ENGLISCH · GERMER
SCHEUSE · THRUN

CPC 464/6128

CONSEJOS Y TRUCOS

Un pozo de ciencia para el usuario del CPC 464/6128

UN LIBRO DATA BECKER
EDITADO POR FERRE MORET, S.A.

ENGLISCH · GERMER
SCHEUSE · THRUN

CPC 464/6128

CONSEJOS Y TRUCOS

Un pozo de ciencia para el usuario del CPC 464/6128

UN LIBRO DATA BECKER
EDITADO POR FERRE MORET, S.A.

Los programas y utilidades que se encuentran en este libro pueden ser adquiridos en Cassette o Diskette, solicitándolos directamente a FERRE MORET, S.A., o a sus distribuidores.

Este libro ha sido traducido por la Sra. Montserrat Rubbel Pons, experta conocedora del CPC 464 y 6128.

Imprime: APSSA, ROCA UMBERT, 26 - L'HOSPITALET DE LL. (Barcelona)

ISBN 84-86437-17-2
Depósito legal B-39.303/85

Advertencia importante

Los circuitos, procedimientos y programas reproducidos en este libro, son divulgados sin tener en cuenta el estado de las patentes. Están destinados exclusivamente para el uso amateur o docente, y no pueden ser utilizados para fines comerciales.

Todos los circuitos, datos técnicos y programas de este libro, han sido elaborados o recopilados con el mayor cuidado por el autor y reproducidos utilizando medidas de control eficaces. No obstante, es posible que exista algún error. FERRE MORET, S.A. se vé por tanto obligada a advertirles, que no puede asumir ninguna garantía, ni responsabilidad jurídica, ni cualquier otra responsabilidad sobre las consecuencias atribuibles a datos erróneos. El autor les agradecerá en todo momento la comunicación de posibles fallos.

I N D I C E

CAPITULO 1 PROLOGO

Cuando recibimos la oferta de escribir este libro, saltamos naturalmente de alegría.

La alegría fue mayor todavía, cuando conocimos los datos de nuestro nuevo ordenador. Cuando lo tuvimos hace aproximadamente unos 3 meses en "nuestras manos" y después de haber trabajado un par de horas con él, acabamos entusiasmados.

Después de algún tiempo vimos con claridad que este ordenador podría definirse como el que muchos habían esperado.

Es al mismo tiempo apropiado para el estudiante, el matemático, el colaborador, el ambicioso, el experto, el que se interesa comercialmente, el programador de Basic, el caprichoso en gráficos, el fanático del Sonido, etc., etc.

El amplio espectro de aplicación se refleja también en el gran campo de temas que en este libro son tratados. Para demostrar que no todo es simple teoría, hemos acompañado este libro con numerosos programas ejemplo (todos ellos comprobables).

Si lee este libro, quisiéramos añadirle el siguiente consejo:
Piense que un ordenador es su buen compañero, trátelo pues como se merece y se lo agradecerá (el nuestro no nos ha dejado ni una sóla vez en la estacada).

Düsseldorf, Agosto de 1984

Los autores

C A P I T U L O 2 G R A F I C A S

2.1 INTRODUCCION

La gráfica es una de las partes más fascinantes del ordenador. Por ello, los constructores del CPC han dado tanta importancia a esta parte de su ordenador. Su CPC 464 y 6128 tiene las siguientes posibilidades gráficas:

Resolución de 640*200 puntos
(=128000 puntos individuales)
20-40-80 Caracteres en 25 líneas
16-4-2 Representación unísona de colores
27 Colores distintos
Instrucciones gráficas de gran rendimiento
Tecnología Window
Caracteres autodefinidos
Caracteres gráficos

El desarrollo en el campo de gráficas ha hecho de los ordenadores el instrumento universal de hoy en día. El hombre es un ser orientado ópticamente. Por éso una serie de números no es fácilmente comprensible para el hombre. Si recibe estos números reflejados gráficamente, captará con mayor facilidad la información que se pretende dar. Ya que el resultado de los cálculos del ordenador consta de muchas cifras, es necesaria una preparación gráfica. Los medios y posibilidades que tiene su CPC en este campo se los vamos a presentar detalladamente en este capítulo.

Nosotros profundizamos nuestras aclaraciones teóricas, como siempre, con ejemplos prácticos. Usted mismo deberá también probar y experimentar.

2.2 LA RESOLUCION DE GRAFICAS - GRAFICAS PARA SER OBSERVADAS

La resolución es la palabra tópico con la que todos los fabricantes intentan revalorar su ordenador. ¿ Qué es esta resolución ? Para poderles aclarar este concepto, tomaremos un ejemplo del tiempo en que el lápiz y el papel gobernaban el mundo. Es muy natural que con un lápiz bien fino pueda realizar mejores diseños que con uno de grueso. Lo mismo ocurre con la resolución. Mediante una resolución más elevada es posible crear gráficos más detallados. Esto podrá realizarse con un mayor número de puntos individuales que Vd. mismo podrá consultar. En su CPC tiene un retículo de 640 puntos en sentido horizontal y 200 puntos en sentido vertical, con lo que podrá fijar 128.000 puntos. Con ellos podrá realizar buenos gráficos de barras, circulares o de líneas. Es evidente que también podrá diseñar dibujos, esbozar juegos y muchas otras cosas.

A continuación les escribiremos cuatro programas que le aclararán las posibilidades de gráficas de su nuevo ordenador. Copie los ejemplos y mediante el RUN déjese llevar al país de las maravillas de las gráficas. En el caso de que los ejemplos se le hiciesen aburridos, modifique los programas o bien prográmelos totalmente de nuevo. Sin embargo, antes de que Vd. efectúe dichas modificaciones, debería leer todo este capítulo hasta el final.

2.2.1 GRAFICAS PARA SER OBSERVADAS

La parte que sigue a este subcapítulo consta de pequeñas rutinas que crean gráficas. Vd. puede únicamente mirárselas o bien encuadrarlas en sus programas.

El primer programa diseña un círculo. Después de introducir origen, radio y anchura de paso tendrá la opción de, o bien diseñar el círculo, o bien rellenar el fondo de color. Si la anchura de paso es demasiado grande, podría ser posible que todo el círculo (fondo) no estuviera rellenado. Por este motivo es preferible elegir la anchura de paso lo más pequeña posible.

```
10 MODE 2
20 INPUT "Origen (X) ";x
30 INPUT "Origen (Y) ";y
40 INPUT "Radio (R) ";r
50 INPUT "Anchura de paso ";s
60 INPUT "Llenar 'F'ondo o 'C'írculo ";a$
70 CLS
80 z=(1 AND a$="f")+(2 AND a$="c")
90 ORIGIN x,y
100 FOR n=1 TO 360 STEP s
110 ex=ex+1
120 xp=r*COS(n)
130 yp=r*SIN(n)
140 PLOT xp,yp,1
150 IF yp=ABS(yp) THEN ey=400-y ELSE ey=y-400
160 IF xp=ABS(xp) THEN ex=640-x ELSE ex=x-640
170 IF z=1 THEN DRAW xp,ey
180 IF z=1 THEN PLOT xp,yp : DRAW ex,yp
190 IF z=2 THEN MOVE 0,0 : DRAW xp,yp
200 NEXT n
210 IF z<>1 GOTO 280
220 FOR n=r TO 640-x
```

```
230 MOVE  n,0 : DRAW  n,400-y
240 MOVE -n,0 : DRAW -n,400-y
250 MOVE  n,0 : DRAW  n,y-400
260 MOVE -n,0 : DRAW -n,y-400
270 NEXT n
280 a$=INKEY$:IF a$="" THEN 280
290 RUN
```

```
Por ejemplo: Origen (x)=320
             Origen (z)=200
             Radio     =30
             Anchura de paso=0.5
```

Como segunda gráfica a ser observada hemos elegido una de tipo seudotridimensional. Del diseño pueden reconocerse muchas imágenes, por ejemplo una pirámide vista desde arriba, o un paso en la profundidad, o también una tela de araña, todo ello según su capacidad de imaginación. Mediante pequeñas modificaciones del programa pueden crearse nuevas imágenes.

```
10 MODE 2
20 ORIGIN 320,200
30 FOR t=0 TO 180 step 10
40 PLOT a,t:DRAW t,-a
50 PLOT-a,t:DRAW-t,-a
60 PLOT 180,180:DRAW 0,0
70 PLOT-180,180:DRAW 0,0
80 PLOT a,t:DRAW -a,t
90 PLOT 180,-180:DRAW 0,0
100 PLOT -180,-180:DRAW 0,0
110 PLOT-t,-a:DRAW t,-a
120 a=a+10
130 NEXT t
```

Si por ejemplo Vd. modifica la línea 30 en:

```
30 FOR T=180 TO 0 STEP -10,
```

conseguirá una imagen completamente distinta.
Realice experimentos con este programa. Conseguirá resultados sorprendentes.

La tercera gráfica se llama calidoscopio. Después de iniciar el programa Vd. comprenderá el por qué. Aquí también entra en juego el factor de la coincidencia. Con el mismo pueden desarrollarse gráficas realmente interesantes.

```
10 MODE 2
20 DEF FN f(x) = INT (RND(1)*x)+1
30 FOR x=1 TO 400 STEP FN f(25)+5
40 PLOT x,0: DRAW 400,x
50 PLOT 0,x: DRAW x,400
60 PLOT x,0: DRAW 0,400-x
70 PLOT 400,x: DRAW 400-x,400
80 PLOT x,0: DRAW 400-x,400
90 PLOT 400,x: DRAW 0,400-x
100 NEXT x
110 LOCATE 55,1
120 PRINT "Otra vez ?"
130 a$=INKEY$:IF a$=""THEN 130
140 IF a$="s" THEN RUN
150 END
```

La cuarta gráfica es un factor del seno circular. El programa diseña, después de introducir una anchura de paso, una curva de seno, pero sin concretar puntos sino con líneas de un punto (0,200) para el punto actual de la curva del seno. La imagen que con ello se crea es nuevamente seudotridimensional. La ilusión óptica se crea mediante la sobrecarga de líneas.

```
10 REM "Seno del círculo"
20 DEG
30 CLG
40 INPUT "Anchura de paso";s
50 DEF FN f(x)=SIN(x)
```

```
60 FOR n=1 TO 640 STEP s
70 x=x+(s*1.125)
80 MOVE 0,200
90 DRAW n,200+FN f(x)*200
100 NEXT n
110 a$=INKEY$: IF a$="" THEN 110
120 RUN
```

Pruebe Vd. este programa con COS(X), con -COS(X) o con -SIN(X).
Es evidente, que la elevada resolución del CPC no podrá utilizarla tan sólo para los llamados ordenadores de arte (arte con el ordenador). Por ejemplo, con la ayuda de un programa pueden realizarse, almacenarse, cargarse, etc. diseños. Vd. mismo puede incluso diseñar su propio escudo sobre pantalla. Si también se monta una rutina de impresión podrá imprimirla y después de enmarcarla, colgarla a la puerta de su "castillo" (es evidente que existen aplicaciones mucho más interesantes).

¿ Se interesa Vd. mucho por las Matemáticas ? Entonces le sería de gran ayuda que su ordenador pudiese ofrecerle una función gráfica. Como Vd. ya sabe, en cuanto a gráficas se refiere pueden valorarse multitud de cosas que de otro modo deberían calcularse laboriosamente.

Para el caso de que Vd. desee programar (por ejemplo programas comerciales), la elevada resolución le será de gran utilidad. Si Vd. por ejemplo desea expresar gráficamente las cuotas de crecimiento de su negocio durante los últimos doce meses, con el CPC le será un juego de niños.

Todas estas excelentes aplicaciones no podrían realizarse si su ordenador tuviese una resolución limitada o bien instrucciones gráficas poco potentes, tal como se da el caso en otros ordenadores.

2.2.2 FIGURAS - LISSAJOUS

Jules Antoin Lissajous, un físico francés, que vivió desde 1822 hasta 1880, hizo un estudio del movimiento de partículas bajo la influencia de movimientos periódicos. Descubrió unas partículas, bajo la influencia de dichos movimientos, se movían en distintas curvas. Hemos escrito un programa que les demostrará el aspecto de dichas curvas.

Después de iniciar el programa deberá efectuar tres entradas:
1.) Frecuencia Y; ésta deberá estar en el campo de 0 a 20.
2.) Frecuencia X; ésta deberá estar en el campo de 0 a 20.
3.) Número de pasos; este valor indica la cantidad de puntos que deberán ser fijados.

```
10 REM Figuras Lissajous
20 DEG
30 MODE 2
40 INPUT "Frecuencia-y ";y
50 INPUT "Frecuencia-x ";x
60 INPUT "Número de pasos ";s
70 CLS
80 FOR a=0 TO 2*PI STEP 2*PI/s
90 PLOT 150*SIN(a*x)+150,100*COS(a*y)+100,1
100 NEXT a
110 END
```

Aclaración del listado:

10	Título
20	Modificación del sistema de cálculo a radián
30	Graduación de la modalidad de la pantalla
40-50	Introducción de la frecuencia Y y X
60	Introducción del numero de pasos
70	Borrado de la pantalla

```
80      Apertura de un bucle
90      Fijación de los puntos
100     Cierre del bucle
110     Final del programa
```

2.3 EDITOR DE GRAFICAS

Para la utilización del programa necesitará un Joystick compatible con un Atari.
Después del inicio del programa podrá mover un pequeño punto, su cursor de gráficas, por toda la pantalla con la ayuda del Joystick. Con la tecla espaciadora, la tecla larga situada debajo de las letras, podrá cambiar entre la modalidad de caracteres y la de movimiento. En la modalidad de movimiento es posible mover el minicursor libremente por toda la pantalla, sin que se olvide de línea alguna. Visto de otro modo, podrá asimismo diseñarse una línea con la tecla "sólo movimiento" con la diferencia, de que el color de esta línea corresponderá exactamente con el color del fondo. Si después de ésto desea borrar algunos puntos o trozos de líneas, deberá conectar a la modalidad de movimiento para pasar por encima de los puntos que desea borrar. Si desea borrar la pantalla, deberá presionar la tecla "C". Una vez haya terminado el diseño podrá almacenar el dibujo mediante la tecla "E". Seguidamente se le preguntará el nombre del diseño y una vez haya entrado dicho dato se le indicará que presione las teclas "REC" y "PLAY". Después se almacenará la gráfica bloque a bloque. Una vez haya finalizado el proceso de almacenamiento se le preguntará si desea continuar diseñando o bien si desea finalizar el programa.

Si desea cargar un diseño, deberá entrar los siguientes datos:

```
10 WINDOW 1,79,24,25:LOAD"
```

Inicie el programa con "RUN" y su diseño será cargado.
Esto evitará que su gráfica sea destrozada por la escritura y que las indicaciones aparezcan únicamente en las últimas dos líneas.

