

Rutinas

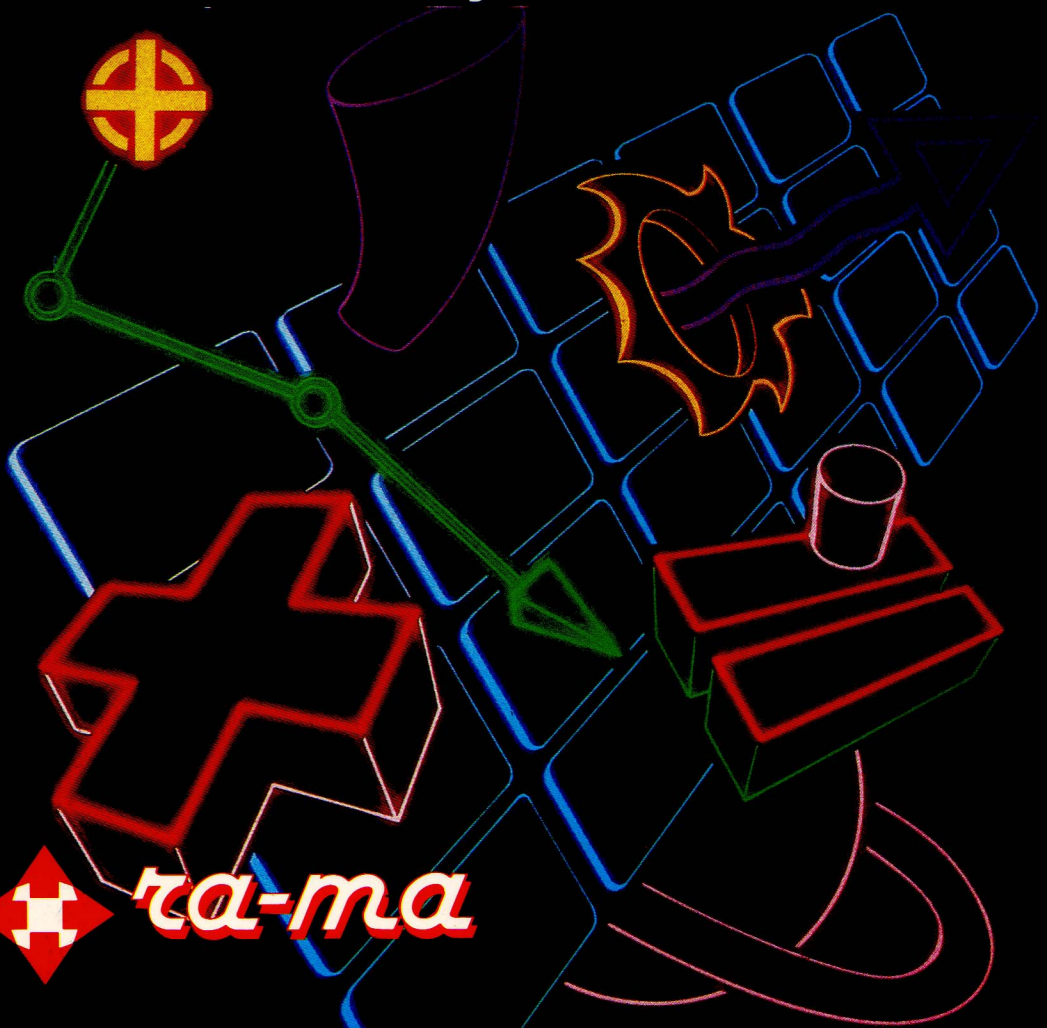
EN CODIGO

MAQUINA

para su

Amstrad

Clive Gifford y Scott Vincent

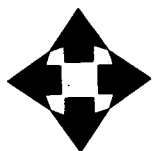


Rutinas

**EN CODIGO
MAQUINA**

para su
Amstrad

Clive Gifford y Scott Vincent



ta-ma

Título de la obra original

MACHINE CODE
ROUTINES FOR YOUR AMSTRAD

Publicado en Gran Bretaña en 1985
por Virgin Books Ltd, 328 Kensal Road,
London W10 5XJ.

Copyright © Scott Vicent y Clive Gifford 1985

ISBN 0863690315

Edición en Español

RUTINAS EN CODIGO MAQUINA
PARA SU AMSTRAD

© 1986 RAMA

Editado por RAMA
Ctra. de Canillas, 144
28043 MADRID
ESPAÑA

Reservados todos los derechos en lengua española.
No está permitida la reproducción parcial o total
de este libro sin consentimiento por escrito del
editor.

Consultas referentes al libro
RAMA, Ctra. Canillas, 144 - 28043 Madrid -
Teléfono: (91) 200.97.46/47

Traducción y Composición: CONORG, S.A.

I.S.B.N.: 84-86381-12-6
Depósito Legal: M. 17824 - 1986

Contenido

Introducción de los autores	5
Reposicionador de Texto	7
RSX Sound (Sonido RSX)	10
Carácter ampliado por 8	16
Carácter ampliado por 24	19
Lector de Cabeceras	23
Impresión de Cuatro Caracteres	27
Inversor de Pantalla	30
Copiador de Texto	34
Rotación a la Derecha	37
Rotación a la Izquierda	41
Fijar Velocidad	44
Desplazamiento Rápido de la Pantalla	46
La Pista de los Diamantes	48
Desplazamiento de Bloques	50
Rejilla de Diseño	52
Explosión Gráfica	55
Analizador de Pantalla	61
Impresión de Nueve Caracteres	63
Itálicas	66
Letras Inclinas	69
Copiador Inteligente	72
Memorizador/Recuperador de Pantallas	74
Utilidades Gráficas	76
OUTs, PEEKs, POKEs, y CALLs útiles	88
Ensambladores y Lenguajes Ensamblador	89
Bibliografía	90
Disposición del Teclado	91

Sobre los autores

Scott Vincent es un programador experimentado en Código Máquina. Habitualmente estudia ordenadores en la Universidad de Sheffield; Scott tiene publicados gran cantidad de libros sobre Código Máquina por conocidas casas de software. Gran parte de los libros que ha escrito, han sido publicados por Interface y Virgin Books.

Clive Gifford tiene aproximadamente 20 libros editados y escribe con frecuencia para un número de revistas de ordenadores, incluyendo **Home Computing Weekly**. Sus títulos publicados incluyen **The Amstrad Programmers Reference Guide** (con Tim Hartnell como co-autor), **Using Computers in Education** y **On-Line With Your Computer**.

Unidos, los dos autores han trabajado en varios proyectos, incluidos bastantes libros de la serie **Games for Your...** publicados por Virgin.

Los autores desean agradecer su colaboración a las siguientes personas: Dave, Dickie, Grannis, Lorraine, Simes y Stuart. También agradecen a Alan y Sue Baggs su ayuda con el préstamo de equipos, así como a John Brown por su acuerdo con todo el proyecto.

Introducción de los autores

Debido a su relación prestaciones/precio, la gama de ordenadores AMSTRAD no tiene igual en el mundo del **Home Computer**. El BASIC suministrado con los equipos AMSTRAD está escrito por Locomotive Software y es excelente, pero no hay BASIC que pueda igualar la velocidad de un programa escrito en Código Máquina. Este lenguaje tiene también otras ventajas. Un programa bien escrito en Código Máquina ocupa mucho menos espacio en la memoria; aún más, pueden escribirse rutinas para ejecutar tareas imposibles de realizar a través del BASIC -como lo demuestra la disponibilidad del sintetizador de voz y nuestra velocidad de cassette más rápida.

Sin embargo, el Código Máquina tiene desventajas. Es tedioso y lento de escribir, no tiene facilidades para comprobar errores (en caso de error en el programa, perderemos el control del mismo) y, por supuesto, para escribir cualquier rutina es necesario estar familiarizado con él. Para muchas personas esto es demasiado complejo y el Código Máquina semeja los vestigios de un dios, demasiado difícil de aprender.

Este libro es para aquellas personas que son incapaces de escribir Código Máquina pero que escriben programas en BASIC. Hemos tratado de escribir rutinas de todo tipo: gráficos, sonido, conservación de memoria, recuperación más rápida desde el cassette y utilidades. Todo ha sido suministrado en la forma de un cargador BASIC, una lista en Ensamblador y una descripción detallada del uso, y en algunos casos, un programa de demostración.

Esperamos que este libro sea una ayuda para aquellos programadores que todavía no se han introducido dentro del complicado Código Máquina, o para aquellos que, habiéndolo hecho, han tenido problemas o no han realizado las mismas rutinas que aquí les ofrecemos.

Feliz teclado.

Clive Gifford
Scott Vincent

Londres, 1985

Reposicionador de texto

Esta subrutina cambia una cadena de caracteres de la posición superior a la inferior. El programa utiliza los primeros caracteres gráficos definibles por el usuario (User-Definable Graphic, UDG) por lo que habrá que tener en cuenta dos puntos.

- 1) . Asegurarse que al menos hay un carácter UDG disponible; el número disponible puede ser cambiado con el comando SYMBOL AFTER.
- 2) Asegurarse de que no se desea utilizar el primer carácter UDG disponible; si Vd. desea emplearlo, haga entonces SYMBOL AFTER, 1 más bajo.

El formato para llamar a esta rutina es el siguiente:

CALL 40000, X, Y, @A\$

X e Y son las coordenadas donde será impreso el primer carácter. Como la cadena será invertida, la impresión de la misma se hará en la pantalla de derecha a izquierda. Esto debe tenerse en cuenta al especificar las coordenadas. X e Y pueden ser cualquier integer o variable numérica, siempre que estén dentro de los márgenes de la pantalla. Según lo anterior **CALL 40000, a, b, @A\$** tiene el mismo efecto que **CALL 40000, 5, 9, @p\$**.

La cadena conteniendo los caracteres, puede ser cualquier variable alfanumérica (string = variable) pero no una serie de caracteres encerrados entre comillas ("). Por ejemplo, **CALL 40000, X, Y, "HELLO", @A\$** no trabajaría, pero esto difícilmente es un problema. **A\$ = "HELLO": CALL X, Y, @A\$** tendría el efecto deseado.

El signo @ permite a la rutina pasar una variable alfanumérica desde y hacia **Código Máquina**, siendo una característica muy útil para el programador que se inicia en esta modalidad. Esta rutina trabajará en los tres modos de **screen** (pantalla) que tiene Amstrad.


```

1 / REPOSICIONADOR DE TEXTO
10 SYMBOL AFTER 256:MEMORY 39999:SYMBOL
AFTER 240
20 FOR n=40000 TO 40076
30 READ a$:POKE n,VAL('%'+a$)
40 NEXT
50 DATA 0D,6E,00,0D,66,01,7E,B7,C8,23
60 DATA 5E,23,56,0D,6E,02,0D,66,04,47
70 DATA C5,E5,CD,75,BB,1A,CD,64,9C,13
80 DATA E1,25,C1,10,F1,C9,D5,CD,A5,BB
90 DATA EB,CD,AE,BB,06,07,23,10,FD,F5
100 DATA CD,06,B9,0E,08,1A,06,08,1F,CB
110 DATA 16,10,FB,13,2B,0D,20,F3,CD,09
120 DATA B9,F1,CD,5A,BB,D1,C9

```

```

Editor Assembler
AMMAS 1.1
Copyright 1985 PICTURESQUE

```

```

          0010 : REPOS
          0020 :
9C40      0030      ORG 40000
BB75      0040  CURSOR DEFL 0BB75H
9C40  DD6E00  0050      LD L,(IX+0)
9C43  DD6601  0060      LD H,(IX+1)
9C46  7E      0070      LD A,(HL)
9C47  B7      0080      OR A
9C48  C8      0090      RET Z
9C49  23      0100      INC HL
9C4A  5E      0110      LD E,(HL)
9C4B  23      0120      INC HL
9C4C  56      0130      LD D,(HL)
9C4D  DD6E02  0140      LD L,(IX+2)
9C50  DD6604  0150      LD H,(IX+4)
9C53  47      0160      LD B,A
9C54  C5      0170  NXTCHR PUSH BC
9C55  E5      0180      PUSH HL
9C56  CD75BB  0190      CALL CURSOR
9C59  1A      0200      LD A,(DE)
9C5A  CD649C  0210      CALL INVCHR
9C5D  13      0220      INC DE
9C5E  E1      0230      POP HL
9C5F  25      0240      DEC H

```

9C60	C1	0250	POP	BC
9C61	10F1	0260	DJNZ	NXTCHR
9C63	C9	0270	RET	
9C64	D5	0280	INVCHR	PUSH DE
BBA5		0290	MATRIX	DEFL 0BBA5H
BBAE		0300	TABLE	DEFL 0BBAEH
B906		0310	ROMENA	DEFL 0B906H
B909		0320	ROMDIS	DEFL 0B909H
BB5A		0330	TXTOUT	DEFL 0BB5AH
9C65	CDA5BB	0340	CALL	MATRIX
9C68	EB	0350	EX	DE,HL
9C69	CDAEBB	0360	CALL	TABLE
9C6C	0607	0370	LD	B,7
9C6E	23	0380	ADD7	INC HL
9C6F	10FD	0390	DJNZ	ADD7
9C71	F5	0400	PUSH	AF
9C72	CD06B9	0410	CALL	ROMENA
9C75	0E08	0420	LD	C,8
9C77	1A	0430	NXTBYT	LD A,(DE)
9C78	0608	0440	LD	B,8
9C7A	1F	0450	NXTBIT	RRA
9C7B	CB16	0460	RL	(HL)
9C7D	10FB	0470	DJNZ	NXTBIT
9C7F	13	0480	INC	DE
9C80	2B	0490	DEC	HL
9C81	0D	0500	DEC	C
9C82	20F3	0510	JR	NZ,NXTBYT
9C84	CD09B9	0520	CALL	ROMDIS
9C87	F1	0530	POP	AF
9C88	CD5ABB	0540	CALL	TXTOUT
9C8B	D1	0550	POP	DE
9C8C	C9	0560	RET	
		0570	END	

RSX Sound (Sonido RSX)

Este sorprendente programa te proporciona un conjunto de comandos similares a los que pueden encontrarse en el ordenador Oric. Para aquéllos que no estén familiarizados con las particularidades del Oric (que pueden ser el 99 por ciento de los lectores), diremos que una de sus excelentes características era la predefinición de sonidos. Si escribías en BASIC el comando **ZAP**, obtenías un sonido "ZAP"; asimismo, si tecleabas **EXPLODE**, obtenías el sonido más real de una explosión. Obviamente, estas facilidades ahorran al programador gran cantidad de tiempo. Muchos de los listados de Oric hacen uso de estos útiles comandos.

Nuestra rutina crea para el Amstrad unas posibilidades similares. Seis sonidos (dos más que en el Oric) han sido predefinidos, todos los cuales consideramos serán muy útiles; principalmente en programas de juegos. Después de rodar el programa cargador en BASIC, deberás ejecutar **CALL 40000** para inicializar la rutina.

A partir de ese momento, dispondrás de seis nuevos comandos en el BASIC. Estos han sido creados con la ayuda de la **ROM** que tiene nuestro Amstrad, la cual permite que puedan crearse nuevas palabras (en el BASIC) mediante una rutina conocida como **RSX**. Estos seis comandos son:

```
| EXPLODE  
| PING  
| SQUELCH  
| ZAP  
| ALIEN  
| SHOOT
```

La barra vertical que precede a los nombres, es parte de las características de **RSX** y es el carácter localizado en la misma tecla que '@' (a la derecha de la 'P').

Feliz "zapping", "squelching", y demás.

```
1      RSX SOUND  
10 SYMBOL AFTER 256:MEMORY 39999:SYMBOL  
AFTER 240
```

```

20 FOR n=40000 TO 40315
30 READ a$:POKE n,VAL('%'+a$)
40 NEXT
50 DATA 01,4A,9C,21,5E,9C,CD,D1,BC,C9
60 DATA 62,9C,C8,82,9C,C8,A1,9C,C8,C0
70 DATA 9C,C8,DF,9C,C8,1B,9D,C8,4C,9D
80 DATA 00,00,00,00,45,58,50,4C,4F,44
90 DATA C5,53,51,55,45,4C,43,C8,50,49
100 DATA 4E,C7,5A,41,D0,41,4C,49,45,CE
110 DATA 53,48,4F,4F,D4,00,CD,A7,BC,3E
120 DATA 01,21,94,9C,CD,BC,BC,21,98,9C
130 DATA CD,AA,BC,C9,01,0F,FF,19,01,01
140 DATA 00,00,00,0F,0F,00,00,CD,A7,BC
150 DATA 06,0F,C5,78,32,BC,9C,21,B7,9C
160 DATA CD,AA,BC,30,FB,C1,10,F0,C9,01
170 DATA 01,00,00,00,00,0F,02,00,CD,A7
180 DATA BC,3E,01,21,D2,9C,CD,BC,BC,21
190 DATA D6,9C,CD,AA,BC,C9,01,0F,FF,05
200 DATA 01,01,00,19,00,00,0F,00,00,CD
210 DATA A7,BC,3E,01,21,05,9D,CD,BF,BC
220 DATA 21,09,9D,CD,AA,BC,06,0F,C5,3E
230 DATA 10,90,32,17,9D,21,12,9D,CD,AA
240 DATA BC,30,FB,C1,10,EE,C9,01,0F,04
250 DATA 02,01,00,01,28,00,00,06,1E,00
260 DATA 02,00,00,00,00,00,0F,02,00,CD
270 DATA A7,BC,3E,FF,21,33,9D,CD,BF,BC
280 DATA 21,3A,9D,CD,AA,BC,21,43,9D,CD
290 DATA AA,BC,C9,02,05,FF,02,05,01,02
300 DATA 01,00,01,64,00,00,07,64,00,02
310 DATA 00,01,6C,00,00,07,64,00,CD,A7
320 DATA BC,3E,01,21,66,9D,CD,BC,BC,21
330 DATA 6A,9D,CD,AA,BC,21,73,9D,CD,AA
340 DATA BC,30,FB,C9,01,0F,FF,02,01,00
350 DATA 00,00,00,0A,0F,04,00,01,01,00
360 DATA 00,00,05,0F,00,00

```

Editor Assembler
 AMMAS 1.1
 Copyright 1985 PICTURESQUE

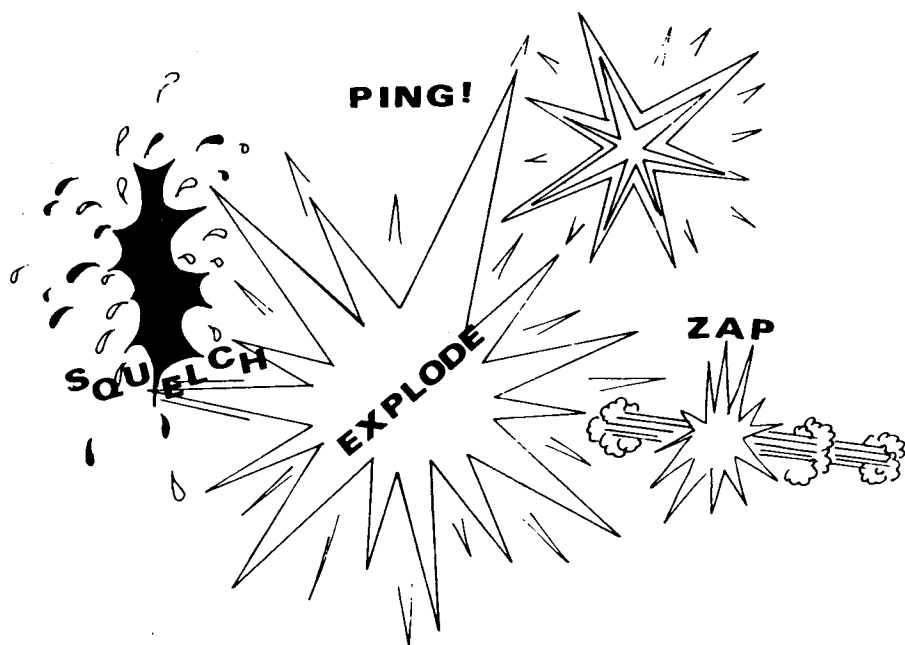
```

                                0001 :      RSX  SOUND
                                0002 :
0C40                            0010          ORG  40000
BCA7                            0020  SNDRES  DEFL  @BCA7H
BCBC                            0030  AMPENV  DEFL  @BCBCH
BCBF                            0040  TONENV  DEFL  @BCBFH
BCAA                            0050  ADDSND  DEFL  @BCAAH
BCD1                            0060  LOGEXT  DEFL  @BCD1H
0C40 014A9C                    0070          LD   BC, TABLE
0C43 215E9C                    0080          LD   HL, SPACE
0C46 CDD1BC                    0090          CALL LOGEXT
0C49 C9                          0100          RET
0C4A 629C                      0110  TABLE  DEFW  NAMETB
0C4C C3829C                    0120          JP   EXPLOD
0C4F C3A19C                    0130          JP   SQUELC
0C52 C3C09C                    0140          JP   PING
0C55 C3DF9C                    0150          JP   ZAP
0C58 C31B9D                    0160          JP   ALIEN
0C5B C34C9D                    0170          JP   SHOOT
0C5E 00                        0180  SPACE  DEFB  0,0,0,0
                                00 00 00
0C62                            0190  NAMETB  DEFM  'EXPLOD'
0C68 C5                        0200          DEFB  'E'+80H
0C69                            0210          DEFM  'SQUELC'
0C6F C8                        0220          DEFB  'H'+80H
0C70                            0230          DEFM  'PIN'
0C73 C7                        0240          DEFB  'G'+80H
0C74                            0250          DEFM  'ZA'
0C76 D0                        0260          DEFB  'P'+80H
0C77                            0270          DEFM  'ALIE'
0C7B C5                        0280          DEFB  'N'+80H
0C7C                            0290          DEFM  'SHOO'
0C80 D4                        0300          DEFB  'T'+80H
0C81 00                        0310          DEFB  0
0C82 CDA7BC                    0320  EXPLOD  CALL  SNDRES
0C85 3E01                      0330          LD   A, 1
0C87 21949C                    0340          LD   HL, AMP1
0C8A CDBCB0                    0350          CALL  AMPENV
0C8D 21989C                    0360          LD   HL, SND1
0C90 CDAABC                    0370          CALL  ADDSND
0C93 C9                        0380          RET
0C94 01                        0390  AMP1   DEFB  1,15,255,25
                                0F FF 19

