. En este sentido el canal «K» no admite modificaciones, pues sus direcciones originales son restauradas cada vez que se ejecuta un INPUT, pero los otros tres sí pueden ser retocados para que apunten a nuestras rutinas, con lo que podríamos conseguir 64 caracteres por línea para el

canal «S» o un canal «P» que controle una impresora específica que no sea la ZX-Printer.

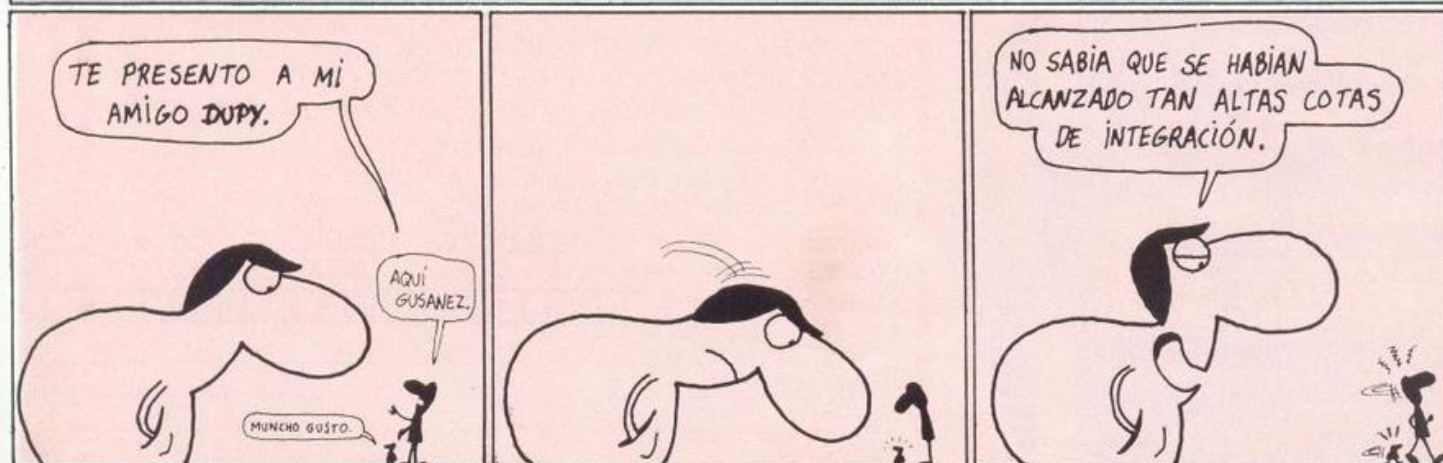
De todas formas, hay que andar con mucho cuidado cuando se trabaja en este tema, pues el operativo del Spectrum, aún siendo toda una obra de arte, no es todo lo seguro que cabría esperar y resulta siempre una caja de sorpresas. Por ejemplo, si queremos cerrar uno de los canales que hayamos implementado de nada nos servirá CLOSE #, de hecho esta instrucción, usada por los canales 4-15, colgará irremisiblemente la máquina si el Interface 1 está ausente.

Queda, pues, demostrado que el crear nuestros propios canales puede ser de gran utilidad a la hora de compenetrar el inflexible BASIC del Spectrum con algunas de nuestras rutinas de código máquina. Hay que volver a mencionar que lo dicho hasta el momento sólo se cumple en parte cuando tenemos conectado el Interface 1, por lo que quienes quieran aprender sobre todo lo que éste implementa concerniente a canales y corrientes, deberán esperar el momento en que dediquemos un capítulo a este interesante periférico.

Luis Gala

## GUSANEZ

por José C. Tomás





# LA MAS IMPORTANTE EDITORIAL DE REVISTAS DE INFORMATICA EN CASTELLANO

*El periódico*  
**INFORMATICO**

EL SEMANARIO PROFESIONAL  
POR EXCELENCIA

**ORDENADOR**  
POPULAR

LA REVISTA LIDER  
DE LOS MICROS

**PC**  
**MAGAZINE**  
EDICION EN CASTELLANO

LA PRIMERA REVISTA EN  
CASTELLANO PARA IBM PC  
Y COMPATIBLES

**MSX**

LA REVISTA IMPRESCINDIBLE  
PARA LOS INTERESADOS EN  
EL STANDAR JAPONES

**commodore**  
*Magazine*

LA DE MAYOR DIFUSION  
PARA ORDENADORES  
COMMODORE

**ZX**  
REVISTA PARA LOS USUARIOS  
DE ORDENADORES SINCLAIR

SINCLAIR

AL ALCANCE DE TODOS

**Todospectrum**

EL NIVEL MAS ALTO  
PARA SINCLAIR

publinformática, s/a

Bravo Murillo, 377 - 28020 MADRID Tel. (91) 733 74 13  
Pelayo, 12 - 08001 BARCELONA  
Tels. (93) 318 02 89 (93) 301 47 00 Ext. 27 28



## GUIA DEL HACKER

# CYBERUN





**E**stamos en el mes de julio, se acercan los exámenes y hace un tiempo espléndido para comernos la moral. En esta situación cayó en mis manos el nuevo programa de ULTIMATE. Solamente por tratarse de esta casa se supone que debe ser una mavarilla y un objetivo idóneo para la guía de Hackers. Se trata del CYBERUN. Cuando se carga por primera vez nos encontramos con que no es lo que esperamos. Los gráficos están bastante bien, pero no son del nivel al que

---

***El objetivo de Cyberun es montar las piezas de una nave espacial y recoger los cristales de Cybernita que nos permitirán escapar del planeta.***

---

nos tienen acostumbrados. Nada más comenzar el juego nos encontramos a los mandos de una curiosa

nave que avanza por el perfil de un planeta. Esta permanece siempre en el centro de la pantalla moviéndose el resto del decorado con un scroll muy suave. Según dicen las instrucciones el objetivo es montar las piezas de la nave que se encuentran repartidas por el planeta y recoger los cristales de Cybernita con los que podremos escapar del campo de fuerza que envuelve el planeta.

Conforme jugamos un par de partidas y aprendemos a defendernos





# GUÍA DEL HACKER

de los extraños habitantes del planeta nos damos cuenta de la gran complejidad del juego. El planeta tiene unas dimensiones impresionantes y un gran laberinto de cavernas subterráneas en las que nos perderemos con facilidad.

Con las ideas así de poco claras sobre lo que hay que hacer para terminar la aventura nos lanzamos a otra no menos compleja. Trataremos de averiguar qué hay detrás de este programa y conseguir algunos POKES que nos ayuden a acabar el juego.

Como siempre, empezaremos por ver cómo funciona el proceso de carga con la finalidad de conseguir que lo haga sin que lance el juego. Así tendremos el programa en memoria para poder analizarlo con ayuda del monitor. Empezamos por cargarlo normalmente y probamos a hacer un break después del primer bloque. La ejecución se detiene con un curioso informe «Out of memory». Sin duda se debe a que ha colocado el RAMTOP demasiado bajo y tiene problemas para crear la zona de variables correspondien-

tes al INTERFACE 1. Cambiamos el color de la tinta y ya podemos ver el listado del primer cargador. Nos encontramos con que aparte del CLEAR que ha originado el error hay dos LOAD " " CODE con una llamada a una rutina en máquina en medio y otra al final. El primer blo-

---

***El programa se lanza justo en la dirección en que comienza el bloque principal, desactiva las interrupciones y salta a una dirección superior.***

---

que tiene que ser la pantalla de presentación y la primera llamada se debe encargar de volcarla desde la dirección en que se ha cargado hacia la memoria de pantalla. Pero no debemos fiarnos de que haga solamente esto, sobre todo al darnos cuenta de que la segunda llamada,

la que lanza el programa, es a una dirección que corresponde con la memoria intermedia de la impresora, donde no ha cargado nada. Cargamos el primer bloque y el monitor para ver que es lo que ocurre realmente. Nos encontramos con que aparte de mover la pantalla prepara un par de bloques en código máquina para usarlos antes de lanzar el programa. Uno de ellos va a parar justo a la dirección 5B80, a la que se llama desde el BASIC. La finalidad de estas rutinas es mover el bloque principal hasta la dirección #5C80 y luego rotarlo todo medio byte a la derecha. Esta es la dirección en que lanza finalmente el programa, lo que empieza a preocuparnos. Cae justo en medio de las variables del sistema BASIC, lo cual imposibilita el utilizar el monitor con el programa cargado en su sitio. (En realidad sí que existe un monitor de PICTURESQUE que es totalmente independiente de estas variables y que podría funcionar, pero no es reubicable y no tiene las prestaciones del MONS). En principio intentaremos cargar el programa en otra

```
10 CLEAR 25200
20 LOAD "CODE 16313: PAPER 0:
PRINT AT 19,0
30 LOAD "CODE 25216
40 LET s=0: FOR i=23296 TO 233
29: READ a: LET s=s+a: POKE i,a:
NEXT i
50 IF s<>4304 THEN STOP
60 RANDOMIZE USR 23296
70 INPUT "Nave montada ?(s/n)
"; LINE a$
80 IF a$="s" THEN POKE 63902,
201: POKE 64494,1: FOR i=4 TO 9:
POKE 64493+i,i: NEXT i
90 IF a$<>"s" THEN INPUT "Par
tida continuada ?"; LINE a$: IF
a$="s" THEN POKE 63902,201
100 INPUT "Inmortal ?"; LINE a$
: IF a$="s" THEN POKE 38278,0
```

```
110 INPUT "Sin combustible ?";
LINE a$: IF a$="s" THEN POKE 53
879,201
120 INPUT "Escapar sin montar e
l cohete ?"; LINE a$: IF a$="s"
THEN POKE 40029,0
130 INPUT "Cualquier orden ?";
LINE a$: IF a$="s" THEN POKE 45
293,0
140 INPUT "Sin enemigos ?"; LIN
E a$: IF a$="s" THEN POKE 37780
,4
900 RANDOMIZE USR 23312
1000 DATA 175,1,128,158,33,255,2
55,237,103,43,13,32,250,16,248,2
01
1010 DATA 243,49,0,254,33,128,98
,17,128,92,1,128,157,237,176,195
,128,92
```



dirección y deshacer la máscara. Cargamos el monitor encima y empezamos a analizar en una dirección cambiada. Aunque se han elegido las direcciones de forma que sea muy sencillo realizar la conversión (para mirar lo que hay en la dirección #5C80 hay que dirigirse a #6280) resulta realmente incómodo. Si tuviéramos que hacer esto durante todo el análisis habría que tener mucho cuidado con las direcciones.