Antes de finalizar añadiremos dos instrucciones: Cuando haya terminado el programa, entre la instrucción "Mode 2". Esta guardará nuevamente las ventanas que se habían definido antes y borrará la gráfica.
Lamentablemente al imprimir este libro no disponíamos todavía del propio Joystick del Armstrad. Este suministra otos valores que los tipos compatibles del Atari. Si posee un Joystick Armstrad, deberá adaptar correspondientemente los valores de consulta en las líneas 90, 120 y 130 hasta 160 de su Joystick.

```
10 MODE 2
20 CLS
30 GOSUB 240
40 x=1:y=1:z=1
50 PLOT x,y,1
60 jo=JOY(0): IF jo<>0 THEN 80
70 a$=INKEY$: IF a$="" THEN GOTO 70
80 PLOT x,y,z
90 y=y+(1 AND jo=1)-(1 AND jo=2)
100 IF y<0 THEN y=0
110 IF y>367 THEN y=367
120 x=x+(1 AND jo=8)-(1 AND jo=4)
130 IF jo=9 THEN x=x+1: y=y+1
140 IF jo=10 THEN x=x+1: y=y-1
150 IF jo=6 THEN x=x-1: y=y-1
160 IF jo=5 THEN x=x-1: y=y+1
170 IF x<0 THEN x=0
180 IF x>639 THEN x=639
190 IF z=0 AND a$=" " THEN z=1: GOTO 210
200 IF z=1 AND a$=" " THEN z=0
210 IF a$="e" THEN GOTO 280
220 IF a$="c" THEN CLS: GOTO 10
230 GOTO 50
240  REM  Colocar  el  minicursor  en  el  ángulo  inferior
       izquierdo
250 ORIGIN 0,33
```

```
260 WINDOW #0,1,79,24,25
270 RETURN
280 WINDOW SWAP 0
290 INPUT "Qué nombre recibirá el cuadro ?",a$
300 SPEED WRITE 1
310 SAVE a$,b,49152,16383
320 PRINT "Nuevo 'G'ráfico o 'F'inal"
330 in$=INKEY$: IF in$="g" THEN RUN
340 IF in$<>"f" THEN GOTO 330
```

2.3.1 EDITOR EN MULTICOLOR

Basándonos sobre el editor de gráficas del apartado anterior, hemos creado aquí una versión ampliada. En general se supone, que para este programa de caracteres necesitará un Joystick compatible con el Atari.
Después de haber iniciado el programa se le preguntará la modalidad en la que desea diseñar. Aquí deberá tenerse en cuenta que la resolución de la pantalla del diseño disminuirá al aumentar el número de colores. Si en la modalidad "0" no le gustan los 14 colores o no son suficientes, entonces podrá elegir antes de iniciar el programa, otros colores para los lápices de diseño. Esto podrá realizarse muy fácilmente mediante la instrucción "INK". La tecla espaciadora nos sirve en este programa como conector de colores. Cada vez que se active la tecla, el programa pasará al siguiente color. En el ángulo inferior izquierdo de la pantalla podrá informarse siempre sobre el color con el que en aquel momento se está trabajando. Tenga presente, que el fondo también es un color con el que Vd. puede diseñar o borrar. Además, también es necesario elegir el color del fondo en el caso de que sólo quiera moverse sin olvidarse de alguna línea.

Tal como ocurre en el editor normal, aquí también podrá almacenar su diseño. Para ello deberá activar la tecla "E". El programa le preguntará el nombre del diseño, que después almacenará en un Cassette.

Con la tecla "C" podrá borrarse la pantalla.

Los dibujos almacenados sobre Cassette podrán volverse a cargar mediante la tecla "L". El programa le preguntará aquí también el nombre del diseño que desea seguir tratando. Esto es posible ya que el programa, después del proceso de carga, vuelve a la modalidad normal de caracteres.

```
10 CLS: MODE 1
20 PRINT " Mode 0 = 14 Colores "
30 PRINT " Mode 1 =  4 Colores "
40 PRINT " Mode 2 =  2 Colores "
50 INPUT " Qué Modalidad ( 0,1,2 ) ";mx
60 IF mx = 0 THEN mo = 13
70 IF mx = 1 THEN mo = 3
80 IF mx = 2 THEN mo = 1
90 IF MX > 2 THEN GOTO 50
100 MODE mx
110 CLS
120 GOSUB 340
130 X=1: Y=1: Z=0
140 PLOT X,Y,1
150 JO=JOY(0): IF JO<>0 THEN GOTO 170
160 A$=INKEY$: IF A$="" THEN GOTO 160
170 PLOT X,Y,Z
180 Y=Y+(1 AND JO=1)-(1 AND JO=2)
190 IF Y<0 THEN Y=0
200 IF Y>367 THEN Y=367
210 X=X+(1 AND JO=8)-(1 AND JO=4)
220 IF JO=9 THEN X=X+1: Y=Y+1
230 IF JO=10 THEN X=X+1: Y=Y-1
240 IF JO=6 THEN X=X-1: Y=Y-1
250 IF JO=5 THEN X=X-1: Y=Y+1
260 IF X<0 THEN X=0
270 IF X>639 THEN X=639
280 IF A$=" " THEN Z=Z+1: IF Z>MO THEN Z=0
290 IF A$=" " THEN FOR Q1=1 TO 6: FOR Q2=1 TO 3: PLOT
Q1,Q2,Z: NEXT Q2: NEXT Q1
300 IF A$="E" THEN GOTO 380
310 IF A$="C" THEN CLS: GOTO 70
320 IF A$="1" THEN GOTO 480
330 GOTO 140
340 REM COLOCAR MINICURSOR EN EL ANGULO INFERIOR IZQUIERDO
350 ORIGIN 0,33
360 WINDOW #0,1,79,24,25
370 RETURN
```

```
380 WINDOW SWAP 0
390 INPUT "COMO SE LLAMARA EL CUADRO ";CUADRO$
400 CUADRO$=CUADRO$+".PIC"
410 SPEED WRITE 1
420 SAVE CUADRO$,B,49152,16383
430 PRINT "NUEVO G'RAFICO, F'INAL O C'ONTINUACION "
440 IN$=INKEY$: IF IN$="" THEN GOTO 440
450 IF  IN$="C" THEN  PRINT "
": GOTO 120
460 IN$=INKEY$: IF IN$="G" THEN RUN
470 IF IN$<>"F" THEN GOTO 460
480 REM LEER CUADRO
490 INPUT "QUE CUADRO ";CUADRO$
500 CUADRO$=CUADRO$+"PIC."
510 WINDOW 1,79,24,25: LOAD CUADRO$
520 PRINT "
530 GOTO 120
```

2.4 PLOTTER DE FUNCIONES

Un Plotter de funciones es uno de los medios más importantes de un matemático, pudiendo ser también de gran ayuda para el estudiante en el momento de realizar sus deberes de matemáticas. Dicho Plotter le facilitará la entrada de una función en la línea 50, que será trazada después de introducirla en el campo X. A continuación el proceso del programa: En primer lugar se requieren datos sobre el valor máximo y mínimo de X y la anchura de paso. Después de entrar dichos datos se calculará el máximo y mínimo de los valores de la función, con el fin de poder diseñar a escala el eje Y. Esto se realiza en la subrutina a partir de la línea 370. Aquí se probarán veinte valores para poder calcular el valor máximo y mínimo de Y. Después del salto de retroceso se diseñará el sistema de coordenadas, dándole un nombre. A continuación se diseñará la función. En primer lugar se pondrá la rama negativa y luego la positiva. Una vez terminado el diseño, el programa se esperará hasta que Vd. haya activado una tecla.

La función utilizada da por resultado una parábola, cuya parte negativa se encuentra doblada sobre el eje X. Efectúe la prueba tomando como mínimo (-2), como máximo (+2) y como anchura de paso (0.1). Después de un tiempo de cálculo relativamente corto se trazará una curva bastante sorprendente.

Como variable utilice siempre la letra X.

```
10 REM ************** PLOTTER DE FUNCIONES **************
20 REM
30 MODE 2
40 CLEAR
50 DEG
60 DEF FN f(x)=-(x^2)
70 PRINT"                        (El cero no está permitido !)"
80 INPUT  "
Minimo (x) ";a
```

```
90 PRINT"                          (El cero no está permitido !)"
100 INPUT "
Máximo (x) ";b
110 IF  a=0 OR  B=0 THEN PRINT"
EL CERO NO ESTA PERMITIDO !": GOTO 70
120 INPUT "Anchura de paso ";c
130 IF a>b THEN 30
140 GOSUB 410
150 CLG
160 PLOT 0,200: DRAW 640,200:PLOT 320,0:DRAW 320,400
170 LOCATE 1,14
180 mp=320/ABS(a)
190 mq=320/ABS(b)
200 IF mp=mq THEN pr=a
210 IF mp<mq THEN pr=a
220 IF mp>mq THEN pr=-b
230 PRINT pr
240 pr$=STR$(ABS(pr))
250 LOCATE (80-LEN (pr$)),14
260 PRINT pr$
270 de$=STR$(de(1))
280 LOCATE 39 -LEN(de$),1
290 PRINT de$
300 de$=STR$(-de(1))
310 LOCATE 39 - LEN (de$),25
320 PRINT de$
330 mn=200/de(1)
340 FOR s=0.1 TO a STEP -c
350 PLOT 320+s*mp,200+FN f(s)*mn,1
360 NEXT s
370 FOR n=0.1 TO b STEP c
380 PLOT 320+n*mq,200+FN f(n)*mn,1
390 NEXT n
400 a$=INKEY$:IF a$="" THEN 400 ELSE 30
410 m1= ABS(a) + ABS(b)
420 det=m1/20
430 m1=m1+2
440 DIM de(22)
450 FOR i=a TO b STEP det
460 var=var+1
```

```
470 de(var)=FN f(i)
480 NEXT i
490 de(21)=FN f(a):de(22)=FN f(b)
500 FOR j=21 TO 1 STEP -1
510 FOR i=1 TO j
520 IF ABS(de(i+1))<ABS(de(i)) THEN 560
530 de=de(i+1)
540 de(i+1)=de(i)
550 de(i)=de
560 NEXT i
570 NEXT j
580 PRINT"Posible función máxima y mínima de este campo:",
de(1)
590 de(1)=ABS(de(1))
600 PRINT "Pulsar tecla"
610 a$=INKEY$:IF a$="" THEN 610
620 RETURN
```

2.5 COLORES Y CARACTERES

Su CPC puede denominarse tanto en el campo de los colores como en el de los caracteres como el superordenador.

Lamentablemente el mayor número de colores (27) no puede alcanzarse en cada modalidad. Así pues al programar tenga presente dicha limitación, ya que posiblemente podría extrañarse sobre un "IMPROPER ARGUMENT". La cantidad de colores que Vd. puede utilizar en una modalidad está detalladamente especificada en su manual. Lea pues estos apartados detenidamente.

Las distintas modalidades (caracteres 20,40 y 80) son asímismo comentadas en el manual al caso. Preste pues también atención a estos capítulos.

A continuación expondremos nuevamente las instrucciones para las modalidades y graduación de los colores, ya que éstas no se han especificado en el manual en forma de tabla.

Modalidad 0	Borra la pantalla y conecta a la modalidad de 20 caracteres.
Modalidad 1	Borra la pantalla y conecta a la modalidad de 40 caracteres.
Modalidad 2	Borra la pantalla y conecta a la modalidad de 80 caracteres.
Border	Fija el color del margen (hasta el 27). Si se indican 2 colores cambiará el color del cuadro entre ambos.
Ink	Asigna a un canal determinado (primer valor) un color (segundo valor).

Pen	Asigna al escribir (primer valor) un color a un canal (segundo valor).
Paper	Asigna al fondo de un canal (primer valor) un color (segundo valor).

2.6 INSTRUCCIONES DE GRAFICAS DE GRAN RENDIMIENTO

En el próximo capítulo se aclararán las siguientes instrucciones:

PLOT y PLOTR
DRAW y DRAWR
MOVE y MOVER
ORIGIN
TEST y TESTR
XPOS y YPOS

La primera instrucción de nuestra lista se llama PLOT X,Y y fija un punto en las coordenadas absolutas X,Y. Absoluto quiere decir en este caso, que la posición del punto depende únicamente del origen actual. En un caso normal el origen está en 0,0 en el ángulo inferior izquierdo de la pantalla. PLOTR X,Y fija asimismo un punto pero relativamente a las coordenadas X,Y. Relativo quiere decir, que los valores de X e Y se calculan relativamente con respecto a la posición del cursor de gráficas.

Una instrucción muy importante es DRAW X,Y. Esta instrucción traza una línea desde la posición del cursor hasta las coordenadas absolutas X,Y. DRAWR traza asímismo una línea hacia las coordenadas X,Y, calculando aquí también las coordenadas relativamente con respecto a la posición del cursor de gráficas.

Para poder mover el cursor de gráficas sin tener que fijar un punto, se utilizará la instrucción MOVE X,Y. Esta mueve el cursor hacia las coordenadas absolutas X,Y hasta el origen. Cuando se utiliza MOVER X,Y, el cursor se moverá hacia las coordenadas X,Y calculadas relativamente hacia la antigua posición.

ORIGIN es la instrucción para cambiar el origen del que se calcularon las coordenadas absolutas. Además, mediante la instrucción ORIGIN puede crearse una ventana de gráficas. La ventana original mide 640 puntos en su anchura y 200

puntos en su altura.

Con la instrucción TEST X,Y podrá leerse el color del punto de las coordenadas X,Y. Con esta instrucción existe asimismo la posibilidad de calcular relativamente las coordenadas. En este caso la instrucción será TESTR.

Para poder llegar a saber la posición actual del cursor de gráficas, su CPC tiene dos funciones extraordinarias: La XPOS y la YPOS. Utilizando PRINT XPOS, YPOS se indicará la posición actual del cursor.

Ahora volvamos nuevamente al concepto "relativo". Para poder calcular la posición real, necesitará únicamente en su instrucción los valores de X e Y para poderlos sumar a los valores X e Y de la posición del cursor, pudiendo así obtener la posición absoluta que Vd. deseaba conseguir.

2.7 TECNOLOGIA WINDOW

Ultimamente la palabra WINDOW se ha convertido en un punto de valoración, tal como ha ocurrido en el capítulo 2.1 sobre la resolución elevada.

¿ Qué son realmente Windows ? El equivalente español para Window es ventana; este concepto indica aproximadamente lo que son los Windows. Un Window es un campo fijo en la pantalla de su ordenador, que podrá tratar como una pantalla independiente. Ordenadores que aplican con sentido la tecnología Window ofrecen al usuario la posibilidad de utilizar varios Windows. Este es el caso de su CPC.

¿ Para qué es buena la tecnología Window ? Aquí vuelve a aparecer una palabra apreciada dentro de la informática que es la Ergonomía. Anteriormente este concepto se refería únicamente al Hardware (o sea, a ordenadores, pantalla, impresora, medios de almacenaje...). Si un sistema demuestra un buen grado de Ergonomía significará, que su configuración, o sea altura del teclado, ángulo de la pantalla, nivel de ruido de la impresora ..., está adaptada de tal manera a las necesidades del hombre, que un trabajo prolongado con dicho sistema no significará una carga corporal demasiado elevada.

Hace ya mucho tiempo que el concepto de Ergonomía se extendió también al Software (o sea, a los programas de ordenadores). Los científicos encontraron que la configuración óptica de un programa, la presentación de datos, los tiempos de respuesta de un sistema y similares tenían una influencia decisiva sobre el cansancio en el trabajo. Además, con un programa gráficamente bien adaptado, puede trabajarse mucho más de prisa y efectivamente mejor que con uno que esté mal creado.