```

9C98	01	0400	SND1	DEFB	1,1,0,0,0
	01 00 00	00			
9C9D	0F	0410		DEFB	15,15,0,0
	0F 00 00				
9CA1	CDA7BC	0420	SQUELC	CALL	SNDRES
9CA4	060F	0430		LD	B,15
9CA6	C5	0440	LOOP2	PUSH	BC
9CA7	78	0450		LD	A,B
9CA8	32BC9C	0460		LD	(NOISE2),A
9CAB	21B79C	0470		LD	HL,SND2
9CAE	CDAABC	0480	FULL2	CALL	ADDSDND
9CB1	30FB	0490		JR	NC,FULL2
9CB3	C1	0500		POP	BC
9CB4	10F0	0510		DJNZ	LOOP2
9CB6	C9	0520		RET	
9CB7	01	0530	SND2	DEFB	1,1,0,0,0
	01 00 00	00			
9CBC	00	0540	NOISE2	DEFB	0,15,2,0
	0F 02 00				
9CC0	CDA7BC	0550	PING	CALL	SNDRES
9CC3	3E01	0560		LD	A,1
9CC5	21D29C	0570		LD	HL,AMP3
9CC8	CDBCBC	0580		CALL	AMPENV
9CCB	21D69C	0590		LD	HL,SND3
9CCE	CDAABC	0600		CALL	ADDSDND
9CD1	C9	0610		RET	
9CD2	01	0620	AMP3	DEFB	1,15,255,5
	0F FF 05				
9CD6	01	0630	SND3	DEFB	1,1,0,25,0
	01 00 19	00			
9CDB	00	0640		DEFB	0,15,0,0
	0F 00 00				
9CDF	CDA7BC	0650	ZAP	CALL	SNDRES
9CE2	3E01	0660		LD	A,1
9CE4	21059D	0670		LD	HL,TONE4
9CE7	CDBFBC	0680		CALL	TONENV
9CEA	21099D	0690		LD	HL,SND4A
9CED	CDAABC	0700		CALL	ADDSDND
9CF0	060F	0710		LD	B,15
9CF2	C5	0720	LOOP4	PUSH	BC
9CF3	3E10	0730		LD	A,16
9CF5	90	0740		SUB	B
9CF6	32179D	0750		LD	(NOISE4),A
9CF9	21129D	0760		LD	HL,SND4B
9CFC	CDAABC	0770	FULL4	CALL	ADDSDND
9CFF	30FB	0780		JR	NC,FULL4
9D01	C1	0790		POP	BC
9D02	10EE	0800		DJNZ	LOOP4

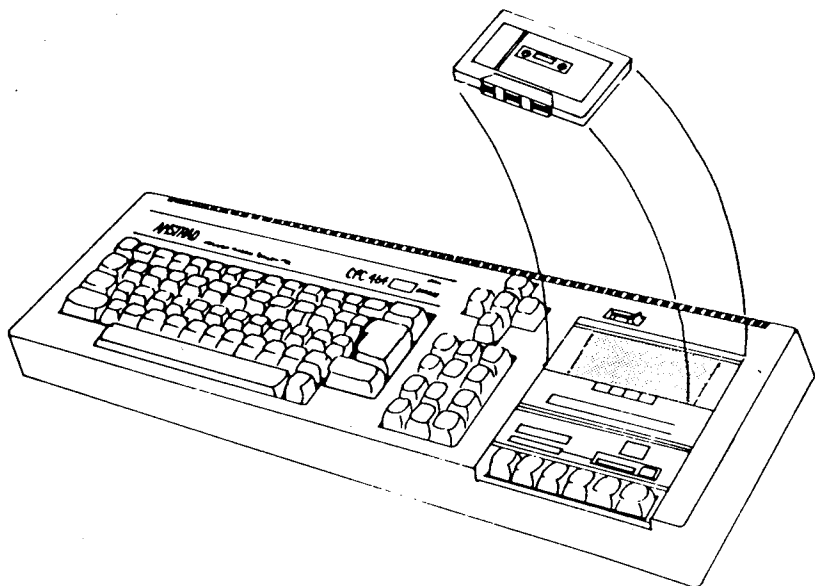
9D04	C9	0810	RET
9D05	01	0820	TONE4 DEF 1,15,4,2
	0F 04 02		
9D09	01	0830	SND4A DEF 1,0,1,40,0
	00 01 28 00		
9D0E	00	0840	DEFB 0,6,30,0
	06 1E 00		
9D12	02	0850	SND4B DEF 2,0,0,0,0
	00 00 00 00		
9D17	00	0860	NOISE4 DEF 0,15,2,0
	0F 02 00		
9D1B	CDA7BC	0870	ALIEN CALL SNDRES
9D1E	3EFF	0880	LD A,255
9D20	21339D	0890	LD HL,TONE5
9D23	C0BFBC	0900	CALL TONENV
9D26	213A9D	0910	LD HL,SND5A
9D29	CDAABC	0920	CALL ADDSND
9D2C	21439D	0930	LD HL,SND5B
9D2F	CDAABC	0940	CALL ADDSND
9D32	C9	0950	RET
9D33	02	0960	TONE5 DEF 2,5,255,2
	05 FF 02		
9D37	05	0970	DEFB 5,1,2
	01 02		
9D3A	01	0980	SND5A DEF 1,0,1,100



```

00 01 64
9D3E 00 0990 DEFB 0,0,7,100
00 07 64
9D42 00 1000 DEFB 0
9D43 02 1010 SND5B DEFB 2,0,1,108
00 01 6C
9D47 00 1020 DEFB 0,0,7,100
00 07 64
9D4B 00 1030 DEFB 0
9D4C CDA7BC 1040 SHOOT CALL SNDRES
9D4F 3E01 1050 LD A,1
9D51 21669D 1060 LD HL,AMP6
9D54 C0BCBC 1070 CALL AMPENV
9D57 216A9D 1080 LD HL,SND6A
9D5A CDAABC 1090 CALL ADDSND
9D5D 21739D 1100 LD HL,SND6B
9D60 CDAABC 1110 FULL6 CALL ADDSND
9D63 30FB 1120 JR NC,FULL6
9D65 C9 1130 RET
9D66 01 1140 AMP6 DEFB 1,15,255,2
0F FF 02
9D6A 01 1150 SND6A DEFB 1,0,0,0,0
00 00 00 00
9D6F 0A 1160 DEFB 10,15,4,0
0F 04 00
9D73 01 1170 SND6B DEFB 1,1,0,0,0
01 00 00 00
9D78 05 1180 DEFB 5,15,0,0
0F 00 00
1190 END

```



Caracter ampliado por 8

Esta rutina transforma tu pequeño e insignificante carácter de 8 por 8 pixels en un **monstruo** 8 veces mayor -de aquí el imaginativo título. La rutina es, efectivamente, bastante simple de escribir. Lo que hace la rutina es, en efecto, tomar el carácter especificado, examinarlo línea a línea y allí donde un pixel contiene el **foreground** (primer plano) **color**, reemplazarlo (el pixel) con el carácter 143 (CHR\$(143)) que representa un bloque sólido. Esto hace que el carácter impreso sea 8 veces más grande que el normal. Hábil, ¿eh?

Esta rutina acepta el siguiente comando:

CALL 40000, X, Y, CHAR, P1, P2

X e Y son las coordenadas superior-izquierda desde donde será impreso el carácter magnificado; CHAR es el código ASCII correspondiente al carácter que vamos a ampliar; P1 y P2 controlan los colores de la mitad superior e inferior del carácter, respectivamente. Con esto es posible obtener un color diferente en cada mitad, creando un sensacional efecto.

La siguiente rutina de demostración imprimirá dos caracteres, 'GO' en Mode 0 con sus mitades en diferente color. Para cambiar los colores, simplemente pulsa la barra espaciadora.

```
10 MODE 0: CLS
20 FOR T = 0 TO 25
30 CALL 40000, 1, 8, 71, T, T+1
40 CALL 40000, 12, 8, 79, T+1, T
50 IF INKEY$ = "" THEN NEXT ELSE 50
60 END
```

```
1 /          CARACTER AMPLIADO POR 8
10 SYMBOL AFTER 256:MEMORY 39999:SYMBOL
AFTER 240
20 FOR n=40000 TO 40076
30 READ a$:POKE n,VAL('%'+a$)
40 NEXT
50 DATA 0D,93,BB,F5,0D,06,B9,0D,7E,04
```

```

60 DATA CD,A5,BB,EB,DD,6E,06,DD,66,08
70 DATA 06,08,C5,E5,CD,75,BB,CB,40,28
80 DATA 05,DD,7E,00,18,03,DD,7E,02,CD
90 DATA 90,BB,E1,2C,0E,80,06,08,1A,A1
100 DATA 28,04,3E,8F,18,02,3E,20,CD,5A
110 DATA BB,CB,39,10,EF,13,C1,10,D1,CD
120 DATA 09,B9,F1,CD,90,BB,C9

```

```

Editor Assembler
AMMAS 1.1
Copyright 1985 PICTURESQUE

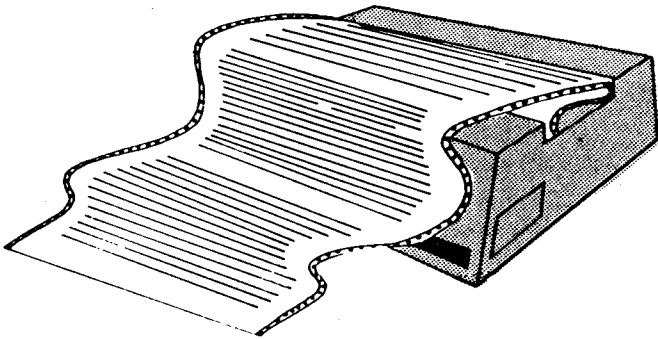
```

```

                                0001 ;      CAR  *8
                                0002 ;
9C40                            0010      ORG  40000
BB93                            0020  GETPEN  DEFL 0BB93H
BB90                            0030  SETPEN  DEFL 0BB90H
BBA5                            0040  MATRIX  DEFL 0BBA5H
BB75                            0050  CURSOR  DEFL 0BB75H
BB5A                            0060  TXTOUT  DEFL 0BB5AH
B906                            0070  ROMENA  DEFL 0B906H
B909                            0080  ROMDIS  DEFL 0B909H
9C40 CD93BB                    0090      CALL  GETPEN
9C43 F5                         0100      PUSH  AF
9C44 CD06B9                    0110      CALL  ROMENA
9C47 DD7E04                    0120      LD   A,(IX+4)
9C4A CDA5BB                    0130      CALL  MATRIX
9C4D EB                         0140      EX   DE,HL
9C4E DD6E06                    0150      LD   L,(IX+6)
9C51 DD6608                    0160      LD   H,(IX+8)
9C54 0608                      0170      LD   B,8
9C56 C5                        0180  NXTROW  PUSH  BC
9C57 E5                        0190      PUSH  HL
9C58 CD75BB                    0200      CALL  CURSOR
9C5B CB40                      0210      BIT   0,B
9C5D 2805                      0220      JR   Z,COL2
9C5F DD7E00                    0230      LD   A,(IX+0)
9C62 1803                      0240      JR   SETCOL
9C64 DD7E02                    0250  COL2   LD   A,(IX+2)
9C67 CD90BB                    0260  SETCOL  CALL  SETPEN
9C6A E1                        0270      POP  HL
9C6B 2C                        0280      INC  L
9C6C 0E80                      0290      LD   C,128

```

9C6E	0608	0300		LD	B,8
9C70	1A	0310	NXTCOL	LD	A,(DE)
9C71	A1	0320		AND	C
9C72	2804	0330		JR	Z,SPACE
9C74	3E8F	0340		LD	A,143
9C76	1802	0350		JR	PRINT
9C78	3E20	0360	SPACE	LD	A,32
9C7A	CD5ABB	0370	PRINT	CALL	TXTOUT
9C7D	CB39	0380		SRL	C
9C7F	10EF	0390		DJNZ	NXTCOL
9C81	13	0400		INC	DE
9C82	C1	0410		POP	BC
9C83	10D1	0420		DJNZ	NXTROW
9C85	CD09B9	0430		CALL	ROMDIS
9C88	F1	0440		POP	AF
9C89	CD90BB	0450		CALL	SETPEN
9C8C	C9	0460		RET	
		0470		END	



Caracter ampliado por 24

Esta es una variación sobre un tema. En lugar de poner un carácter 143 (CHR\$(143)) por cada pixel activado, esta rutina permite que en su lugar sea puesto un extravagante bloque de 3 por 3 de estos caracteres, con lo cual creamos un carácter 3 veces mayor que el anterior y 24 veces mayor que el original. Obviamente, el efecto de la rutina se verá limitado si se usa en **Mode 0**, donde únicamente hay 20 columnas a lo ancho de la pantalla.

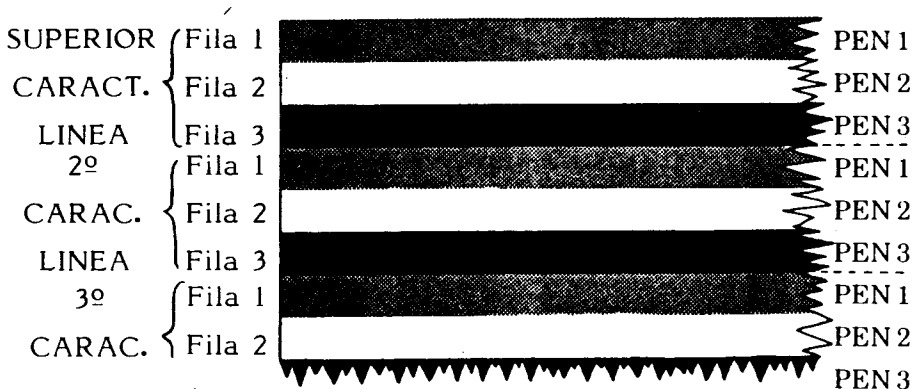
Como puede verse en el formato mostrado abajo, la rutina no tiene otra diferencia (para su uso):

CALL 40000, X, Y, CHAR, P1, P2, P3

Mediante la 'P' extra, conseguimos una mayor diferencia en el color del carácter. Existe una serie de posibilidades para usar este color extra, nosotros hemos decidido usarlo de la siguiente manera. Cada línea del carácter (ampliado) tiene una longitud de 24 caracteres por 3 de alto. El color de la primera línea de este rectángulo (24 caracteres por 1 de alto) es siempre adjudicado por **P1**, la segunda por **P2** y la tercera por **P3**.

Probablemente, es más fácil mostrar esto en un diagrama, tal como el representado a continuación:

COLORES DE PEN SOBRE TIMES 24 PRINT



HASTA COMPLETAR EL CARACTER

```

1 /          CHARACTER AMPLIADO POR 24
10 SYMBOL AFTER 256:MEMORY 39999:SYMBOL
AFTER 240
20 FOR n=40000 TO 40097
30 READ a#:POKE n,VAL('%'+a#)
40 NEXT
50 DATA CD,93,BB,F5,CD,06,B9,DD,7E,06
60 DATA CD,A5,BB,EB,DD,6E,08,DD,66,0A
70 DATA 06,08,C5,0E,80,06,03,C5,E5,CD
80 DATA 75,BB,CB,40,28,0E,CB,48,28,05
90 DATA DD,7E,04,18,08,DD,7E,00,18,03
100 DATA DD,7E,02,CD,90,BB,E1,06,08,1A
110 DATA A1,28,04,3E,8F,18,02,3E,20,CD
120 DATA 5A,BB,CD,5A,BB,CD,5A,BB,CB,39
130 DATA 10,E9,2C,C1,10,C5,13,C1,10,BC
140 DATA CD,09,B9,F1,CD,90,BB,C9

```

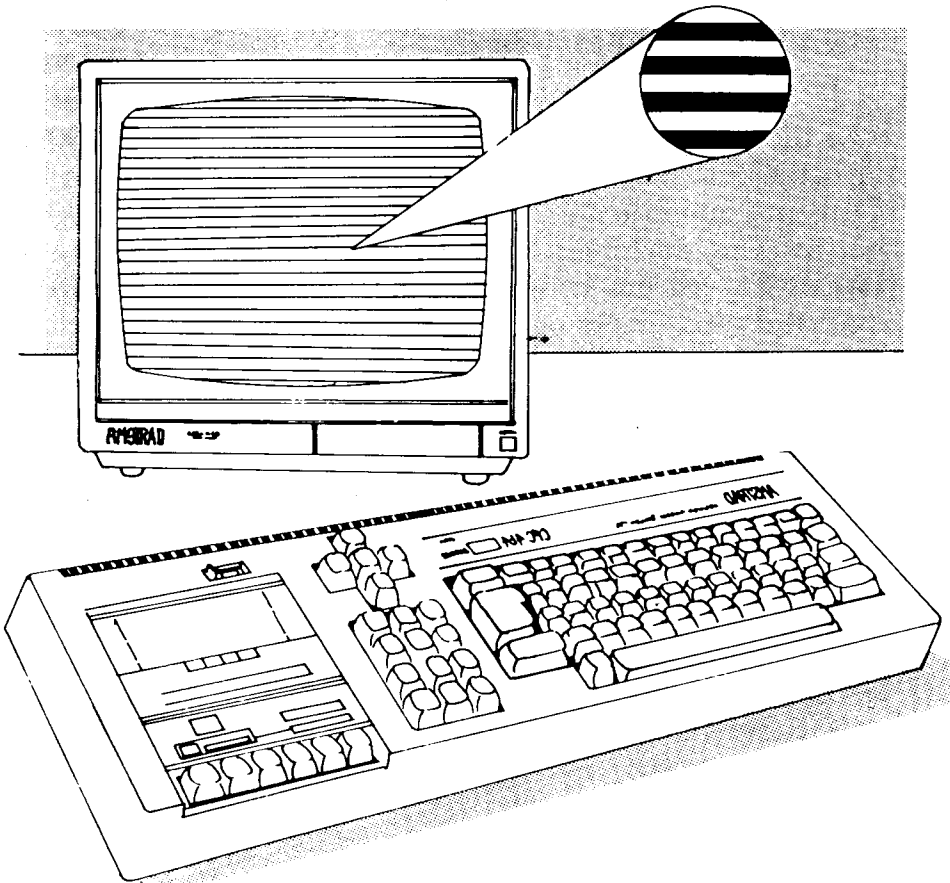
Editor Assembler
AMMAS 1.1
Copyright 1985 PICTURESQUE

```

          0001 ;      CAR  *24
          0002 ;
9C40      0010          ORG  400000
BB93      0020 GETPEN  DEFL 0BB93H
BB90      0030 SETPEN  DEFL 0BB90H
BBA5      0040 MATRIX  DEFL 0BBA5H
BB75      0050 CURSOR  DEFL 0BB75H
BB5A      0060 TXTOUT  DEFL 0BB5AH
B906      0070 ROMENA  DEFL 0B906H
B909      0080 ROMDIS  DEFL 0B909H
9C40 CD93BB 0090          CALL GETPEN
9C43 F5      0100          PUSH AF
9C44 CD06B9 0110          CALL ROMENA
9C47 DD7E06 0120          LD   A,(IX+6)
9C4A CDA5BB 0130          CALL MATRIX
9C4D EB      0140          EX   DE,HL
9C4E DD6E08 0150          LD   L,(IX+8)
9C51 DD660A 0160          LD   H,(IX+10)
9C54 0608    0170          LD   B,8
9C56 C5      0180 NXTROW  PUSH BC
9C57 0E80    0190          LD   C,128
9C59 0603    0200          LD   B,3

```

905B	C5	0210	AGAIN	PUSH	BC
905C	E5	0220		PUSH	HL
905D	CD75BB	0230		CALL	CURSOR
9060	CB40	0240		BIT	0,B
9062	280E	0250		JR	Z,COL2
9064	CB48	0260		BIT	1,B
9066	2805	0270		JR	Z,COL3
9068	DD7E04	0280		LD	A,(IX+4)
906B	1808	0290		JR	SETCOL
906D	DD7E00	0300	COL3	LD	A,(IX+0)
9070	1803	0310		JR	SETCOL
9072	DD7E02	0320	COL2	LD	A,(IX+2)
9075	CD90BB	0330	SETCOL	CALL	SETPEN
9078	E1	0340		POP	HL
9079	0608	0350		LD	B,8
907B	1A	0360	NXTCOL	LD	A,(DE)
907C	A1	0370		AND	C
907D	2804	0380		JR	Z,SPACE
907F	3E8F	0390		LD	A,143
9081	1802	0400		JR	PRINT
9083	3E20	0410	SPACE	LD	A,32
9085	CD5ABB	0420	PRINT	CALL	TXTOUT
9088	CD5ABB	0430		CALL	TXTOUT
908B	CD5ABB	0440		CALL	TXTOUT
908E	CB39	0450		SRL	C
9090	10E9	0460		DJNZ	NXTCOL
9092	2C	0470		INC	L
9093	C1	0480		POP	BC
9094	10C5	0490		DJNZ	AGAIN
9096	13	0500		INC	DE
9097	C1	0510		POP	BC
9098	10BC	0520		DJNZ	NXTROW
909A	CD09B9	0530		CALL	ROMDIS
909D	F1	0540		POP	AF
909E	CD90BB	0550		CALL	SETPEN
90A1	C9	0560		RET	
		0570		END	



Lector de cabeceras

Esta corta rutina en Código Máquina, en conjunción con un programa en BASIC, te permitirá obtener los detalles que necesites sobre un programa grabado en cassette. No es necesario que grabes completamente todo el programa; la rutina únicamente necesita la cabecera de donde obtiene los siguientes parámetros de gran utilidad para identificar el programa:

Nombre de archivo (File Name)

Número de bloque (si el archivo es más largo de uno)

Especifica si es el último bloque de un programa

Tipo de archivo (File Type, esto es, BASIC, binario o ASCII)

Especifica si el programa está protegido

Longitud de los datos

Localización de los datos

Longitud total del archivo (File Length)

Dirección de entrada (únicamente en archivos binarios)

En esta rutina no es necesario el comando **CALL**, ya que existe un programa en BASIC ocupándose del Código Máquina. La rutina lee la cabecera del programa, poniéndola dentro de un buffer de 28 bytes especialmente creado para ello. El Código Máquina devuelve entonces el control al programa en BASIC, el cual examina el buffer (mediante el comando **PEEK**) mostrando en la pantalla las diferentes informaciones descritas anteriormente.