Sin embargo, nada más comenzar el análisis nos encontramos con un rayo de esperanza. Lo primero que hace es desactivar las interrupciones y saltar a una dirección mucho mayor (#9A66) en la que ya no habrá problemas para manejarlo en su sitio.

Esta forma de lanzar el programa justo en la dirección en que comienza el bloque principal y hacer rápi-

TABLA 1

POKE 63902,201	Partida continuada.
POKE 37745,X	Número de enemigos (máximo 9).
POKE 63951,X	Número de vidas iniciales.
POKE 64207,0	Vidas infinitas.
POKE 38278,0	Inmortal.
POKE 53617,201	Evita la muerte por quedarnos sin combustible.
POKE 40029,0	Permite escapar sin hacer nada.
POKE 45293,0	No importa el orden.
POKE 37780,4	Sin enemigos.
POKE 45369,0	Basta con acercarse para coger una cosa.
POKE 51168,195	Permite bajar la vela a los subterráneos.
POKE 53879,201	No consume combustible.

Para estos POKES hay que jugar con el de partida continuada.

POKE 64494,1	Tenemos las pinzas desde el principio.
POKE 64497,4	Propulsión horizontal.
POKE 64498,5	Propulsión vertical.
POKE 64499,6	Ruedas.
POKE 64500,7	Plasma.
POKE 64501,8	Bombas.
POKE 64502,9	Soporte de la vela.

## DISPONEMOS DE TAPAS ESPECIALES PARA SUS EJEMPLARES DE

# Todospectrum

SIN NECESIDAD DE ENCUADERNACION

PRECIO UNIDAD  
**650** ptas.

Para hacer su pedido, rellene este cupón HOY MISMO y envíelo a:

**Todospectrum**

Bravo Murillo, 377  
Tel. 733 96 62 - 28020 MADRID

Por favor envíenme ..... tapas para la encuadernación de mis ejemplares de TODOSPECTRUM, al precio de 650 pts. más gastos de envío.

El importe lo abonaré

☐ POR CHEQUE ☐ CONTRA REEMBOLSO ☐ CON MI TARJETA DE CREDITO ☐ AMERICAN EXPRESS ☐ VISA ☐ INTERBANK

Número de mi tarjeta:

Fecha de caducidad ..... Firma

NOMBRE .....

DIRECCION .....

CIUDAD ..... C. P. ....

PROVINCIA .....

(cada tapa es para 6 ejemplares)



# GUIA DEL HACKER

damente un salto a la dirección en que realmente debe empezar el juego es clásica de **ULTIMATE**. Que yo recuerde la ha utilizado en casi todos sus programas excepto en los dos escritos con la técnica **Filmation 1**, que coincidieron con la época en que utilizo el método de protección **SPEEDLOCK**, en el que no quedaba claramente definido el bloque principal del programa. En general suele colocar a continuación las tablas de datos y el mapa. De todas formas nunca habían apurado tanto la memoria como para sobrecribir el área de las variables del sistema **BASIC**.

Rápidamente, montamos un nuevo cargador que aunque no carga todo el programa, lo hace en la dirección correcta y nos permitirá incluso ejecutar paso a paso algunas rutinas. Pasamos a ver lo que ocurre en #9A66. Se trata de la zona de inicialización general del programa. Llama a una subrutina en #F86B y luego salta a #F422, la dirección del bucle principal. En la rutina a la que llama nos encontramos con que a partir de #FA00 crea una serie de tablas de desplazamiento que servirán para acelerar el proceso de scroll de la pantalla. Lo que hace es colocar en la dirección #FAXX el byte XX rotado dos pixels a la derecha, y los dos bits que sacamos pasan a la dirección #FBXX entrando por la izquierda. Lo mismo hace a partir de #FC00 pero rotando cuatro veces y a partir de #FE00 rotando seis veces. A la vista de estas tablas se deduce que el movimiento más pequeño es de dos pixels. La principal ventaja de esto es que sólo tiene que desplazar el contorno del planeta, ya que el interior es una cuadrícula que al moverla dos posiciones coincide con ella misma. De la misma forma crea a partir de #F900 la tabla de rotaciones en la que para cada byte tiene almacenado su simétrico. Todas estas tablas ocupan en total 1792 bytes, pero a cambio se consigue una gran velocidad en los desplazamientos y giros.

Continuamos el análisis en #F422. Aquí está el bucle principal

de juego. Se extiende hasta #F48A y sólo se abandona cuando nos han matado o hemos conseguido abandonar el planeta. Enseguida surge la idea de anular los dos saltos condicionales que controlan estas dos salidas. Pero los resultados no son demasiado buenos. Modificando el primero (**NO**Peando #F481) nos siguen matando de vez en cuando aunque sin descontar la vida. Sin embargo, cuando se acaba el combustible nos empiezan a matar continuamente quedando el programa bloqueado. Si **POKE**amos el segundo poniendo en #F487 un cero, nada

puntos. Como funciona con las interrupciones deshabilitadas esto provocará el bloqueo del ordenador. De esta forma podremos saber cuál es el efecto de la parte de programa que ha sido ejecutada. Así llegamos a la conclusión de que la primera rutina a la que se llama (#F26C) gestiona toda la parte correspondiente al menú. A partir de aquí debe de estar la inicialización de variables y este es un punto del que normalmente podemos sacar bastante partido. Por esta razón nos adelantamos en la rutina de la dirección #F3BD. Lo primero que hace es poner a cero los



más empezar la partida nos felicita por haber conseguido el objetivo de nuestra misión. Sólo sirve para descubrir que en total hay cinco planetas que se suceden cíclicamente y que cada vez que abandonamos uno obtenemos 50.000 puntos.

El bucle se cierra sobre #F46C mientras jugamos y sobre #F440 cuando nos acaban de matar. Lo que hay antes de esta última dirección debe de ser la parte de inicialización de la partida y el menú. Para ir aislando la función de cada parte del programa probamos a colocar una instrucción **HALT** en distintos

dos primeros bits de los 290 bytes a partir de la dirección #5C94. Para saber para que sirven modificamos esto y nos encontramos con que se trata de todos los soportes donde pueden estar colocadas las cosas que buscamos. A continuación inicializa de la misma forma dos zonas de memoria llamando dos veces a la misma subrutina. Esta dualidad nos hace pensar en los dos jugadores. Deben de ser las variables que se conservan cuando le toca jugar al otro jugador, esto es: los puntos, la posición en que estamos, las partes de la nave que tenemos montadas, etc.



Sería una buena idea el que todo esto se pudiera conservar de una partida a otra. Para ello bastaría con colocar una instrucción RET al principio de la rutina. Pero no funcionará a no ser que cuando se cargue el programa de la cinta ya estén las variables en su sitio. Rápidamente vamos a las direcciones donde deben estar y nos encontramos con que si que tienen el valor con que son inicializadas. Poniendo en la dirección #F39E el dato #C9 se juega una «partida continuada», en la que si nos matan continuamos la siguiente partida en la situación en que está-

distintos valores en el registro HL. Tiene que ser algo relacionado con los dos jugadores. Esta zona de memoria corresponde con la que hemos destruido para poder funcionar con el monitor. Quizá pudiéramos pasar sin saber cuál es su finalidad, pero puede ser importante y no cuesta tanto cargar la primera copia que hicimos y analizar la rutina desplazada. Nos ponemos a ello. Lo primero con lo que nos encontramos es que pone a cero los bits.7 de una serie de bytes y luego realiza un extraño sorteo. Analizando detenidamente y con una serie de PO-

de dieciséis. Cada objeto se sortea dentro de un grupo distinto. Así todas las partes de la nave están en la superficie y relativamente cerca de la posición de partida. En cambio las partes de la unidad de propulsión se suelen encontrar en los pasadizos.

Sabiendo ya como esta codificado todo esto no ofrece dificultad adentrarse en el resto del programa en busca de los POKES que nos permitan acabar fácilmente el juego (o avanzar en el ya que no tiene final). Anulando las llamadas a las rutinas en el bucle principal averiguamos cual es la utilidad de cada una de ellas. La que esta en la dirección #C374 se encarga del control de nuestra nave y del teclado, la de #CC77 del control del combustible, la de #8D42 del movimiento de todos los objetos de la pantalla, y la de #F23B de generar números aleatorios.

Los POKES finalmente encontrados los tenéis en la tabla 1 y para usarlos basta con ponerlos en el programa 1 a partir de la línea 500. En este cargador ya se han incluido algunos más interesantes, entre ellos el que nos permite tener la nave totalmente montada antes de empezar la partida. Esto implica tener que jugar una partida continuada, puesto que sino en la inicialización nos quitaría todo. Para los que intentéis continuar buscando más cosas tened en cuenta que las direcciones están calculadas para este cargador concreto que coloca el bloque principal desplazado 1536 bytes respecto a su dirección de ejecución. En esta ocasión no se ha construido un cargador de los que nos preguntan por los POKES antes de cada partida por el problema de que la memoria está utilizada en su totalidad y no hemos podido encontrar un sitio donde colocarlo durante la ejecución. Por último una advertencia respecto el POKE que nos permite recoger los pedazos de la unidad de propulsión en cualquier orden. Si se coge más de uno simultáneamente podéis tener problemas para colocar el último.

Manuel Arana



bamos.

Si seguimos con la rutina de inicialización nos encontramos con que coloca en las direcciones #FSED y #F656 el dato #05. No hay duda de que tiene que ser el número de vidas de los dos jugadores. Para asegurarnos cambiamos este dato por otro mayor y en efecto tenemos más vidas. Como siempre buscamos el punto en que se decrementa y conseguimos el POKE de vidas infinitas.