¿ Por qué aplicar Windows para un ordenador, que se ubicará en el sector de los ordenadores personales como

en el caso de su CPC 464 y 6128 ? Aquí podemos acogernos al último punto, o sea, a la efectividad y rapidez. Los dos temas de ordenadores más utilizados (juegos y programas comerciales) se aprovechan mucho de ello.

Los Windows se aplican muy a menudo en los programas comerciales. Esto quedará reflejado en la administración de ficheros del capítulo 7.1. Aquí se utilizan Windows para entrar informaciones en la parte superior de la pantalla, que son continuamente requeridas por el usuario (datos actuales y número de campo). La información no se modificará en dichos Windows. Sin embargo, podrá definir con la instrucción ORIGIN un Window de gráficas. En un cálculo de tablas escrito por Vd. se demostrarán por un lado los datos en forma de tabla y por el otro, en el Window de gráficas definido, se reproducirán en forma de un diagrama de barras o de un diagrama circular.

El segundo campo en el que los Windows pueden ser muy útiles es el de los juegos. Reserve un campo para una calle, que Vd. quiera tener siempre en el borde inferior de la pantalla mediante una instrucción de Window. De este modo no tendrá que diseñar siempre de nuevo la calle al caso, ya que estas cosas requieren mucho tiempo, siendo programadas en Basic, y podrían retardar todo su juego hasta el aburrimiento.

Vd. debería experimentar con la instrucción WINDOW. Una posibilidad la tiene en la administración de ficheros, aunque también podría escribir sus propios pequeños programas. En el manual ya se han impreso algunos programas ejemplo. ¿ Qué tal sería realizar el ejemplo con 8 Windows, en los que se tuvieran 8 gráficas distintas ? O también podrían mover un Window por la pantalla, en el que modificase los valores angulares. Existen realmente muchas posibilidades de aplicación para los Windows.

Con el fin de poderle dar un par de estímulos en la experimentación con los Windows, encontrarán a continuación dos programas bien sencillos. Realice tranquilamente dichos

programas, incluso aunque crea que no tienen mucho sentido. Lo importante es que aprenda a manejar los Windows y así los pueda aplicar de forma óptima en sus programas ulteriores.

1. El programa siguiente crea 8 Windows y los llena con caracteres, que Vd. no podrá identificar por haberse modificado el color de fondo.
Vd. debería modificar los valores de las líneas 500 - 510. Estos son los valores para los Windows. La primera línea es la dirección X (valores entre 1 y 80) y la segunda contiene los valores Y (entre 1 y 25). Evidentemente podría crear también más Windows sobre la pantalla. En este caso debería aumentar los valores en los bucles de 8 y fijarles el valor deseado.

```
10 MODE 2
20 FOR x=1 TO 8
30 READ columna (x)
40 NEXT x
50 FOR y=1 TO 8
60 READ línea (y)
70 NEXT y
80 PAPER 5
90 FOR x=1 TO 8
100 FOR y=1 TO 8
110 WINDOW columna (x), columna (x)+5,línea (y),línea (y)+5
120 PRINT "**********"
130 NEXT y
140 NEXT x
500 DATA 1,80,20,60,30,45,27,50
510 DATA 9,19,12,25,1,3,18,1
```

Aclaración del programa:

10	Ajustar la modalidad de los 80 caracteres.
20-40	Bucle para leer los valores de la dirección X.
50-70	Bucle para leer los valores de la dirección Y.
80	Ajustar el color de fondo.
10-100	Abrir el bucle para X/Y.

110 Creación del Window.
120 Emisión de una cadena de caracteres.
130-140 Final de los bucles.
500-510 Líneas DATA.

2. El segundo programa rellena la pantalla mediante un carácter y borra después una tercera parte bajo la activación de teclas. Una rutina de este tipo puede montarse a la perfección en un programa orientado comercialmente, ya que en el mismo es a menudo necesario borrar partes enteras de la pantalla, cosa que con la ayuda de un WINDOW puede realizarse rápidamente.

```
10 MODE 2
20 WINDOW #2,1,80,1,8
30 WINDOW #3,1,80,9,17
40 WINDOW #4,1,80,17,24
50 FOR n=2 TO 4
60 FOR mn=1 TO 640
70 PRINT "*";
80 NEXT mn
90 NEXT n
100 FOR x=2 TO 4
110 WINDOW SWAP x
120 a$=INKEY$:IF a$="" THEN 120
130 CLS
140 NEXT x
150 MODE 2
```

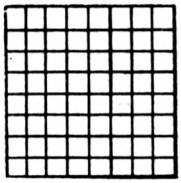
Aclaración del programa:

10	Ajustar la modalidad de los 80 caracteres.
20-40	Definición de 3 Windows.
50-90	Rellenar la pantalla con "*".
100	Apertura de un bucle para cambiar los Windows.
110	Modificar el Window.
120	Esperar a la activación de teclas.
130	Borrar el Window.
140	Final del bucle, se conecta al siguiente Window.
150	Retirar los 3 Windows.

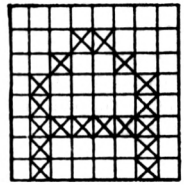
2.8 CARACTERES AUTODEFINIDOS

La última propiedad a ser resaltada en el campo de las gráficas son los llamados caracteres autodefinidos.

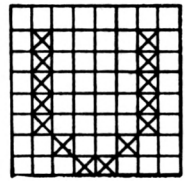
¿ Qué tipo de caracteres son éstos ? Lamentablemente, este tema tan importante ha sido muy brevemente descrito en el manual.
Un carácter consta de puntos fijos y no fijos. El retículo en el que se definen los caracteres se llama matriz. Una de estas matrices la encontrarán en el cuadro 1. Allí se trata de una matriz 8*8 (8 casillas verticales, 8 casillas horizontales). Otros tamaños de matrices que son también usuales son 7*8, 7*9 y cuando se trata de sistemas profesionales incluso 10*10 o bien 12*12. Así pues el CPC está en el campo medio superior. El tamaño de la matriz es interesante, ya que un mayor número de puntos (cuando se trata del indicador correcto de datos) significa poder leer mejor las letras. Esto es lo mismo que con la resolución elevada, cuyas ventajas han sido tratadas en el capítulo 2.1.



Cuadro 1



Cuadro 2



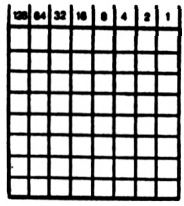
Cuadro 3

En los cuadros 2 y 3 hemos "fijado" puntos en la matriz. Se han creado letras. De este modo se forma cada letra en la pantalla de su ordenador.

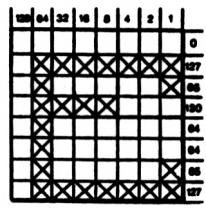
El CPC tiene 255 caracteres de este tipo, de los cuales únicamente podrá modificar 223, es decir, crearlos de nuevo.

¿ Cómo se realiza ésto ? En primer lugar debería coger una hoja de papel cuadriculado y diseñar la matriz 8*8. Una vez haya realizado ésto, probará sobre el papel los puntos que desea fijar y los que no. Todo ello es necesario, ya que sus caracteres autodefinidos podrían verse al final algo oscuros. Ahora se trata de calcular los distintos puntos. De una manera u otra este carácter deberá entrar en su CPC. Esto es algo difícil, ya que necesitamos la llamada aritmética binaria (ésta se ha mencionado en el capítulo 4 sobre la introducción del lenguaje máquina). Observe el cuadro 4.

Allí podrá ver nuevamente una matriz 8*8. Esta ha sido algo ampliada. Sobre la barra vertical encontrará las cifras 1-2-4 - 128. En las líneas horizontales encontrará algo de sitio en el que más adelante insertaremos cifras. En dicha matriz ampliada insertaremos los puntos para su carácter. Observe únicamente la primera línea (línea horizontal) y sume las cifras que están sobre los puntos fijados de esta línea. El resultado de la suma será insertado en el espacio previsto para ello y situado al lado de la línea. Proceda de esta manera con todas las 8 líneas. Las 8 cifras que han recibido son los valores para su carácter. Esto se lo demostraremos nuevamente en el cuadro 5 a base de la letra E.



Cuadro 4



Cuadro 5

Una vez haya localizado estos valores, el resto será relativamente fácil. Ahora únicamente tendrá que reservar espacio para su carácter y ordenarlo mediante la instrucción de una tecla.

El espacio para su carácter (o para varios) lo podrá reservar mediante la instrucción SYMBOL AFTER. A continuación deberá seguirle una cifra en el campo de 32-255. Si elige dicho valor, por ejemplo, con 240 podrá modificar todos los caracteres que tengan un código CHR$ mayor que 239 (es decir 16 partes). Los códigos CHR$ podrá conocerlos del manual de su ordenador. Cuanto menor sea el valor que tome, tanto más espacio de memoria necesitará. Sin embargo, con la gran memoria de que dispone el CPC ésto no será problemático. Además, la cifra más pequeña tiene la ventaja de poder disponer de espacio cuando a Vd. se le ocurran un par de caracteres que desee transformar. Tenga presente, que una vez haya utilizado la instrucción SYMBOL AFTER no podrá volverla a utilizar, ya que los antiguos caracteres se borrarían nuevamente.

¿ Para qué sirven pues los caracteres autodefinidos ? Los caracteres autodefinidos tienen la ventaja de poder definir su propia composición de caracteres (futurista, técnico matemático, químico ...). Los caracteres que Vd. mismo configure serán de gran importancia para los juegos que Vd. desee realizar.

Todo buen teatro vive del aspecto de los actores durante su representación, y no estará bien visto si con una pequeña "i" desea derribar grandes "Bes". Esto nos lo podemos imaginar. Aquí podríamos convertir nuestra "i" en un cañón y nuestra "B" en un extraterrestre de aspecto peligroso. Aquí no se ha puesto límite a su fantasía.

Lamentablemente, el método de la adición de cifras binarias es una cuestión algo larga y penosa. Por este motivo brindaremos en las siguientes páginas un caramelo, es decir, un programa que lo solucionará todo para Vd.

2.8.1 GENERADOR PRACTICO DE CARACTERES

Con el siguiente programa tiene Vd. la posibilidad de crear sus propios caracteres con las teclas del cursor y dentro de una matriz de 8*8.

Después de entrar y de iniciar el programa aparecerá un rectángulo y en el ángulo superior izquierdo un pequeño punto. En este programa será ésto su "cursor". Mediante las teclas del cursor podrá moverlo de un lado a otro dentro de dicho rectángulo. En el caso de que desee fijar un punto, utilice la tecla "COPY" y así podrá fijar una "*" en el lugar en el que se encuentre el cursor. Si por el contrario desea borrar un punto, entonces deberá poner el cursor a la izquierda de la estrellita que desee borrar. Seguidamente presionará la tecla espaciadora. Si a continuación desea convertir el diseño que ha terminado en códigos, deberá activar la tecla "E" apareciendo poco tiempo después los ocho códigos del diseño en el borde inferior de la pantalla. Seguidamente se le planteará la pregunta sobre qué carácter desea modificar. Aquí deberá utilizar la tecla mediante la cual deberá llamar a su carácter y luego "ENTER". Es importante que Vd. memorice los códigos de sus nuevos caracteres para que más tarde se acuerde de qué tecla está ocupada y por qué carácter. Como último se planteará en la secuencia de su programa la pregunta sobre si Vd. desea definir otros caracteres. Dicha pregunta la contestará Vd. con "S" de sí o bien con "N" de no, según si desea continuar definiendo caracteres o no.

Aquí deberá tenerse en cuenta que, si se inicia con "RUN", todos los caracteres anteriormente definidos serán borrados. De lo contrario, si Vd. desea mantener estos caracteres deberá iniciar el programa con "GOTO 40".

```
10 SYMBOL AFTER 32
20 IF y=0 THEN y=1
30 DIM b$(8): DIM c$(8) : DIM C(8)
40 MODE 1
50 GOTO 240
60 x=1 : y=1 : z=0
70 LOCATE x,y
80 PLOT x*16,(128-y*16)+8,1
90 a$=INKEY$ : IF a$="" THEN 90
100 PLOT x*16 , (128-y*16)+8,0
110 z=0
120 y=y+(1 AND a$=CHR$(241))-(1 AND a$=CHR$(240))
130 IF y>8 THEN y=8
140 IF y=0 THEN y=1
150 x=x+(1 AND a$=CHR$(243))-(1 AND a$=CHR$(242))
160 IF x>8 THEN x=8
170 IF x=0 THEN x=1
180 IF a$=" " THEN z=2
190 IF a$ = CHR$(224) THEN z=1
200 IF z=1 THEN PRINT "*"
210 IF z=2 THEN PRINT " "
220 IF a$="f" THEN 320
230 GOTO 70
240 WINDOW #0,10,17,10,18
250 ORIGIN 143, 127
260 PLOT 0,0,1
270 DRAW 0,130
280 DRAW 132,130
290 DRAW 132,0
300 DRAW 0,0
310 GOTO 60
320 FOR I=1 TO 8:B$(I)="": NEXT I
330 PLOT 0,0
340 MOVER 10,120
350 FOR m=1 TO 8
360 FOR n=1 TO 8
370 IF TESTR (0,0)=1 THEN t$="1" ELSE t$="0"
380 b$(m)=b$(m)+t$
390 PLOTR 1,1,1
395 MOVER 15,-1
400 NEXT n
```

```
410 MOVER -8*16,-16
420 NEXT m
430 FOR x=1 TO 8:c$(x)="&x"+b$(x)
440 c(x)=VAL(c$(x)): NEXT x
450 WINDOW #1,1,39,20,25
460 WINDOW SHAP 0,1
470 FOR i=1 TO 8: PRINT c(i);: NEXT i : PRINT
480 INPUT "Qué carácter debe ser modificado";a$
490 a=ASC(a$)
500 SYMBOL a,c(1),c(2),c(3),c(4),c(5),c(6),c(7),c(8)
510 PRINT"Deberán modificarse otros caracteres?"
520 in$=INKEY$: IF in$="" THEN 520
530 WINDOW SWAP 0,1
540 IF in$="s" THEN 40
550 MODE 1
560 PRINT "Final del programa"
```

Aclaraciónes del listado:

En las líneas 10-50 se encuentra la llamada inicialización, es decir, que allí se realizarán todos los preparativos para definir los caracteres. Luego se efectuará un salto en la subrutina, donde se habrá diseñado la matriz 8*8. La ventana (Window) se habrá elegido una línea mayor para que el diseño no sea empujado hacia arriba cuando se fije un punto en el ángulo inferior derecho.

Después de realizado el salto de retroceso, el cursor del texto se fijará en el ángulo superior izquierdo. Lo mismo equivaldrá para el cursor de caracteres. En la línea 90 se realizará una encuesta del teclado que será valorada en las líneas 120-220. Si se activa la tecla "E" (línea 220) se efectuará un salto hacia la valoración del diseño. En dicha valoración se comprobará primero en una línea si un punto está fijado o no (línea 370). Si se ha fijado un punto, se añadirá a la variable B$(M) un uno; de lo contrario se le añadiría un cero. De esta manera se crea para cada línea una cifra binaria de ocho posiciones.

En las líneas 430 y 440 se convertirán las cifras binarias en valores decimales y se memorizarán en las variables C(W). Seguidamente se creará una ventana de textos en la que se darán los códigos y se plantearán las siguientes preguntas. En la línea 500 se tomará el carácter que Vd. habrá creado en la sentencia de caracteres. Por último se almacenarán todos los Windows mediante "MODE 1".