Puede que te resulte difícil encontrar inmediatamente un uso para este programa, pero los de este tipo son muy populares. Los paquetes de software que se anuncian como "rompedores de programas" y que permiten introducirse dentro del software comercial siempre tienen entre sus programas un **lector de cabeceras** (Header Reader). Otro uso para este programa es en aquéllos que has grabado en el cassette, pero te olvidaste de documentar (sí, esto es algo que nos ocurre a todos, sobre todo a las dos de la madrugada, recién acabado nuestro programa "mega-bytico"). Con esta rutina podrás obtener toda la información concerniente al programa que grabaste, sin tener que volver a cargarlo de nuevo.


```

1 /          LECTOR CABEDERAS
10 SYMBOL AFTER 256:MEMORY 39999:SYMBOL
AFTER 240
20 FOR n=40000 TO 40049
30 READ a#:POKE n,VAL('&'+a#)
40 NEXT
50 CLS
60 LOCATE 14,12:PRINT'Comienza la cinta'
70 CALL 40000
80 CLS:PRINT
90 er=PEEK(40050):buff=40051
100 IF er=255 GOTO 130
110 IF er=0 THEN PRINT'*ESCAPE*' ELSE PR
INT'*TAPE ERROR*'
120 END
130 PRINT'programa: ';
140 x=buff
150 PRINT CHR$(PEEK(x));
160 x=x+1:IF PEEK(x)<>0 AND x<buff+16 GO
TO 150
170 PRINT
180 a=PEEK(buff+23):b=PEEK(buff+17)
190 IF a<>0 AND b<>0 GOTO 240
200 PRINT:PRINT'bloque: ';PEEK(buff+16);
210 IF a<>0 THEN PRINT'(primer bloque)';
220 IF b<>0 THEN PRINT'(ultimo bloque)';
230 PRINT
240 PRINT:PRINT'tipo programa: ';
250 n=PEEK(buff+18)
260 IF n=0 THEN PRINT'BASIC';
270 IF n=1 THEN PRINT'BINARY';
280 IF n=2 THEN PRINT'SCREEN IMAGE';
290 IF n=3 THEN PRINT'ASCII';
300 IF PEEK(buff+28)<>0 THEN PRINT'(pro
tegido)' ELSE PRINT
310 PRINT:PRINT'direccion datos: ';PEEK
(buff+21)+256*PEEK(buff+22)
320 PRINT:PRINT'longitud datos: ';PEEK
(buff+19)+256*PEEK(buff+20)
330 IF n<>1 GOTO 370
340 PRINT:PRINT'introduce direccion: ';
350 nn=PEEK(buff+26)+256*PEEK(buff+27)
360 IF nn=0 THEN PRINT' INCOMPLETO' ELSE
PRINT nn

```

```

370 PRINT:PRINT'longitud total:':
PEEK(buff+24)+256*PEEK(buff+25)
380 LOCATE 3,18:PRINT'Pulsa cualquier tec
la para la siguiente cabecera'
390 WHILE INKEY#='':WEND
400 GOTO 50
410 DATA 21,73,9C,11,1C,00,3E,2C,CD,A1
420 DATA BC,F5,CD,03,BB,F1,38,04,32,72
430 DATA 9C,C9,3E,FF,32,72,9C,3E,00,32
440 DATA 8F,9C,3A,85,9C,CB,47,28,03,32
450 DATA 8F,9C,E6,0E,CB,3F,32,85,9C,C9

```

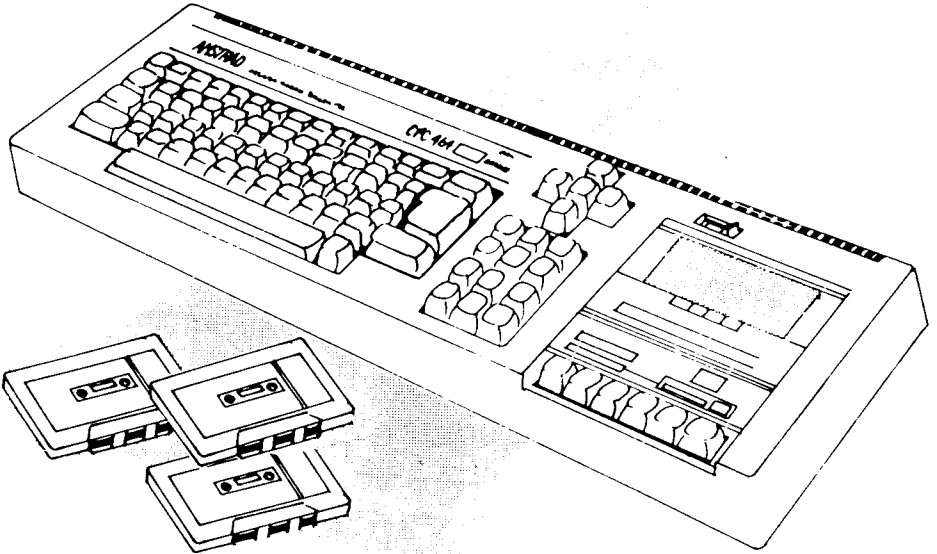
Editor Assembler
AMMAS 1.1
Copyright 1985 PICTURESQUE

```

0001 :      LECT CABECERAS
0002 :
9C40      0010      ORG      40000
BCA1      0020      CSREAD  DEFL 0BCA1H
BB03      0030      KMRES   DEFL 0BB03H
9C40 21739C 0040      LD      HL,BUFFER
9C43 111C00 0050      LD      DE,28
9C46 3E2C   0060      LD      A,2CH
9C48 CDA1BC 0070      CALL  CSREAD
9C4B F5      0080      PUSH  AF
9C4C CD03BB 0090      CALL  KMRES
9C4F F1      0100      POP   AF
9C50 3804   0110      JR    C,OK
9C52 32729C 0120      LD    (ERROR),A
9C55 C9     0130      RET
9C56 3EFF   0140      OK   LD    A,255
9C58 32729C 0150      LD    (ERROR),A
9C5B 3E00   0160      LD    A,0
9C5D 328F9C 0170      LD    (PRTECT),A
9C60 3A859C 0180      LD    A,(BUFFER+18)
9C63 CB47   0190      BIT  0,A
9C65 2803   0200      JR    Z,NOPROT
9C67 328F9C 0210      LD    (PRTECT),A
9C6A E60E   0220      NOPROT AND 14.
9C6C CB3F   0230      SRL  A
9C6E 32859C 0240      LD    (BUFFER+18),A

```

```
9071 09          0250          RET
9072 00          0260 ERROR DEFB 0
001C            0270 BUFFER DEFS 25
908F 00          0280 PRTECT DEFB 0
          0290          END
```



Impresión de cuatro caracteres

Si deseas un bloque de caracteres impresos formando un cuadrado (particularmente usado en gráficos definidos por el usuario, UDG), entonces ésta es la rutina que buscabas. Rápidamente y sin esfuerzo, esta rutina pondrá en la pantalla un bloque de caracteres.

El formato es el siguiente:

CALL 40000, X, Y, C

X e Y son las coordenadas correspondientes a la situación del bloque en la pantalla. X puede estar en cualquier posición entre 0 y 639, mientras que las posiciones para Y estarán entre 0 y 399. Observa que el tamaño del bloque limita la posición más alta de estos márgenes, aproximadamente en 20. Las coordenadas X e Y corresponden al pixel de la esquina superior izquierda del bloque de caracteres.

C corresponde al código ASCII del primer carácter, correspondiendo los otros tres a los siguientes códigos dentro del ASCII. Esto es extremadamente limpio y útil para la creación de gráficos definidos por el usuario (UDG) donde a menudo se utilizan caracteres diferentes para cada una de las partes de un bloque más grande.

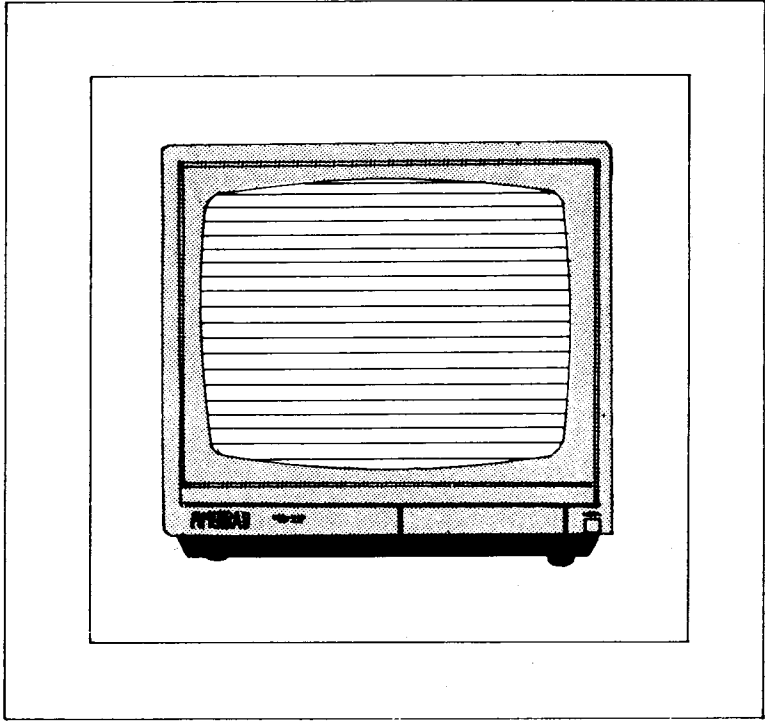
```
1 /      IMPRESION CUATRO CARACTERES
10 SYMBOL AFTER 256:MEMORY 39999:SYMBOL
AFTER 240
20 FOR n=40000 TO 40048
30 READ a$:POKE n,VAL("&"a$)
40 NEXT
50 DATA 00,4E,00,00,6E,02,00,66,03,00
60 DATA 5E,04,00,56,05,06,02,D5,E5,C5
70 DATA CD,C0,BB,C1,79,0C,C5,CD,FC,BB
80 DATA C1,79,0C,C5,CD,FC,BB,C1,E1,D1
90 DATA 3E,10,2B,3D,20,FC,10,E1,C9
```

Editor Assembler
 AMMAS 1.1
 Copyright 1985 PICTURESQUE

```

                                0001 :      IMP  4CARAC
                                0002 :
9C40      0010      ORG   40000
BBC6      0020  ASKCUR  DEFL  0BBC6H
BBC0      0030  MOVABS  DEFL  0BBC0H
BBFC      0040  GRACHR  DEFL  0BBFCH
9C40  DD4E00  0050      LD   C,(IX+0)
9C43  DD6E02  0060      LD   L,(IX+2)
9C46  DD6603  0070      LD   H,(IX+3)
9C49  DD5E04  0080      LD   E,(IX+4)
9C4C  DD5605  0090      LD   D,(IX+5)
9C4F  0602    0100      LD   B,2
9C51  05      0110  NXTCHR  PUSH  DE
9C52  E5      0120      PUSH  HL
9C53  C5      0130      PUSH  BC
9C54  CDC0BB  0140      CALL  MOVABS
9C57  C1      0150      POP   BC
9C58  79      0160      LD   A,C
9C59  0C      0170      INC  C
9C5A  C5      0180      PUSH  BC
9C5B  CDFCBB  0190      CALL  GRACHR
9C5E  C1      0200      POP   BC
9C5F  79      0210      LD   A,C
9C60  0C      0220      INC  C
9C61  C5      0230      PUSH  BC
9C62  CDFCBB  0240      CALL  GRACHR
9C65  C1      0250      POP   BC
9C66  E1      0260      POP   HL
9C67  D1      0270      POP   DE
9C68  3E10    0280      LD   A,16
9C6A  2B      0290  NEWROW  DEC   HL
9C6B  3D      0300      DEC  A
9C6C  20FC    0310      JR   NZ,NEWROW
9C6E  10E1    0320      DJNZ NXTCHR
9C70  C9      0330      RET
                                0340      END

```



Inversor de pantalla

Esta rutina rota la pantalla 180 grados independientemente de cuál sea su posición; esto hace que la parte superior se convierta en la inferior. Únicamente trabaja en **Mode 1**, a pesar de lo cual puede ser de gran efectividad, como muestra nuestro programa de demostración.

Para acceder a esta rutina, simplemente escribe:

CALL 40000

```
1      INVERSOR DE PANTALLA
10     SYMBOL AFTER 256:MEMORY 39999:SYMBOL
      AFTER 240
20     FOR n=40000 TO 40143
30     READ a#:POKE n,VAL('%'+a#)
40     NEXT
50     DATA 21,00,00,CD,1A,BC,EB,2E,18,26
60     DATA 27,CD,1A,BC,23,7C,C6,38,67,0E
70     DATA 64,D5,E5,06,50,C5,1A,D5,56,06
80     DATA 08,5F,1F,7B,CB,12,1F,10,F8,06
90     DATA 04,0F,CB,0A,10,FB,72,D1,12,1C
100    DATA 20,0A,14,7A,E6,07,20,04,7A,D6
110    DATA 08,57,7D,2D,B7,20,0A,7C,25,E6
120    DATA 07,20,04,7C,C6,08,67,C1,10,C9
130    DATA E1,D1,7A,C6,08,57,E6,38,20,14
140    DATA 7A,D6,40,57,7B,C6,50,5F,30,0A
150    DATA 14,7A,E6,07,20,04,7A,D6,08,57
160    DATA 7C,D6,08,67,E6,38,FE,38,20,14
170    DATA 7C,C6,40,67,7D,D6,50,6F,30,0A
180    DATA 7C,25,E6,07,20,04,7C,C6,08,67
190    DATA 0D,20,86,C9
```

Editor Assembler
AMMAS 1.1
Copyright 1985 PICTURESQUE

```
0001 ;      INVE PANTALLA
0002 ;
9C40 0010      ORG 40000
```

BC1A		0020	CHRPOS	DEFL	0BC1AH
9C40	210000	0030		LD	HL,0
9C43	CD1ABC	0040		CALL	CHRPOS
9C46	EB	0050		EX	DE,HL
9C47	2E18	0060		LD	L,24
9C49	2627	0070		LD	H,39
9C4B	CD1ABC	0080		CALL	CHRPOS
9C4E	23	0090		INC	HL
9C4F	7C	0100		LD	A,H
9C50	C638	0110		ADD	A,56
9C52	67	0120		LD	H,A
9C53	0E64	0130		LD	C,100
9C55	D5	0140	NXTLIN	PUSH	DE
9C56	E5	0150		PUSH	HL
9C57	0650	0160		LD	B,80
9C59	C5	0170	NXTBYT	PUSH	BC
9C5A	1A	0180		LD	A,(DE)
9C5B	D5	0190		PUSH	DE
9C5C	56	0200		LD	D,(HL)
9C5D	0608	0210		LD	B,8
9C5F	5F	0220	INVERT	LD	E,A
9C60	1F	0230		RRA	
9C61	7B	0240		LD	A,E
9C62	CB12	0250		RL	D
9C64	1F	0260		RRA	
9C65	10F8	0270		DJNZ	INVERT
9C67	0604	0280		LD	B,4
9C69	0F	0290	ROTATE	RRCA	
9C6A	CB0A	0300		RRC	D
9C6C	10FB	0310		DJNZ	ROTATE
9C6E	72	0320		LD	(HL),D
9C6F	D1	0330		POP	DE
9C70	12	0340		LD	(DE),A
9C71	1C	0350		INC	E
9C72	200A	0360		JR	NZ,DEBTOK
9C74	14	0370		INC	D
9C75	7A	0380		LD	A,D
9C76	E607	0390		AND	7
9C78	2004	0400		JR	NZ,DEBTOK
9C7A	7A	0410		LD	A,D
9C7B	D608	0420		SUB	8
9C7D	57	0430		LD	D,A
9C7E	7D	0440	DEBTOK	LD	A,L
9C7F	2D	0450		DEC	L
9C80	B7	0460		OR	A
9C81	200A	0470		JR	NZ,HLBTOK
9C83	7C	0480		LD	A,H
9C84	25	0490		DEC	H

9C85	E607	0500	AND	7
9C87	2004	0510	JR	NZ,HLBTOK
9C89	7C	0520	LD	A,H
9C8A	C608	0530	ADD	A,8
9C8C	67	0540	LD	H,A
9C8D	C1	0550	HLBTOK POP	BC
9C8E	10C9	0560	DJNZ	NXTBYT
9C90	E1	0570	POP	HL
9C91	D1	0580	POP	DE
9C92	7A	0590	LD	A,D
9C93	C608	0600	ADD	A,8
9C95	57	0610	LD	D,A
9C96	E638	0620	AND	56
9C98	2014	0630	JR	NZ,DELNOK
9C9A	7A	0640	LD	A,D
9C9B	D640	0650	SUB	64
9C9D	57	0660	LD	D,A
9C9E	7B	0670	LD	A,E
9C9F	C650	0680	ADD	A,80
9CA1	5F	0690	LD	E,A
9CA2	300A	0700	JR	NC,DELNOK
9CA4	14	0710	INC	D
9CA5	7A	0720	LD	A,D
9CA6	E607	0730	AND	7
9CA8	2004	0740	JR	NZ,DELNOK
9CAA	7A	0750	LD	A,D
9CAB	D608	0760	SUB	8
9CAD	57	0770	LD	D,A
9CAE	7C	0780	DELNOK LD	A,H
9CAF	D608	0790	SUB	8
9CB1	67	0800	LD	H,A
9CB2	E638	0810	AND	56
9CB4	FE38	0820	CP	56
9CB6	2014	0830	JR	NZ,HLLNOK
9CB8	7C	0840	LD	A,H
9CB9	C640	0850	ADD	A,64
9CBB	67	0860	LD	H,A
9CBC	7D	0870	LD	A,L
9CBD	D650	0880	SUB	80
9CBF	6F	0890	LD	L,A
9CC0	300A	0900	JR	NC,HLLNOK
9CC2	7C	0910	LD	A,H
9CC3	25	0920	DEC	H
9CC4	E607	0930	AND	7
9CC6	2004	0940	JR	NZ,HLLNOK
9CC8	7C	0950	LD	A,H
9CC9	C608	0960	ADD	A,8

900B	67	0970		LD	H,A
900C	0D	0980	HLLNOK	DEC	C
900D	2086	0990		JR	NZ,NXTLIN
900F	C9	1000		RET	
		1010		END	

Copiador de texto

No hay gran cosa que decir sobre esta rutina, excepto que puede aplicársele la máxima que dice "lo pequeño es bonito". Esta rutina puede "volcar" el texto de la pantalla a una impresora conectada a tu Amstrad.

Esto es útil en gran número de situaciones en las cuales copiar directamente una pantalla llena de números o datos, es mucho más fácil que alterar tu programa con cientos de **PRINT#8s** para sacar cada una de las líneas de texto a la impresora. Con esta rutina obtienes al mismo tiempo que los datos en la pantalla, una copia de esta pantalla en la impresora.