El resto de la rutina cambia de sitio algunas variables y al final llama dos veces a la dirección #6113 con

KEs de ayuda llegamos a la conclusión de que se trata de la elección de la posición de las piezas de la nave y de la unidad de propulsión. En total tiene que sortear 16 cosas: siete partes de la nave, ocho partes de la unidad de propulsión y la vela, que sólo puede ser cogida cuando ya tenemos el soporte. La vela en realidad está formada por dos objetos, pero sólo se sortea la parte de abajo, quedando determinada la posición de la parte de arriba. Para realizar el sorteo tiene en total 44 posibles posiciones divididas en 5 grupos, uno de dos, otro de cuatro, otro de seis y dos



# MANDELBROT ataca de nuevo

No creemos en duendes, pero haberlos haylos, y el mes pasado se ensañaron con el artículo de Gerardo Izquierdo dedicado al conjunto de Mandelbrot. Tras una ardua batalla con la sección de montaje, tres de los cinco listados del artículo fueron derrotados y publicados en absoluto desorden, por lo que, esta vez con un fuerte apoyo de artillería por parte de la redacción, los reproducimos de nuevo íntegramente.

```

100 inicializa
110 REPEAT lazo
120   rellena
130   aumenta
140 END REPEAT lazo
150 :
160 DEFINE PROCEDURE rellena
170   dx=dx/256
180   dy=dy/256
190   FOR i=0 TO 255
200     xx = XI + dx * i
210     yy1 = yi + 255 * dy
220     yy2 = yi + 127 * dy
230     bisecy INT (i/2,0,127,HUIDA (xx,yy1),HUIDA (xx,yy2))
240     yy1 = yi + 127 * dy
250     yy2 = yi
260     bisecy INT (i/2,127,255,HUIDA (xx,yy1),HUIDA (xx,yy2))
270   END FOR i
280 END DEFINE rellena
290 :
300 DEFINE PROCEDURE bisecy (x%,y%,y2%,c1%,c2%)
310   LOCAL xcc,ycl,yc2
320   xcc = XI + dx * x%
330   ycl = yi + dy * (255 - INT((y1%+y2%)/2))
340   yc2 = yi + dy * (255 - INT((y1%+y2%)/2)-1)
350   IF c1% = c2% THEN
360     linea x%,y1%,x%,y2%,c1%
370   ELSE
380     IF (y2%-y1%) = 1 THEN
390       BLOCK 2,1,248,y1%,color% (c1%)
400       BLOCK 2,1,248,y2%,color% (c2%)
410     ELSE
420       bisecy x%,y1%,INT((y1%+y2%)/2),c1%,HUIDA(xcc,ycl)
430       bisecy x%,INT((y1%+y2%)/2)+1,y2%,HUIDA(xcc,yc2),c2%
440     END IF
450   END IF
460 END DEFINE bisecy
470 :
480 DEFINE PROCEDURE linea (xa%,ya%,xb%,yb%,c%)
490   BLOCK (248*(xb%-xa%+1)),(yb%-ya%+1),(248*xa%),ya%,color% (c%)
500 END DEFINE linea
510 :
520 DEFINE PROCEDURE inicializa
530   MODE 8:WINDOW 512,256,0,0
540   PAPER 2:CLS:CSIZE 3,1
550   INPUT " 4x?''XI'' 4y?''yi'' 4dx?''dx'' 4dy?''dy''
560   PAPER 0:CLS
570 END DEFINE inicializa
580 :
590 DEFINE FUNCTION color% (x%)
600   IF x%=1000 THEN RETURN (0)
610   IF x% > 300 THEN RETURN (7)
620   IF x% > 100 THEN RETURN (6)
630   IF x% > 32 THEN RETURN (4)
640   RETURN (1)
650 END DEFINE color%
660 :
670 DEFINE PROCEDURE aumenta
680   x% = 0
690   y% = 0

```

```

700   dx% = 512
710   dy% = 256
720   OVER -1
730   box
740   REPEAT loza
750     tecla=CODE(INKEY$(1))
760     box
770     SELECT ON tecla
780       = 192:IF x% > 0 THEN x%=x%-2
790       = 200:IF x% + dx% < 512 THEN x%=x%+2
800       = 216:IF y% + dy% < 256 THEN y%=y%+1
810       = 208:IF y% > 0 THEN y%=y%-1
820       = 193:IF dx% > 10 THEN dx%=dx%-2
830       = 201:IF dx% < 512 THEN dx%=dx%+2
840       = 217:IF dy% < 256 THEN dy%=dy%+1
850       = 209:IF dy% > 5 THEN dy%=dy%-1
860       = 10:nuevo:OVER 0:RETURN
870       = REMAINDER
880     END SELECT
890     box
900   END REPEAT loza
910 END DEFINE aumenta
920 :
930 DEFINE PROCEDURE box
940   BLOCK dx%-1,1,x%,y%,7
950   BLOCK 2,dy%,x%+dx%-2,y%,7
960   BLOCK dx%-1,1,x%,y%+dy%-1,7
970   BLOCK 2,dy%,x%,y%,7
980 END DEFINE box
990 :
1000 DEFINE PROCEDURE nuevo
1010   ddx = dx% / 512
1020   ddy = dy% / 256
1030   XI = XI + x% * ddx
1040   yi = yi + (256-y%-dy%+1)*ddy
1050   dxi = dxi * dx% / 512
1060   dyi = dyi * dy% / 256
1070 END DEFINE nuevo

```

```

310   y=y+dy
320   x=x0
330   FOR i=0 TO 511 STEP 2*factor
340     x=x+dx
350     BLOCK 2*factor,factor,i,256-factor-j,color%(HUIDA(x,y))
360   END FOR i
370 END FOR j
380 END DEFINE rellena
390 :
400 DEFINE FUNCTION color% (x%)
410   IF x%=1000 THEN RETURN (0)
420   IF x% > 300 THEN RETURN (7)
430   IF x% > 100 THEN RETURN (6)
440   IF x% > 20 THEN RETURN ((1*x% DIV 3) MOD 4) + 2)
450   RETURN (1)
460 END DEFINE color%
470 :
480 DEFINE PROCEDURE aumenta
490   x% = 0
500   y% = 0
510   dx% = 512
520   dy% = 256
530   OVER -1
540   box
550   REPEAT loza
560     tecla=CODE(INKEY$(1))
570     box
580     SELECT ON tecla
590       = 192:IF x% > 0 THEN x%=x%-2
600       = 200:IF x% + dx% < 512 THEN x%=x%+2
610       = 216:IF y% + dy% < 256 THEN y%=y%+1
620       = 208:IF y% > 0 THEN y%=y%-1
630       = 193:IF dx% > 10 THEN dx%=dx%-2
640       = 201:IF dx% < 512 THEN dx%=dx%+2
650       = 217:IF dy% < 256 THEN dy%=dy%+1
660       = 209:IF dy% > 5 THEN dy%=dy%-1
670       = 10:nuevo:OVER 0:RETURN
680       = REMAINDER
690     END SELECT

```

```

700   box
710   END REPEAT loza
720 END DEFINE aumenta
730 :
740 DEFINE PROCEDURE box
750   BLOCK dx%-1,1,x%,y%,7
760   BLOCK 2,dy%,x%+dx%-2,y%,7
770   BLOCK dx%-1,1,x%,y%+dy%-1,7
780   BLOCK 2,dy%,x%,y%,7
790 END DEFINE box
800 :
810 DEFINE PROCEDURE nuevo
820   ddx = dx% / 512
830   ddy = dy% / 256
840   XI = XI + x% * ddx
850   yi = yi + (256-y%-dy%+1)*ddy
860   dxi = dxi * dx% / 512
870   dyi = dyi * dy% / 256
880 END DEFINE nuevo

```



```

;-----
; Esta rutina assembler, lista para ser llamada por el BASIC
; permite ampliar zonas del conjunto de Mandelbrot.
;
; Para ello, se teclea HUIDA x,y y devuelve el numero que
; hace falta para que su modulo pase de 2.
;
; Ver Investigacion y Ciencia de Octubre de 1.985
;
; Copyright Gerardo Izquierdo 10-85
;-----

;-----
; Igualdades iniciales
;-----

;----- Vectores -----
ut_scr equ $C8
ut_err0 equ $CA
bp_init equ $110
ca_gtfp equ $114
bv_chrix equ $114
ri_exec equ $11C
ri_execb equ $11E

;----- Trap #2 -----
fo_open equ $1
fo_close equ $2

;----- Trap #3 -----
sd_fill equ $2E

;----- Errores -----
err_bp equ -15

;----- SuperBasic -----
bv_rip equ $58

;-----
; Inicializar el procedimiento
;-----

inicio lea.l proc_def,a1
move bp_init,a2
jsr (a2)
rts

;-----
; Tabla de definicion de procedimientos
;-----

proc_def dc.w 0 ; 0 procedimientos
dc.w 0
dc.w 1 ; 1 funcion
dc.w mandel-4 ; comienzo
dc.b 5,'HUIDA' ; nombre
align
dc.w 0,0

;-----
; Cargar los valores de x,y
;-----

mandel
move.l #60,d1 ; Reservamos sitio para 10 numeros
move.w bv_chrix,a2
jsr (a2)
move.w ca_gtfp,a2 ; poner en el stack los parametros
jsr (a2)

cmp.w #2,d3 ; #Hay dos parametros?
bne errr_bp ; NO! error
clr.l -4(a6,a1.l) ; ponemos 2 zeros (x1 e y1).
clr.l -8(a6,a1.l)
clr.l -12(a6,a1.l)
move.l a1,a4
add.l #12,a4
sub.l #18,a1 ; actualizamos el puntero de pila
move.l #540000000,2(a6,a1.l) ; ponemos 4
move.w #0803,0(a6,a1.l)
clr.l d4 ; empezamos a contar

lazo!
addq.w #1,d4 ; a adimos 1 al contador
moveq #0,d7 ; siempre cero
lea.l calc_l,a3 ; lista de calculos
move.w ri_execb,a2 ; orden de calcular
jsr (a2) ; R-4 en 0(A6,A1.l)
tst.l d0