2.8.2 DIERESIS ALEMANAS PARA EL CPC 464 Y 6128

Si Vd. desea utilizar su CPC para al procesador de textos o una administración de ficheros, normalmente se busca una posibilidad de poder representar asímismo las diéresis alemanas que el ordenador de casa tampoco domina.

Ya que el CPC puede definir libremente tanto la representación de cada carácter sobre la pantalla como destinar para cada una de las teclas el código que se desee, ya tendremos las condiciones previas que se requieren para ello.

Dos instrucciones son las que deberán aplicarse: Con SIMBOL podremos conferir a un carácter ASCII de la pantalla cualquier aspecto. Como seguramente sabrán, cada carácter se compone de una matriz de puntos 8*8. Si deseamos definir un nuevo carácter, dibujaremos una de estas matrices marcando los puntos fijados. Si elegimos el tipo de representación binaria, podremos tomar directamente la muestra Bit. Veamos a continuación el ejemplo de la Ä mayúscula.

	12345678
1	01011010
2	00111100
3	01100110
4	01100110
5	01111110
6	01100110
7	01100110
8	00000000

Un 1 significa un punto fijo y un 0 un punto no fijo. Antes de que pasemos a la definición, tendremos que saber los códigos ASCII que reservaremos para las diéresis. Para ello no existe ninguna norma; La asignación más normal es destinar las diéresis grandes detrás de la 'Z' y las pequeñas, incluyendo la doble 'ß' alemana, detrás de la 'z'. Para los códigos ASCII resultará la siguiente asignación:

Ä	91	&5B
ö	92	&5C
Ü	93	&5D
ä	123	&7B
ö	124	&7C
ü	125	&7D
ß	126	&7E

Si elegimos este tipo de codificación, las diéresis serán correctamente representadas si disponemos de una impresora Epson. Normalmente estos códigos ASCII se encuentran ocupados por los paréntesis angulares y contorneados respectivamente por los Backslash. Modificaremos la ocupación de teclas de tal manera, que los 'paréntesis angulares' correspondan sobre la ä, el 'Backslash' (sobre la tecla CTRL) sobre la ö y el 'paréntesis angular' sobre el ü. La ß alemana la pondremos sobre el 'doble paréntesis'. Con estos datos podremos escribir el siguiente pequeño programa, que realizará las modificaciones correspondientes:

```
100 REM DEFINICION DE LAS DIERESIS ALEMANAS
        Y DE LA OCUPACION DEL TECLADO
110 SYMBOL AFTER 90
120 SYMBOL 91, &X01011010,&X00111100,&X01100110,&X01100110,
               &X01111110,&X01100110,&X01100110,&X00000000
```

```
130 SYMBOL 92, &X10111010,&X01101100,&X11000110,&X11000110,
               &X11000110,&X01101100,&X00111000,&X00000000
140 SYMBOL 93, &X01100110,&X00000000,&X01100110,&X01100110,
               &X01100110,&X01100110,&X00111100,&X00000000
150 SYMBOL 123,&X01001000,&X00000000,&X01111000,&X00001100,
               &X01111100,&X11001100,&X01110110,&X00000000
160 SYMBOL 124,&X00100100,&X00000000,&X00111100,&X01100110,
               &X01100110,&X01100110,&X00111100,&X00000000
170 SYMBOL 125,&X01000100,&X00000000,&X01100110,&X01100110,
               &X01100110,&X01100110,&X00111110,&X00000000
180 SYMBOL 126,&X00111000,&X01101100,&X01101100,&X01101100,
               &X01100110,&X01110110,&X01101100,&X01100000
190 DEF KEY 22,1,124,92
200 KEY DEF 19,1,125,93
210 KEY DEF 17,1,123,91
220 KEY DEF 26,1,126,96
```

En el momento de efectuar la entrada de datos tenga presente que el Editor intercepte los ceros que van por delante, es decir, que &X00110011 sea listado como &X11001-1. Si Vd. pone este programa al principio de su programa principal o bien lo llama como subprograma, podrá utilizar en adelante el vocabulario alemán.

2.9 CARACTERES GRAFICOS

Su CPC tiene además otra propiedad muy positiva. Para que ésta no se "atrofie" pasaremos a comentarla un poco más de cerca. Es la posibilidad de los caracteres gráficos.

Seguramente se preguntarán por qué tememos que a los caracteres gráficos no se les dé la importancia debida. Lamentablemente su utilización no es demasiado sencilla y muchos usuarios de ordenadores no están después de un año en condiciones de utilizar los caracteres gráficos de su ordenador, siempre y cuando éste esté dotado con los mismos. ¡ A este tipo de usuarios no debe pertenecer Vd.!

¿ Por qué caracteres gráficos ? En el capítulo parcial anterior hemos mencionado los caracteres autodefinidos. Algo parecido son también los caracteres gráficos de su CPC, sólo que éstos no deberán ser definidos en el "pié", sino que el ordenador se los tendrá preparados para Vd.

Vd. podrá conocer los caracteres y sus códigos del apéndice de su manual. En el mismo se encuentran todos los caracteres descritos, que su ordenador tiene a disposición. Aquí podrá encontrar también los caracteres gráficos, tales como marcianitos, una bomba, etc.

¿ Cómo "conseguirá" estos caracteres ? Aquí deberá pasar por la función CHR$ (argumento). Así pues, si Vd. desea por ejemplo dejar aparecer una bomba sobre su pantalla, deberá entrar los siguientes datos: PRINT CHR$(252).

De este mismo modo podrá proceder con cada uno de los caracteres gráficos. Evidentemente podrá mover dichos caracteres, por ejemplo, con la ayuda de LOCATE y PRINT. Realice la siguiente prueba:

```
10   MODE 2
20   FOR Y=1 TO 25
30   LOCATE40,Y:PRINT CHR$(252)
40   LOCATE 40,Y:PRINT " "
50   NEXT Y
```

Este corto programa deja caer la bomba del margen superior al inferior de su pantalla. Esto se realiza mediante un bucle así como con las instrucciones LOCATE y PRINT. Con un poco de fantasía podrá imaginarse la cantidad de buenos juegos que pueden programarse. Con un poco de habilidad, ésto podría llegar hasta obtener la calidad de las películas de trucos.

A continuación encontrarán un nuevo programa ejemplo, que demuestra ante todo la rapidez de este método. El programa finaliza curiosamente con un aviso de error, cosa que no es preocupante. Quizá Vd. mismo pueda llegar a descubrir el por qué. Un consejo: Después de la interrupción déjese dar el valor de Y. Este será la clave para su búsqueda de errores.

```
10 MODE 2
20 FOR X=1 TO 25
30 t=36
40 q=12
50 LOCATE t,x:PRINT STRING$(&8,CHR$(249))
60 NEXT x
70 y=1
80 FOR x=q TO q+4
90 LOCATE y,x:PRINT STRING$(&8,CHR$(250))
100 NEXT x
110 FOR x=q TO q+4
120 LOCATE y,x: PRINT STRING$(&8," ")
130 NEXT x
140 y=y+8
150 GOTO 80
```

Aclaraciones del programa:

10	Ajuste de la modalidad de los 80 caracteres.
20	Apertura del bucle para el movimiento hacia abajo.
30	Valor inicial para el movimiento hacia abajo del marcianito.
40	Valor Y para los marcianitos fijos.
50-60	Bucle para la emisión de los marcianitos verticales.
70	La Y se fijará sobre el 1.
80-100	Bucle para el movimiento horizontal a pasos de a cuatro.
110-130	Bucle para borrar el último bloque de marcianitos.
140	Aumento del valor para que los marcianitos puedan moverse en bloques hacia delante.
150	Salto de retroceso para el bucle de movimiento.

2.10 LA MEMORIA DE PANTALLA

La memoria principal de su CPC tiene una capacidad de 64K. Esto son 65536 Bytes. Para sus programas Basic están a su disposición aproximadamente 43K.
La memoria completa está dividida en cuatro bloques, cada uno de 16K. Estos bloques se llaman Banks y están enumerados correlativamente de 0 a 3. El Bank 0 representa aquí el campo de la dirección de memoria de 0 (dec.) hasta 16383 (dec.) y el Bank 3 representa el campo de 49152 (dec.) hasta 65535 (dec.). Si Vd. conecta su ordenador, la memoria de pantalla ocupará el Bank 3. Sea cual fuere su posición en alguno de los tres módulos, se reservarán básicamente 16K para la memoria de la pantalla. Introduzca las cuatro líneas siguientes e inicie el programa con "RUN".

```
10 REM Pantalla 1
20 MODE 2 : PRINT "a"
30 FOR t = &C000 to &FFFF STEP &800
40 PRINT BIN$(PEEK(t) ,8) : NEXT t
```

Si todo lo ha realizado bien, recibirá la siguiente información:

```
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 1 1 1 1 0 0 0
0 0 0 0 1 1 0 0
0 1 1 1 1 1 0 0
1 1 0 0 1 1 0 0
0 1 1 1 0 1 1 0
0 0 0 0 0 0 0 0
```

Claramente podrá reconocer la muestra Bit de la que consta la "a" minúscula, la cual ha sido llevada a la línea 20 del ángulo superior izquierdo de la pantalla.

Modifiquemos el pequeño programa según sigue:

```
10 REM Pantalla 2
20 FOR t = &C000 to &FFFF STEO &800
30 FOR a = 0 TO 3
40 PRINT BIN$(PEEK(t+a),8);" ";
50 NEXT a : PRINT
60 NEXT t
```

Introduzca en la modalidad directa "MODE 2" seguido de "RUN". Vd. recibirá cuatro bloques de muestras Bit de los que podremos reconocer claramente las letras R E A D. Esto no es de extrañar, ya que en la primera línea se encuentra la palabra "Ready" la cual se creó mediante el comando MODE x.
¿ Qué ocurrirá si elegimos otra modalidad ? Ya que como sabemos el probar va por encima del estudiar, introduzca nuevamente "MODE 1" y "RUN". El resultado no es como esperado. Aunque podamos reconocer una "R" y una "e", no se habrá modificado nada en el número de los Bits fijados en comparación con "MODE 2". Cada letra está dividida en dos bloques de muestras Bit. Pero contrariamente a lo esperado, los 4 Bits de la derecha de cada uno de los bloques estarán a "0". ¿ Por qué aparece la letra el doble de grande a pesar de no haber entrado más Bits que los que hay en la modalidad de los caracteres 80 ? Una nueva prueba deberá ayudarnos a descubrir este fenómeno.

Introduzca:
```
PEN 2
MODE 1
RUN
```

Nuevamente podrán reconocerse las dos letras iniciales del aviso Ready en la primera línea. Aquí también se encuentra cada letra dividida en dos bloques de muestras Bit. La diferencia está en que esta vez la mitad izquierda de un bloque se ha fijado a cero y los Bits marcadores que

representan nuestras letras se encuentran en el lado derecho.
La explicación a estas preguntas deberá buscarse en la configuración de la memoria de la pantalla y en función del MODE elegido. El campo de la pantalla de 16K está destinado para 2 funciones. En primer lugar se fijará la muestra Bit del carácter representado y en segundo lugar serán responsables del color los mismos Bits donde deberá representarse el carácter. A lo más tardar en este punto tendrá que efectuarse la diferencia entre las distintas modalidades.

2.10.1 MODALIDAD 2

En la modalidad 2 el asunto es relativamente sencillo. Cada Bit fijado corresponde a un punto de imagen, que se emitirá en el color del carácter. Un "1" significa el color del carácter 24 (amarillo claro), un "0" en la muestra Bit señaliza el color 1 (azul). Para poderle demostrar que también el "azul" es un color de carácter, introduzca las siguientes líneas en la modalidad directa:

```
MODE 2
PAPER 1 : PEN 0
CLS
```

La pantalla se habrá puesto amarilla y el color del carácter habrá cambiado a azul. Vuelva a iniciar el programa "pantalla 2" mediante el "RUN". Cada punto de imagen fijado se señalizará con un "0" en el bloque de muestras Bit. El color azul se habrá convertido en el nuevo color de caracteres.
Recopilemos: En la modalidad 2 cada Bit contiene la información en el campo de memoria de la pantalla sobre el color que deberá tomarse para diseñar el punto de la imagen correspondiente.

```
0 = Azul
1 = Amarillo
```

2.10.2 MODALIDAD 1

Conecte su CPC en la modalidad 1. En este tipo de funcionamiento con 40 caracteres por línea tendrán a disposición cuatro colores para los caracteres. En cada Byte de la memoria de la pantalla se enclavará la información sobre qué color deberá tener cada punto de imagen.
Para poderles demostrar cómo almacena su CPC el color y el punto de imagen a ser fijado, deberemos observar un sólo Byte.

Nº de Bit	I 7	I 6	I 5	I 4	II 3	I 2	I 1	I 0 I

En la modalidad 1 cada Byte está dividido en 2 tetraedros de 4 Bits cada uno, que aquí se encuentran identificados mediante la línea doble de separación. Los dos Bits con la misma posición en ambos tetraedros se unen en un par. Un par lo forman siempre los Bits (3,7), (2,6), (1,5) y (0,4). Con estos cuatro pares deberán contactarse los ocho puntos de imagen existentes. Ya que aparentemente ésto es imposible, se unirán dos de los puntos de imagen contíguos y se les asignará uno de los cuatro pares de Bits mencionados. De esta manera es posible asignar a cada par de puntos de imagen que aparece en la pantalla uno de los 4 posibles colores.
Los Bits 3,7 tendrán la información para el par de puntos de imagen de la izquierda.
En la siguiente tabla podrán apreciar los colores que representan los estados de los Bits de información.

Muestra de los Bits de información		Color
0	0	Azul
0	1	Amarillo claro
1	0	Verde-azul claro
1	1	Rojo claro

Lo que acabamos de aprender, pongámoslo a la práctica. ¿ Se acuerda todavía del aspecto un tanto extraño de los bloques de muestras de Bits cuando los quisimos leer en la modalidad 1 ? En el caso de que no se acuerde, busque el programa "Pantalla 2" y pruebe nuevamente ambos ejemplos. Observemos la primera línea del primer bloque de muestras de Bits:

1 1 1 1 0 0 0 0

Junte los pares de Bits de información y compruebe en la tabla el color que indican.

Par de Bits :	3,7	2,6	1,5	0,4
Valor :	0 1	0 1	0 1	0 1
Color :	Amarilo	Amarillo	Amarillo	Amarillo

Coja una hoja de papel de cálculo en la que dibujará 4 cuadrados con la longitud de ángulo 8 por 8 cuadritos contíguos. Cada cuadrado lo dividirá en cuatro columnas verticales (cada una de 2 cuadritos de ancho) y 8 líneas. Cada par de cuadritos sobre el papel corresponderá a dos puntos de pantalla contíguos. Rellene cada par de cuadritos cuando el par de Bits de información correspondiente indique, que aquí se ha fijado el color amarillo. Déje los cuadritos vacíos cuando los Bits de información señalen sobre el azul. En el caso de que en nuestro ejemplo obtenga otro valor que 00=(azul) o bien 01=(amarillo), entonces habrá cometido un error. De este modo descifrará cada línea de los bloques de muestras Bits presentados sobre pantalla. Como resultado obtendrá una "R" ancha y una "e" ancha sobre el papel.
Cambie el color de los caracteres con el PEN 2. A continuación introduzca : CLS y RUN. Como de costumbre, deberá unir los pares de Bits de información pudiendo constatar, que todos tienen el valor "1 0". Dando una ojeada sobre la tabla de colores podremos constatar, que esta combinación crea el

color "verde-azul claro".
Con estos conocimientos posiblemente ya no le será difícil crear caracteres propios de cuatro colores. Tenga siempre presente que cada carácter tiene únicamente la mitad de anchura de un carácter normal de escritura en la modalidad 1. Además, tampoco es posible adaptar estos caracteres con SYMBOL o SYMBOL AFTER en la tabla de los mismos. Caracteres de cuatro colores y caracteres multicolores, sobre cuyo lenguaje pasaremos a comentar en el próximo apartado, sólo pueden ser insertados en pantalla. En primer lugar deberán ser almacenados en líneas DATA, que durante el proceso del programa serán leídas oportunamente.
La fórmula para el cálculo exacto de la posición en pantalla, en la que se insertará un carácter multicolor, podrá ser encontrada al final de este apartado.