Para usar esta rutina, únicamente escribe **CALL 40000**, después rueda el programa del cual quieres sacar copias de la pantalla.

```
1 /          COPIADOR DE TEXTO
10 SYMBOL AFTER 256:MEMORY 39999:SYMBOL
AFTER 240
20 FOR n=40000 TO 40054
30 READ a$:POKE n,VAL('%'+a$)
40 NEXT
50 DATA 2E,01,CD,17,BC,0C,04,C5,26,01
60 DATA E5,CD,75,BB,E1,CD,60,BB,CD,2E
70 DATA BD,38,FB,CD,31,BD,24,10,ED,2C
80 DATA 3E,0D,CD,2E,BD,38,FB,CD,31,BD
90 DATA 3E,0A,CD,2E,BD,38,FB,CD,31,BD
100 DATA C1,0D,20,01,C9
```

```
Editor Assembler
AMMAS 1.1
Copyright 1985 PICTURESQUE
```

```
0001 ;          COPI TEXTO
0002 ;
9C40 0010 ORG 40000
```

BC17	0020	CHRLIM	DEFL	0BC17H
BB75	0030	CURSOR	DEFL	0BB75H
BB60	0040	RDCHAR	DEFL	0BB60H
BD2E	0050	BUSY	DEFL	0BD2EH
BD31	0060	SENDPR	DEFL	0BD31H
9C40 2E01	0070		LD	L,1
9C42 CD17BC	0080		CALL	CHRLIM
9C45 0C	0090		INC	C
9C46 04	0100		INC	B
9C47 C5	0110	NXTROW	PUSH	BC
9C48 2601	0120		LD	H,1
9C4A E5	0130	NXTCOL	PUSH	HL
9C4B CD75BB	0140		CALL	CURSOR
9C4E E1	0150		POP	HL
9C4F CD60BB	0160		CALL	RDCHAR
9C52 CD2EBD	0170	WAIT1	CALL	BUSY
9C55 38FB	0180		JR	C,WAIT1
9C57 CD31BD	0190		CALL	SENDPR
9C5A 24	0200		INC	H
9C5B 10ED	0210		DJNZ	NXTCOL
9C5D 2C	0220		INC	L
9C5E 3E0D	0230		LD	A,13
9C60 CD2EBD	0240	WAIT2	CALL	BUSY
9C63 38FB	0250		JR	C,WAIT2
9C65 CD31BD	0260		CALL	SENDPR
9C68 3E0A	0270		LD	A,10
9C6A CD2EBD	0280	WAIT3	CALL	BUSY
9C6D 38FB	0290		JR	C,WAIT3
9C6F CD31BD	0300		CALL	SENDPR
9C72 C1	0310		POP	BC
9C73 0D	0320		DEC	C
9C74 20D1	0330		JR	NZ,NXTROW
9C76 C9	0340		RET	
	0350		END	

Para completar esta rutina, tenemos una demostración que es uno de los procesadores de texto más cortos que hayas visto. Este programa acepta inputs de 80 caracteres en screen **Mode 0**, dejándolas en la pantalla hasta que es introducido un espacio o es alcanzada la parte inferior de la pantalla. Cuando alguna de estas cosas ocurre, la rutina **Copiador Texto** es llamada y la pantalla es copiada en la impresora. Pon el código de impresión que desees en la línea 1 (línea de doble espacio, énfasis de impresión). Este programa puede no ser el **Wordstar**, pero siempre tendrás la posibilidad de mejorarlo por ti mismo.

```
1 / Añade aqui tus codigos de control p  
ara la impresora  
10 MODE 2:INK 1,26:INK 0,0:BORDER 13:PEN  
1:PAPER 0:CLS  
20 FOR T=1 TO 24  
30 LINE INPUT A$:IF A$=' ' THEN CALL 400  
00  
40 NEXT  
50 CALL 40000
```

Rotación a la derecha

Esta rutina desplazará (scroll) hacia la derecha una línea de texto, haciendo desaparecer todos los pixels por la derecha y reaparecer por la izquierda (creando un efecto de rotación sobre la pantalla).

Pensando que esta rutina únicamente trabaja en **Mode 1**, tiene un número de parámetros para hacerla completamente flexible en su uso.

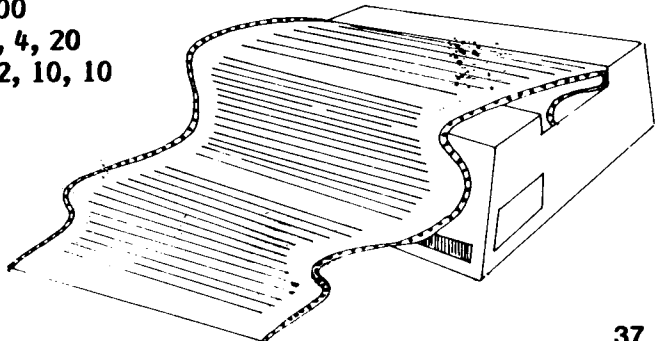
CALL 40000, X, Y, N

X equivale a las columnas, mientras que Y corresponde a la línea desde donde comenzará el desplazamiento. N equivale al número de caracteres que queremos desplazar hacia la derecha en esa línea. Con esto, en lugar de producir desplazamientos indiscriminados de grandes áreas de la pantalla, podemos desplazar pequeñas porciones de texto o UDGs y bloque de gráficos, en juegos podemos mover de manera precisa símbolos gráficos.

Puedes tener un número de áreas de la pantalla desplazándose simultáneamente, aunque en este caso es necesario para cada una de ellas una llamada **CALL**. Ya que con un **CALL** únicamente desplazamos un pixel cada columna, es necesario el uso de loops para mantener el movimiento de forma constante.

Aquí tenemos un ejemplo de la rutina desplazando dos áreas de la pantalla en **Mode 1**. Antes de correr el programa, llena la pantalla con caracteres distintos unos de otros, o escribe cualquier descripción.

```
10 FOR T = 1 to 500
20 CALL 40000, 5, 4, 20
30 CALL 40000, 12, 10, 10
40 NEXT
```



```

1 / ROTACION A DERECHA
10 SYMBOL AFTER 256:MEMORY 39999:SYMBOL
AFTER 240
20 FOR n=40000 TO 40094
30 READ a$:POKE n,VAL('&'+a$)
40 NEXT
50 DATA FE,03,C0,00,6E,02,2D,00,4E,00
60 DATA 0D,7E,04,81,06,02,67,0D,1A,BC
70 DATA 23,41,CB,20,05,0E,08,E5,C5,CB
80 DATA 3E,7E,F5,5D,54,7D,2D,B7,20,0A
90 DATA 7C,25,E6,07,20,04,7C,C6,08,67
100 DATA 1A,CB,3E,38,04,CB,9F,18,02,CB
110 DATA 0F,CB,5E,28,02,CB,FF,12,10,0B
120 DATA F1,38,04,CB,9E,18,02,CB,DE,CB
130 DATA 5F,28,02,CB,FE,C1,E1,7C,C6,08
140 DATA 67,0D,20,BD,C9

```

Editor Assembler
AMMAS 1.1
Copyright 1985 PICTURESQUE

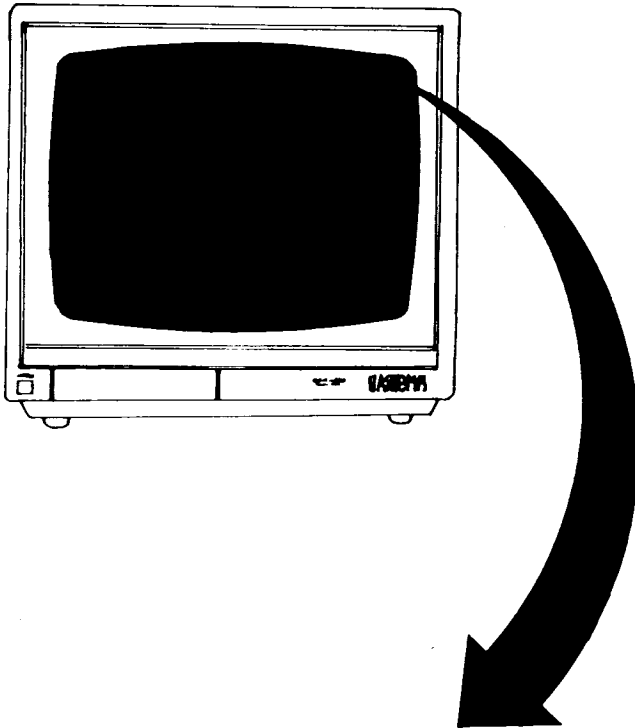
```

          0001 ;      ROTA DERECHA
          0002 ;
9C40      0010      ORG    40000
BC1A      0020 CHRPOS DEFL 0BC1AH
9C40 FE03      0030      CP    3
9C42 C0        0040      RET  NZ
9C43 DD6E02    0050      LD   L,(IX+2)
9C46 2D        0060      DEC  L
9C47 DD4E00    0070      LD   C,(IX+0)
9C4A DD7E04    0080      LD   A,(IX+4)
9C4D 81        0090      ADD  C
9C4E D602      0100      SUB  2
9C50 67        0110      LD   H,A

```

9C51	CD1ABC	0120		CALL	CHRPOS
9C54	23	0130		INC	HL
9C55	41	0140		LD	B,C
9C56	CB20	0150		SLA	B
9C58	05	0160		DEC	B
9C59	0E08	0170		LD	C,8
9C5B	E5	0180	NXTROW	PUSH	HL
9C5C	05	0190		PUSH	BC
9C5D	CB3E	0200		SRL	(HL)
9C5F	7E	0210		LD	A,(HL)
9C60	F5	0220		PUSH	AF
9C61	5D	0230	NXTBYT	LD	E,L
9C62	54	0240		LD	D,H
9C63	7D	0250		LD	A,L
9C64	2D	0260		DEC	L
9C65	B7	0270		OR	A
9C66	200A	0280		JR	NZ,HLOK
9C68	7C	0290		LD	A,H
9C69	25	0300		DEC	H
9C6A	E607	0310		AND	7
9C6C	2004	0320		JR	NZ,HLOK
9C6E	7C	0330		LD	A,H
9C6F	C608	0340		ADD	A,8
9C71	67	0350		LD	H,A
9C72	1A	0360	HLOK	LD	A,(DE)
9C73	CB3E	0370		SRL	(HL)
9C75	3804	0380		JR	C,SETBIT
9C77	CB9F	0390		RES	3,A
9C79	1802	0400		JR	TSTBIT
9C7B	CBDF	0410	SETBIT	SET	3,A
9C7D	CB5E	0420	TSTBIT	BIT	3,(HL)
9C7F	2802	0430		JR	Z,BITOK
9C81	CBFF	0440		SET	7,A
9C83	12	0450	BITOK	LD	(DE),A
9C84	10DB	0460		DJNZ	NXTBYT
9C86	F1	0470		POP	AF
9C87	3804	0480		JR	C,WRAP
9C89	CB9E	0490		RES	3,(HL)
9C8B	1802	0500		JR	ENDPIX
9C8D	CBDE	0510	WRAP	SET	3,(HL)
9C8F	CB5F	0520	ENDPIX	BIT	3,A
9C91	2802	0530		JR	Z,ROWFIN
9C93	CBFE	0540		SET	7,(HL)
9C95	C1	0550	ROWFIN	POP	BC
9C96	E1	0560		POP	HL
9C97	7C	0570		LD	A,H

9C98	C608	0580	ADD	A,8
9C9A	67	0590	LD	H,A
9C9B	0D	0600	DEC	C
9C9C	20BD	0610	JR	NZ,NXTROW
9C9E	C9	0620	RET	
		0630	END	



Rotación a la izquierda

Esencialmente esta rutina es la misma que la última, únicamente que el desplazamiento (scroll) se produce hacia el lado opuesto.

La declaración **CALL** es la misma:

CALL 40000, X, Y, N

Esta rutina (como la anterior) trabaja únicamente en **Mode 1** y también necesita ser llamada desde un loop si se quiere mantener el movimiento de forma continua. Para más detalles consultar la rutina **ROTACION DERECHA**.

```
1 /          ROTACION A IZQUIERDA
10 SYMBOL AFTER 256:MEMORY 39999:SYMBOL
AFTER 240
20 FOR n=40000 TO 40087
30 READ a#:POKE n,VAL('%'+a#)
40 NEXT
50 DATA FE,03,C0,DD,6E,02,2D,DD,66,04
60 DATA 25,CD,1A,BC,DD,46,00,CB,20,05
70 DATA 0E,08,E5,C5,CB,26,7E,F5,5D,54
80 DATA 2C,20,0A,24,7C,E6,07,20,04,7C
90 DATA D6,08,67,1A,CB,26,38,04,CB,A7
100 DATA 18,02,CB,E7,CB,66,28,02,CB,C7
110 DATA 12,10,DD,F1,38,04,CB,A6,18,02
120 DATA CB,E6,CB,67,28,02,CB,C6,C1,E1
130 DATA 7C,C6,08,67,0D,20,BF,C9
```

Editor Assembler
 AMMAS 1.1
 Copyright 1985 PICTURESQUE

```

                                0001 ;      ROTA IZQUIERDA
                                0002 ;
0C40      0010      ORG      40000
BC1A      0020 CHRPOS  DEFL  0BC1AH
0C40 FE03      0030      CP      3
0C42 C0        0040      RET     NZ
0C43 DD6E02    0050      LD      L,(IX+2)
0C46 2D        0060      DEC     L
0C47 DD6604    0070      LD      H,(IX+4)
0C4A 25        0080      DEC     H
0C4B CD1ABC    0090      CALL   CHRPOS
0C4E DD4600    0100      LD      B,(IX+0)
0C51 CB20      0110      SLA    B
0C53 05        0120      DEC     B
0C54 0E08      0130      LD      C,8
0C56 E5        0140 NXTROW  PUSH   HL
0C57 C5        0150      PUSH  BC
0C58 CB26      0160      SLA    (HL)
0C5A 7E        0170      LD      A,(HL)
0C5B F5        0180      PUSH  AF
0C5C 5D        0190 NXTBYT  LD      E,L
0C5D 54        0200      LD      D,H
0C5E 2C        0210      INC    L
0C5F 200A      0220      JR     NZ,HLOK
0C61 24        0230      INC    H
0C62 7C        0240      LD      A,H
0C63 E607      0250      AND    7
0C65 2004      0260      JR     NZ,HLOK
0C67 7C        0270      LD      A,H
0C68 D608      0280      SUB    8
0C6A 67        0290      LD      H,A
0C6B 1A        0300 HLOK    LD      A,(DE)
0C6C CB26      0310      SLA    (HL)
0C6E 3804      0320      JR     C,SETBIT
0C70 CBA7      0330      RES    4,A
0C72 1802      0340      JR     TSTBIT
0C74 CBE7      0350 SETBIT SET    4,A
0C76 CB66      0360 TSTBIT BIT    4,(HL)
0C78 2802      0370      JR     Z,BITOK
0C7A CBC7      0380      SET    0,A
0C7C 12        0390 BITOK  LD      (DE),A
0C7D 10DD      0400      DJNZ  NXTBYT
0C7F F1        0410      POP   AF

```

9C80	3804	0420	JR	C,WRAP
9C82	CBA6	0430	RES	4,(HL)
9C84	1802	0440	JR	ENDFIX
9C86	CBE6	0450	WRAP SET	4,(HL)
9C88	CB67	0460	ENDFIX BIT	4,A
9C8A	2802	0470	JR	Z,ROWFIN
9C8C	CBC6	0480	SET	0,(HL)
9C8E	C1	0490	ROWFIN POP	BC
9C8F	E1	0500	POP	HL
9C90	7C	0510	LD	A,H
9C91	C608	0520	ADD	A,8
9C93	67	0530	LD	H,A
9C94	0D	0540	DEC	C
9C95	20BF	0550	JR	NZ,NXTROW
9C97	C9	0560	RET	
		0570	END	

Fijar velocidad

Este programa puede ser titulado '**Turbo Load and Save**'; puede ser utilizado no únicamente para aumentar la velocidad de grabación y recuperación de nuestros programas, sino también para reducirla -quizás para una copia de seguridad o para aparentar que un programa es más largo de lo que es en realidad.

Bien, suficiente sobre cómo y con qué trabaja este programa. Para alterar las velocidades, esta rutina hace uso de otra localizada en la ROM que tiene el Amstrad. Con el BASIC del Amstrad tú puedes únicamente seleccionar SPEED WRITE 0, (1000 baudios) y SPEED WRITE 1, (2000 baudios). Con esta rutina podrás seleccionar dentro de un margen de velocidades mucho más amplio. Sin embargo, a partir de cierta velocidad cargar y salvar programas se hace cada vez más inseguro. Nosotros encontramos, después de una constante experimentación, que la velocidad más rápida con fiabilidad era de 3300 baudios, pero que esa fiabilidad únicamente se obtenía si la carga y recuperación del programa se realizaba en el mismo ordenador.

Dirigirse al '**Firmware Manual**', capítulo 14.125, ofrecido por Amsoft para una explicación completa sobre términos tales como "longitud mitad-cero" (half-zero length) y el funcionamiento del cassette. Todo lo que diremos aquí es que el formato de esta rutina es:

CALL 4000, H, P

H es la longitud de medio cero, pero todo lo que aquí necesitamos conocer es que $333,333$ (un tercio de millón) dividido por este valor, nos dará un valor aproximado de la velocidad en baudios. Según esto, un valor de $H = 167$ nos dará una velocidad aproximada de 2000 baudios.

P equivale al nivel de 'Precompensación' (dirigirse de nuevo al **Firmware Manual** para una explicación detallada). Teóricamente, este valor podía situarse entre 0 y 255, pero en la práctica valores entre 2 y 100 son más probables. Los niveles de precompensación que hemos encontrado más adecuados han sido entre 10 y 30.

En el caso de no especificar ningún parámetro después del estamento CALL, el ordenador asignará valores de 100 para H y 10 para P. Esto da una velocidad aproximada de 3300 baudios.

Experimenta con esta rutina, ya que para tu equipo pueden existir otros márgenes de velocidad distintos a los aquí indicados.

```

1      /      FIJAR VELOCIDAD
10     SYMBOL AFTER 256:MEMORY 39999:SYMBOL
      AFTER 240
20     FOR n=40000 TO 40026
30     READ a$:POKE n,VAL("&" +a$)
40     NEXT
50     DATA FE,00,20,07,21,64,00,3E,0A,18
60     DATA 0C,FE,02,C0,DD,7E,00,DD,6E,02
70     DATA DD,66,03,CD,68,BC,C9

```

Editor Assembler
 AMMAS 1.1
 Copyright 1985 PICTURESQUE

```

                                0001 ;      FIJA VELOCIDAD
                                0002 ;
9C40                                0010      ORG  40000
BC68                                0020 SPEED DEF 0BC68H
9C40 FE00                            0030      CP   0
9C42 2007                            0040      JR   NZ,GTVALS
9C44 216400                          0050      LD   HL,100
9C47 3E0A                            0060      LD   A,10
9C49 180C                            0070      JR   CONT
9C4B FE02                            0080 GTVALS CP   2
9C4D C0                              0090      RET  NZ
9C4E DD7E00                          0100      LD   A,(IX+0)
9C51 DD6E02                          0110      LD   L,(IX+2)
9C54 DD6603                          0120      LD   H,(IX+3)
9C57 CD68BC                          0130 CONT  CALL SPEED
9C5A C9                              0140      RET
                                0150      END

```

Desplazamiento rápido de la pantalla

Para escribir muchos tipos de programas de acción, así como para conseguir excitantes efectos visuales para casi cualquier programa, un desplazamiento (scroll) rápido, y sin salto, de la pantalla es una de las más útiles adiciones a tu colección de rutinas en Código Máquina.

Una vez llamada esta rutina, puede desplazar toda la pantalla el espacio de una línea hacia arriba o hacia abajo. El formato para CALL es:

CALL 40000, C, UD

C corresponde al color de la tinta que tomará la línea "vacía" que aparecerá arriba o abajo, dependiendo del tipo de scroll que uses. El parámetro UD define la dirección del movimiento; '0' para abajo y '1' para arriba.