```

```

bne.s retorno
add.w #6,a1 ; restauramos pila
tst.l -4(a6,a1.l) ; miramos signo de R-4
bge.s final ; si >=0 terminar
cmp.w #1000,d4 ; #1000 iteraciones?
blt.s lazo! ; NO seguir

;-----
; Devolvemos el resultado y terminamos
;-----
final
moveq #0,d0
move.l a4,a1 ; tomamos el indice bueno de la pila
subq #2,a1 ; a adimos 2 actetos
move.l a1,bv_ripladl ; salvamos el nuevo valor
move.w d4,0(a6,a1.l) ; cargamos ahí el numero de iter.
moveq #3,d4 ; el tipo es entero
rts ; terminamos

;-----
; Error y retorno
;-----
errr_bp
moveq.l #err_bp,d0
retorno rts

;-----
; Definiciones de los calculos
;-----

l_x equ -12
l_y equ -6
l_xl equ -24
l_yl equ -18
s_xl equ -23
s_yl equ -17
l_4 equ -30

calc_l dc.b l_xl ; x1
dc.b $16 ; x1 x1
dc.b $16 ; x1 x1 x1
dc.b $E ; x1x1 x1
dc.b l_yl ; y1 x1x1 x1
dc.b $16 ; y1 y1 x1x1 x1
dc.b $E ; y1y1 x1x1 x1
dc.b $C ; x1x1-y1y1 x1
dc.b l_x ; x x1x1-y1y1 x1
dc.b $A ; x1x1-y1y1+x x1
dc.b s_xl ; x1
dc.b l_yl ; y1 x1
dc.b $E ; x1y1
dc.b $16 ; x1y1 x1y1
dc.b $A ; 2x1y1
dc.b l_y ; y 2x1y1
dc.b $A ; 2x1y1+y
dc.b $16 ; 2x1y1+y 2x1y1+y
dc.b s_yl ; 2x1y1+y

; En estos momentos, x1 e y1 tienen el nuevo valor.
dc.b $16 ; y1 y1
dc.b $E ; y1y1
dc.b l_xl ; x1 y1y1
dc.b $16 ; x1 x1 y1y1
dc.b $E ; x1x1 y1y1
dc.b $A ; x1x1+y1y1 = R
dc.b l_4 ; 4 R
dc.b $C ; R-4
dc.b 0 ; fin

```

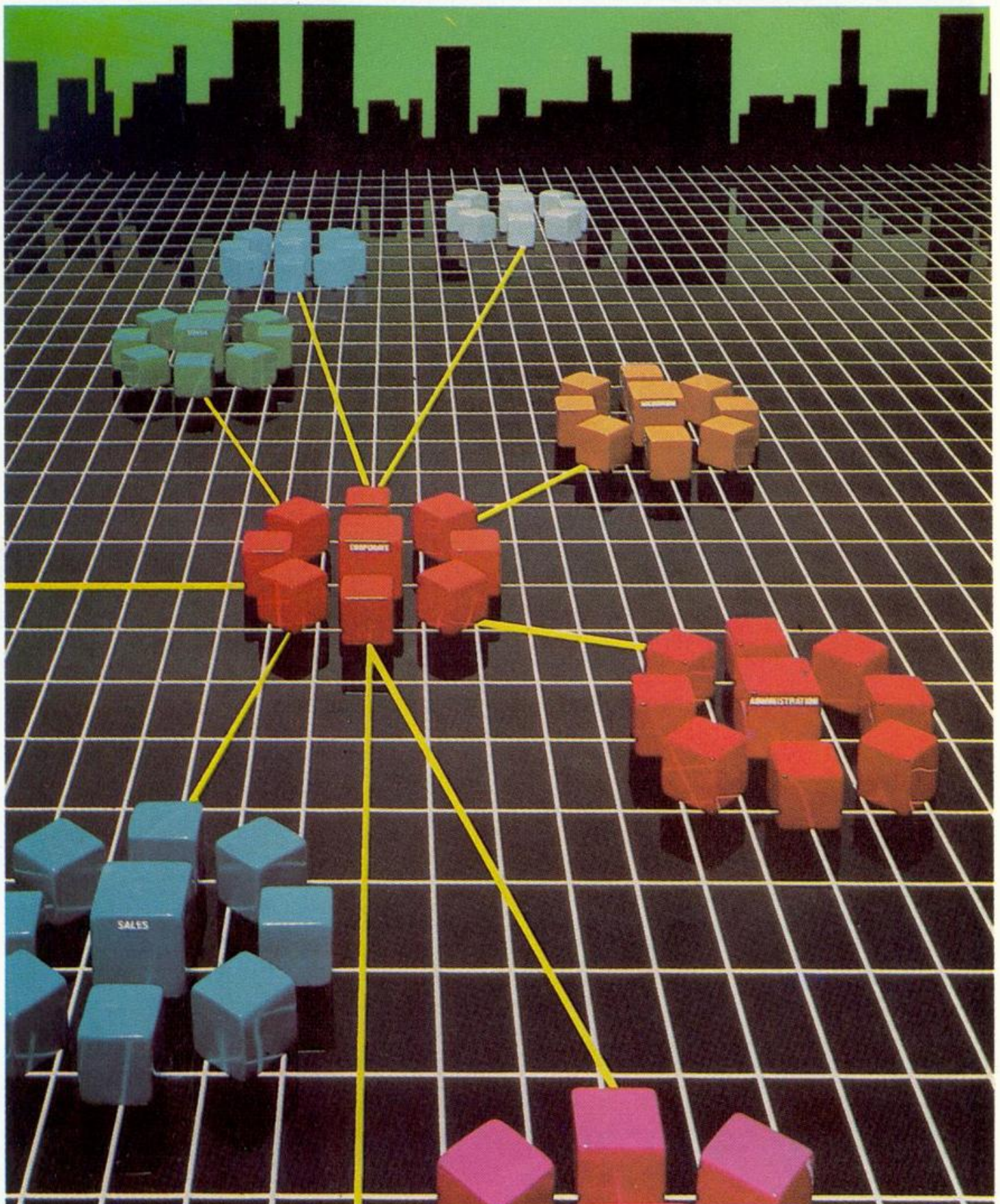
```

;-----
;
; zona de variables
;-----
;
; y
;-----
;
; x
;-----
;
; y1
;-----
;
; x1
;-----
;
; 4
;-----
;

```



# PROGRAMAS





# PROGRESIONES

Joaquín Herreros es el autor de este programa que permite resolver los problemas que se presentan con más frecuencia al trabajar con progresiones, tanto aritméticas como geométricas.

Las opciones posibles son: hallar la razón de una progresión, hallar un término determinado, hallar el número de términos y hallar la suma de todos los términos.

```
7 BORDER 0: PAPER 0: INK 7: C
LS
15 GO SUB 1100
19 GO SUB 20: GO TO 30
20 CLS : PRINT "
```

```
": FOR a=1 TO
20: PRINT AT a,0;" ": PRINT AT a
,31;" ": NEXT a: PRINT "
```

```
": RETUR
N
```

```
30 PRINT AT 7,2;"1.-PROGRESION
ES ARITMETICAS.": PRINT AT 14,2;
"2.-PROGRESIONES GEOMETRICAS."
```

```
40 IF INKEY$="1" THEN LET b=1
: GO SUB 20: GO TO 100
50 IF INKEY$="2" THEN LET b=2
: GO SUB 20: GO TO 100
60 GO TO 40
```

```
100 PRINT AT 4,2;"1.- HALLAR LA
RAZON.": PRINT AT 8,2;"2.- HALL
AR UN TERMINO DETER-": PRINT AT
9,6;"MINADO DE LA PROGRESION.":
PRINT AT 13,2;"3.- HALLAR EL NUM
ERO DE TER-": PRINT AT 14,6;"MIN
OS DE LA PROGRESION.": PRINT AT
18,2;"4.- HALLAR LA SUMA DE TODO
S": PRINT AT 19,6;"LOS TERMINOS.
"
```

```
110 IF INKEY$="1" THEN LET c=5
: GO SUB 20: GO SUB b*1000: GO T
O b*1000+c*100
```

```
120 IF INKEY$="2" THEN LET c=6
: GO SUB 20: GO SUB b*1000: GO T
O b*1000+c*100
```

```
130 IF INKEY$="3" THEN LET c=7
: GO SUB 20: GO SUB b*1000: GO T
O b*1000+c*100
```

```
140 IF INKEY$="4" THEN LET c=8
: GO SUB 20: GO SUB b*1000: GO T
O b*1000+c*100
```

```
150 GO TO 110
1000 PRINT AT 2,3;"LA PROGRESION
ES DEL TIPO:": PRINT AT 4,3;"t(
1),t(2),t(3),t(n-1),t(n)": PRINT
AT 6,3;"siendo:"
```

```
1001 PRINT AT 8,1;"t(2)=t(1)+r":
PRINT AT 10,1;"t(3)=t(2)+r t(3
)=t(1)+(3-1)xr": PRINT AT 12,1;"
t(4)=t(3)+r t(4)=t(2)+(4-2)xr":
PRINT AT 14,1;"....."
```

```
.....": PRINT AT 16,1;"
t(n)=t(n-1)+r": PRINT AT 19,8; F
LASH 1;"t(a)=t(b)+(a-b)xr"
```

```
1002 PRINT #0;" PULSA UNA TECLA
PARA CONTINUAR "
```

```
1003 PAUSE 0
1004 GO SUB 20: GO SUB 1010: RET
URN
```

```
1010 PRINT AT 2,2;"LA SUMA DE TO
DOS LOS TERMINOS";AT 3,2;"DE LA
PROGRESION: ";AT 5,2;"t(1),t(2),t
(3),t(n-1),t(n)";AT 7,2;"ES: ";AT
9,2;"S=t(1)+t(2)+.....+t(n-1)+t
(n)";AT 11,2;"Y SUSTITUYENDO QUE
NA:"
```

```
1020 PRINT AT 13,10;"S=t(1)+t(n)
xn";AT 14,16;"2": PLOT 96,63: DR
AW 70,0
```



# PROGRAMAS

- 1.- HALLAR LA RAZON.
- 2.- HALLAR UN TERMINO DETERMINADO DE LA PROGRESION.
- 3.- HALLAR EL NUMERO DE TERMINOS DE LA PROGRESION.
- 4.- HALLAR LA SUMA DE TODOS LOS TERMINOS.