2.10.3 MODALIDAD 0

Esta modalidad funcional es muy significante para la capacidad de su CPC de representar muchos colores en el espacio más reducido. Lamentablemente la posible diversidad de colores irá en detrimento de la disolución de los puntos individuales. Tenga presente, que si desea recalar cada punto de imagen en 16 colores necesitará una memoria de pantalla de 64K. Tal como se realiza en la modalidad 1, se podrá valer nuevamente del truco de recalar varios puntos de imagen contiguos con el mismo valor del color. Si en la modalidad 2 de cuatro colores de puntos de imagen contiguos se ha contactado un mismo color para todos, para la modalidad multicolor serán 4 puntos de imagen. Con otras palabras puede decirse, que cada Byte de la memoria de la pantalla dirige exactamente 4 puntos de imagen contiguos en su color. De la tabla siguiente podrán deducir los colores que están a su disposición.

Tabla de colores para la modalidad multicolor

Bin.	Hex.	Dec.	Color
0000	0	00	Azul
0001	1	01	Amarillo claro
0010	2	02	Verde-azul claro
0011	3	03	Rojo claro
0100	4	04	Blanco luminoso
0101	5	05	Negro
0110	6	06	Azul claro
0111	7	07	Magenta claro
1000	8	08	Verde azulado
1001	9	09	Amarillo
1010	A	10	Azul pastel
1011	B	11	Rosa
1100	C	12	Verde claro
1101	D	13	Verde pastel
1110	E	14	Azul/amarillo luminoso
1111	F	15	Rosa/azul claro luminoso

Mediante un programa pequeño probaremas de pasar mediante magia los 16 colores sobre la pantalla.

```
10 REM Ejemplo de colores
20 DATA &00,&11,&22,&33,&44,&55,&66,&77,
        &88,&99,&AA,&BB,&CC,&DD,&EE,&FF
30 MODE 0
40 FOR d = 0 TO 31 STEP 2
50 READ x
60 FOR t = &C000 to &FFFF STEP &800
70 POKE t+d,x
80 NEXT t : NEXT d
90 GOTO 90
```

La instrucción "RUN" inicia nuestro pequeño programa. Sorprendentemente Vd. no verá los 16 colores esperados, sino un revoltijo de muestras. ¿ Qué es lo que hemos hecho mal ? Tal como se dijo para la modalidad 1, los valores del color que estén codificados en un Byte no deberán buscarse en 4 Bits que estén relacionados. Mas bién el código se encontrará repartido por todo el Byte. La información para los 4 puntos de imagen de la derecha se encontrará escondida en los Bits 1,3,5,7 y el valor de colores de la izquierda en los Bits 0,2,4,6. Pero aquí no habremos terminado. La secuencia a la que pertenecen estos Bits de información de colores no corresponde a aquella que Vd. acaba de leer. A continuación la secuencia correcta:

Puntos de imagen de la izquierda = 1,5,3,7
Puntos de imagen de la derecha = 0,4,2,6

Con lo que acabamos de aprender nos será posible conectar con todos los colores disponibles.

Bit Nr. 7 6 5 4 3 2 1 0		Izq.	Dcha.	Byte
0 0 0 0 0 0 0 0	=	00	00	0
1 1 0 0 0 0 0 0	=	01	01	192
0 0 0 0 1 1 0 0	=	02	02	12
1 1 0 0 1 1 0 0	=	03	03	204
0 0 1 1 0 0 0 0	=	04	04	48
1 1 1 1 0 0 0 0	=	05	05	240
0 0 1 1 1 1 0 0	=	06	06	60
1 1 1 1 1 1 0 0	=	07	07	252
0 0 0 0 0 0 1 1	=	08	08	3
1 1 0 0 0 0 1 1	=	09	09	195
0 0 0 0 1 1 1 1	=	10	10	15
1 1 0 0 1 1 1 1	=	11	11	207
0 0 1 1 0 0 1 1	=	12	12	51
1 1 1 1 0 0 1 1	=	13	13	243
0 0 1 1 1 1 1 1	=	14	14	63
1 1 1 1 1 1 1 1	=	15	15	255

Modifique el programa "ejemplo de colores" de tal manera, que los valores indicados para el Byte sean entrados en la línea DATA. Las cifras que constan bajo el Byte son números decimales, lo cual indica que Vd. no deberá introducir el carácter "&". A continuación exponemos la nueva línea DATA:

```
20 DATA 0,192,12,204,48,240,60,252,3,
        195,15,207,51,243,63,255
```

Si Vd. inicia el programa modificado de esta manera, obtendrá los 16 colores en la secuencia correcta. Si lo observamos detenidamente podremos contar 15 barras. Esto es debido, a que la barra 1 ha sido dibujada en el color 0, o sea, el color de fondo.

2.10.4 POSICION DE UN CARACTER

En los últimos apartados se han comentado a menudo los caracteres multicolor, su composición y como puede uno mismo calcularlos. En el presente apartado se presentará un pequeño programa auxiliar mediante el cual Vd. podrá posicionar sobre la pantalla la muestra de colores al caso.

```
 10 REM Posición de un carácter
 20 MODE 0
 30 INPUT "Qué columna";s
 40 LOCATE 1,1
 50 INPUT "Qué línea ";z
 60 LOCATE 1,1
 70 INPUT "Muestra de color     ";color
 80 LOCATE 1,1
 90 base = &C000
100 FOR dirección = base TO base + &3FFF STEP &800
110 POKE dirección + (s-1) + ((z-1)*80), color
120 NEXT dirección
130 GOTO 30
```

Este programa inserta pequeños ángulos del tamaño del cursor en la modalidad 2 dentro de la posición definida por Vd. mediante línea y columna. La pregunta sobre la muestra de color no es idéntica con la tabla de colores de su manual del CPC. Aquí deberá introducir un valor que Vd. mismo habrá calculado. Esto no debería presentarle ningún problema si se ha leído detenidamente todo el capítulo 2.10.3. En la línea 90 del programa se asignará a las variables "base" un valor. Aquí se trata de la dirección estándar donde empieza la memoria de la pantalla. En el capítulo 6 de este libro encontrará una rutina de máquina con la que podrá modificar el inicio de la memoria de la pantalla. Correspondiendo al campo de memoria modificado de la pantalla deberá adaptarse también la dirección base. La anchura de paso y la dirección final no deberá necesaria-

mente tenerlas en cuenta, ya que se fijan automáticamente. La línea 110 es el núcleo. Aquí se llevará la mancha de color al lugar adecuado. Tenga presente, que la posición del borde superior izquierdo de la pantalla (la llamada posición HOME) posee las coordenadas 1,1. El programa no dispone en la forma aquí presentada de ninguna rutina en la que se controlen los valores entrados en cuanto a su validez.

STEP &800

Seguramente no se le habrá escapado que en todos los programas de los que leemos o introducimos caracteres sobre el espacio de imagen, aparece siempre la anchura de paso "STEP &800". A continuación le facilitaremos las correspondientes aclaraciones. Contrariamente a muchos otros ordenadores en los que las ocho líneas de las que consta cada carácter están también los ocho Bytes contiguos de la memoria de la pantalla, en su CPC se alinearán en forma de cadena todas las primeras líneas de todas las posiciones de pantalla 2000\(25 líneas cada una de ellas con 80 caracteres). En la segunda cadena se encuentran todas las segundas líneas de las posiciones de escritura 2000. La primera dirección de la primera cadena es idéntica con el inicio de la memoria de la imagen, o sea que tendrá la dirección &C000 o 49152(decimal). La próxima cadena empieza con &C800 o 51200(decimal). Todos los valores siguientes se obtienen sumando 2048 al valor que precede. Nuestro "STEP &800" no es otra cosa, que el sistema de escritura hexadecimal de 2048.

	Hex.	Decimal
1. Línea	C000	49152
2. Línea	C800	51200
3. Línea	D000	53248
4. Línea	D800	55296
5. Línea	E000	57344
6. Línea	E800	59392

7. Linea F000 61440
8. Linea F800 63488

Esta distribución de la memoria de la imagen presenta una nueva cuestión. Tenemos 2000 posiciones de escritura a disposición y cada cadena es 2048 Bytes de largo. ¿ Qué ocurre con los 48 Bytes no visibles al final de cada bloque ? La respuesta es bien sencilla. Estos Bytes no están utilizados, teniendo sin embargo una pequeña limitación. Si modificamos el Offset de la pantalla para crear un Scrolling horizontal, entonces aparecerán estos Bytes. Le aconsejamos, pues, ir con cuidado, si es que Vd. desea utilizar este espacio para rutinas de máquina propias. Una descripción detallada del Offset con la posibilidad del Scrolling horizontal podrá encontrarla en el capitulo 6, apartado "direcciones de salto".

2.11 CARACTERES ALTERNATIVOS DE DISTINTO MODO

En el capítulo 2.8 ha podido conocer una posibilidad con la que podrá crear sus propios grupos de caracteres. Con las instucciones SYMBOL y SYMBOL AFTER estará en condiciones de crear sus propios caracteres y de utilizarlos. Para aquél que se inicie, ésto puede ser suficiente. Existen sin embargo situaciones en las que se desearía poder disponer de otra modalidad para definir sus propios caracteres. Esto se cuestionaría, cuando el juego de caracteres ya no es suficiente y tan sólo se desea modificar un carácter. Otros campos de aplicación son con seguridad programas, que están completamente escritos en lenguaje máquina o bien cuando sólo se desean modificar 4 ó 5 puntos dentro de un carácter. A todos aquellos lectores que más tarde o temprano se encuentran con esta evidencia se les ha dedicado este apartado.
Introduzca el siguiente programa:

```
10 REM Leer conjunto de caracteres
20 MODE 1
30 FOR t = &AB80 TO &AB87
40 PRINT BIN$(PEEK(t),8)
50 NEXT t
```

Sobre la pantalla obtendremos la muestra Bit con la flecha indicando hacia arriba. Esto es el carácter con el valor CHR$(240). De este modo podrán elegirse todos los caracteres cuyo código de carácter sea 240 o mayor. Con respecto a los otros símbolos, ésto no es tan fácil. Por este motivo prescindimos de efectuar en este punto la oportuna descripción. Tan sencillo como es servirse de cualquiera de los últimos 16 caracteres, tan fácil será poderlos modificar. Escriba la siguiente línea Basic:

```
POKE &AB87,255
```

Inicie de nuevo el programa con la instrucción "RUN" que se

encuentra en la memoria. La flecha dirigida hacia arriba tiene ahora una "base" sobre la que puede apoyarse. La flecha con la base también podrá emitirse por pantalla.

```
PRINT CHR$(240) : PRINT
```

La segunda instrucción de Print sirve para que la palabra "Ready" no aparezca directamente bajo nuestro símbolo y así podamos admirar la base que acabamos de crear. A continuación exponemos un listado de signos sobre los que podemos disponer libremente con este método.

Hex. de	hasta	Decimal de	hasta	CHR$( )
AB80	AB87	43904	43911	240
AB88	AB8F	43912	43919	241
AB90	AB97	43920	43927	242
AB98	AB9F	43928	43935	243
ABA0	ABA7	43936	43943	244
ABA8	ABAF	43944	43951	245
ABB0	ABB7	43952	43959	246
ABB8	ABBF	43960	43967	247
ABC0	ABC7	43968	43975	248
ABC8	ABCF	43976	43983	249
ABD0	ABD7	43984	43991	250
ABD8	ABDF	43992	43999	251
ABE0	ABE7	44000	44007	252
ABE8	ABEF	44008	44015	253
ABF0	ABF7	44016	44023	254
ABF8	ABFF	44024	44031	255

CAPITULO 3 S O N I D O

3.1 INTRODUCCION

En el próximo capítulo se aclararán algunos conceptos importantes y técnicas de la creación de ruidos y de música. Dichos conceptos serán recopilados en la secuencia bajo la cual irán apareciendo.

---Conexión del CPC 464 y 6128 a una instalación estereofónica.
---Bases de la música y del sonido.
---Aclaración de las instrucciones del Sonido.
---Env (envolvente del volumen).
---Ent (envolvente del tono).
---Editor de Sonido.
---Demos para la música de fondo.

Los comentarios siguientes le parecerán algo teóricos; sin embargo hemos intentado amenizar este tema mediante muchos ejemplos. Prepare pues su CPC para poder entrar los programas más importantes.

En el caso de posibles incomprensiones le rogamos que consulte tranquilamente su manual con el fin de volver a leer los apartados no del todo comprendidos.

Aunque Vd. sea completamente amusical, no debe desesperarse. La música de ordenadores es un tema muy físico-matemático. Aunque a Vd. no le guste ésto demasiado estamos seguros que se alegrará sobre las bonitas melodías listadas al final de este capítulo.

3.2 CONEXION DEL CPC A UNA INSTALACION ESTEREOFONICA

Como habrán podido comprobar, el altavoz que se ha instalado en su CPC no es el idóneo para la capacidad de sonido. Para poder remediar este punto se ha explicado aquí detenidamente, cómo poder conectar su CPC a una instalación estereofónica.

Caso de que hubiesen probado conectar auriculares en la parte posterior del CPC se habrán dado ya cuenta, que éste no ha emitido sonido alguno. Este hecho se debe a que la señal no está amplificada en la salida y por lo tanto sus auriculares no han sido conectados.

Los auriculares necesitan una señal preamplificada o un tono para poderlo emitir. Así pues para utilizar los auriculares es necesario intercalar un amplificador. Su instalación estereofónica está equipada ya con dicho amplificador, asimismo cualquier grabadora e incluso radio.

¿ Cómo recibirá Vd. la señal del CPC en su amplificador ? Para ello se requiere una conducción blindada que lleve en el lateral del CPC un jack de un diámetro de 3.5 mm. Tenga presente, que aquí se trata de un modelo tripolar, tal como se utiliza para los auriculares Walkman. En el lado del amplificador se necesitará otro enchufe. Este dependerá del casquillo de entrada de su micrófono. Así pues deberá ser la entrada del micrófono, porque ésta es la que reacciona mejor frente a señales no preamplificadas. En el caso que Vd. no tuviese una entrada de este tipo en su amplificador, podría utilizar del mismo modo la entrada de un tocadiscos o bien de una grabadora.

Conecte pues ambos aparatos, pero no olvide de desconectar

antes los mismos para evitar tener complicaciones.