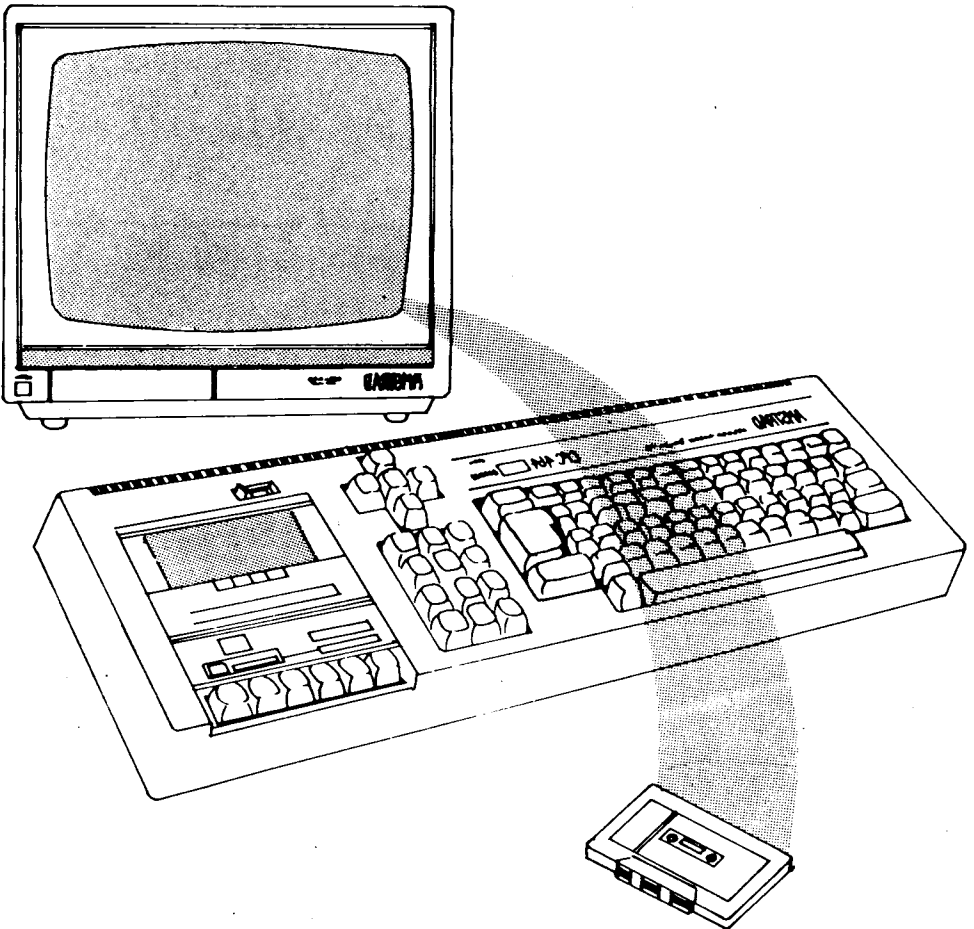
```
1 /      DESPLAZAMIENTO DE PANTALLA
10 SYMBOL AFTER 256:MEMORY 39999:SYMBOL
AFTER 240
20 FOR n=40000 TO 40015
30 READ a#:POKE n,VAL('&'+a#)
40 NEXT
50 DATA FE,02,C0,DD,46,00,DD,7E,02,CD
60 DATA 2C,BC,CD,4D,BC,C9
```

Editor Assembler
AMMAS 1.1
Copyright 1985 PICTURESQUE

```
0001 ;      DESP PANTALLA
0002 ;
9C40 0010 ORG 40000
BC2C 0020 ENCODE DEFL 0BC2CH
BC4D 0030 HWROLL DEFL 0BC4DH
9C40 FE02 0040 CP 2
```

```
9C42 C0      0050
9C43 DD4600  0060
9C46 DD7E02  0070
9C49 CD2CBC  0080
9C4C CD4DBC  0090
9C4F C9      0100
           0110
```

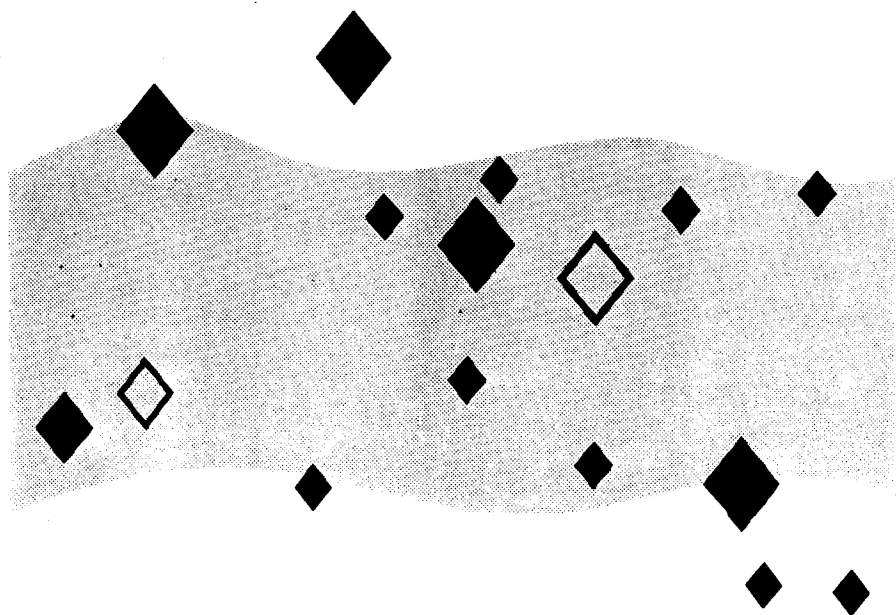
```
RET  NZ
LD   B,(IX+0)
LD   A,(IX+2)
CALL ENCODE
CALL HWROLL
RET
END
```



La pista de los diamantes

Para demostrar la rapidez y suavidad de este scroll (desplazamiento), aquí tenemos un corto programa con cuya ejecución podrás divertirte mucho. Está basado en esos juegos en los cuales se desplaza una pista y debemos evitar vernos fuera de sus límites. Nuestro juego es más complicado que eso. Tú estás viajando a través del famoso **Central African Diamond Trail** (pista o sendero de diamantes del Africa Central) en busca de sus escurridizas gemas. El sendero es peligroso y contiene muchos riesgos potencialmente fatales -indicados por una cruz roja. Los diamantes de suave brillo azul son los que has venido a buscar y podrás conseguirlos pasando por encima de ellos con tu coche, esto te dará 50 puntos extra. También conseguirás 10 puntos por cada décima de milla que viajes y el juego continúa hasta que te estrelles.

El juego utiliza el fast scroll (desplazamiento rápido) para mover hacia ti la carretera suave y rápidamente. El juego también indica la puntuación máxima y la que vamos obteniendo. Adelante... ponte a jugar.



```

10 RANDOMIZE TIME:hi=0
20 CLS:sc=0:x=3+INT(RND*24):y=10:c=x+y/2
30 x=3+INT(RND*24):c=x+y/2
40 LOCATE 1,1:PRINT STRING$(x,CHR$(143))
;STRING$(y,' ');STRING$(40-x-y,CHR$(143))
)
50 IF RND<0.3 THEN PEN 3:LOCATE x+1+INT(
RND*y),1:PRINT CHR$(203):PEN 1
60 IF RND<0.15 THEN PEN 2:LOCATE x+1+INT
(RND*y),1:PRINT CHR$(227):PEN 1
70 LOCATE c,22:PRINT ' '
80 c=c+(INKEY(8)=0)-(INKEY(1)=0)
90 CALL 40000,0,0
100 LOCATE 15,25:PRINT'PUNTUACION' =';sc
110 t=TEST(c*16-8,56)
120 IF t=1 OR t=3 GOTO 180
130 IF t=2 THEN SOUND 1,40,5,7:sc=sc+50
140 sc=sc+10:IF sc MOD 300=0 THEN y=y-
1-(y=5)
150 LOCATE c,22:PRINT''
160 x=x+INT(RND*3)-1+(x>24)-(x<3)
170 GOTO 40
180 CLS:LOCATE 9,10:PRINT'CCCCRRRRRAAAASS
SSHHHH!!!!'
190 SOUND 1,1000,200,7
200 LOCATE 13,12:PRINT'Tu puntuacion';sc
210 IF sc<hi GOTO 230
220 LOCATE 14,15:PRINT'PUNTUACION MAS AL
TA':hi+sc
230 LOCATE 13,20:PRINT'PUNTUACION =';hi
240 LOCATE 6,25:PRINT'Pulsa espacio para
jugar nuevamente'
250 WHILE INKEY#<>' ':WEND
260 GOTO 20

```

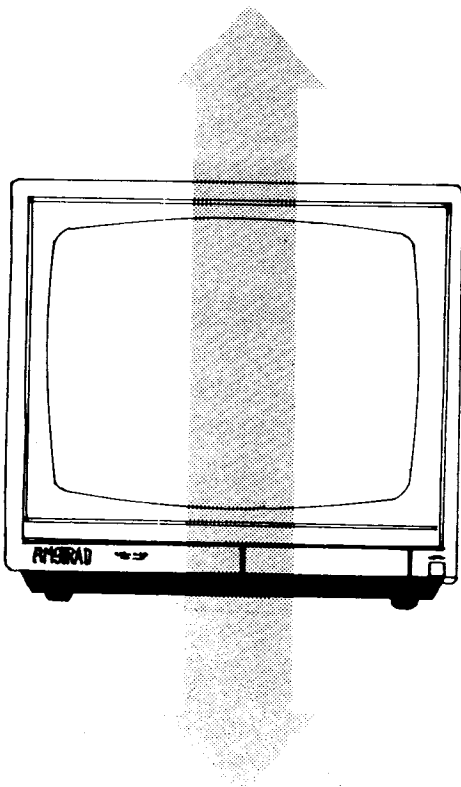
Desplazamiento de bloques

Esta práctica utilidad puede desplazar una porción de la pantalla hacia arriba o abajo. La 'caja' (box) necesita ser definida mediante el estamento **CALL**, asimismo es necesario definir el color y la dirección del movimiento; 1 hacia arriba y 0 hacia abajo.

Cuando se ejecuta el comando **CALL**, la caja se desplazará (scroll) una línea completa. La rutina ejecuta un desplazamiento extremadamente rápido. El formato de **CALL** es:

CALL 4000, L, R, T, B, P, D

L corresponde a la posición izquierda, R es la derecha, T corresponde a la parte superior y B a la inferior de la 'caja'. Esta caja o rectángulo es definida de una forma similar a cuando se utiliza el comando **BASIC WINDOW**. P es el color de la tinta y D determina la dirección del desplazamiento; 0 para abajo y 1 hacia arriba.



```

1 / DESPLAZAMIENTO BLOQUES
10 SYMBOL AFTER 256:MEMORY 39999:SYMBOL
AFTER 240
20 FOR n=40000 TO 40031
30 READ a$:POKE n,VAL('%'+a$)
40 NEXT
50 DATA FE,06,C0,DD,46,00,DD,7E,02,DD
60 DATA 5E,04,DD,6E,06,DD,56,08,DD,66
70 DATA 0A,1D,2D,15,25,CD,2C,BC,CD,50
80 DATA BC,C9

```

```

Editor Assembler
AMMAS 1.1
Copyright 1985 PICTURESQUE

```

```

          0001 :      DESP BLOQUES
          0002 :
9C40      0010      ORG    40000
BC2C     0020 ENCODE DEFL 0BC2CH
BC50     0030 SWROLL DEFL 0BC50H
9C40 FE06      0040      CP    6
9C42 C0        0050      RET  NZ
9C43 DD4600    0060      LD    B,(IX+0)
9C46 DD7E02    0070      LD    A,(IX+2)
9C49 DD5E04    0080      LD    E,(IX+4)
9C4C DD6E06    0090      LD    L,(IX+6)
9C4F DD5608    0100      LD    D,(IX+8)
9C52 DD660A    0110      LD    H,(IX+10)
9C55 1D        0120      DEC  E
9C56 2D        0130      DEC  L
9C57 15        0140      DEC  D
9C58 25        0150      DEC  H
9C59 CD2CBC    0160      CALL ENCODE
9C5C CD50BC    0170      CALL SWROLL
9C5F C9        0180      RET
          0190      END

```

Rejilla de diseño

Si tú realizas dibujos en tu Amstrad, o diseñas pantallas de texto y gráficos para usar en programas, entonces encontrarás que una matriz de rectángulos o rejilla, que cubra la pantalla a un toque de botón durante el diseño, desapareciendo posteriormente a otro toque de botón sin dejar rastro, ni dañando el dibujo, es algo muy útil.

Bien, esto es exactamente lo que hace la rutina **Rejilla Diseño** mediante el uso de una función especial del Código Máquina llamada **OR Exclusiva** o de forma abreviada **XOR** (extrañamente no forma parte del planeta Zilog).

La ejecución de **CALL 40000** mostrará instantáneamente una matriz de 8 por 8 pixels en la parte superior de la pantalla que estás diseñando. Esta rutina trabaja en cualquier modo de **screen** y se ajusta de tal forma que contiene muchas más líneas en **Mode 2** y muchas menos en **Mode 1**.

Una segunda ejecución de **CALL 40000** eliminará la rejilla de tu pantalla, pero dejará 'sin tocar' tu dibujo, esto es debido a la característica **XOR** del Código Máquina.

La mejor forma de utilizarla es en medio de un programa artístico donde esta rutina es llamada apretando un botón simplemente. Siguiendo desde este punto, aquí tenemos un corto y simple programa artístico en BASIC que te permite realizar dibujos mediante el joystick en **screen Mode 1**. Pulsando el botón de disparo una vez aparecerá la rejilla, una segunda pulsación la hará desaparecer. Estoy seguro que tú eres capaz de escribir un programa de dibujo mucho más complejo; el propósito del mío es simplemente demostrativo.

La rejilla fue deliberadamente diseñada para 8 por 8 pixels, de manera que corresponda completamente con un carácter de texto. Esta rutina puede ser usada no únicamente para dibujar en programas gráficos, sino incluso para una detallada presentación en pantalla de texto o semitexto.

```

1 / REJILLA DE DISEÑO
10 SYMBOL AFTER 256:MEMORY 39999:SYMBOL
AFTER 240
20 FOR n=40000 TO 40083
30 READ a$:POKE n,VAL('&'+a$)
40 NEXT
50 DATA 3E,01,CD,59,BC,CD,11,BC,3C,0E
60 DATA 0A,CB,21,3D,20,FB,3E,01,CD,2C
70 DATA BC,11,07,00,41,C5,21,00,00,01
80 DATA C7,00,F5,D5,CD,62,BC,D1,F1,06
90 DATA 08,13,10,FD,C1,10,EA,69,26,00
100 DATA 29,29,29,2B,4D,44,21,00,00,1E
110 DATA 19,F5,D5,C5,E5,11,00,00,CD,5F
120 DATA BC,E1,06,08,23,10,FD,C1,D1,F1
130 DATA 1D,20,EA,C9

```

Editor Assembler
AMMAS 1.1
Copyright 1985 PICTURESQUE

```

                                0001 : REJI DISEÑO
                                0002 :
9C40 0010 ORG 40000
BC59 0020 XORMOD DEFL 0BC59H
BC11 0030 MODE DEFL 0BC11H
BC2C 0040 INKENC DEFL 0BC2CH
BC5F 0050 SCRHOR DEFL 0BC5FH
BC62 0060 SCRVER DEFL 0BC62H
9C40 3E01 0070 LD A,1
9C42 CD59BC 0080 CALL XORMOD
9C45 CD11BC 0090 CALL MODE
9C48 3C 0100 INC A
9C49 0E0A 0110 LD C,10
9C4B CB21 0120 WIDTH SLA C
9C4D 3D 0130 DEC A
9C4E 20FB 0140 JR NZ,WIDTH
9C50 3E01 0150 LD A,1
9C52 CD2CBC 0160 CALL INKENC
9C55 110700 0170 LD DE,7
9C58 41 0180 LD B,C
9C59 C5 0190 NXTVER PUSH BC

```

9C5A	210000	0200		LD	HL,0
9C5D	01C700	0210		LD	BC,199
9C60	F5	0220		PUSH	AF
9C61	D5	0230		PUSH	DE
9C62	CD62BC	0240		CALL	SCRVER
9C65	D1	0250		POP	DE
9C66	F1	0260		POP	AF
9C67	0608	0270		LD	B,8
9C69	13	0280	MOVEX	INC	DE
9C6A	10FD	0290		DJNZ	MOVEX
9C6C	C1	0300		POP	BC
9C6D	10EA	0310		DJNZ	NXTVER
9C6F	69	0320		LD	L,C
9C70	2600	0330		LD	H,0
9C72	29	0340		ADD	HL,HL
9C73	29	0350		ADD	HL,HL
9C74	29	0360		ADD	HL,HL
9C75	2B	0370		DEC	HL
9C76	4D	0380		LD	C,L
9C77	44	0390		LD	B,H
9C78	210000	0400		LD	HL,0
9C7B	1E19	0410		LD	E,25
9C7D	F5	0420	NXTHOR	PUSH	AF
9C7E	D5	0430		PUSH	DE
9C7F	C5	0440		PUSH	BC
9C80	E5	0450		PUSH	HL
9C81	110000	0460		LD	DE,0
9C84	CD5FBC	0470		CALL	SCRHOR
9C87	E1	0480		POP	HL
9C88	0608	0490		LD	B,8
9C8A	23	0500	MOVEY	INC	HL
9C8B	10FD	0510		DJNZ	MOVEY
9C8D	C1	0520		POP	BC
9C8E	D1	0530		POP	DE
9C8F	F1	0540		POP	AF
9C90	1D	0550		DEC	E
9C91	20EA	0560		JR	NZ,NXTHOR
9C93	C9	0570		RET	
		0580		END	

Explosión gráfica

Si alguna vez has escrito juegos de acción, entonces te apuesto a que a menudo has deseado tener una buena, rápida y espectacular rutina de explosión. Reconozco que anteriormente a escribir ésta, había hecho muchos intentos en BASIC, pero todos carecían del suave desplazamiento y calidad del Código Máquina.

Esta rutina, como podemos ver abajo, es seguida por una buena cantidad de parámetros:

CALL 4000, L, R, T, B, TY, D

Los cuatro primeros parámetros fijan el rectángulo donde debe producirse la explosión. Como podrás comprobar, para obtener un buen efecto de explosión, necesitarás un rectángulo de tamaño razonable, L y R determinan la posición izquierda y derecha del rectángulo en la pantalla, mientras que T y B establecen la parte superior e inferior. El rectángulo se define de la misma forma que el comando Amstrad **WINDOW**.

A través del comando TY podemos seleccionar dos tipos de explosión disponibles. '0' selecciona la explosión que cubre completamente el rectángulo. '1' selecciona el otro tipo, con el cual únicamente hacemos 'explotar' algunos caracteres dentro del rectángulo. Si, por ejemplo, tienes una nave espacial realizada con dos caracteres dentro de un rectángulo de 3 por 3, entonces con la explosión tipo 1 únicamente explotarán los dos caracteres; sin embargo, con la tipo 0, 'explotarán' los 9 caracteres contenidos en el rectángulo de 3 por 3.

El último parámetro corresponde al retardo. Las áreas más pequeñas explotan antes que las más grandes, luego el retardo debe ajustarse de acuerdo con esto. Deberás experimentar con la rutina para encontrar el retardo que se ajusta al tamaño del rectángulo y al programa. Cuanto más alto sea el valor, mayor será el retardo y más lenta la explosión.

La rutina no es efectiva cuando se emplea en grandes áreas, pero puede trabajar en los tres modos de screen. Para realizar la explosión utiliza todos los colores disponibles en el modo en que se encuentra.


```

1 / EXPLOSION
10 SYMBOL AFTER 256:MEMORY 39999:SYMBOL
AFTER 240
20 FOR n=40000 TO 40226
30 READ a$:POKE n,VAL('%'+a$)
40 NEXT
50 DATA FE,06,C0,CD,06,B9,DD,6E,06,DD
60 DATA 66,0A,2D,25,DD,7E,08,94,4F,CD
70 DATA 1A,BC,AF,81,10,FD,47,DD,7E,04
80 DATA DD,4E,06,0D,91,4F,EB,21,00,00
90 DATA DD,7E,02,B7,20,45,C5,D5,3E,08
100 DATA F5,C5,D5,7E,23,B6,23,CB,74,28
110 DATA 03,21,00,00,12,1C,20,0A,14,7A
120 DATA E6,07,20,04,7A,D6,08,57,10,E5
130 DATA D1,7A,C6,08,57,C1,F1,3D,20,D8
140 DATA 7A,D6,40,57,7B,C6,50,5F,30,0A
150 DATA 14,7A,E6,07,20,04,7A,D6,08,57
160 DATA 0D,20,BF,D1,C1,3E,0F,F5,C5,D5
170 DATA DD,7E,00,3C,76,3D,20,FC,3E,08
180 DATA F5,C5,D5,7E,23,B6,23,CB,74,28
190 DATA 03,21,00,00,EB,A6,EB,12,1C,20
200 DATA 0A,14,7A,E6,07,20,04,7A,D6,08
210 DATA 57,10,E2,D1,7A,C6,08,57,C1,F1
220 DATA 3D,20,D5,7A,D6,40,57,7B,C6,50
230 DATA 5F,30,0A,14,7A,E6,07,20,04,7A
240 DATA D6,08,57,0D,20,BC,D1,C1,F1,3D
250 DATA 20,AB,3E,00,DD,5E,04,DD,6E,06
260 DATA DD,56,08,DD,66,0A,1D,2D,15,25
270 DATA CD,44,BC,CD,09,B9,C9

```

Editor Assembler
AMMAS 1.1
Copyright 1985 PICTURESQUE

```

0001 ; EXP GRAFICA
0002 ;
9C40 0010 ORG 40000
BC1A 0020 CHRPOS DEFL 0BC1AH
B906 0030 ROMENA DEFL 0B906H
B909 0040 ROMDIS DEFL 0B909H
BC44 0050 FILBOX DEFL 0BC44H
9C40 FE06 0060 CP 6