LA SUMA DE TODOS LOS TERMINOS DE LA PROGRESION:

$t(1), t(2), t(3), t(n-1), t(n)$

ES:

$S = t(1) + t(2) + \dots + t(n-1) + t(n)$

Y SUSTITUYENDO QUEDA:

$$S = \frac{t(n) \times r - t(1)}{r - 1}$$

```
1030 PRINT #0;"PULSA UNA TECLA PARA CONTINUAR"
1035 PAUSE 0
1040 RETURN
1100 BORDER 0: PAPER 0: INK 7: CLS
1110 PRINT AT 7,0;"          PROGRAMA
      REALIZADO POR:"
1120 PRINT ': PRINT "          JO
      AQUIN ARGON          "
1130 PRINT ': PRINT "          Y PUBLIC
      ADD FOR LA REVISTA  "
1140 PRINT ': PRINT "          T
      ODOSPECTRUM          "
1150 PRINT #0;"          PULSA UN
      A TECLA          ": PAUSE 0: RETURN
1500 CLS : PRINT " La formula que
      vamos a emplear para hallar la
      razon es:" : PRINT ': PRINT ; FL
```

```
ASH 1;"r=t(a)-t(b)/(a-b)"; FLASH
0;" siendo:"
1501 LET d=b: PRINT ': PRINT "t(
a) y t(b) dos terminos conoci-
dos de la progresion.": PRINT ': P
RINT "a y b los lugares que esto
s ocupan en ella."
1502 PRINT ': PRINT " INTRODUCE
EL LUGAR DEL PRIMER "; "TERMINO
CONOCIDO Y LUEGO SU VA- "; "LOR,
Y SEGUIDAMENTE HAZ LO MISMO"; "C
ON EL OTRO TERMINO."
1503 INPUT " DAME EL LUGAR Y EL
VALOR DEL PRIMER TERMINO: "; b,
tb
1505 IF b<=0 THEN GO TO 1503
1510 INPUT " HAZ LO MISMO CON E
L SIGUIENTE TERMINO: "; a, ta
1512 IF a<=0 THEN GO TO 1510
1513 IF a=b AND ta<>tb THEN GO
TO 1503
1514 IF a=b AND ta=tb THEN LET
l=0
1515 IF d=2 THEN RETURN
1516 LET z=ta-tb: LET w=a-b: LET
l=z/w: CLS : PRINT ': PRINT " L
a solucion a tu problema es: ":
PRINT ': PRINT " RAZON = "; l
1517 FOR q=1 TO b-1
1518 LET tb=tb-1
1519 NEXT q
```


## VALENTE computación

MADRID BUENOS AIRES

PROGRAMAS PARA QL DESDE 2.500

JUEGOS : Match Point & Chess &  
Games Cartridge & Hiper Drive &  
Night Flight & Snooker & etc.  
UTILITARIOS: Teaprint & QL Paint &  
GraphiQL & Toolkit & QL Doctor &  
Gapeel & Lisp & Pascal & Monitor &  
Forth & BCPL & Editor Assembler &  
Generador Sprites & S. Astrologer  
COMERCIALES : Administración de  
Financ & Home Accounter Manager &  
Contabilidad General & Archiver &  
Life & Business Organizer & etc.

SPECTRUM PLUS ..... 27.900  
COPIADOR "PHOENIX II-E"..... 9.000  
Grandes oportunidades en programas y  
perifericos de SPECTRUM y QL.  
ENVIOS CONTRA REEMBOLSO A TODA ESPAÑA

Santa Engracia, 88 ☎ 445 32 85  
28010 MADRID /  IGLESIA



## PROGRAMAS

```
1520 PRINT ': PRINT " El primer
termino de la progre- sion es: "
;tb
1521 PRINT ': PRINT " DAME EL NU
MERO DE TERMINOS QUE TIENE LA P
ROGRESION Y TE DARE EL VALOR D
E CADA UNO DE ELLOS. SI NO, MET
E "; FLASH 1;"CERO"; FLASH 0;"."
1522 INPUT ;n
1523 IF n=0 THEN RUN 15
1524 IF n<0 THEN GO TO 1522
1525 CLS
1526 PRINT ': PRINT " TERMINO
VALOR ": PRINT AT
1,0; OVER 1;"
-----
"; OVER 0: LET abc
=6: LET abd=26
1527 FOR q=1 TO n
1530 IF q>=10 THEN LET abc=5: I
F q>=100 THEN LET abc=4: IF q>=
```

```
1000 THEN LET abc=3: IF q>=1000
0 THEN LET abc=2
1535 IF tb>=10 THEN LET abd=25:
IF tb>=100 THEN LET abd=24: IF
tb>=1000 THEN LET abd=23: IF t
b>=10000 THEN LET abd=22: IF tb
>=100000 THEN LET abd=21: IF tb
>=1000000 THEN LET abd=20
1538 PRINT ': PRINT TAB abc;q;TA
B abd;tb
1539 LET tb=tb+1
1540 NEXT q
1543 PRINT #0;"PULSA UNA TECLA P
ARA CONTINUAR"
1545 PAUSE 0
1550 RUN 15
1600 CLS : PRINT " La formula qu
e vamos a emplear para hallar el
termino descono- cido es:" PRI
NT ': PRINT ; FLASH 1;"t(a)=t(b)
```

# TodoSpectrum

**ANUNCIESE  
por  
MODULOS**

**MADRID  
(91) 733 96 62  
BARCELONA  
(93) 301 47 00**



# PROGRAMAS

```

+(a-b)*r"; FLASH 0;" siendo:"
1601 LET d=b: PRINT ": PRINT "t(
a) el termino que queremos ha-ll
ar de la progresion.": PRINT ":
PRINT "t(b) el termino conocido.
"
1602 PRINT ": PRINT "a y b los l
ugares que estos ele-mentos ocup
an en la progresion.": PRINT ":
PRINT " INTRODUCE POR ESTE ORDEN
:": PRINT ": PRINT " -EL LUGAR Q
UE OCUPA EL TERMINO DESCONOCIDO.
": PRINT " -EL LUGAR QUE OCUPA E
L TERMINO QUE CONOCEMOS.": PRINT
" -EL VALOR DE ESTE."
1603 INPUT "DAME ESTOS DATOS: ";
a
1604 IF a<=0 THEN GO TO 1603
1605 INPUT "DAME ESTOS DATOS: ";
b
1606 IF b<=0 THEN GO TO 1605
1607 INPUT "DAME ESTOS DATOS: ";
w
1608 IF d=2 THEN RETURN
1609 IF a=b THEN LET v=w: LET r
=0: GO TO 1614
1610 INPUT "DAME LA RAZON: ";r
1611 IF (a<b AND r>0) OR (a>b AN
D r<0) THEN GO SUB 9100: GO TO
1603
1612 LET z=a-b: LET c=z*r: LET v
=w+c
1614 CLS : PRINT ": PRINT " El t
ermino que esta en ";a;" posi- c
ion es: ";v
1615 FOR f=1 TO a-1
1616 LET v=v-r
1617 NEXT f
1618 PRINT ": PRINT " El primer
termino de la progre-sion es: ";
v
1619 PRINT ": PRINT " DAME EL NU
MERO DE TERMINOS QUE TIENE LA PR
OGRESION Y TE DARE EL VALOR DE CA
DA UNO DE ELLOS. SI NO, METE ";
FLASH 1;"CERO"; FLASH 0;"."
1620 INPUT ;n
1621 IF n=0 THEN RUN 15
1622 IF n<0 THEN GO TO 1616

```

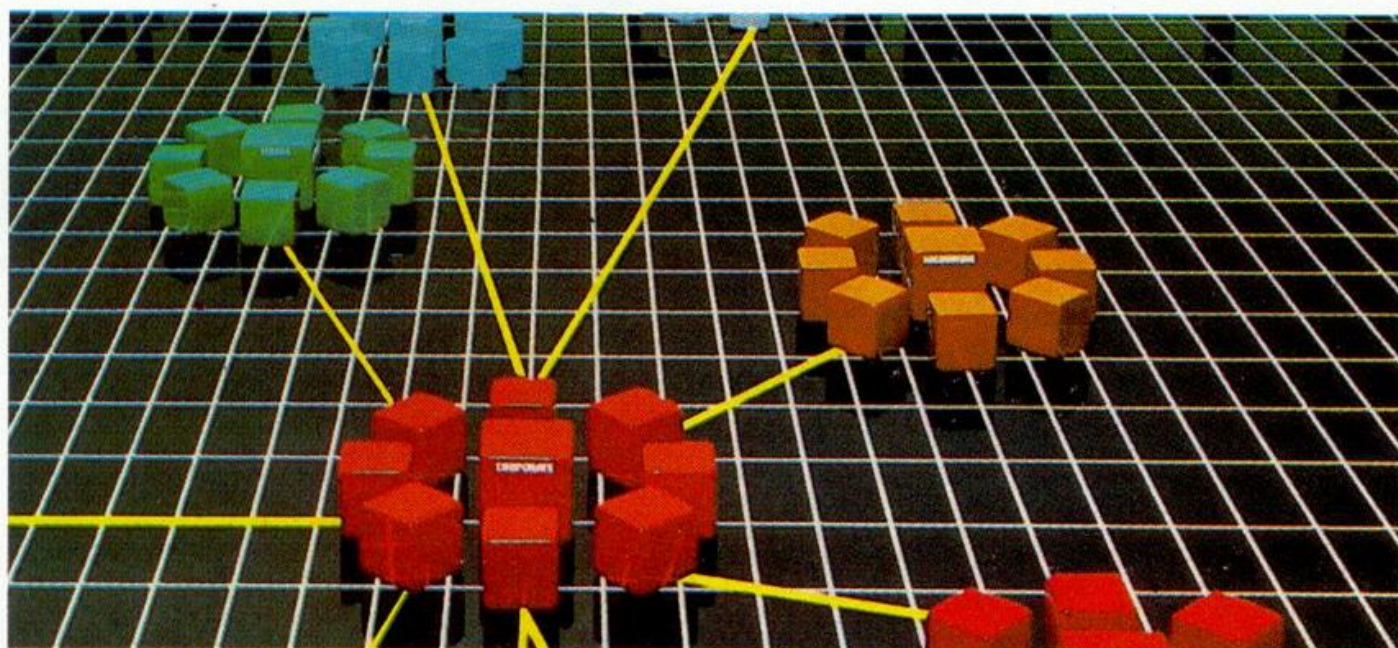
```

1623 CLS
1624 PRINT ": PRINT " TERMINO
VALOR ": PRINT AT
1,0; OVER 1;"
"; OVER 0: LET abc
=6: LET abd=26
1625 FOR q=1 TO n
1626 IF q>=10 THEN LET abc=5: I
F q>=100 THEN LET abc=4: IF q>=
1000 THEN LET abc=3: IF q>=1000
0 THEN LET abc=2
1627 IF v>=10 THEN LET abd=25:
IF v>=100 THEN LET abd=24: IF v
>=1000 THEN LET abd=23: IF v>=1
0000 THEN LET abd=22: IF v>=100
000 THEN LET abd=21: IF v>=1000
000 THEN LET abd=20
1629 PRINT ": PRINT TAB abc;q;TA
B abd;v
1630 LET v=v+r
1633 NEXT q
1635 PRINT #0;"PULSA UNA TECLA P
ARA CONTINUAR"
1640 PAUSE 0
1650 RUN 15
1700 CLS : PRINT " La formula qu
e vamos a utilizaren este caso,
va a ser:": PRINT ": PRINT ; FLA
SH 1;"n=2*S/(t(1)+t(n))"; FLASH
0;" siendo:"
1701 PRINT ": PRINT "n el numer
o de terminos.": PRINT ": PRINT
"S la suma de todos los terminos
de la progresion.": PRINT ": PR
INT "t(1) y t(n) los terminos ex
tre- mos y conocidos de la suces
ion."
1702 PRINT ": PRINT " INTRODUCE:
": PRINT ": PRINT "-EL EXTREMO I
NFERIOR t(1).": PRINT "-EL EXTRE
MO SUPERIOR t(n).": PRINT "-LA S
UMA DE TODOS LOS TERMINOS."
1705 INPUT "t(1) ";t1
1706 INPUT "t(n) ";tn
1707 INPUT "SUMA ";s
1708 IF (t1=s OR tn=s) AND t1<>t
n THEN GO SUB 9100: GO TO 1702
1709 IF s=0 AND ABS t1<>ABS tn T
HEN GO SUB 9100: GO TO 1702