Después de que haya unido y conectado nuevamente ambos aparatos, deberá nivelar el sonido del amplificador. De lo contrario tendría una sorpresa desagradable con respecto a la intensidad del sonido.

El nivelado del sonido del amplificador podrá conseguirlo introduciendo la siguiente instrucción:

```
SOUND 7,284,32767,7
```

Esta instrucción crea el sonido para el valor de notas internacional "a", utilizado por todos los músicos para afinar sus instrumentos. Es la nota a partir de la cual podrán calcularse todas las demás.

3.3 FUNDAMENTOS DEL SONIDO

La música, así como la creación de ruidos, es uno de los campos más importantes que un buen ordenador personal debe poder abarcar.

En cuanto a esta constatación podemos considerar al CPC 464 y 6128 como a un ordenador personal de los mejores del mercado, ya que con sus excelentes propiedades de sonido puede competir con cualquier otro ordenador.

Las bases del sonido son en primer lugar el lenguaje y los sonidos de animales, ya que sin estas propiedades de los seres vivos no existiría ninguna aplicación razonable de la creación de sonidos. Con el fin de poderle demostrar la capacidad de su CPC, quisiéramos mencionar en este punto el campo auditivo del hombre. Un adulto medio tiene un campo auditivo de aproximadamente 16 Hz hasta 16 Khz. Sin embargo su CPC puede crear tonos en la frecuencia de 30 Hz hasta 125 Khz. Así pues podrán comprobar que no todas las notas que su CPC pueda crear pueden ser oídas por Vd. En el caso de que tuviera un perro en casa y Vd. diese la instrucción SOUND 1,1,100 no se extrañe que el perro reaccione de inmediato, ya que él sí que puede oir este tono. Dicho en otras palabras, el perro tiene un mayor campo auditivo que el hombre.

Como ya sabrá, una pieza musical no es otra cosa que una fricción mútua de tonos. Aquí no pueden ser tonos disarmónicos que se acoplen entre si, no siendo en general algo que suene tan bien como una pieza musical.

Una pieza musical puede asímismo sonar mal, aunque sus notas estén tocadas en la secuencia adecuada. Normalmente depende también del instrumento con el que se toca.

Sobre este punto no debe preocuparse, ya que con el CPC 464 y 6128 tiene un excelente instrumento a disposición.

Una de las bases más importantes para poder comprender la creación de sonidos es el conocimiento sobre el aspecto de un tono.

La norma actual válida es aquélla, que un tono se compone de ondas, respectivamente vibraciones. Consecuentemente un tono es más elevado que otro si tiene una frecuencia más alta. Esto quiere decir, que se realizan más vibraciones en el mismo espacio de tiempo. La unidad de medida para la frecuencia es el Hertz, que simplificado anotamos como Hz.

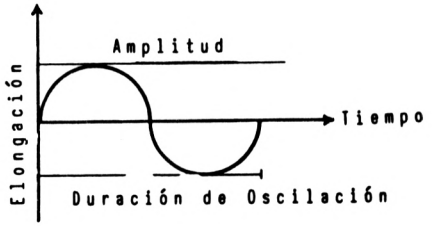
A continuación un ejemplo:

La nota "do" tiene una frecuencia (número de oscilaciones) de 261,626 Hz.

La "a" internacional tiene una frecuencia de 440 Hz y es por lo tanto más elevada que la nota "do" mencionada anteriormente.

Con el fin de poderles demostrar más visiblemente el aspecto de un tono, hemos dibujado una oscilación con todas las denominaciones más importantes.

Estas denominaciones serán luego aclaradas más detenidamente.



A continuación facilitamos las aclaraciones de los conceptos de izquierda a derecha.

1. Elongation (distanciamiento momentáneo desde la posición de reposo)

Este concepto no es demasiado importante para la creación del sonido, ya que no puede aplicarse de modo práctico. Este valor es únicamente la denominación del eje Y.

2. Amplitude (mayor Elongation o anchura de oscilación)

Este valor es en el sonido el valor aplicado al volumen. Esto quiere decir, cuanto mayor sea la Amplitude tanto más intenso será el tono.

3. Tiempo o duración del tono

Este factor jugará un papel importante en la creación de los envolventes. En nuestra gráfica es la denominación del eje X.

4. Duración de la oscilación

Este valor es el que se aplica al tiempo para una oscilación completa. La duración de la oscilación depende de la frecuencia. A medida que aumenta la frecuencia disminuye la duración de oscilación.

Aquellos lectores que no estén "iniciados" en la física de las oscilaciones seguramente se preguntarán cómo pueden crearse dichas oscilaciones por el CPC. Esto es muy complejo para poderlo aclarar aquí por completo, sin embargo podremos facilitarles unos conocimientos básicos.

Un altavoz que sea necesario en el ordenador para la creación del tono consta, dicho en pocas palabras, de un imán, de una bobina y de un baffle. Si se manda en un ritmo determinado corriente eléctrica por la bobina, ésta o bien

atraerá al imán o lo repelerá. Si lo repele, el imán moverá el baffle creando así una oscilación que será transmitida por el aire. Esta oscilación se irá extendiendo por el aire llegando momentos después a su oido.

Mediante el ritmo de la corriente eléctrica podrá determinarse asimismo la frecuencia del tono y con ello su intensidad.

Para más aclaración sobre la creación del tono es preferible que consulten un diccionario según los conceptos mencionados anteriormente. Aquí encontrarán una explicación más detallada.

3.4 COMANDOS DE SONIDO

Los comandos más importantes de sonido del CPC son:

SOUND
ENV
ENT
RELEASE

En cuanto al primer comando, el SOUND es el comando básico para la creación del tono del CPC. Tiene los siguientes parámetros (en su secuencia después del comando):

1. Estado del canal: Este valor indica al ordenador el canal del tono que se está utilizando y las demás actividades que deberán realizarse, como por ejemplo un Rendez-vous o un paro.

Los distintos valores para estas acciones serán calculados de los siguientes Bytes:

Bit	Valor	Comando
0	1	El tono se mandará al canal A.
1	2	El tono se mandará al canal B.
2	4	El tono se mandará al canal C.
3	8	Rendez-vous con el canal A.
4	16	Rendez-vous con el canal B.
5	32	Rendez-vous con el canal C.
6	64	Paro.
7	128	Flush.

Si desea mandar un tono al mismo tiempo a los tres canales, el estado del canal deberá tener el valor 7. Esto quiere decir, que deberá calcular conjuntamente todos los valores para los comandos deseados.

Con un Rendez-vous podrá sintonizar entre sí la secuencia de tonos de los tres canales de los mismos, es decir, que

podrá tocar una melodía con pausas sin que el altavoz emita un ruido desagradable.

Si desea tocar una melodía y dejar oir los tonos alternativamente por los tres canales de tonalidad, deberá concertar un Rendez-vous. Un Rendez-vous de este tipo puede tener el siguiente aspecto:

	1	2	3	4	5
Canal A:	NO	NO	NO	RB	RB
Canal B:	NO	RC	NO	NO	RA
Canal C:	NO	RB	RA	NO	NO

En este esquema el NO significa que se toca una nota y la R concierta un Rendez-vous con la letra del canal correspondiente.

A continuación les expondremos un esquema del que podrán deducir el curso temporal de la estructura del Rendez-vous descrita anteriormente. Acciones que tengan la misma duración están situadas una debajo de la otra. El NO significa nuevamente una nota y WA una pausa.

Canal A	NO	NO	NO	WA	NO	NO	WA	WA
Canal B	NO	NO	NO	NO	NO	NO	WA	WA
Canal C	NO	NO	WA	WA	WA	NO	NO	NO

Deberá tenerse en cuenta que, cuando se concierte un Rendez-vous, se creará una pausa en el canal de reacción. Dicho canal será aquél que siga tocando.

Otra posibilidad para poder sincronizar los canales es fijar un paro. Con dicho paro podrá interrumpir el canal que ya esté completo, pudiéndolo llamar de nuevo mediante la instrucción RELEASE. La ventaja frente al Rendez-vous es que el canal de tonalidad podrá ser llenado en un paro, mientras que el Rendez-vous no permitirá la entrada en el

canal de otros tonos hasta que no se haya realizado dicho Rendez-vous.

2. Valor de las notas: El segundo valor según los comandos de Sonido es el de las notas. Este podrá deducirlo del apéndice de su manual. Normalmente necesitará tan sólo los valores de la octava cero; todos los demás valores podrá calcularlos de la siguiente manera:

Valor de la nota = Valor de la nota de aquélla que está en la octava 0 / 2^octava

Aquí un ejemplo: Si desea calcular el valor de "do" en la octava -2, deberá efectuar el siguiente cálculo: 478/2^-2 ó bien 478/0.25

3. Duración del tono: Esta deberá indicarse en centésimas de segundo.

Si el valor es negativo, indicará la cantidad de veces que se repetirá la duración del tono del ENV. Si el valor es cero, el tono se tocará de acuerdo con la duración del mismo del ENV.

4. Volumen: Esta se indicará de 0-7 si no se fija un ENV. Si el ENV se ha concretado, los valores se tomarán de 0-15.

Los dos valores siguientes se detallarán más adelante. Estos se refieren al ENV y al ENT.

El último valor está para la creación del ruido. Si éste se aplica en los tres canales, los tres recibirán el mismo ruido.

El siguiente comando que depende estrechamente del comando del sonido, es el RELEASE. Con dicho comando podrá anularse un paro que se haya fijado. Para ello deberá indicarse únicamente el número del canal (A=1, B=2,C=4).

Antes de que pasemos a la parte más importante que es la aclaración de los envolventes, deseamos alegrar un poco el ánimo mediante una pequeña canción. Esta se llama: "Oh when

the Saints go marching in". Copie el siguiente programa y almacénelo primero sobre Cassette, ya que más adelante lo volveremos a utilizar.

```
10 READ A,B
20 IF A=-1 THEN RESTORE: GOTO 10
30 SOUND 1,A,B
40 SOUND 2,0.5*A,B
50 SOUND 4,0.25*A,B
60 GOTO 10
70 DATA 478,50,379,50,358,50,319,200,0,5,319
,50,478,50,379,50,358,50,319,200,0,5,319,100
80 DATA 478,50,379,50,358,50,319,100,379,100
,478,100,379,100,426,200
90 DATA 0,5,426,50,379,50,0,5,379,50,426,50,478
,150,0,5,478,50
100 DATA 379,100,319,100,0,5,319,50,358,150,0,5
,358,100,379,50,358,50,319,100,379,100
110 DATA 478,100,426,100,478,200,0,5,478,50,-1,1
```

3.5 EL COMANDO ENV Y LA INTENSIDAD DEL SONIDO ENVOLVENTE

El comando ENV sirve para modificar la intensidad de un tono, de modo que un tono se crea tal y como sale de un instrumento.

Existe la posibilidad de crear un sonido mejorado al incrementarse o disminuirse el volumen en el momento de tocar una pieza determinada.

Al aplicarse un ENV deberá tenerse en cuenta, que el volumen esté a 0 al darse la comando del sonido (a no ser que Vd. desee otro valor), para que el envolvente aparezca en toda su configuración.

Al envolvente se le asignarán los llamados Labels, conocidos también como números ENV, que se insertarán en el comando del sonido para llamar al envolvente.

Este Label es el primer valor del comando ENV. A éste le siguen tres valores que están determinados para el número de pasos, anchura de paso y tiempo de pausa. Estos tres valores pueden ser aplicados cinco veces consecutivamente.

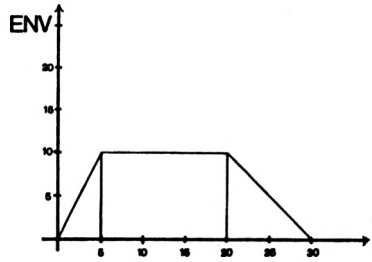
El número de pasos dará la cantidad de éstos en los que el volumen deberá aumentar o bien disminuir. La anchura de paso dará, multiplicándola por el tiempo de pausa, la duración del tono. Si este valor es negativo entonces bajará el volumen. El tiempo de pausa es aquél en el que se mantiene el volumen actual. Este se calculará en 1/100 segundos. Este tiempo de espera puede ser también denominado como duración del paso.

Con respecto a nuestra canción intente el siguiente ENV:

ENV 1,5,2,1,1,0,16,5,-3,2

Vd. quedará maravillado del resultado que obtendrá. A continuación explicaremos como hemos llegado a obtener los valores arriba mencionados.

En su manual existe una hoja en la que pueden diseñar las curvas de los envolventes. Tenga presente de anotarse para cada parte de la curva los tres valores correspondientes. Ya le hemos diseñado una curva de este tipo con todos sus valores para que Vd. pueda comprender perfectamente el desarrollo de un ENV.



3.6 LA ENT Y EL SONIDO ENVOLVENTE

El sonido envolvente es, en principio, el mismo que el volumen envolvente, aunque las posibilidades no sean tan grandes como en el volumen envolvente, ya que en el sonido envolvente únicamente podrá conseguirse un grado reducido de modificaciones. Una de estas modificaciones puede ser un Vibrato, o sea una modificación mínima de la elevación del tono proporcionando una sensación como si el tono "temblase".

La estructura del comando con el número de paso, la anchura del paso y el tiempo de pausa corresponde al ENV, aunque generalmente se requieran tan sólo tres partes, o sea subir, parar y bajar, mientras que en el ENT se necesitan normalmente cinco.

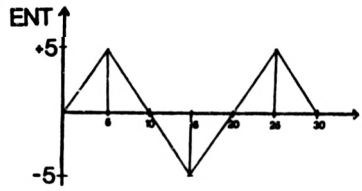
Un tipo de Vibrato se crea mediante el siguiente comando. No olvide modificar en el programa "oh when the Saints" las líneas del sonido, de modo que el ENT también sea contactado. El valor que deberá insertar es uno. Este será el número ENT, semejante al Label del comando ENV. Deberá tener presente, que si aplica aquí un número negativo el envolvente se repetirá.

A continuación exponemos el comando que deberá dar:

```
ENT 1,5,1,1,10,-1,1,10,1,1,5,-1,1
```

Asímismo hemos diseñado una curva para el ENT con el fin de podérselo demostrar gráficamente. Aquí se han elegido las mismas unidades como en la curva del ENV.

En el caso de que desee hacer Vd. mismo un envolvente, podrá hacerlo de acuerdo con el mismo esquema que hemos hecho nosotros.



Para finalizar ambos envolventes diremos todavía unas palabras: Aquí debe mencionarse que los envolventes no son realmente curvas. Más bien deberán ser denominados como oscilaciones rectangulares. Este hecho depende de que no puede calcularse cada valor sino únicamente los puntos angulares de las partes curvas. El camino más corto entre dos puntos es pues la recta. Con ésta difícilmente puede dibujarse una curva.

Mediante los dos comandos de envolventes podrá teoréticamente modificar cada tono de tal manera, que a penas podrá reconocerlo. Esto puede ser una ventaja para un sintetizador. El siguiente editor de sonido podría servirle de piedra fundamental. Modifique los tonos mediante las teclas del cursor y el comando ENV, dejando aumentar los valores para la anchura del paso o bien el número de pasos al presionar "el cursor y la tecla de mayúsculas".

El modo más sencillo de entender por completo los envolventes es experimentando con los valores, ya que mediante cálculos difícilmente podrá apreciarse el sonido real.