```

9C42	C0	0070		RET	NZ
9C43	0D06B9	0080		CALL	ROMENA
9C46	0D6E06	0090		LD	L,(IX+6)
9C49	0D660A	0100		LD	H,(IX+10)
9C4C	2D	0110		DEC	L
9C4D	25	0120		DEC	H
9C4E	0D7E08	0130		LD	A,(IX+8)
9C51	94	0140		SUB	H
9C52	4F	0150		LD	C,A
9C53	0D1ABC	0160		CALL	CHRPOS
9C56	AF	0170		XOR	A
9C57	81	0180	BYTES	ADD	A,C
9C58	10FD	0190		DJNZ	BYTES
9C5A	47	0200		LD	B,A
9C5B	0D7E04	0210		LD	A,(IX+4)
9C5E	0D4E06	0220		LD	C,(IX+6)
9C61	0D	0230		DEC	C
9C62	91	0240		SUB	C
9C63	4F	0250		LD	C,A
9C64	EB	0260		EX	DE,HL
9C65	210000	0270		LD	HL,0
9C68	0D7E02	0280		LD	A,(IX+2)
9C6B	B7	0290		OR	A
9C6C	2045	0300		JR	NZ,PART2
9C6E	C5	0310		PUSH	BC
9C6F	D5	0320		PUSH	DE
9C70	3E08	0330	NXTROW	LD	A,8
9C72	F5	0340	NXTLIN	PUSH	AF
9C73	C5	0350		PUSH	BC
9C74	D5	0360		PUSH	DE
9C75	7E	0370	NXTBYT	LD	A,(HL)
9C76	23	0380		INC	HL
9C77	B6	0390		OR	(HL)
9C78	23	0400		INC	HL
9C79	CB74	0410		BIT	6,H
9C7B	2803	0420		JR	Z,INROM
9C7D	210000	0430		LD	HL,0
9C80	12	0440	INROM	LD	(DE),A
9C81	1C	0450		INC	E
9C82	200A	0460		JR	NZ,GOTBYT
9C84	14	0470		INC	D
9C85	7A	0480		LD	A,D
9C86	E607	0490		AND	7
9C88	2004	0500		JR	NZ,GOTBYT
9C8A	7A	0510		LD	A,D
9C8B	D608	0520		SUB	8
9C8D	57	0530		LD	D,A

9C8E	10E5	0540	GOTBYT	DJNZ	NXTBYT
9C90	D1	0550		POP	DE
9C91	7A	0560		LD	A,D
9C92	C608	0570		ADD	A,8
9C94	57	0580		LD	D,A
9C95	C1	0590		POP	BC
9C96	F1	0600		POP	AF
9C97	3D	0610		DEC	A
9C98	20D8	0620		JR	NZ,NXTLIN
9C9A	7A	0630		LD	A,D
9C9B	D640	0640		SUB	64
9C9D	57	0650		LD	D,A
9C9E	7B	0660		LD	A,E
9C9F	C650	0670		ADD	A,80
9CA1	5F	0680		LD	E,A
9CA2	300A	0690		JR	NC,GOTROW
9CA4	14	0700		INC	D
9CA5	7A	0710		LD	A,D
9CA6	E607	0720		AND	7
9CA8	2004	0730		JR	NZ,GOTROW
9CAA	7A	0740		LD	A,D
9CAB	D608	0750		SUB	8
9CAD	57	0760		LD	D,A
9CAE	0D	0770	GOTROW	DEC	C
9CAF	20BF	0780		JR	NZ,NXTROW
9CB1	D1	0790		POP	DE
9CB2	C1	0800		POP	BC
9CB3	3E0F	0810	PART2	LD	A,15
9CB5	F5	0820	FADE	PUSH	AF
9CB6	C5	0830		PUSH	BC
9CB7	D5	0840		PUSH	DE
9CB8	0D7E00	0850		LD	A,(IX+0)
9CBB	3C	0860		INC	A
9CBC	76	0870	PAUSE	HALT	
9CBD	3D	0880		DEC	A
9CBE	20FC	0890		JR	NZ,PAUSE
9CC0	3E08	0900	NEWROW	LD	A,8
9CC2	F5	0910	NEWLIN	PUSH	AF
9CC3	C5	0920		PUSH	BC
9CC4	D5	0930		PUSH	DE
9CC5	7E	0940	NEWBYT	LD	A,(HL)
9CC6	28	0950		INC	HL
9CC7	B6	0960		OR	(HL)
9CC8	28	0970		INC	HL
9CC9	CB74	0980		BIT	6,H
9CCB	2803	0990		JR	Z,ROM
9CCD	210000	1000		LD	HL,0

9CD0	EB	1010	ROM	EX	DE,HL
9CD1	A6	1020		AND	(HL)
9CD2	EB	1030		EX	DE,HL
9CD3	12	1040		LD	(DE),A
9CD4	1C	1050		INC	E
9CD5	200A	1060		JR	NZ,FNDBYT
9CD7	14	1070		INC	D
9CD8	7A	1080		LD	A,D
9CD9	E607	1090		AND	7
9CDB	2004	1100		JR	NZ,FNDBYT
9CDD	7A	1110		LD	A,D
9CDE	D608	1120		SUB	8
9CE0	57	1130		LD	D,A
9CE1	10E2	1140	FNDBYT	DJNZ	NEWBYT
9CE3	D1	1150		POP	DE
9CE4	7A	1160		LD	A,D
9CE5	C608	1170		ADD	A,8
9CE7	57	1180		LD	D,A
9CE8	C1	1190		POP	BC
9CE9	F1	1200		POP	AF
9CEA	3D	1210		DEC	A
9CEB	20D5	1220		JR	NZ,NEWLIN
9CED	7A	1230		LD	A,D
9CEE	D640	1240		SUB	64
9CF0	57	1250		LD	D,A
9CF1	7B	1260		LD	A,E
9CF2	C650	1270		ADD	A,80
9CF4	5F	1280		LD	E,A
9CF5	300A	1290		JR	NC,FNDROW
9CF7	14	1300		INC	D
9CF8	7A	1310		LD	A,D
9CF9	E607	1320		AND	7
9CFB	2004	1330		JR	NZ,FNDROW
9CFD	7A	1340		LD	A,D
9CFE	D608	1350		SUB	8
9D00	57	1360		LD	D,A
9D01	0D	1370	FNDROW	DEC	C
9D02	20BC	1380		JR	NZ,NEWROW
9D04	D1	1390		POP	DE
9D05	C1	1400		POP	BC
9D06	F1	1410		POP	AF
9D07	3D	1420		DEC	A
9D08	20AB	1430		JR	NZ,FADE
9D0A	3E00	1440		LD	A,0
9D0C	DD5E04	1450		LD	E,(IX+4)
9D0F	DD6E06	1460		LD	L,(IX+6)
9D12	DD5608	1470		LD	D,(IX+8)

9D15	DD660A	1480	LD	H, (IX+10)
9D18	1D	1490	DEC	E
9D19	2D	1500	DEC	L
9D1A	15	1510	DEC	D
9D1B	25	1520	DEC	H
9D1C	CD44BC	1530	CALL	FILBOX
9D1F	CD09B9	1540	CALL	ROMDIS
9D22	C9	1550	RET	
		1560	END	

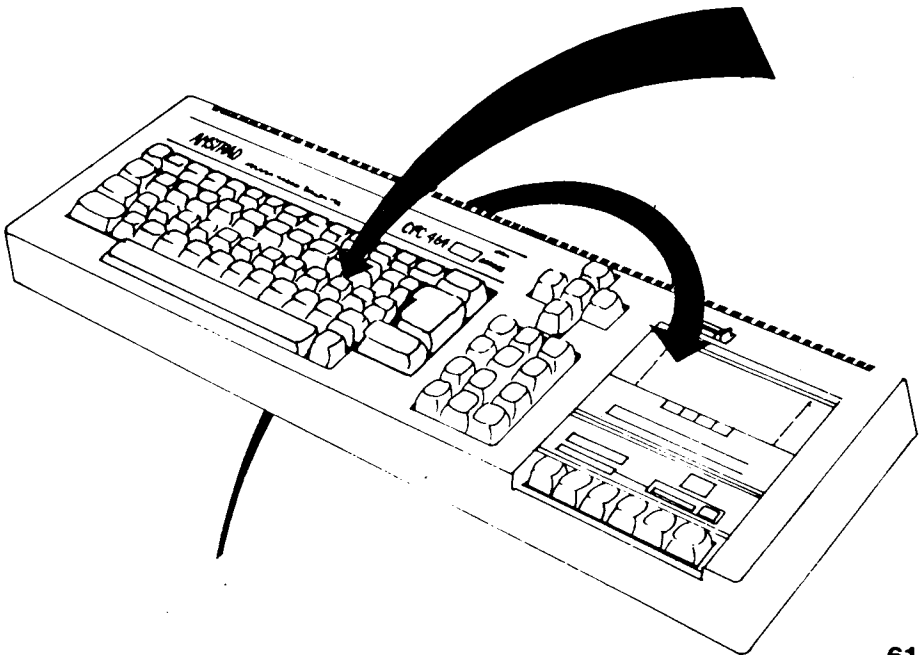
Analizador de pantalla

Esta rutina realiza una función de las más útiles; comprueba que un carácter está situado en una determinada posición de la pantalla. Esta función es frecuentemente usada en juegos gráficos con movimiento, y algunos home computers tienen una instrucción BASIC -tal como SCRN en el Spectrum- para realizar la labor de nuestra rutina.

La rutina trabaja en los tres modos de screen que tiene el Amstrad y debe ser llamada con el formato que se muestra a continuación:

C% = 0: CALL 40000, A, B, @ C%

A y B son las coordenadas de la posición que será 'chequeada' (comparada) en la pantalla, mientras que C% es la variable que contiene el código ASCII del carácter que reside en la posición especificada. C% debe ser un entero (de aquí el signo '%' después de la variable) y necesita ser definido antes que CALL, de otra forma se generará un error.



```

1 / ANALIZADOR DE PANTALLA
10 SYMBOL AFTER 256:MEMORY 39999:SYMBOL
AFTER 240
20 FOR n=40000 TO 40019
30 READ a$:POKE n,VAL('&'a$)
40 NEXT
50 DATA DD,6E,02,DD,66,04,CD,75,BB,CD
60 DATA 60,BB,DD,6E,00,DD,66,01,77,C9

```

Editor Assembler
AMMAS 1.1
Copyright 1985 PICTURESQUE

```

0001 ; ANAL PANTALLA
0002 ;
9C40 0010 ORG 40000
BB75 0020 CURSOR DEFL 0BB75H
BB60 0030 ROCHAR DEFL 0BB60H
9C40 DD6E02 0040 LD L,(IX+2)
9C43 DD6604 0050 LD H,(IX+4)
9C46 CD75BB 0060 CALL CURSOR
9C49 CD60BB 0070 CALL ROCHAR
9C4C DD6E00 0080 LD L,(IX+0)
9C4F DD6601 0090 LD H,(IX+1)
9C52 77 0100 LD (HL),A
9C53 C9 0110 RET
0120 END

```

Impresión de nueve caracteres

Una rutina anterior, **Impresión de Cuatro Caracteres**, ha sido rediseñada y ampliada para proporcionar un bloque de 3 por 3 caracteres que puede ser situado en cualquier lugar de la pantalla gráfica sin la necesidad de tener que usar TAG y TAGOFF, o cualquier otro comando complejo.

El principio para esta rutina es el mismo que para su predecesora. CALL 40000, X, Y, C iniciará la impresión del bloque en la pantalla gráfica, las coordenadas X e Y corresponden a la parte superior izquierda del bloque.

Naturalmente, la diferencia principal es el número de caracteres en pantalla. C sigue siendo el código ASCII correspondiente al primer carácter pero ahora siguen otros 8 caracteres más. Esto únicamente requerirá un poco más de esfuerzo en el planteamiento de tu diseño.

Como en la primera rutina, el principal uso de ésta es la de poder mover con facilidad un grupo de UDGs (User Definable Graphic) alrededor de la pantalla.

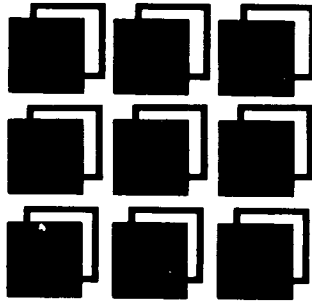
```
1 /      IMPRESION 9 CARACTERES
10 SYMBOL AFTER 256:MEMORY 39999:SYMBOL
AFTER 240
20 FOR n=40000 TO 40055
30 READ a#:POKE n,VAL('&'+a#)
40 NEXT
50 DATA 00,4E,00,00,6E,02,00,66,03,00
60 DATA 53,04,00,56,05,06,03,05,E5,C5
70 DATA 0D,C0,BB,C1,79,0C,C5,CD,FC,BB
80 DATA C1,79,0C,C5,CD,FC,BB,C1,79,0C
90 DATA C5,CD,FC,BB,C1,E1,D1,3E,10,2B
100 DATA 3D,20,FC,10,DA,C9
```


Editor Assembler
 AMMAS 1.1
 Copyright 1985 PICTURESQUE

```

                                0001 ;      IMP  9CARAC
                                0002 ;
9C40                            0010      ORG   40000
BBC6                            0020  ASKCUR  DEFL  @BBC6H
BBC0                            0030  MOVABS  DEFL  @BBC0H
BBFC                            0040  GRACHR  DEFL  @BBFCH
9C40 DD4E00                    0050      LD   C, (IX+0)
9C43 DD6E02                    0060      LD   L, (IX+2)
9C46 DD6603                    0070      LD   H, (IX+3)
9C49 DD5E04                    0080      LD   E, (IX+4)
9C4C DD5605                    0090      LD   D, (IX+5)
9C4F 0603                      0100      LD   B,3
9C51 D5                        0110  NXTCHR  PUSH  DE
9C52 E5                        0120      PUSH HL
9C53 C5                        0130      PUSH BC
9C54 CDC0BB                    0140      CALL MOVABS
9C57 C1                        0150      POP  BC
9C58 79                        0160      LD   A,C
9C59 0C                        0170      INC  C
9C5A C5                        0180      PUSH BC
9C5B CDFCBB                    0190      CALL GRACHR
9C5E C1                        0200      POP  BC
9C5F 79                        0210      LD   A,C
9C60 0C                        0220      INC  C
9C61 C5                        0230      PUSH BC
9C62 CDFCBB                    0240      CALL GRACHR
9C65 C1                        0250      POP  BC
9C66 79                        0260      LD   A,C
9C67 0C                        0270      INC  C
9C68 C5                        0280      PUSH BC
9C69 CDFCBB                    0290      CALL GRACHR
9C6C C1                        0300      POP  BC
9C6D E1                        0310      POP  HL
9C6E D1                        0320      POP  DE
9C6F 3E10                      0330      LD   A,16
9C71 2B                        0340  NEWROW  DEC  HL
9C72 3D                        0350      DEC  A
9C73 20FC                      0360      JR   NZ,NEWROW
9C75 10DA                      0370      DJNZ NXTCHR
9C77 C9                        0380      RET
                                0390      END

```



Este pequeño programa de demostración hace rebotar una pelota alrededor de la pantalla, mostrándonos la velocidad de la rutina **Impresión de Nueve Caracteres**. Recuerda, cuando observes la pelota moviéndose por la pantalla, que lo hace a través de los gráficos, no del texto.

```
10 SYMBOL 240,0,0,0,0,0,0,0,0,0
20 SYMBOL 241,0,0,0,0,0,0,0,0,0
30 SYMBOL 242,0,0,0,0,0,0,0,0,0
40 SYMBOL 243,0,0,0,0,0,0,0,0,0
50 SYMBOL 244,60,126,255,255,255,255,126
,60
60 SYMBOL 245,0,0,0,0,0,0,0,0,0
70 SYMBOL 246,0,0,0,0,0,0,0,0,0
80 SYMBOL 247,0,0,0,0,0,0,0,0,0
90 SYMBOL 248,0,0,0,0,0,0,0,0,0
100 BORDER 2:CLS
110 x=320:y=200:a=16:b=16
120 CALL 40000,x,y,240
130 x=x+a:y=y+b
140 IF x<10 OR x>600 THEN a=-a:SOUND 1,4
0,5,7
150 IF y<50 OR y>390 THEN b=-b:SOUND 1,5
0,5,7
160 GOTO 120
```

Itálicas

Mediante el comando **SYMBOL**, es posible crear un conjunto completo de nuevos caracteres. Considera, sin embargo, la cantidad de trabajo que conllevaría el realizar todas las letras mayúsculas, minúsculas, números y símbolos, poniendo cada uno de ellos dentro de una matriz de 8 por 8; estoy seguro de que estarás de acuerdo conmigo en que este programa es mucho más fácil y rápido de usar.

El formato es:

CALL 40000, R

R corresponde a la línea que deseas cambiar a caracteres itálicos. Originalmente hicimos que la rutina cambiara toda la pantalla, pero vimos que esto reducía su uso y flexibilidad. Pensamos también en la posibilidad de especificar la línea y la columna, de manera que se pudiese 'italizar' un carácter, pero este sistema sería muy lento en el caso de una frase, necesitando gran cantidad de accesos a la rutina. El parámetro resultante -el que especifica la línea- es lo que consideramos como mejor compromiso, este es el tipo de situaciones con las que te puedes encontrar cuando escribas tus propias rutinas en Código Máquina.

A diferencia de la mayoría de nuestras rutinas, ésta únicamente trabaja en screen Mode 1. Esto es debido a que la rutina accede a la pantalla directamente. Los caracteres están inclinados hacia la derecha, como en el caso de los caracteres itálicos, haciendo de esta manera que una pequeña porción del carácter nº 40 se deslice fuera de la pantalla.

```

1 /          ITALICAS
10 SYMBOL AFTER 256:MEMORY 39999:SYMBOL
AFTER 240
20 FOR n=40000 TO 40071
30 READ a$:POKE n,VAL('&' +a$)
40 NEXT
50 DATA B7,C8,DD,6E,00,2D,26,27,CD,1A
60 DATA BC,23,0E,07,C5,E5,CB,3E,06,4F
70 DATA 5D,54,7D,2D,B7,20,0A,7C,25,E6
80 DATA 07,20,04,7C,C6,08,67,1A,CB,3E
90 DATA 38,04,CB,9F,18,02,CB,DF,CB,5E
100 DATA 28,02,CB,FF,12,10,DB,CB,9E,E1
110 DATA 0D,20,D0,7C,C6,08,67,C1,0D,20
120 DATA C7,C9

```

```

Editor Assembler
AMMAS 1.1
Copyright 1985 PICTURESQUE

```

```

          0001 ;          ITALICAS
          0002 ;
9C40      0010      ORG    40000
BC1A      0020  CHRPOS  DEFL 0BC1AH
9C40 B7      0030      OR    A
9C41 C8      0040      RET   Z
9C42 DD6E00  0050      LD    L,(IX+0)
9C45 2D      0060      DEC   L
9C46 2627    0070      LD    H,39
9C48 CD1ABC  0080      CALL  CHRPOS
9C4B 23      0090      INC   HL
9C4C 0E07    0100      LD    C,7
9C4E C5      0110  NXTROW  PUSH  BC
9C4F E5      0120  SCROLL  PUSH  HL
9C50 CB3E    0130      SRL   (HL)

```

9C52	064F	0140		LD	B,79
9C54	5D	0150	NXTBYT	LD	E,L
9C55	54	0160		LD	D,H
9C56	7D	0170		LD	A,L
9C57	2D	0180		DEC	L
9C58	B7	0190		OR	A
9C59	200A	0200		JR	NZ,HLOK
9C5B	7C	0210		LD	A,H
9C5C	25	0220		DEC	H
9C5D	E607	0230		AND	7
9C5F	2004	0240		JR	NZ,HLOK
9C61	7C	0250		LD	A,H
9C62	C608	0260		ADD	A,8
9C64	67	0270		LD	H,A
9C65	1A	0280	HLOK	LD	A,(DE)
9C66	CB8E	0290		SRL	(HL)
9C68	8804	0300		JR	C,SETBIT
9C6A	CB9F	0310		RES	3,A
9C6C	1802	0320		JR	TSTBIT
9C6E	CBDF	0330	SETBIT	SET	3,A
9C70	CB5E	0340	TSTBIT	BIT	3,(HL)
9C72	2802	0350		JR	Z,BITOK
9C74	CBFF	0360		SET	7,A
9C76	12	0370	BITOK	LD	(DE),A
9C77	10DB	0380		DJNZ	NXTBYT
9C79	CB9E	0390		RES	3,(HL)
9C7B	E1	0400		POP	HL
9C7C	0D	0410		DEC	C
9C7D	20D0	0420		JR	NZ,SCROLL
9C7F	7C	0430		LD	A,H
9C80	C608	0440		ADD	A,8
9C82	67	0450		LD	H,A
9C83	C1	0460		POP	BC
9C84	0D	0470		DEC	C
9C85	20C7	0480		JR	NZ,NXTROW
9C87	C9	0490		RET	
		0500		END	

Letras inclinadas

Esta rutina tiene mucho en común con la anterior, **Itálicas**. Ella produce la inclinación hacia atrás de todos los caracteres contenidos en una línea determinada. La instrucción **CALL** es exactamente la misma que para la rutina **Itálicas**:

CALL 40000, R

La única diferencia es que ahora es la 1ª columna en lugar de la 40ª la que pierde parcialmente su carácter.

```
1 /      LETRAS INCLINADAS
10 SYMBOL AFTER 256:MEMORY 39999:SYMBOL
AFTER 240
20 FOR n=40000 TO 40068
30 READ a$:POKE n,VAL('%'+a$)
40 NEXT
50 DATA B7,C8,DD,6E,00,2D,26,00,CD,1A
60 DATA BC,0E,07,C5,E5,CB,26,06,4F,5D
70 DATA 54,2C,20,0A,24,7C,E6,07,20,04
80 DATA 7C,D6,08,67,1A,CB,26,38,04,CB
90 DATA A7,18,02,CB,E7,CB,66,28,02,CB
100 DATA C7,12,10,DD,CB,A6,E1,0D,20,D2
110 DATA 7C,C6,08,67,C1,0D,20,C9,C9
```

Editor Assembler
AMMAS 1.1
Copyright 1985 PICTURESQUE.

```
0001 :      LETR INCLINADAS
0002 :
9C40  0010  ORG  40000
BC1A  0020  CHRPOS  DEFL 0BC1AH
9C40  B7    0030  OR   A
9C41  C8    0040  RET  Z
9C42  DD6E00 0050  LD   L,(IX+0)
9C45  2D    0060  DEC  L
9C46  2600  0070  LD   H,0
```

9C48	CD1ABC	0080		CALL	CHRPOS
9C4B	0E07	0090		LD	C,7
9C4D	C5	0100	NXTROW	PUSH	BC
9C4E	E5	0110	SCROLL	PUSH	HL
9C4F	CB26	0120		SLA	(HL)
9C51	064F	0130		LD	B,79
9C53	5D	0140	NXTBYT	LD	E,L
9C54	54	0150		LD	D,H
9C55	2C	0160		INC	L
9C56	200A	0170		JR	NZ,HLOK
9C58	24	0180		INC	H
9C59	7C	0190		LD	A,H
9C5A	E607	0200		AND	7
9C5C	2004	0210		JR	NZ,HLOK
9C5E	7C	0220		LD	A,H
9C5F	D608	0230		SUB	8
9C61	67	0240		LD	H,A
9C62	1A	0250	HLOK	LD	A,(DE)
9C63	CB26	0260		SLA	(HL)
9C65	3804	0270		JR	C,SETBIT
9C67	CBA7	0280		RES	4,A
9C69	1802	0290		JR	TSTBIT
9C6B	CBE7	0300	SETBIT	SET	4,A
9C6D	CB66	0310	TSTBIT	BIT	4,(HL)
9C6F	2802	0320		JR	Z,BITOK
9C71	CBC7	0330		SET	0,A
9C73	12	0340	BITOK	LD	(DE),A
9C74	10DD	0350		DJNZ	NXTBYT
9C76	CBA6	0360		RES	4,(HL)
9C78	E1	0370		POP	HL
9C79	0D	0380		DEC	C
9C7A	20D2	0390		JR	NZ,SCROLL
9C7C	7C	0400		LD	A,H
9C7D	C608	0410		ADD	A,8
9C7F	67	0420		LD	H,A
9C80	C1	0430		POP	BC
9C81	0D	0440		DEC	C
9C82	20C9	0450		JR	NZ,NXTROW
9C84	C9	0460		RET	
		0470		END	

normal	NORMAL
<i>ITALICS</i>	<i>italics</i>
back slant	BACK SLANT
<i>ITALICS</i>	<i>italics</i>
back slant	BACK SLANT

Copiador inteligente

¡No, esta rutina no tiene ningún título ni nivel académico, pero sin embargo, es mucho más inteligente que esas simples rutinas para copiar que andan por ahí! Con esta rutina podremos copiar un área de la memoria en otra. El formato es el siguiente:

CALL 40000, A1, A2, L

A1 corresponde a la dirección inicial del bloque que va a ser movido, mientras que L representa su longitud. A2 es la dirección inicial donde va a ser copiado. La longitud viene expresada en bytes y, por ejemplo, un área de memoria desde 40000 hasta 40240 tiene una longitud de 241 bytes.