```



# PROGRAMAS



```

1710 IF s=0 THEN GO TO 9000
1711 LET s=2*s: LET x=t1+tn: IF
x=0 THEN GO SUB 9100: GO TO 170
2
1712 LET s=s/x: IF s<>INT s THEN
GO TO 1705
1713 CLS: PRINT "El numero de e
lementos que tienetu progresion
es: ";s;" y la progre-sion es:"
PRINT ': PRINT " TERMINO
VALOR ": OVER 1: PRIN
T AT 4,0;"
-----
----- ": OVER 0

```

```

1714 LET abc=6: LET abd=26
1715 LET n=s-1: LET l=t1
1720 FOR f=1 TO s
1721 IF f>=10 THEN LET abc=5: I
F f>=100 THEN LET abc=4: IF f>=
1000 THEN LET abc=3: IF f>=1000
0 THEN LET abc=2
1723 IF t1>=10 THEN LET abd=25:
IF t1>=100 THEN LET abd=24: IF
t1>=1000 THEN LET abd=23: IF t
1>=10000 THEN LET abd=22: IF t1
>=100000 THEN LET abd=21: IF t1
>=1000000 THEN LET abd=20
1725 PRINT ': PRINT TAB abc;f;TA
B abd;t1
1726 IF n=0 THEN GO TO 1730
1727 LET t1=t1+(tn-1)/n

```

```

1730 NEXT f
1740 PRINT #0;"PULSA UNA TECLA P
ARA CONTINUAR"
1750 PAUSE 0
1760 RUN 15
1800 CLS: PRINT " La formula po
r la cual vamos a hallar la suma
de todos los ter-minos de una p
rogresion aritme- tica es:"
1801 PRINT ': PRINT " S=
(t1+tn) x n ": PRINT "
2
: PLOT 90,128: DRAW 52,0
1802 PRINT ': PRINT " t1 es el
primer termino de la progresion.
": PRINT ': PRINT " tn es el u
ltimo termino de la progresion."
: PRINT ': PRINT " n es el nume
ro de terminos."
1803 PRINT ': PRINT " INTRO
DUCE LOS DATOS ": PRINT "
POR ESTE ORDEN "
1804 INPUT "t1 ";t1
1805 INPUT "tn ";tn
1806 INPUT "n ";x
1807 IF x<=0 THEN GO TO 1806
1810 LET h=(t1+tn)/2: LET s=h*x
1813 LET l=t1: LET k=tn-1: LET c
=k/(x-1)
1814 IF c<>INT c THEN GO SUB 91
00: GO TO 1804

```



# PROGRAMAS

Tu progresion es:

	TERMINO	VALOR
	1	19.0139
79	2	22.8652
53	3	27.4966
	4	33.0660
26	5	39.7635
36	6	47.8176
25	7	57.5030

La formula que vamos a emplear para hallar la razon es:

$r = t(a) - t(b) / (a - b)$  siendo:

**t(a)** y **t(b)** dos terminos conocidos de la progresion.

**a** y **b** los lugares que estos ocupan en ella.

INTRODUCE EL LUGAR DEL PRIMER TERMINO CONOCIDO Y LUEGO SU VALOR, Y SEGUIDAMENTE HAZ LO MISMO CON EL OTRO TERMINO.

```
1815 CLS : PRINT " La suma de to
dos los terminos de la progresi
on cuyo primer e- lemento es ";t
1;" y ";tn;" es el ultimo,es:"
PRINT ': PRINT "
S
```

```
1820 PRINT ': PRINT "y la progre
sion es:" PRINT ': PRINT " TE
RMINO VALOR ": OV
ER 1: PRINT AT 9,0;"
```

": OVER 0

```
1821 LET abd=26
1825 FOR f=1 TO x
1827 IF t1>=10 THEN LET abd=25:
IF t1>=100 THEN LET abd=24: IF
t1>=1000 THEN LET abd=23: IF t
1>=10000 THEN LET abd=22: IF t1
>=100000 THEN LET abd=21: IF t1
>=1000000 THEN LET abd=20
```

```
1830 PRINT ': PRINT " t";f;t
AB abd;t1
```

```
1831 IF x=1 THEN GO TO 1840
```

```
1834 LET t1=t1+c
```

```
1835 NEXT f
```

```
1840 PRINT #0;"PULSA UNA TECLA P
ARA CONTINUAR"
```

```
1850 PAUSE 0
```

```
1860 RUN 15
```

```
2000 PRINT AT 2,3;"LA PROGRESION
ES DEL TIPO:" PRINT AT 4,3;"t(
1),t(2),t(3),t(n-1),t(n)": PRINT
```

AT 6,3;"siendo:"

```
2001 PRINT AT 8,1;"t(2)=t(1)xr":
PRINT AT 11,1;"t(3)=t(2)xr t(3
)=t(1)xr": PRINT AT 10,25;"(3-1)
": PRINT AT 14,1;".....
.....": PRINT AT 16,1
;"t(n)=t(n-1)xr": PRINT AT 19,8;
FLASH 1;"t(a)=t(b)xr ":AT 1
8,8;" (a-b)"
```

```
2002 PRINT #0;" PULSA UNA TECLA
PARA CONTINUAR "
```

```
2003 PAUSE 0
```

```
2004 GO SUB 20: GO SUB 2010: RET
URN
```

```
2010 PRINT AT 2,2;"LA SUMA DE TO
DOS LOS TERMINOS";AT 3,2;"DE LA
PROGRESION:";AT 5,2;"t(1),t(2),t
(3),t(n-1),t(n)":AT 7,2;"ES:";AT
9,2;"S=t(1)+t(2)+.....+t(n-1)+t
(n)":AT 11,2;"Y SUSTITUYENDO QUE
DA:"
```

```
2020 PRINT AT 13,10;"S=t(n)xr-t(
1)": PRINT AT 14,16;"r-1": PLOT
96,63: DRAW 86,0
```

```
2030 PRINT #0;"PULSA UNA TECLA P
ARA CONTINUAR"
```

```
2035 PAUSE 0
```

```
2040 RETURN
```

```
2500 CLS : PRINT " La formula qu
e vamos a utilizarva a ser:" PR
INT AT 3,7;"a-b": PRINT AT 4,10;
```



# PROGRAMAS

```

"t(a)": PRINT AT 5,10;"t(b)"
2501 PLOT 62,140: DRAW 10,-10: D
RAW 12,20: DRAW 26,0
2502 PLOT 81,135: DRAW 28,0
2503 PLOT 114,136: DRAW 5,0: PLO
T 114,134: DRAW 5,0
2504 PLOT 123,133: DRAW 0,3: PLO
T 124,137: DRAW 2,0
2505 PRINT "siendo:"
2510 GO SUB 1501
2511 IF a=b THEN LET l=1
2512 IF a<>b AND ta=tb THEN GO
TO 2500
2513 LET z=ta/tb: LET r=a-b: LET
l=z^(1/r)
2514 CLS
2516 FOR a=1 TO b-1
2517 LET tb=tb/l
2518 NEXT a
2519 PRINT " La solucion a la pr
ogresion cu-yos datos me has dad
o es:": PRINT ': PRINT " RAZON="
;l
2520 PRINT ': PRINT " El primer
termino de la progre-sion es: ";
tb
2521 PRINT ': PRINT " DAME EL NU
MERO DE TERMINOS DE LA PROGRESI
ON Y TE LA DARE COM- PLETA. PARA
EMPEZAR OTRA VEZ ME-TE "; FLASH
1;"CERO"; FLASH 0;". "
2522 INPUT n
2523 IF n<0 THEN GO TO 2522
2524 IF n=0 THEN RUN 15
2530 CLS : PRINT " Tu progresion
es:"
2533 PRINT ': PRINT " TERMINO
VALOR ": OVER 1:
PRINT AT 2,3;"-----";AT 2,24;"
-----": OVER 0: LET abc=6: LET a
bd=26
2534 FOR w=1 TO n
2536 IF w>=10 THEN LET abc=5: I
F w>=100 THEN LET abc=4: IF w>=
1000 THEN LET abc=3: IF w>=1000
0 THEN LET abc=2
2537 IF tb>=10 THEN LET abd=25:
IF tb>=100 THEN LET abd=24: IF

```

```

tb>=1000 THEN LET abd=23: IF t
b>=10000 THEN LET abd=22: IF tb
>=100000 THEN LET abd=21: IF tb
>=1000000 THEN LET abd=20
2538 PRINT ': PRINT TAB abc;w;TA
B abd;tb
2539 LET tb=tb*l
2540 NEXT w
2550 PRINT #0;"PULSA UNA TECLA P
ARA CONTINUAR."
2560 PAUSE 0
2570 RUN 15
2600 CLS : PRINT " La formula me
diante la cual va-mos a hallar d
icho termino es:": PRINT ': PRIN
T AT 3,19;"(a-b)": PRINT AT 4,8;
"t(a)=t(b)xr": PRINT AT 4,25;"si
endo:"
2601 GO SUB 1601
2610 IF a=b THEN LET r=1: GO TO
2630
2611 INPUT "DAME LA RAZON: ";r
2633 FOR f=1 TO b-1
2635 LET w=w/r
2640 NEXT f
2641 LET l=w
2642 FOR f=1 TO a-1
2644 LET w=w*r
2646 NEXT f
2650 CLS : PRINT " El termino qu
e esta en ";a;" posicion es: ";w
2660 PRINT ': PRINT " DAME EL NU
MERO DE TERMINOS Y TEDARE LA PRO
GRESION COMPLETA. SI NO METE CER
O."
2670 INPUT z
2680 IF z<0 THEN GO TO 2670
2685 IF z=0 THEN RUN 15
2686 LET abc=6: LET abd=26
2690 PRINT ': PRINT " TERMINO
VALOR ": OVER 1:
PRINT AT 7,0;"-----
-----": OVER 0
2691 FOR f=1 TO z
2692 IF f>=10 THEN LET abc=5: I
F f>=100 THEN LET abc=4: IF f>=
1000 THEN LET abc=3: IF f>=1000
0 THEN LET abc=2