En el caso de que desee simular un instrumento deberá saber con exactitud el sonido que éste tiene. Un instrumento de cuerda tiene normalmente una pequeña reververación de las oscilaciones de las cuerdas. Para poder crear dicha reververación, la tercera escala del ENV, o sea al decrecer, debería ser más prolongada que al efectuarse la subi-

da. Cuando se trata de instrumentos de percusión, como es el caso de una batería o de un timbal, este eco no existe. Aquí el decrecimiento puede efectuarse en un período de tiempo más corto que al realizarse la subida.

Si se desea imitar un instrumento de viento es ya más difícil calcular el envolvente correspondiente. Existen muy distintas formas de tocar la trompeta o el trombón. Sin embargo lo más importante es aumentar brúscamente el volumen, ya que estando condicionada la columna de aire por las oscilaciones (según la longitud del cuerpo de resonancia) el tono bajará más o menos suavemente. El decrecimiento depende de la forma de tocar.

3.7 EL EDITOR DE SONIDO

Como ya habrán podido comprobar, la programación de música es una cuestión realmente difícil si únicamente se trabaja a base de cálculos puros. Por este motivo hemos escrito un programa en esta parte del capítulo para que Vd. pueda crear un tono mediante una simple presión de teclas. Aquí se sobreentiende un tono y no un ruido. La ocupación del teclado corresponde a la escala de notas, es decir de "do" a "do". Los medios tonos, que son aquéllos de la escala de notas con la "A" delante de las notas, se conseguirán mediante la tecla Shift y la nota correspondiente.

¿ Cuántas cosas podrá hacer con este programa ?

1. Programar melodías.
2. Almacenar melodías.
3. Tocar melodías.
4. Cargar melodías.

1.) Bajo este punto podrá mediante la presión de una tecla, según indicado anteriormente, crear tonos. Estos tonos pueden "retenerse" por su CPC y al ser llamados podrá volverlos a tocar. Existe la selección entre nueve octavas distintas. Esto ocurre si presiona una tecla entre el 1 y el 9 (la ocupación del teclado se aclarará más adelante mediante una tabla). Si presiona la tecla "P" podrá dejar aparecer sobre pantalla la última tecla que habrá presionado. En cuanto al punto del programa "programar melodías", existe una limitación con respecto al número de tonos. Está limitado a mil tonos para que el espacio de memoria no esté sobrecargado y no existan así complicaciones con el almacenaje. El número de los tonos ya programados se indicará en el ángulo superior derecho de la pantalla. La duración del

tono podrá determinarla presionando la tecla durante más tiempo. Aquí surgirá una pequeña pausa al efectuarse la programación y cuando toque la melodía pasará desapercibida.

Para volver del punto 1 al menú deberá presionar la tecla "ENTER".

2.) Bajo este punto podrá almacenar una melodía que ya haya programado. Para ello deberá entrar el nombre de esta melodía y poner, siguiendo las instrucciones, la grabadora en el punto de grabar. Las notas se almacenarán en forma de un listado de variables. Con el punto "4" podrá volverlas a cargar. Evite sin embargo almacenar las melodías demasiado cerca una de otra, ya que podrían solaparse. Además, la búsqueda de una melodía es así mucho más fácil.

3.) Aquí se tocará una melodía acabada de programar o bien acabada de cargar. Los tonos se tocarán en la secuencia de su aparición pero sin pausas. Si desea también programar una pausa deberá presionar en el punto de "programación" un espaciado en lugar de una nota.

4.) Aquí se cargará una melodía que antes haya sido grabada sobre cinta. La melodía cargada podrá volverla a llamar mediante el punto "3". Sin embargo aquí no existe la posibilidad de modificar la melodía; ésto podrá conseguirse mediante pequeñas modificaciones del programa. Dichas modificaciones no deberían presentarle problema alguno después de la aclaración del listado.

Otras posiciones que requieren modificaciones son:
(a) El programa trabaja únicamente con un canal de tonos. Mediante otros dos comandos de sonido y una nueva ocupación de teclado podrá evitarlo.
(b) Tampoco existe la posibilidad de modificar el volumen del programa; sin embargo podrá efectuar esta modificación manualmente con el regulador de volumen de su CPC respectivamente de su instalación estereofónica.

Con el fin de poder aprovechar completamente este programa, es aconsejable conectar el CPC a una instalación estereofónica, ya que de lo contrario, el sonido no es demasiado bueno.

A continuación y como prometido detallamos una tabla de todas las ocupaciones de teclado y sus correspondientes significados, así como algunos breves comentarios.

Tabla de ocupación:

Tecla	Significado	Comentario
c	Nota do	
C	Media nota do	
d	Nota re	
D	Media nota re	
e	Nota mi	
f	Nota fa	
F	Media nota fa	
g	Nota sol	
G	Media noto sol	
a	Nota la	
A	Media nota la	
h	Nota si	
Espaciador	Pausa	
p	Presión de tecla	La primera vez se conecta, luego se desconecta

1	Octava -3	Número de octavas según la tabla del manual
2	Octava -2	
3	Octava -1	
4	Octava 0	
5	Octava 1	
6	Octava 2	
7	Octava 3	
8	Octava 4	
9	Octava 5	No indicado en el manual
ENTER	Fin	Salto de retroceso al menú

```
10 DIM tono(1000)
20 MODE 2
30 LOCATE 5,10: PRINT "Programar melodía    :1"
40 LOCATE 5,12: PRINT "Almacenar melodía    :2"
50 LOCATE 5,14: PRINT "Tocar melodía        :3"
60 LOCATE 5,16: PRINT "Cargar melodía       :4"
70 LOCATE 5,20: INPUT "Su elección ";wa
80 ON wa GOTO 100,490,330,390
90 END
100 CLS
110 PRINT "P R O G R A M A R   M E L O D I A "
120 n=0
130 FOR tonos=1 TO 1000
140 a$= INKEY$: IF a$="" THEN 140
150 a=a+(478 AND a$="c")+(426 AND a$="d")
160 a=a+(379 AND a$="e")+(358 AND a$="f")
170 a=a+(319 AND a$="g")+(284 AND a$="a")
180 a=a+(253 AND a$="b")+(451 AND a$="C")
190 a=a+(402 AND a$="D")+(338 AND a$="F")
200 a=a+(301 AND a$="G")+(268 AND a$="A")
210 IF a$="p" AND pr=0 THEN pr=1: GOTO 140
220 IF a$="p" AND pr=1 THEN pr=0
230 IF a$=CHR$(13) THEN 20
240 nn=VAL(a$):IF nn=0 THEN n=n ELSE GOTO 320
```

```
250 IF pr=1 THEN PRINT a$;
260 tono(tonos)=(a*2^n)
270 SOUND 1,tono(tonos)
280 LOCATE 3,3: PRINT tonos
290 a=0
300 NEXT
310 GOTO 20
320 n=4-nn:GOTO 140
330 CLS
340 LOCATE 5,15: PRINT "La melodía será tocada"
350 FOR n=1 TO tonos
360 SOUND 1,tono(n)
370 NEXT
380 GOTO 20
390 CLS
400 LOCATE 5,5: PRINT "C A R G A R   M E L O D I A "
410 LOCATE 1,20
415 INPUT"Qué melodía he de cargar?";na$
420 OPENIN na$
430 INPUT#9,tonos
440 FOR n=1 TO tonos
450 INPUT#9,tono(n)
460 NEXT n
470 CLOSEIN
480 GOTO 20
490 CLS
500 LOCATE 5,5
510 INPUT "Cómo ha de llamarse la melodía? ";na$
520 OPENOUT na$
530 PRINT#9,tonos
540 FOR n=1 TO tonos
550 PRINT#9,tono(n)
560 NEXT
570 CLOSEOUT
580 GOTO 20
```

Aclaraciones del listado:

Línea 20-90 : Menú y selección de la modalidad (1-4)
Línea 100-310 : Modalidad de programación
Línea 130 : Apertura del bucle para el número de tonos
Línea 140 : Preguntar teclado
Línea 150-230 : Valoración de las teclas presionadas
Línea 240 : Valoración sobre la modificación de la octava
Línea 250 : Emisión de la tecla presionada, si se desea
Línea 260 : Anotación del valor en la variable actual del Arrays
Línea 270 : Creación de tonos mediante el comando del sonido
Línea 310 : Salto de retroceso al menú
Línea 320 : Modificación de la octava, salto de retroceso a la modalidad de programación
Línea 330-380 : Modalidad de interpretación
Línea 350 : Apertura del bucle para el número de tonos
Línea 360 : Interpretación del tono actual
Línea 380 : Salto de retroceso al menú
Línea 390-480 : Cargar melodía
Línea 420 : Abrir fichero de entrada del Cassette
Línea 430 : Lectura del número de tonos
Línea 440 : Apertura del bucle para la lectura del tono
Línea 450 : Lectura de los distintos tonos
Línea 470 : Cerrar fichero de entrada del Cassette
Línea 480 : Salto de retroceso al programa principal
Línea 490-580 : Almacenar una melodía
Línea 510 : Entrada del nombre de la melodía
Línea 520 : Abrir fichero de salida sobre Cassette
Línea 530 : Grabar número de tonos sobre cinta
Línea 540 : Apertura del bucle para el grabado de los tonos sobre cinta
Línea 550 : Grabar los distintos tonos sobre cinta
Línea 570 : Cerrar fichero de salida sobre Cassette
Línea 580 : Salto de retroceso al programa principal

3.8 PROGRAMAS EJEMPLO

Con los siguientes pequeños Demos podrá conferir a sus propios programas el sonido adecuado. Hemos elegido estos ejemplos para que los pueda incluir en las dos melodías sin grandes esfuerzos. No debería serle difícil, tal como se demuestra en la gráfica, crear mediante pequeñas modificaciones un nuevo Demo de los que ya tiene.

Todos los Demos serán titulados según lo que hayamos reconocido al escucharlos. Podría ser que Vd. escuchase algo muy distinto - según sea su facultad de imaginación y experiencia.

A continuación una breve nota: Los números de las líneas no son condicionantes; pueden ser modificados sin reserva alguna, siempre y cuando las direcciones de salto sean también cambiadas. Esto no debería suponerle problema alguno con la comando RENUM implementada en el manual "Schneider Basic", que también modifica las direcciones de salto.

Además añadiremos un pequeño consejo: Quizás podrán mejorar los Demos con los envolventes o bien crear una gráfica que corresponda al tono, por ejemplo con caracteres definidos por Vd. mismo. Deje correr su propia fantasía. Con este método conseguirá siempre los mejores resultados.

```
10 REM Sirena de policía
20 FOR n=100 TO 200 STEP 10
30 SOUND 1,n,2
40 NEXT
50 FOR n=200 TO 100 STEP -10
60 SOUND 1,n,2
70 NEXT
80 GOTO 20
```

```
10 REM Carácter de ocupado
20 SOUND 1,100,100,15
30 SOUND 1,0,100
40 GOTO 20
```

```
10 REM Guerra de las galaxias
20 FOR n=90 TO 125 STEP INT(RND(1)*10)+1
30 SOUND 1,n,2,15
40 NEXT
50 SOUND 1,0,INT (RND(1)*20)
60 GOTO 20
```

```
10 REM Escala de tonos
20 FOR n=127 TO 450 STEP 12
30 SOUND 1,n
40 NEXT n
50 FOR n=450 TO 127 STEP -12
60 SOUND 1,n
70 NEXT n
80 GOTO 20
```

```
10 REM Explosión
20 FOR n=15 TO 1 STEP -1
30 SOUND 1,426,40,n,,,1
40 NEXT
```

```
10 REM Alarma
20 FOR n=500 TO 100 STEP -15
30 SOUND 1,n,4
40 NEXT
50 SOUND 1,0,30
60 GOTO 20
```

```
10 REM Blanco alejado
20 FOR n=100 TO 800 STEP 15
30 SOUND 1,n,2,15
40 NEXT
50 FOR x=1 TO 7 STEP 0.5
60 SOUND 1,400,5,x,,,1
70 NEXT
80 SOUND 1,0,50
90 GOTO 20
```

```
10 REM Melodia preferida
20 a=INT (RND(1)*3500)+284
30 SOUND 1,a
40 b=INT (RND(1)*3400)+284
50 SOUND 2,b
60 c=INT (RND(1)*3300)+284
70 SOUND 4,c
80 GOTO 20
```

El programa siguiente es en principio igual que el programa de música "Oh when the Saints". Este programa toca la melodía "Happy Birthday" y está únicamente configurado con notas y prolongaciones de notas. Para mejorar el sonido le aconsejamos arreglarlo con ENV y ENT.

```
10 REM Happy Birthday
20 READ a,b
30 IF b=-1 THEN END
40 SOUND 1,a,b
50 GOTO 20
60 DATA 319,50,319,50,284,100,319,100,239,100
70 DATA 253,150,319,50,319,50,284,100,319,100
80 DATA 213,100,239,150,319,50,319,50,159,100
90 DATA 190,100,239,100,253,100,284,100,179,50
100 DATA 179,50,190,100,239,100,213,100,239,150
110 DATA 0,-1
```

Aclaración del listado:

10	Título
20	Lectura de las notas, resp. de los valores de las mismas
30	Preguntar si se ha terminado la pieza de música
40	Tocar la nota A con la prolongación de la nota B
50	Salto de retroceso hacia el 20
60-110	Datos para las notas y prolongaciones de las mismas

3.8.1 EL DESPERTADOR SONORO

Para aligerar las aclaraciones parcialmente teóricas sobre el sonido hemos puesto aquí un programa, que podrán utilizar con provecho si pertenece a aquel tipo de personas que no acostumbran a levantarse a tiempo por las mañanas y debido a ello pierden ya sea su autobús o el tren. En este programa se trata de un reloj digital que tiene un dispositivo que actúa de despertador. Nuestro reloj digital tiene además la función de indicar cada segundo mediante un tono suave y cada minuto con otro de más fuerte. La mayor ventaja es, que cada hora toca una corta melodía, que simulará ante sus conocidos como si de un reloj suizo se tratase.

Nuestro reloj se ha creado sencillo pero con unas pequeñas modificaciones, sería posible hacer de un despertador sencillo un segundo Big Ben. Para poder realizar estas pequeñas modificaciones necesita únicamente las notas de la pieza deseada para que el despertador las toque. Vd. puede dejarse despertar por una melodía o bien por un tono mucho más "desagradable" que el que hemos elegido.

Aclaraciones sobre su utilización:

Después de iniciar el programa se le preguntará en primer lugar la hora actual exacta según la siguiente secuencia: Hora, minutos y segundos.

Estos valores los introducirá uno por uno presionando por separado la tecla ENTER. A continuación se le preguntará la hora a la que desea ser despertado. La introducción de estos datos se realizará del mismo modo que con los datos del reloj. Aquí deberá tener en cuenta, que cuando presione únicamente ENTER la hora de despertarse esté fijada a las cero horas y así ser despertado a esta hora mientras el programa esté en la memoria.

Si desea interrumpir el sonido del despertador, deberá presionar una tecla. El tono se parará y el reloj seguirá funcionando normalmente.