'Inteligente', tal y como es usado en el título, no es nuestro adjetivo preferido, sino más bien un término que describe cierta rutina. Una copia simple es aquella que se limita a transferir un área de memoria a otra sin hacer ningún tipo de comprobaciones; una copia inteligente permite que el área de destino 'solape' la de procedencia.

Por ejemplo, supongamos que deseas mover el área de memoria comprendida entre las direcciones 30000 y 30010, y quieres que la nueva dirección inicial esté en 30005, es decir, solapada. En esta situación, un copiadore simple realizaría la copia directamente, perdiendo el contenido de la mitad del área de memoria, mientras que un copiadore **inteligente**, comprobará la coincidencia de las dos áreas, copiando el nuevo material dentro de la nueva área, sin sobrescribir ninguna parte del mismo.

```
1 /          COPIADOR INTELIGENTE
10 SYMBOL AFTER 256:MEMORY 39999:SYMBOL
AFTER 240
20 FOR n=40000 TO 40039
30 READ a#:POKE n,VAL('&'+a#)
40 NEXT
50 DATA FE,03,C0,00,4E,00,00,46,01,00
60 DATA 5E,02,00,56,03,00,6E,04,00,66
70 DATA 05,B7,E5,ED,52,E1,38,03,ED,B0
80 DATA C9,09,EB,09,EB,2B,1B,ED,B8,C9
```

Editor Assembler
 AMMAS 1.1
 Copyright 1985 PICTURESQUE

```

          0001 :      COPI INTELIG
          0002 :
9C40      0010      ORG   40000
9C40 FE03   0020      CP    3
9C42 C0     0030      RET  NZ
9C43 DD4E00 0040      LD   C,(IX+0)
9C46 DD4601 0050      LD   B,(IX+1)
9C49 DD5E02 0060      LD   E,(IX+2)
9C4C DD5603 0070      LD   D,(IX+3)
9C4F DD6E04 0080      LD   L,(IX+4)
9C52 DD6605 0090      LD   H,(IX+5)
9C55 B7     0100      OR   A
9C56 E5     0110      PUSH HL
9C57 ED52   0120      SBC  HL,DE
9C59 E1     0130      POP  HL
9C5A 3803   0140      JR   C,LDDEC
9C5C EDB0   0150      LDIR
9C5E C9     0160      RET
9C5F 09     0170 LDDEC  ADD  HL,BC
9C60 EB     0180      EX  DE,HL
9C61 09     0190      ADD  HL,BC
9C62 EB     0200      EX  DE,HL
9C63 2B     0210      DEC  HL
9C64 1B     0220      DEC  DE
9C65 EDB8   0230      LDOR
9C67 C9     0240      RET
          0250      END

```

Memorizador/recuperador de pantallas

Esta rutina toma una pantalla (sobre 16K en términos de memoria) y la almacena en un lugar de la memoria. Posteriormente puede ser llamada para mostrarla de nuevo en la pantalla.

Permítenos explicarte cómo funciona. Para salvar la pantalla y transferirla a otra área de la memoria (comenzando, en efecto, en la dirección 27392) debemos escribir **CALL 43800**. Para recuperar la pantalla escribe **CALL 43812** -todo lo que hubiera en la pantalla será borrado y reemplazado por la nueva que estaba en la memoria.

La rutina ofrece una característica adicional. Si utilizas **CALL 43824**, la pantalla almacenada aparecerá sobre la pantalla actual sin borrarla. Obviamente, puede generarse cierta confusión allí donde coincidan dos caracteres distintos pertenecientes a las dos pantallas. Sin embargo, las áreas libres de una pantalla pueden ser utilizadas por la otra de una forma efectiva en distintos tipos de programas, aventuras, juegos, negocios, etc.

Examinando la lista del BASIC, puedes comprobar que en lugar de nuestro habitual comando **MEMORY 39999** (reservando unos cuantos millares de bytes para nuestras rutinas en Código Máquina, que comienzan en 40000) esta rutina reserva memoria a partir de **27390** en adelante. Esto es, como es natural, para permitir que todo el contenido de pantalla sea memorizado y si usas esta rutina en alguno de tus programas, deberás incluir en el mismo el comando **MEMORY 27390**.

Todavía hay otro punto que debes tener en cuenta al utilizar este programa. Antes de llamar a la rutina con **CALL**, deberás usar el comando **MODE** debido a que, si la pantalla ha sido desplazada (scroll) o alterada particularmente, podemos encontrarnos con que la nueva pantalla puede comenzar en un lugar erróneo. Obviamente, el mode en que estás actualmente determinará exactamente el comando que debes ejecutar. Una rutina que esté trabajando en mode 2 debe tener un comando **MODE 2** anterior a **CALL**.

```

1  MEMO/RECU DE PANTALLAS
10 SYMBOL AFTER 256:MEMORY 27390:SYMBOL
AFTER 240
20 FOR n=43800 TO 43843
30 READ a$:POKE n,VAL('%'+a$)
40 NEXT
50 DATA 01,00,40,11,00,6B,21,00,C0,ED
60 DATA B0,C9,01,00,40,11,00,C0,21,00
70 DATA 6B,ED,B0,C9,01,00,40,11,00,C0
80 DATA 21,00,6B,1A,AE,12,13,23,0B,78
90 DATA B1,20,F6,C9

```

```

Editor Assembler
AMMAS 1.1
Copyright 1985 PICTURESQUE

```

```

0001 ; MEM PANTALLAS
0002 ;
AB18 0010 ORG 43800
AB18 010040 0020 LD BC,4000H
AB1B 11006B 0030 LD DE,6B00H
AB1E 2100C0 0040 LD HL,0C000H
AB21 EDB0 0050 LDIR
AB23 C9 0060 RET
AB24 010040 0070 LD BC,4000H
AB27 1100C0 0080 LD DE,0C000H
AB2A 21006B 0090 LD HL,6B00H
AB2D EDB0 0100 LDIR
AB2F C9 0110 RET
AB30 010040 0120 LD BC,4000H
AB33 1100C0 0130 LD DE,0C000H
AB36 21006B 0140 LD HL,6B00H
AB39 1A 0150 LOOP LD A,(DE)
AB3A AE 0160 XOR (HL)
AB3B 12 0170 LD (DE),A

AB3C 13 0180 INC DE
AB3D 23 0190 INC HL
AB3E 0B 0200 DEC BC
AB3F 78 0210 LD A,B
AB40 B1 0220 OR C
AB41 20F6 0230 JR NZ,LOOP
AB43 C9 0240 RET
0250 END

```

Utilidades gráficas

Este conjunto de utilidades ha sido creado re-escribiendo un número de rutinas de las contenidas en este libro, de tal manera que se puedan utilizar como una rutina más larga. Con esta nueva posibilidad gráfica.

Esta rutina, una vez cargada en la memoria, permitirá al programador utilizar los nuevos comandos en los programas de BASIC que el mismo realice.

Teclear RUN, una vez metida la rutina en memoria, nos proporcionará los siguientes comandos:

! INVERT	Inversión de carácter
! EXPLODE	Explosión gráfica
! WRAPLEFT	Scroll a la izquierda
! WRAPRIGHT	Scroll a la derecha
! PRINTFOUR	Print de cuatro caracteres

Estos nuevos comandos BASIC toman el lugar de los CALL 40000, pero como es lógico, deben ser seguidos de varios parámetros. Para comprobar este punto y tener una mejor idea de cómo trabaja individualmente cada comando, revisar la explicación dada en el libro sobre la rutina en cuestión.

Este conjunto de utilidades puede ser utilizado para mejorar la presentación de un programa, sea éste un juego, educacional o incluso (quizás con la excepción de !EXPLODE) un programa comercial.

```
1          UTILIDADES GRAFICAS
10 SYMBOL AFTER 256:MEMORY 39999:SYMBOL
AFTER 240
20 DIM x(6)
30 FOR c=0 TO 6:READ a$:x(c)=VAL("&" + a$)
: NEXT
40 c=0:sum=0
50 FOR n=40000 TO 40609
60 READ a$:v=VAL("&" + a$)
```

```

70 sum=sum+v:POKE n,v
80 IF (n+1-40000) MOD 100<>0 GOTO 110
90 IF sum<>x(c) THEN PRINT'*ERROR IN DAT
A*':CHR$(7):PRINT'Comprobar lineas':200+
100*c;'a':290+100*c:END
100 sum=0:c=c+1
110 NEXT
120 IF sum<>x(c) THEN PRINT'*ERROR IN DA
TA*':CHR$(7):PRINT'Comprobar linea 800'
130 /
140 /          CHECKSUMS
150 /
160 DATA 2BCA,2EC6,2881,2803,2C6B,3029,0
450
170 /
180 /          MACHINE CODE
190 /
200 DATA 01,4A,9C,21,5B,9C,CD,D1,BC,C9
210 DATA 5F,9C,C3,87,9C,C3,D7,9C,C3,BA
220 DATA 9D,C3,12,9E,C3,71,9E,00,00,00
230 DATA 00,49,4E,56,45,52,04,45,58,50
240 DATA 4C,4F,44,C5,57,52,41,50,4C,45
250 DATA 46,04,57,52,41,50,52,49,47,48
260 DATA 04,50,52,49,4E,54,46,4F,55,D2
270 DATA 00,FE,03,C0,00,6E,00,00,66,01
280 DATA 7E,B7,C8,23,5E,23,56,00,6E,02
290 DATA 00,66,04,47,C5,E5,CD,75,BB,1A
300 DATA CD,AE,9C,13,E1,25,C1,10,F1,C9
310 DATA 05,CD,A5,BB,EB,CD,AE,BB,06,07
320 DATA 23,10,FD,F5,CD,06,B9,0E,08,1A
330 DATA 06,08,1F,CB,16,10,FB,13,2B,0D
340 DATA 20,F3,CD,09,B9,F1,CD,5A,BB,D1
350 DATA C9,FE,06,C0,CD,06,B9,DD,6E,06
360 DATA 00,66,0A,2D,25,DD,7E,08,94,4F
370 DATA CD,1A,BC,AF,81,10,FD,47,DD,7E
380 DATA 04,DD,4E,06,0D,91,4F,EB,21,00
390 DATA 00,DD,7E,02,B7,20,45,C5,D5,3E
400 DATA 08,F5,C5,D5,7E,23,B6,23,CB,74
410 DATA 28,03,21,00,00,12,1C,20,0A,14
420 DATA 7A,E6,07,20,04,7A,D6,08,57,10
430 DATA E5,D1,7A,D6,08,57,C1,F1,3D,20
440 DATA 08,7A,D6,40,57,7B,C6,50,5F,30
450 DATA 0A,14,7A,E6,07,20,04,7A,D6,08
460 DATA 57,0D,20,BF,D1,C1,3E,0F,F5,C5
470 DATA 05,DD,7E,00,3C,76,3D,20,FC,3E
480 DATA 08,F5,C5,D5,7E,23,B6,23,CB,74
490 DATA 28,03,21,00,00,EB,A6,EB,12,1C

```

500 DATA 20,0A,14,7A,E6,07,20,04,7A,D6
510 DATA 08,57,10,E2,D1,7A,C6,08,57,C1
520 DATA F1,3D,20,D5,7A,D6,40,57,7B,C6
530 DATA 50,5F,30,0A,14,7A,E6,07,20,04
540 DATA 7A,D6,08,57,0D,20,BC,D1,C1,F1
550 DATA 3D,20,AB,3E,00,DD,5E,04,DD,6E
560 DATA 06,DD,56,08,DD,66,0A,1D,2D,15
570 DATA 25,CD,44,BC,CD,09,B9,C9,FE,08
580 DATA C0,DD,6E,02,2D,DD,66,04,25,CD
590 DATA 1A,BC,DD,46,00,CB,20,05,0E,08
600 DATA E5,C5,CB,26,7E,F5,5D,54,2C,20
610 DATA 0A,24,7C,E6,07,20,04,7C,D6,08
620 DATA 67,1A,CB,26,38,04,CB,A7,18,02
630 DATA CB,E7,CB,66,28,02,CB,C7,12,10
640 DATA DD,F1,38,04,CB,A6,18,02,CB,E6
650 DATA CB,67,28,02,CB,C6,C1,E1,7C,C6
660 DATA 08,67,0D,20,BF,C9,FE,03,C0,DD
670 DATA 6E,02,2D,DD,4E,00,DD,7E,04,81
680 DATA D6,02,67,CD,1A,BC,23,41,CB,20
690 DATA 05,0E,08,E5,C5,CB,3E,7E,F5,5D
700 DATA 54,7D,2D,B7,20,0A,7C,25,E6,07
710 DATA 20,04,7C,C6,08,67,1A,CB,3E,38
720 DATA 04,CB,9F,18,02,CB,DF,CB,5E,28
730 DATA 02,CB,FF,12,10,DB,F1,38,04,CB
740 DATA 9E,18,02,CB,DE,CB,5F,28,02,CB
750 DATA FE,C1,E1,7C,C6,08,67,0D,20,BD
760 DATA C9,DD,4E,00,DD,6E,02,DD,66,03
770 DATA DD,5E,04,DD,56,05,06,02,D5,E5
780 DATA C5,CD,C0,BB,C1,79,0C,C5,CD,FC
790 DATA BB,C1,79,0C,C5,CD,FC,BB,C1,E1
800 DATA D1,3E,10,2B,3D,20,FC,10,E1,C9

Editor Assembler
 AMMAS 1.1
 Copyright 1985 PICTURESQUE

```

                                0001 ;      UTIL  GRAFICAS
                                0002 ;
9C40      0010      ORG    40000
BCD1      0020 LOGEXT  DEFL  0BCD1H
BB75      0030 CURSOR  DEFL  0BB75H
BBA5      0040 MATRIX  DEFL  0BBA5H
BBAE      0050 MTABLE  DEFL  0BBAEH
B906      0060 ROMENA  DEFL  0B906H
B909      0070 ROMDIS  DEFL  0B909H
BB5A      0080 TXTOUT  DEFL  0BB5AH
BC1A      0090 CHRPOS  DEFL  0BC1AH
BC44      0100 FILBOX  DEFL  0BC44H
BBC0      0110 MOVABS  DEFL  0BBC0H
BBFC      0120 GRACHR  DEFL  0BBFCH
9C40 014A9C 0130      LD    BC, TABLE
9C43 215B9C 0140      LD    HL, SPACE
9C46 CDD1BC 0150      CALL LOGEXT
9C49 C9      0160      RET
9C4A 5F9C   0170 TABLE DEFW  NAMETB
9C4C C3879C 0180      JP   INVERT
9C4F C3D79C 0190      JP   EXPLOD
9C52 C3BA9D 0200      JP   LWRAP
9C55 C3129E 0210      JP   RWRAP
9C58 C3719E 0220      JP   GRPNT4
9C5B 00     0230 SPACE  DEFB  0,0,0,0
      00 00 00
9C5F      0240 NAMETB  DEFM  'INVER'
9C64 04     0250      DEFB  'T'+80H
9C65      0260      DEFM  'EXPLOD'
9C6B 05     0270      DEFB  'E'+80H
9C6C      0280      DEFM  'WRAPLEF'
9C73 04     0290      DEFB  'T'+80H
9C74      0300      DEFM  'WRAPRIGH'
9C7C 04     0310      DEFB  'T'+80H
9C7D      0320      DEFM  'PRINTFOU'
9C85 02     0330      DEFB  'R'+80H
9C86 00     0340      DEFB  0
9C87 FE03   0350 INVERT  CP   3
9C89 C0     0360      RET  NZ
9C8A DD6E00 0370      LD   L, (IX+0)
9C8D DD6601 0380      LD   H, (IX+1)
9C90 7E     0390      LD   A, (HL)
9C91 B7     0400      OR   A
9C92 C8     0410      RET  Z

```


9C93	23	0420		INC	HL
9C94	5E	0430		LD	E, (HL)
9C95	23	0440		INC	HL
9C96	56	0450		LD	D, (HL)
9C97	DD6E02	0460		LD	L, (IX+2)
9C9A	DD6604	0470		LD	H, (IX+4)
9C9D	47	0480		LD	B, A
9C9E	05	0490	NXTCHR	PUSH	BC
9C9F	E5	0500		PUSH	HL
9CA0	CD75BB	0510		CALL	CURSOR
9CA3	1A	0520		LD	A, (DE)
9CA4	CDAE9C	0530		CALL	INVCHR
9CA7	13	0540		INC	DE
9CA8	E1	0550		POP	HL
9CA9	25	0560		DEC	H
9CAA	C1	0570		POP	BC
9CAB	10F1	0580		DJNZ	NXTCHR
9CAD	C9	0590		RET	
9CAE	D5	0600	INVCHR	PUSH	DE
9CAF	CD45BB	0610		CALL	MATRIX
9CB2	EB	0620		EX	DE, HL
9CB3	CD4EBB	0630		CALL	MTABLE
9CB6	0607	0640		LD	B, 7
9CB8	23	0650	ADD7	INC	HL
9CB9	10FD	0660		DJNZ	ADD7
9CBB	F5	0670		PUSH	AF
9CBC	CD06B9	0680		CALL	ROMENA
9CBF	0E08	0690		LD	C, 8
9CC1	1A	0700	NXTBYT	LD	A, (DE)
9CC2	0608	0710		LD	B, 8
9CC4	1F	0720	NXTBIT	RRA	
9CC5	CB16	0730		RL	(HL)
9CC7	10FB	0740		DJNZ	NXTBIT
9CC9	13	0750		INC	DE
9CCA	2B	0760		DEC	HL
9CCB	0D	0770		DEC	C
9CCD	20F3	0780		JR	NZ, NXTBYT
9CCE	CD09B9	0790		CALL	ROMDIS
9CD1	F1	0800		POP	AF
9CD2	CD5ABB	0810		CALL	TXTOUT
9CD5	D1	0820		POP	DE
9CD6	C9	0830		RET	
9CD7	FE06	0840	EXPLOD	CP	6
9CD9	C0	0850		RET	NZ
9CDA	CD06B9	0860		CALL	ROMENA
9CDD	DD6E06	0870		LD	L, (IX+6)
9CE0	DD660A	0880		LD	H, (IX+10)