```



# PROGRAMAS

```

2693 IF 1>=1 THEN LET abd=26: I
F 1>=10 THEN LET abd=25: IF 1>=
100 THEN LET abd=24: IF 1>=1000
THEN LET abd=23: IF 1>=10000 T
HEN LET abd=22: IF 1>=100000 TH
EN LET abd=21: IF 1>=1000000 TH
EN LET abd=20
2694 PRINT ': PRINT TAB abc;f;TA
B abd;l
2695 LET l=1*r
2696 NEXT f
2697 PRINT #0;"PULSA UNA TECLA P
ARA CONTINUAR."
2698 PAUSE 0
2699 RUN 15
2700 CLS
2701 PRINT " En esta ocasion vas
a introdu- cir los datos como s
igue:"
2702 PRINT ': PRINT "--PRIMERO: I
NTRODUCE EL LUGAR QUE OCUPA UN T
ERMINO CUALQUIERA EN LA PROGRES
ION Y LUEGO SU VALOR."
2703 PRINT ': PRINT "--SEGUNDO: I
NTRODUCE EL LUGAR QUE OCUPA CUAL
QUIER OTRO TERMINO DE LA PROGRES
ION Y SU VALOR."
2704 PRINT ': PRINT "--TERCERO: I
NTRODUCE LA SUMA DE TODOS LOS
TERMINOS PERTENECIEN- TES A LA P
ROGRESION."
2705 INPUT "DAME EL LUGAR Y EL V
ALOR DE UNO DE LOS TERMINOS: ";a
,ta
2706 IF a<=0 THEN GO TO 2705
2710 INPUT "DAME EL LUGAR Y EL V
ALOR DE OTRODE LOS TERMINOS: ";b
,tb
2711 IF b<=0 THEN GO TO 2710
2712 IF a=b AND ta<>tb THEN GO
TO 2790
2720 INPUT "DAME LA SUMA DE TODO
S LOS TERMI-NOS DE LA PROGRESION
: ";s
2721 LET z=tb/ta: LET x=b-a
2722 IF ta=tb THEN LET x=1
2723 LET r=z^(1/x)
2725 FOR d=1 TO a-1
2726 LET ta=ta/r
2730 NEXT d
2731 LET f=1
2735 LET y=0
2749 LET f=f+1
2750 LET y=r*y+1
2751 IF s=ta*y THEN GO TO 2756
2752 IF ABS s>ABS (ta*y) THEN G
O TO 2740
2753 IF s<>ta*y THEN GO TO 2790
2755 GO TO 2740
2756 CLS : PRINT " El primer ter
mino de la progre-sion es: ";ta;
"; y la razon: ";r;".
2757 PRINT '
2760 PRINT " El numero de termin
os que tienela progresion es: ";
f-1;" y estos son:"
2765 PRINT ': PRINT AT 7,3;"TERM
INO";AT 7,25;"VALOR": OVER 1: PR
INT AT 7,3;"-----";AT 7,25;"__
--"
2770 FOR q=1 TO f-1
2771 IF q>=1 THEN LET abc=6: IF
q>=10 THEN LET abc=5: IF q>=10
0 THEN LET abc=4
2772 IF ta>=1 THEN LET abd=27:
IF ta>=10 THEN LET abd=26: IF t
a>=100 THEN LET abd=25: IF ta>=
1000 THEN LET abd=24: IF ta>=10
000 THEN LET abd=23: IF ta>=100
000 THEN LET abd=22
2775 PRINT ': PRINT TAB abc;q;TA
B abd;ta
2777 LET ta=ta*r
2780 NEXT q
2785 PRINT #0;"PULSA UNA TECLA P
ARA CONTINUAR"
2788 PAUSE 0
2789 RUN 15
2790 CLS : PRINT " Has introduci
do mal algun dato. Repite la ope
racion.": GO TO 2702
2800 CLS

```



# PROGRAMAS

```
2801 PRINT " La formula mediante
la cual va-mos a calcular la su
ma de todos los terminos de la p
rogresion vaa ser:"
```

```
2802 PRINT ': PRINT "S=t(1)+t(2)
+t(3)+...+t(n-1)+t(n)"
```

```
2803 PRINT ': PRINT " De donde,
despejando, queda:" : PRINT ': PR
INT "                2          n-1
```

```
": PRINT "          S=t(1)x(1+r+r
+...+r          )          "
```

```
2804 PRINT ': PRINT " DAME EL PR
IMER TERMINO DE LA PROGRESION,
LA RAZON Y DESPUES EL NUMERO D
E TERMINOS."
```

```
2805 INPUT "PRIMER TERMINO: ";ta
```

```
2810 INPUT "RAZON: ";r
```

```
2820 INPUT "NUMERO DE TERMINOS:
```

```
";n
```

```
2823 IF n<=0 THEN GO TO 2820
```

```
2825 LET y=0
```

```
2830 FOR x=1 TO n
```

```
2835 LET y=r*y+1
```

```
2840 NEXT x
```

```
2845 LET s=ta*y
```

```
2850 PRINT ': PRINT " La suma de
todos los terminos de dicha pr
ogresion es: ";s;" y los termino
s son: "
```

```
2855 PRINT ': PRINT TAB 3;"TERMI
NO";TAB 25;"VALOR": OVER 1: PRIN
T AT 20,3;"-----";AT 20,25;"__
```

```
---"
```

```
2860 FOR x=1 TO n
```

```
2861 IF x>=1 THEN LET abc=6: IF
x>=10 THEN LET abc=5: IF x>=10
0 THEN LET abc=4
```

```
2862 IF ta>=1 THEN LET abd=27:
IF ta>=10 THEN LET abd=26: IF t
a>=100 THEN LET abd=25: IF ta>=
1000 THEN LET abd=24: IF ta>=10
000 THEN LET abd=23: IF ta>=100
000 THEN LET abd=22
```

```
2865 PRINT ': PRINT TAB abc;x;TA
B abd;ta
```

```
2866 LET ta=ta*r
```

```
2870 NEXT x
```

```
2875 PRINT #0;"PULSA UNA TECLA P
ARA CONTINUAR"
```

```
2880 PAUSE 0
```

```
2885 RUN 15
```

```
9000 CLS : PRINT " La solucion c
on ese dato es in-determinada. L
o mismo puede ha- ber n, que n-1
, que n-2, que n+nelementos en l
a progresion.": INVERSE 1: PRINT
': PRINT "          EJEMPLO:
```

```
"
```

```
9001 PRINT ': INVERSE 0: PRINT "
PROGRESION 1          PROGRESION 2
": PRINT ': PRINT "TERMINO VALO
R          TERMINO VALOR": PRINT AT 9
,0; OVER 1;"-----"  --
```

```
"
```

```
9002 FOR p=1 TO 3
```

```
9003 PRINT ': PRINT TAB 1;"t";p
```

```
9004 NEXT p
```

```
9005 LET k=1
```

```
9006 FOR p=1 TO 5
```

```
9007 PRINT AT 10+k,19;"t";p: LET
k=k+2
```

```
9008 NEXT p
```

```
9009 PRINT AT 11,9;t1;AT 11,27;t
1
```

```
9010 PRINT AT 13,9;"0": LET k=t1
/2: PRINT AT 13,27;k
```

```
9011 PRINT AT 15,9;tn: PRINT AT
15,27;"0"
```

```
9012 PRINT AT 17,27;-k: PRINT AT
19,27;tn
```

```
9040 PRINT #0;" PULSA UNA TECLA
PARA CONTINUAR "
```

```
9050 PAUSE 0
```

```
9060 RUN 15
```

```
9100 CLS : PRINT " Has introduci
do mal los datos. No es posible
ninguna progresionaritmetica con
ellos.": PRINT '': PRINT FLAS
H 1;"          REPITE LA OPERACION
```

```
": RETURN
```





# APARTADO DE CORREOS

Dirige tus cartas a:  
Todospectrum  
Bravo Murillo, 377, 5.º-A  
28020 Madrid

## BASIC DESDE CODIGO MAQUINA, PASANDO POR REUS

Ante todo felicitaros por la revista, de la que cada vez estoy más contento, y en particular por los últimos números, que me parece que han cambiado y para bien; pero bueno, esto os lo deben decir todos.

Iré al grano, he realizado el programa titulado «BASIC DESDE EL CODIGO MAQUINA» que creo que es muy interesante, ya que hay ciertas operaciones que desde este último son muy engorrosas. El programa funciona bien sólo en parte, de pronto se detiene con el mensaje «Program finished». Creo que el motivo es el RST 8. Si analizamos el programa vemos que empieza con el registro HL apuntando a la dirección del texto y luego hace una llamada a la subrutina llamada «CODIGO» que es la que se encarga de realizar el trabajo, y cuando estamos de vuelta se realiza el RST 8. Esta es la rutina de error del ZX, que después de ejecutar sus instrucciones retorna a la dirección señalada indirectamente por (ERR\_SP),

normalmente MAIN\_4, termina saltando al editor BASIC. Yo he solucionado el problema eliminando las líneas 40 y 50, y sustituyéndolas simplemente por un RET.

Espero efectuéis la correspondiente corrección en Todospectrum. Sólo me queda saludaros y desearos prosperidad en esta vuestra revista.

Angel Olivart  
Reus (Tarragona)

Gracias por los cumplidos. El problema que has detectado se debe a que el autor del programa incluyó en el listado unas líneas de demostración para que pudiera ser llamado directamente desde el BASIC sin problemas. Puede ser perfectamente utilizado para nuestros propios fines usando exclusivamente las líneas 60 - 300, es decir, la rutina «CODIGO». Eso sí, antes de llamar esta rutina hay que cargar en HL la dirección de inicio de la tabla donde se encuentre el texto BASIC almacenado, con un 13 (ENTER) al final. Por esto, si la llamáramos directamente desde el BASIC, daría problemas al tener HL un valor no predecible.

## PROTECCION DE FICHEROS

Poseo un Spectrum Plus con dos unidades de microdrive y tengo confeccionado un programa de ficheros para dichas unidades. El programa, en BASIC, consiste en la creación de un índice para todos los artículos de revistas y libros que poseo, manejando a un mismo tiempo varias variables de cadena (revista, número, página, contenido, etc.) e incluyendo opciones de creación, lectura, ampliación y borrado de ficheros, así como catálogo de los mismos. En el número 8 de su revista apareció un programa con el título «Una clave, please»; el cual he querido incorporar al programa confeccionado por mí sin poder lograr mi objetivo.

¿Cómo podría incorporar el programa «Una clave, please» para proteger el acceso a mi programa o para no poder leer los datos escritos en el fichero sin introducir la clave preestablecida?; lo he intentado varias veces según las instrucciones aparecidas y no lo consigo, pues no comprendo del todo los apartados (del 1 al 9) que vds. editan ni las imputaciones (Ramtop y direcciones) que solicita dicho programa. ¿Cómo podría actualizar los ficheros empleando las dos unidades de microdrive y utilizando la sentencia MOVE?; pienso que este método puede ser más rápido y eficaz que el adoptado por mí.

Fernando Clavijo  
Málaga

Si realmente quieres proteger tu programa de la curiosidad de la gente, de poco te valdrá la rutina a la que haces referencia, pues, aunque es evidente que una vez se esté ejecutando impedirá el continuar a quien no conozca la clave correspondiente, no es menos cierto que resultará totalmente inútil desde el momento en que alguien rompa la ejecución del cargador antes de que se salte a esta rutina, con



lo que, tras suprimir la llamada correspondiente, el supuesto pirata tendría vía libre para acceder a tus ficheros.

Existen, sin embargo, determinados métodos, bastante clásicos por otra parte, que impedirán el acceso a tus programas a quien no tenga ciertos conocimientos de código máquina ni sepa la contraseña BASIC en forma de bytes, incluídas las variables del sistema, de modo que tras cargarlo comience la ejecución en determinada línea del programa (sin que pueda evitarse con MERGE la autoejecución), donde pueda pedirse la contraseña al amparo de determinada protección anti-BREAK.

Para usar este sistema con tu programa deberás hacer NEW, cargar el programa, insertar en él las siguientes líneas 9500-9550 y salvarlo a cinta con GOTO 9500. Cuando sea cargado (con LOAD "CODE") se autoejecutará en la línea 9520, pidiéndonos la clave (en el ejemplo «Patatita») y haciendo RUN sólo en el caso de que la clave sea la correcta.

```
9500 POKE 23613,0
9510 SAVE "nombre" CODE
23613,PEEK 23641+256*PEEK
23642-23613
9520 IF AS<>"Patatita"
THEN STOP
9540 POKE 23613,84
9550 RUN
```

En cuanto al uso que pretendes dar a MOVE para la actualización de tus ficheros, no nos ha quedado muy clara tu pregunta. En cualquier caso, te podemos decir que la sintaxis de esta instrucción es MOVE #a TO #b, donde "a" y "b" son los números de las dos corrientes entre las que va a efectuarse la operación. Lo que hace es leer un carácter de "a" e imprimirlo en "b", repitiendo esta operación hasta que se cumpla una determinada condición (p.e. fin de fichero) que depende del tipo de canal a que esté asociada la corriente "a".

## ERROR EN EL QL ESPAÑOL

Existe una clara errata en el firmware del QL (mejor dicho, sólo en el modelo español), precisamente en la instrucción POINT. Esta instrucción debería iluminar un solo punto en la pantalla gráfica, pero en lugar de esto ilumina dos puntos en diagonal. ¿Existe algún método para evitar esto?, si es así, ¿podrían facilitármelo?

¿Hay algún algoritmo o método para simular en el QL una instrucción semejante a la instrucción POINT del ZX Spectrum o la instrucción TEST del Amstrad, que informa si un punto está iluminado o el color que posee éste respectivamente?

He elaborado una serie de programas para el QL que me interesaría mandaros. Mi pregunta es: ¿es necesario que os los mande grabados en microdrives, u os basta con un listado de los mismos?, lo pregunto por que los cartuchos no son precisamente muy baratos.

Sin más que darles las gracias por su atención se despide:

Juan I. Rodríguez  
Castilleja de la Cuesta  
(Sevilla)

En cuanto a la instrucción POINT y al hipotético TEST, lo mejor será que te construyas un par de rutinas para arreglar la primera y hacer realidad la segunda. Aunque lo ideal sería usar código máquina para ello, también puedes hacerlo desde el BASIC mediante PEEKs y POKEs. Por ello pasamos a explicarte la forma en que están «mapeadas» las pantallas en la memoria del QL en sus dos modos gráficos.

El QL utiliza para almacenar la pantalla en memoria nada menos que 32 ks, entre las direcciones 131072 y 163839 (20000 y 27FFF hexadecimal) en ambos modos de presentación. Los puntos quedan distribuidos de izquierda a derecha y de arriba a abajo como sigue:

En el modo de alta resolución a cada ocho puntos le correspon-

den dos bytes de memoria (a los ocho primeros puntos les corresponden, pues, las direcciones 131072 y 130173), distribuidos a su vez de la siguiente forma: a cada punto le corresponde un bit de cada uno de los dos bytes, de forma que al primer punto le corresponde el bit 7 de la dirección 130172 y también el bit 7 de la 130173, al segundo punto el bit 6 de las mismas direcciones, hasta llegar al punto noveno, al que le corresponderán el bit 7 de las direcciones 130174-5, y así sucesivamente. Los bits de las direcciones pares indican si está activado el color verde, mientras que los impares indican si lo está el rojo. Si lo están ambos, el color que aparece es el blanco, mientras que si los dos están desactivados (valen cero) el color resultante (en realidad, la ausencia de color) es el negro.

En el modo de baja resolución (ochó colores) el sistema es muy parecido, pero a cada dos bytes sólo le corresponden cuatro puntos. En este caso, a cada punto se le asignan cuatro bits, dos de cada byte de los que le toquen. Así al primer punto le corresponden los bits 7 y 6 de la dirección 130172 y también el 7 y el 6 de 130173, al segundo punto los bits 5 y 4 de las direcciones 130174-5, etc., etc. En este caso podemos formar, con el bit de la izquierda de los dos correspondientes a la dirección par y los dos de la impar, un número binario que nos indicará el color utilizado (0=negro, 1=azul, 2=rojo... 7=blanco), mientras que el bit restante indica si está o no activado el parpadeo o flash.

Aunque, como tú dices, los cartuchos no son muy baratos, tampoco puede decirse que sean excesivamente caros; puedes utilizar uno ya usado, si tus programas en verdad valen la pena serás generosamente recompensado.



**Intercambio o vendo** software para Spectrum 48 K. Amplia lista con las últimas novedades. Poseo copión turbo. Interesado en conseguir el compilador «The Colt» con instrucciones. Escribir a: Bernardo Pou Fernández. Avda. Manuel del Palacio, 5, 4.º C. Tel.: (968) 85 67 99. 36003 Pontevedra.

**Por cambio de ordenador** vendo: colección completa de revistas, cintas con programas comerciales, libros, magnetofón Gold King, así como disco más interface para el Sinclair Spectrum. Envío relación. Tel.: (958) 27 24 93. Preguntar por Jose.

**Deseo contactar** con usuarios del QL para intercambio de artículos de prensa, noticias o ideas, así como programas. Juan R. García Martell. Dr. Juan de Padilla, 42. Tel.: (928) 31 20 90/36 60 04 (tardes). 35002 Las Palmas de Gran Canaria.

**Estoy interesado** en entrar en contacto con usuarios de ZX-Spectrum. Escribir a Mario Sáenz de Santamaría. Río Ebro, 27, 7.º C. Miranda de Ebro (Burgos).

**Intercambio programas** de todo tipo para Spectrum o Amstrad. A. J. Rodríguez Salas. Nervión, 8, 1.º B. 18015 Granada.

**Vendo procesador** de textos especial para la GP-50-S o compatibles. Permite la impresión de textos en 64 columnas sin reducción de caracteres ni modificaciones en el hardware. Manuel Cagiao. Apartado 2144. 15080 La Coruña. Tel.: (981) 78 29 52 (20 h.).

**Vendo/cambio programas** último grito para Spectrum y también ordenador ZX-Spectrum Plus a estrenar, con todos los accesorios. Sin usar y con garantía por seis meses. 30.000 ptas. Iglesias, Troitosende, Novigilde, Santiago de Compostela (La Coruña).





# Catálogo de Software para ordenadores personales IBM



Todo el Software disponible en el mercado reunido en un catálogo de 800 fichas

1.ª ENTREGA  
**550** FICHAS  
+ FICHERO

Resto en dos entregas  
trimestrales de 150 fichas  
cada una

**OFERTA  
ESPECIAL DE  
SUSCRIPCION  
8.000 PTAS.  
(IVA INCLUIDO)**

**PRECIO TOTAL DE LA SUSCRIPCION 8.000 PTAS.**

COPIE O RECORTE ESTE CUPON DE PEDIDO

## CUPON DE PEDIDO

SOLICITE HOY MISMO EL  
CATALOGO DE SOFTWARE A:

**infodis, s.a.**

Bravo Murillo, 377, 5.º A  
28020 MADRID

O EN CONCESIONARIOS IBM

El importe lo abonaré POR CHEQUE ☐ CONTRA REEMBOLSO ☐ CON MI  
TARJETA DE CREDITO ☐

Cargue 8.000 ptas. a mi tarjeta American Express ☐ Visa ☐ Interbank ☐

Número de mi tarjeta

NOMBRE

CALLE

CIUDAD  C. P.

PROVINCIA  TELEFONO

ref: CATALOGO DE SOFTWARE

CS-2





# COMIENZA LA AVENTURA



DESDE  
LOS 16 AÑOS

## DT-80

Audaz. Segura. Una auténtica "trail". Ligera como una gacela. Fuerte como el león. Para dominar tanto a la jungla urbana, como a las dunas de Dakar. Su línea sabe de aventuras, de horizontes abiertos. Siente en tu rostro el azote del viento de la libertad. Descubre sus prestaciones. Y no pongas límites a tu independencia.

**MOTUL**  
MOTOR OIL



**YAMAHA VA POR DELANTE**