9CE3	20	0890		DEC	L
9CE4	25	0900		DEC	H
9CE5	0D7E08	0910		LD	A, (IX+8)
9CE8	94	0920		SUB	H
9CE9	4F	0930		LD	C, A
9CEA	0D1ABC	0940		CALL	CHRPOS
9CED	AF	0950		XOR	A
9CEE	81	0960	BYTES	ADD	A, C
9CEF	10FD	0970		DJNZ	BYTES
9CF1	47	0980		LD	B, A
9CF2	0D7E04	0990		LD	A, (IX+4)
9CF5	0D4E06	1000		LD	C, (IX+6)
9CF8	0D	1010		DEC	C
9CF9	91	1020		SUB	C
9CFA	4F	1030		LD	C, A
9CFB	EB	1040		EX	DE, HL
9CFC	210000	1050		LD	HL, 0
9CFF	0D7E02	1060		LD	A, (IX+2)
9D02	B7	1070		OR	A
9D03	2045	1080		JR	NZ, PART2
9D05	C5	1090		PUSH	BC
9D06	D5	1100		PUSH	DE
9D07	3E08	1110	NXTROW	LD	A, 8
9D09	F5	1120	NXTLIN	PUSH	AF
9D0A	C5	1130		PUSH	BC
9D0B	D5	1140		PUSH	DE
9D0C	7E	1150	NXTBTE	LD	A, (HL)
9D0D	23	1160		INC	HL
9D0E	B6	1170		OR	(HL)
9D0F	23	1180		INC	HL
9D10	0B74	1190		BIT	6, H
9D12	2803	1200		JR	Z, INROM
9D14	210000	1210		LD	HL, 0
9D17	12	1220	INROM	LD	(DE), A
9D18	1C	1230		INC	E
9D19	200A	1240		JR	NZ, GOTBYT
9D1B	14	1250		INC	D
9D1C	7A	1260		LD	A, D
9D1D	E607	1270		AND	7
9D1F	2004	1280		JR	NZ, GOTBYT
9D21	7A	1290		LD	A, D
9D22	D608	1300		SUB	8
9D24	57	1310		LD	D, A
9D25	10E5	1320	GOTBYT	DJNZ	NXTBTE
9D27	D1	1330		POP	DE
9D28	7A	1340		LD	A, D
9D29	C608	1350		ADD	A, 8

902B	57	1360		LD	D,A
902C	C1	1370		POP	BC
902D	F1	1380		POP	AF
902E	3D	1390		DEC	A
902F	200B	1400		JR	NZ,NXTLIN
9031	7A	1410		LD	A,D
9032	D640	1420		SUB	64
9034	57	1430		LD	D,A
9035	7B	1440		LD	A,E
9036	C650	1450		ADD	A,80
9038	5F	1460		LD	E,A
9039	300A	1470		JR	NC,GOTROW
903B	14	1480		INC	D
903C	7A	1490		LD	A,D
903D	E607	1500		AND	7
903F	2004	1510		JR	NZ,GOTROW
9041	7A	1520		LD	A,D
9042	D608	1530		SUB	8
9044	57	1540		LD	D,A
9045	0D	1550	GOTROW	DEC	C
9046	20BF	1560		JR	NZ,NXTROW
9048	D1	1570		POP	DE
9049	C1	1580		POP	BC
904A	3E0F	1590	PART2	LD	A,15
904C	F5	1600	FADE	PUSH	AF
904D	C5	1610		PUSH	BC
904E	D5	1620		PUSH	DE
904F	DD7E00	1630		LD	A,(IX+0)
9052	3C	1640		INC	A
9053	76	1650	PAUSE	HALT	
9054	3D	1660		DEC	A
9055	20FC	1670		JR	NZ,PAUSE
9057	3E08	1680	NEWROW	LD	A,8
9059	F5	1690	NEWLIN	PUSH	AF
905A	C5	1700		PUSH	BC
905B	D5	1710		PUSH	DE
905C	7E	1720	NEWBYT	LD	A,(HL)
905D	23	1730		INC	HL
905E	B6	1740		OR	(HL)
905F	23	1750		INC	HL
9060	CB74	1760		BIT	6,H
9062	2803	1770		JR	Z,ROM
9064	210000	1780		LD	HL,0
9067	EB	1790	ROM	EX	DE,HL
9068	A6	1800		AND	(HL)
9069	EB	1810		EX	DE,HL
906A	12	1820		LD	(DE),A

9D6B	1C	1830		INC	E
9D6C	200A	1840		JR	NZ,FNOBYT
9D6E	14	1850		INC	D
9D6F	7A	1860		LD	A,D
9D70	E607	1870		AND	7
9D72	2004	1880		JR	NZ,FNOBYT
9D74	7A	1890		LD	A,D
9D75	D608	1900		SUB	8
9D77	57	1910		LD	D,A
9D78	10E2	1920	FNOBYT	DJNZ	NEWBYT
9D7A	D1	1930		POP	DE
9D7B	7A	1940		LD	A,D
9D7C	C608	1950		ADD	A,8
9D7E	57	1960		LD	D,A
9D7F	C1	1970		POP	BC
9D80	F1	1980		POP	AF
9D81	3D	1990		DEC	A
9D82	20D5	2000		JR	NZ,NEWLIN
9D84	7A	2010		LD	A,D
9D85	D640	2020		SUB	64
9D87	57	2030		LD	D,A
9D88	7B	2040		LD	A,E
9D89	C650	2050		ADD	A,80
9D8B	5F	2060		LD	E,A
9D8C	300A	2070		JR	NC,FNDROW
9D8E	14	2080		INC	D
9D8F	7A	2090		LD	A,D
9D90	E607	2100		AND	7
9D92	2004	2110		JR	NZ,FNDROW
9D94	7A	2120		LD	A,D
9D95	D608	2130		SUB	8
9D97	57	2140		LD	D,A
9D98	0D	2150	FNDROW	DEC	C
9D99	20BC	2160		JR	NZ,NEWROW
9D9B	D1	2170		POP	DE
9D9C	C1	2180		POP	BC
9D9D	F1	2190		POP	AF
9D9E	3D	2200		DEC	A
9D9F	20AB	2210		JR	NZ,FADE
9DA1	3E00	2220		LD	A,0
9DA3	DD5E04	2230		LD	E,(IX+4)
9DA6	DD6E06	2240		LD	L,(IX+6)
9DA9	DD5608	2250		LD	D,(IX+8)
9DAC	DD660A	2260		LD	H,(IX+10)
9DAF	1D	2270		DEC	E
9DB0	2D	2280		DEC	L
9DB1	15	2290		DEC	D

9DB2	25	2300		DEC	H
9DB3	CD44BC	2310		CALL	FILBOX
9DB6	CD09B9	2320		CALL	ROMDIS
9DB9	C9	2330		RET	
9DBA	FE03	2340	LWRAP	CP	3
9DBC	C0	2350		RET	NZ
9DBD	DD6E02	2360		LD	L,(IX+2)
9DC0	2D	2370		DEC	L
9DC1	DD6604	2380		LD	H,(IX+4)
9DC4	25	2390		DEC	H
9DC5	CD1ABC	2400		CALL	CHRPOS
9DC8	DD4600	2410		LD	B,(IX+0)
9DCB	CB20	2420		SLA	B
9DCD	05	2430		DEC	B
9DCE	0E08	2440		LD	C,8
9DD0	E5	2450	ROWL	PUSH	HL
9DD1	C5	2460		PUSH	BC
9DD2	CB26	2470		SLA	(HL)
9DD4	7E	2480		LD	A,(HL)
9DD5	F5	2490		PUSH	AF
9DD6	5D	2500	BYTEL	LD	E,L
9DD7	54	2510		LD	D,H
9DD8	2C	2520		INC	L
9DD9	200A	2530		JR	NZ,HLLOK
9DDB	24	2540		INC	H
9DDC	7C	2550		LD	A,H
9DDD	E607	2560		AND	7
9DDF	2004	2570		JR	NZ,HLLOK
9DE1	7C	2580		LD	A,H
9DE2	D608	2590		SUB	8
9DE4	67	2600		LD	H,A
9DE5	1A	2610	HLLOK	LD	A,(DE)
9DE6	CB26	2620		SLA	(HL)
9DE8	3804	2630		JR	C,SETBTL
9DEA	CBA7	2640		RES	4,A
9DEC	1802	2650		JR	TSTBTL
9DEE	CBE7	2660	SETBTL	SET	4,A
9DF0	CB66	2670	TSTBTL	BIT	4,(HL)
9DF2	2802	2680		JR	Z,BITLOK
9DF4	CBC7	2690		SET	0,A
9DF6	12	2700	BITLOK	LD	(DE),A
9DF7	10DD	2710		DJNZ	BYTEL
9DF9	F1	2720		POP	AF
9DFA	3804	2730		JR	C,WRAPL
9DFC	CBA6	2740		RES	4,(HL)
9DFE	1802	2750		JR	ENDFXL
9E00	CBE6	2760	WRAPL	SET	4,(HL)

9E02	CB67	2770	ENDPXL	BIT	4,A
9E04	2802	2780		JR	Z,ROWFNL
9E06	CBC6	2790		SET	0,(HL)
9E08	C1	2800	ROWFNL	POP	BC
9E09	E1	2810		POP	HL
9E0A	7C	2820		LD	A,H
9E0B	C608	2830		ADD	A,8
9E0D	67	2840		LD	H,A
9E0E	0D	2850		DEC	C
9E0F	20BF	2860		JR	NZ,ROWL
9E11	C9	2870		RET	
9E12	FE03	2880	RWRAP	CP	3
9E14	C0	2890		RET	NZ
9E15	DD6E02	2900		LD	L,(IX+2)
9E18	2D	2910		DEC	L
9E19	DD4E00	2920		LD	C,(IX+0)
9E1C	DD7E04	2930		LD	A,(IX+4)
9E1F	81	2940		ADD	C
9E20	D602	2950		SUB	2
9E22	67	2960		LD	H,A
9E23	CD1ABC	2970		CALL	CHRPDS
9E26	23	2980		INC	HL
9E27	41	2990		LD	B,C
9E28	CB20	3000		SLA	B
9E2A	05	3010		DEC	B
9E2B	0E08	3020		LD	C,8
9E2D	E5	3030	ROWR	PUSH	HL
9E2E	C5	3040		PUSH	BC
9E2F	CB3E	3050		SRL	(HL)
9E31	7E	3060		LD	A,(HL)
9E32	F5	3070		PUSH	AF
9E33	5D	3080	BYTER	LD	E,L
9E34	54	3090		LD	D,H
9E35	7D	3100		LD	A,L
9E36	2D	3110		DEC	L
9E37	B7	3120		OR	A
9E38	200A	3130		JR	NZ,HLROK
9E3A	7C	3140		LD	A,H
9E3B	25	3150		DEC	H
9E3C	E607	3160		AND	7
9E3E	2004	3170		JR	NZ,HLROK
9E40	7C	3180		LD	A,H
9E41	C608	3190		ADD	A,8
9E43	67	3200		LD	H,A
9E44	1A	3210	HLROK	LD	A,(DE)
9E45	CB3E	3220		SRL	(HL)
9E47	3804	3230		JR	C,SETBTR

9E49	CB9F	3240		RES	3,A
9E4B	1802	3250		JR	TSTBTR
9E4D	CBDF	3260	SETBTR	SET	3,A
9E4F	CB5E	3270	TSTBTR	BIT	3,(HL)
9E51	2802	3280		JR	Z,BITROK
9E53	CBFF	3290		SET	7,A
9E55	12	3300	BITROK	LD	(DE),A
9E56	10DB	3310		DJNZ	BYTER
9E58	F1	3320		POP	AF
9E59	3804	3330		JR	C,WRAPR
9E5B	CB9E	3340		RES	3,(HL)
9E5D	1802	3350		JR	ENDPXR
9E5F	CBDE	3360	WRAPR	SET	3,(HL)
9E61	CB5F	3370	ENDPXR	BIT	3,A
9E63	2802	3380		JR	Z,ROWFNR
9E65	CBFE	3390		SET	7,(HL)
9E67	C1	3400	ROWFNR	POP	BC
9E68	E1	3410		POP	HL
9E69	7C	3420		LD	A,H
9E6A	C608	3430		ADD	A,8
9E6C	67	3440		LD	H,A
9E6D	0D	3450		DEC	C
9E6E	20BD	3460		JR	NZ,ROWR
9E70	C9	3470		RET	
9E71	DD4E00	3480	GRPNT4	LD	C,(IX+0)
9E74	DD6E02	3490		LD	L,(IX+2)
9E77	DD6603	3500		LD	H,(IX+3)
9E7A	DD5E04	3510		LD	E,(IX+4)
9E7D	DD5605	3520		LD	D,(IX+5)
9E80	0602	3530		LD	B,2
9E82	D5	3540	NXTCDE	PUSH	DE
9E83	E5	3550		PUSH	HL
9E84	C5	3560		PUSH	BC
9E85	CDC0BB	3570		CALL	MOVABS
9E88	C1	3580		POP	BC
9E89	79	3590		LD	A,C
9E8A	0C	3600		INC	C
9E8B	C5	3610		PUSH	BC
9E8C	CDFCBB	3620		CALL	GRACHR
9E8F	C1	3630		POP	BC
9E90	79	3640		LD	A,C
9E91	0C	3650		INC	C
9E92	C5	3660		PUSH	BC
9E93	CDFCBB	3670		CALL	GRACHR
9E96	C1	3680		POP	BC
9E97	E1	3690		POP	HL

9E98	D1	3700	POP	DE
9E99	3E10	3710	LD	A,16
9E9B	2B	3720	MOVEGR	DEC HL
9E9C	3D	3730	DEC	A
9E9D	20FC	3740	JR	NZ,MOVEGR
9E9F	10E1	3750	DJNZ	NXTCDE
9EA1	C9	3760	RET	
		3770	END	

OUTs, PEEKs, POKEs

y CALLs útiles

El mamotréptico título lo dice todo. Este capítulo está dedicado a un número de útiles rutinas y posibilidades que pueden ser accedidas mediante los comandos arriba indicados, todos estos comandos y su uso han sido ya explicados, pero únicamente dando un pequeño número de ejemplos. Aquí tenemos el resto.

- 1) Hay un número de sistemas para impedir que los demás tengan acceso a nuestros listados pero la mayoría están relacionados con la imposibilidad de 'romper' el programa. Este **POKE** permite romper el programa, pero no listarlo. Esto es debido a que **POKE** introduce un carácter de control en la primera línea del programa.
Escribir la línea 1 de la forma siguiente: **1 REM*****
Teclear ahora **POKE &176,255**
A partir de este momento, tu programa debe ser imposible de listar.
- 2) Un rápido **CALL** aquí para limpiar el buffer del teclado:
CALL &BFF9, &C9, &1CED, &CF00.
- 3) No sé cuántos de vosotros habréis encontrado molesto tener que establecer un **LOAD** o **CAT** cada vez que se quiere posicionar la cinta del cassette usando el botón de **PLAY**. **OUT 512,6** habilita botón de **PLAY**, permitiéndote que lo uses nuevamente, mientras que **OUT 512,0** lo deshabilita de nuevo.
- 4) Mediante el comando **OUT 256,N** puede obtenerse una forma simple de scroll horizontal, donde N es la posición del carácter. Prueba **OUT 256,20** para hacer un desplazamiento desde el centro de la pantalla en modo 1. El scroll produce un efecto de rotación de la pantalla sobre su eje longitudinal, donde los caracteres que desaparecen por el lado izquierdo reaparecen por el derecho.
- 5) Teclea **CALL &BB7E** para anular el cursor de la pantalla. Para activarlo usa **CALL &BB18**.
- 6) **CALL &BB18** es igual al comando **BASIC**:
WHILE INKEY\$ = "" : WEND
Tanto la línea de programa como la llamada (**CALL**) esperan que sea pulsada una tecla.
- 7) **CALL &BD19** realiza una acción conocida como **frame flyback**. Esto simplemente da un mejor aspecto y un movimiento más suave a la animación de tus pantallas.

Ensambladores y Lenguaje

Ensamblador

Te habrás dado cuenta que en este libro no únicamente están incluidos listados de **BASIC**, sino que también los hay en **Lenguaje Ensamblador**. Para aquéllos que no están familiarizados con este lenguaje, existe un método simple de explicar las distintas instrucciones en Código Máquina que hay disponibles. Por ejemplo, una instrucción que hace regresar al ordenador desde el Código Máquina al **BASIC**. Esta instrucción es mostrada en los programas cargadores **BASIC** como **C9** cuando estos programas utilizan la numeración hexadecimal (base 16). El código equivalente en ensamblador para esta instrucción es **RET**, que obviamente es una abreviatura de **RETURN**. Echa un vistazo a las listas en **Lenguaje Ensamblador** y comprobarás que esta instrucción está en penúltimo lugar.

Si quieres obtener más información sobre el **Lenguaje Ensamblador**, te sugiero que mires una de las listas mencionadas en la bibliografía. Lo que sí estoy interesado en mencionar aquí es otro grupo de instrucciones. Este grupo puede ser visto en los listados de **Lenguaje Ensamblador** y son instrucciones tales como **END**, **ORG**, **DEFM** y **PRNT**, todas ellas son comandos usados por el ensamblador (el programa que te permite escribir Código Máquina en **Lenguaje Ensamblador**). El uso de un ensamblador hace más fácil la escritura del Código Máquina, así como seguir el programa escrito por otra persona.

Estos comandos del ensamblador, o directores como también son conocidos, ayudan al programador de Código Máquina y deben ser incluidos en los listados en **Lenguaje Ensamblador**. Para que no te confundas cuando mires un listado en **Lenguaje Ensamblador**, abajo tienes detallados estos comandos:

ORG END EQU DEFL DEFB DEFW
DEFW DEFS DEFM PRNT ENT

BIBLIOGRAFIA

Entre los numerosos libros que tratan la programación en Código Máquina del Z-80 en general y específicamente del Amstrad, nosotros hemos elegido algunos títulos para consultar.

Amstrad Machine Language For The Absolute Beginner

de Joe Pritchard, Melbourne House. ISBN 0 86161 1934

En nuestra opinión, la mejor guía para iniciarse en este sujeto (¡por supuesto, antes de que la nuestra fuese publicada!). En serio, el libro es una completa guía para el que se inicia en Código Máquina con el Amstrad. A diferencia de otras muchas guías de Código Máquina, ésta es muy específica para el Amstrad, incluyendo un número de las 'llamadas' (calls) disponibles en la **ROM**. El libro tiene una pobre presentación; justo los listados de impresora y algún que otro diagrama. En el texto hay algunos fallos, pero lo más preocupante son los errores existentes en los listados de programas. A pesar de todo, una buena compra para aquéllos que deseen ir más allá de los primeros inicios con el Código Máquina.

Amstrad CPC 464 Machine Code

de Steve Webb. Virgin Books. ISBN 0 86369 082 3

Este delgado libro es ideal para el novel en Código Máquina ¡quién piensa que **PUSHing** y **POPing** son formas nuevas de baile! En lugar de lanzarse dentro de complicadas explicaciones sobre los conceptos y principios del Código Máquina, el libro se mantiene dentro de un nivel más práctico introduciéndote en los **Z80 Op codes** (códigos de operación de Z80) que tú puedes usar. Es posible que puedas necesitar otro libro para suplementar las enseñanzas de éste, pero para el principiante total merece la pena considerar su compra.

The Concise Firmware Specification

Order Code: SOFT 158. Amsoft.

Con un precio de 20 libras (unas 4.000 Ptas.) se trata de un folleto de hojas sueltas que puede hacerte pensar que Amsoft no trata bien al público. No es este el caso. Por ese dinero, obtienes una referencia técnica completa donde están todos los detalles de las rutinas contenidas en la **ROM**, cómo trabajan y qué parámetros requieren. Un número de rutinas de nuestro libro habrían sido bastante más difíciles de escribir si no hubiésemos tenido esta guía.

Disposición del Teclado

El siguiente diagrama muestra el código numérico de todas las teclas:

TECLADO PRINCIPAL

66	64	65	57	56	49	48	41	40	33	32	25	24	16	79
68	67	59	58	50	51	43	42	35	34	27	26	17	18	
70	69	60	61	53	52	44	45	37	36	29	28	19		
21	71	63	62	55	54	46	38	39	31	30	22	21		
47											23			

TECLADO NUMERICO

10	11	3
20	12	4
13	14	5
15	7	6

TECLAS DE CURSOR

	0	
8	9	1
	2	

Rutinas

EN CODIGO

MAQUINA

para su

Amstrad

EL BASIC es el lenguaje para ordenadores más sencillo de aprender, y usualmente es dominado de forma rápida por el programador novel. Sin embargo, al tiempo que progresan, muchos programadores encuentran que partes de sus programas que requieren una rápida ejecución, tales como desplazamientos rápidos de la pantalla, no pueden ser realizados en BASIC.

Ahora tú podrás controlar el verdadero poder de tu Amstrad mediante el Código Máquina con impresionantes resultados.

Este libro contiene 20 rutinas, cada una escrita de forma que pueden ser incorporadas en tus propios programas con el mínimo esfuerzo. Sin que sean necesarios conocimientos previos de lenguaje Ensamblador ni Código Máquina.

Cada rutina es totalmente descrita mediante ejemplos y explicaciones de su uso. Con este libro serás capaz de lograr extraordinarios efectos tales como suaves desplazamientos de la pantalla, rotación de caracteres multicolores, nuevos comandos de sonido, caracteres itálicos y más efectos -rutinas que en BASIC son imposibles de realizar, y que mejorarán tus "cansados" programas, tanto si son juegos, educativos o aplicaciones de negocios.

Sobre nosotros

Son bienvenidas las solicitudes de catálogos, así como otro tipo de propuestas.

Escriban a:

RA-MA
Editorial
Crta. de Canillas, 144
28043 MADRID

ISBN 84-86381-12-6