En el caso de que el reloj no sintonizase correctamente, deberá modificar en la línea 150 la anchura del paso del comando EVERY. Concretamente ésto quiere decir que si el reloj atrasa reducirá el valor de EVERY; si adelanta, lo aumentará.

```
5 REM Despertador sonoro
10 MODE 2
20 INPUT "Horas     ";z
30 INPUT "Minutos   ";y
40 INPUT "Segundos ";x
50 INPUT "Hora a despertar (h)";z2
60 INPUT "Hora a despertar (m)";y2
70 INPUT "Hora a despertar (s)";x2
80 MODE 2
90 LOCATE 3,9 :PRINT CHR$(150);STRING$(&D ,CHR$(154));
CHR$(156)
100  LOCATE   3,10:PRINT   CHR$(149):LOCATE   17  ,10:PRINT
CHR$(149)
110   LOCATE   3,11:PRINT   CHR$(149):LOCATE  17  ,11:PRINT
CHR$(149)
120  LOCATE   3,12:PRINT   CHR$(149):LOCATE   17  ,12:PRINT
CHR$(149)
130 LOCATE 3,13:PRINT CHR$(147):STRING$(&D ,CHR$(154));
CHR$(153)
140 LOCATE 5,5:PRINT "Despertador musical"
150 EVERY 50,2 GOSUB 170
160 GOTO 160
170 SOUND 1,284,5,5:x=x+1
180 IF x=60 THEN x=0:y=y+1:SOUND 2,71,5,6
190 IF y=60 THEN y=0:z=z+1:GOSUB 270
200 IF z=z2 AND x=x2 AND y=y2 THEN GOSUB 320
210 LOCATE 5,11:PRINT z
220 LOCATE 8,11:PRINT":"
230 LOCATE 9,11:PRINT y
240 LOCATE 12,11:PRINT ":"
250 LOCATE 13,11:PRINT x
260 RETURN
270 READ no
```

```
280 IF no=-1 THEN RETURN
290 SOUND 4,no,30,7
300 GOTO 270
310 DATA 253,253,169,169,150,150,169,190,190,201,201,225,
225,253,169,169,190,190,201,201,225,169,169,190,201,201
315 DATA 225,253,253,169,169,150,150,169,190,190,201,201,
225,225,223,-1
320 SOUND 7,100,1000,15
330 IF INKEY$="" THEN 330
340 SOUND 135,0
350 RETURN
```

Aclaración del listado:

Línea 10-80	Fijación de la modalidad de la pantalla, introducción de los valores requeridos (hora (h,m,s)); hora a despertar (h,m,s).
Línea 90-140	Diseño del contorno del reloj.
Línea 150	Mediante el comando EVERY se llamará cada segundo a la subrutina a partir de 170.
Línea 170	Creación del tono de los segundos y marcado de un segundo.
Línea 180	Preguntar si se ha alcanzado el minuto; si es así, creación del tono de los minutos.
Línea 190	Preguntar si se ha alcanzado la hora; si es así, salto a la subrutina a partir de 270.
Línea 200	Preguntar si se ha alcanzado la hora de despertar; si es así, salto a la subrutina a partir de 320.
Línea 210-260	Indicación de la hora en el contorno.
Línea 270	Lectura de las notas.

Línea 280	Preguntar si se ha alcanzado el final de la melodía.
Línea 290	Tocar la melodía con los tres canales de tonos.
Línea 310	Línea de datos con los valores de las notas.
Línea 320	Creación del tono del despertador.
Línea 330	En este lugar espera el programa hasta que se haya presionado una tecla.
Línea 340	Con este tipo de comando de sonido se interrumpirá el tono del despertador mandando un paro a los canales del sonido.

C A P I T U L O 4 LENGUAJE MAQUINA

4.1 INTRODUCCION AL LENGUAJE MAQUINA

Esta introducción no es un curso de lenguaje máquina extenso. Es únicamente para darle una orientación en la amplia programación de su CPC 464 y 6128. Como ya sabrá, existen algunos problemas en la programación en Basic, que únicamente pueden solucionarse con pequeñas rutinas de máquina. Por ejemplo, realizar una rutina de clasificación o de búsqueda, que en Basic requiere a veces algunos minutos e incluso horas, para poder encontrar algo determinado de un gran número de datos. El mismo programa escrito en lenguaje máquina requiere únicamente una centésima del tiempo que el programa Basic necesita.

El cerebro del CPC 464 y 6128 es un ladrillo Z 80, siendo uno de los procesadores de 8 Bits más usuales en la técnica de los ordenadores personales. Este Z 80 tiene un juego de instrucciones que supera a las 600, que para poderlas aclarar por completo no disponemos de espacio suficiente en este libro.

Para su CPC existe únicamente una posibilidad para abarcar cifras, siendo el código binario. Los números representados binariamente constan de las cifras "0" y "1", las cuales pueden representar en una secuencia determinada cualquier número deseado. Una parte de una de estas cifras, o sea un "0" o bien un "1" determinado, se llama BIT. La palabra Bit proviene del inglés, de la expresión Binary Digit (cifra binaria).

Estos Bits están generalmente dispuestos en grupos de ocho. Uno de estos grupos de ocho se llama Byte. Veamos uno de estos Bytes: 01010101.

Sin saberlo previamente uno puede difícilmente imaginarse, que este Byte representa el número "85". Para poderlo comprender deberá saberse, que el Bit de la derecha representa el valor 2 elevado a 0, el siguiente Bit de la izquierda tendrá el valor de 2 elevado a 1, el siguiente 2 elevado a 2 y el Bit completamente a la izquierda 2

elevado a 7. Si deseamos calcular el número decimal "85" de la cifra binaria "01010101", únicamente tendrá que multiplicar el primer Bit de la derecha, es decir el "1", por 2 elevado a cero, el segundo Bit, o sea el "0", por 2 elevado a 1, el tercer Bit por 2 elevado a 2, ... y el octavo Bit completamente a la izquierda por 2 elevado a 7. Si suma todos estos resultados obtendrá el número "85". Puesto a la práctica se verá de la siguiente manera:

```
1 * 2↑0 =   1
0 * 2↑1 =   0
1 * 2↑2 =   4
0 * 2↑3 =   0
1 * 2↑4 =  16
0 * 2↑5 =   0
1 * 2↑6 =  64
0 * 2↑7 =   0
          ---
```

Sumado todo junto resulta 85

Escrito en variables un Byte se ve de la siguiente manera:

b7 b6 b5 b4 b3 b2 b1 b0

La b0 está para el Bit completamente a la derecha y el b7 para el Bit completamente a la izquierda.

El número decimal puede calcularse según sigue:

```
b7 * 2↑7  +  b6 * 2↑6  +  b5 * 2↑5  +  b4 * 2↑4  +
b3 * 2↑3  +  b2 * 2↑2  +  b1 * 2↑1  +  b0 * 2↑0
```

Para calcularlo más facilmente hemos representado aquí todas las potencias de dos desde 2 elevado a 0 hasta 2 elevado a 7:

2↑0=1 2↑1=2 2↑2=4 2↑3=8 2↑4=16 2↑5=32 2↑6=64 2↑7=128

Si observan más de cerca la representación binaria comprenderán también, porque la numeración va de derecha a izquierda, de cero a siete y no de uno a ocho. Esto depende de las potencias de dos relativas a las posiciones correspondientes.

Para la conversión de números binarios a decimales deberá entrar únicamente PRINT &X, estando el (bin) a disposición para su número binario. Para la conversión de decimales a binarios deberá utilizar la función PRINT BIN$(dez). Aquí estará el (dez) para su número decimal, que deberá ser convertido.

```
Ejemplo:  Print &x01010101
          85

          Print BIN$(85)
          01010101
```

Deberá tenerse en cuenta que todos los ceros que se encuentran delante del primer uno de la izquierda generalmente son ignorados.

4.2 EL SISTEMA NUMERICO HEXADECIMAL

Otro sistema numérico muy importante es el sistema hexadecimal o sedecimal. Es el modo más extendido de programar el ordenador en el lenguaje máquina. El sistema hexadecimal se basa sobre 16 cifras: Representativamente, éstas se destinan para los valores decimales de 0 a 15.

hex.	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
dez.	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

La gran ventaja del sistema hexadecimal es la reducción de la longitud de la entrada de datos así como del tiempo empleado para los programas de máquina y demás datos. Imaginese que tuviera que entrar la cifra "255" y tuviese la elección entre "11111111" en el código binario o bien "FF" en el código hexadecimal. Con seguridad preferiría el sistema más corto de "FF", como lo haría actualmente cualquier programador.

Volvamos nuevamente al sistema hexadecimal. La ventaja de este sistema está en que puede reoresentarse un Byte (o sea ocho Bits) con dos caracteres.

Si un número hexadecimal tiene la longitud de un Byte, se convertirá en el sistema decimal de la siguiente manera:

Puede multiplicar la primera parte de la cifra por 16 y sumándole la segunda parte.

A continuación un ejemplo:
La cifra es A1. Tome la primera parte, o sea la A, 16 veces. Esto dará por resultado 160, ya que la "A" tiene el valor de "10" según nuestra tabla. Súmele un 1 y así conseguirá el número 161.

Cuando se trata de números con dos Bytes (16 Bits) de longitud, éstos se dividirán en dos partes, o sea el Highbyte y el Lowbyte. Tomemos el número A1E1; la A1 es el Highbyte y el E1 el Lowbyte. Generalizando, puede decirse que el Byte izquierdo de una cifra de dos Bytes es siempre el Highbyte, y el derecho el Lowbyte. El cálculo es relativamente sencillo; en primer lugar puede Vd. calcular cada Byte tal como se ha descrito anteriormente; luego tome el Highbyte 256 veces y súmele el Lowbyte.

A continuación sigue nuevamente un ejemplo:

La cifra es A1E1. En primer lugar calcularemos los Bytes:
Highbyte=161;
Lowbyte=225.
Tome el Highbyte 256 veces y añádale el Lowbyte.
El resultado será 41441.

A continuación se especifica el cálculo sin aclaración alguna:

```
Highbyte(161)*256 +Lowbyte(225)=41441
```

Si desea ahorrarse estos cálculos, su CPC 464 y 6128 tiene para ello una función extraordinaria para la conversión en números hexadecimales de decimales y viceversa. A continuación se especifican ambas funciones:
De hexadecimal a decimal: PRINT &hex , indicando aquí hex para su número hexadecimal.

De decimal a hexadecimal: PRINT HEX$(dez), indicando aquí dez para su número decimal.

Sin embargo aquí tendremos una limitación: Los números hexadecimales deberían ser únicamente convertidos hasta 7FFF con la primera función, ya que en la conversión de valores más elevados se resta del resultado 65536, dando así una cifra negativa.

Ejemplo:
PRINT &A1E1, aquí se espera el resultado mencionado anteriormente, o sea 41441.
Sin embargo se emite como resultado -24095 (41441-65536= -24095).

Para evitar ésto, podrá utilizar esta pequeña rutina:

```
10 INPUT A$:B$="&"+A$:IF VAL(B$)<0 THEN PRINT VAL (B$)
+65536:END
20 PRINT VAL(B$)
```

Hemos utilizado una variable del String para que Vd. no tenga que escribir siempre "&" en el momento de introducir el número hexadecimal.

Memorice bien los conceptos Highbyte y Lowbyte, ya que en próximos capítulos aparecerán a menudo por codificar los espacios de memoria en el lenguaje máquina, llamados también direcciones en Highbyte y Lowbyte. Con ello se explica también que la dirección mayor es 65535, ya que el Highbyte puede tener como valor mayor el 255 o aceptar el hexadecimal &FF, siendo lo mismo para el Lowbyte. De esta manera y según la fórmula HIGHBYTE*256 + LOWBYTE el valor mayor es 65535.

Para concluir detallamos a continuación una tabla de todos los números de 0 a 16 en los tres sistemas:

Decimal	Hexadecimal	Binario
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101

6	6	0110
7	7	0111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111
16	10	10000

Con ayuda de esta tabla y los conocimientos facilitados anteriormente no le será muy difícil poder seguir nuestras próximas aclaraciones.

4.3 TECNICAS DE PROGRAMACION

Para programar desde el Basic el programa máquina podrá utilizar varios métodos: El primero, y en cuanto a tiempo se refiere, el más largo es aquél que carga en una memoria particular los distintos códigos para las instrucciones. Un modo mucho más confortable es el de utilizar el llamado monitor de lenguaje máquina. Al final de este capítulo hemos creado un monitor provisional para el lenguaje máquina que le será de gran ayuda para la programación de programas máquina. Este carga únicamente en los espacios de memoria particular los valores necesarios, pero le evita el gran trabajo de tenerlos que teclear. Una aclaración detallada vendrá más adelante.

En cuanto al primer método y como Vd. ya sabrá, la instrucción POKE X,A escribe el valor A en el espacio de memoria (dirección) X. En el lenguaje máquina las instrucciones están asignadas a una dirección determinada, tal como están las instrucciones en el Basic en cuanto a los números de línea. Así pues si desea escribir un programa a partir de la dirección &2000 deberá cargar en la memoria particular el valor de la primera instrucción en el espacio de memoria &2000; los valores de la instrucción, siempre y cuando se necesiten, se insertarán en los espacios de memoria &2001 y &2002. Si la instrucción sólo tiene una longitud de un Byte, en el espacio de memoria &2001 encontraremos la segunda instrucción.

¿ Qué tienen los códigos que pertenecen a las instrucciones ? El ordenador sólo entiende números binarios. Por este motivo una instrucción deberá constar de números binarios. A cada instrucción se le han asignado uno o varios de estos números (códigos). De este modo resultará una lista muy larga de cifras para un programa. Dichas cifras deberán insertarlas en los espacios de memoria en el orden de secuencia que se presente. Esto se realizará mediante la instrucción POKE.

Para poder leer un espacio de memoria, es decir, para llegar a saber su valor, deberá utilizar la instrucción PEEK (X). Aquí se le facilitará el valor del espacio de

memoria X. Si desea iniciar un programa ya escrito se utilizará la instrucción CALL X. Esta llamará al programa a partir del espacio de memoria X.

En el próximo apartado se le presentarán algunos problemas; no tema, nosotros tampoco hemos aprendido en tres días el lenguaje máquina.

ENSAMBLADOR

El ensamblador es un lenguaje de programación que utiliza abreviaciones para las propias instrucciones del lenguaje máquina. Estas abreviaciones se llaman MNEMONICOS. Así pues, por ejemplo, para la instrucción de carga (load) se utiliza la abreviación "ld". Ya que la máquina no entiende esta abreviación, se requiere un programa para transformar estas instrucciones en números binarios. Un programa de este tipo es un ensamblador. Con uno de estos ensambladores puede asímismo darse a las direcciones importantes nombres simbólicos, los llamados Labels (en inglés: caracteres). Los Labels se utilizarán más tarde para dar a las direcciones de las variables que utilizaremos un nombre determinado.

Tal como podremos deducir de nuestro manual, se está preparando un programa de ensamblador por la Firma Amstrad. En el caso de que no disponga de un programa de ensamblador deberá sacar de la tabla de instrucciones del apéndice el valor correspondiente de la instrucción utilizada.

REGISTRO Y PRIMERAS INSTRUCCIONES

En primer lugar comentaremos la configuración del registro del CPU (cerebro del ordenador).

Existen varios registros de gran importancia:

--El registro "a", denominado también acumulador. Se utiliza principalmente para el almacenaje y elaboración de valores actuales, es decir, para valores que se tratan en aquel momento. Esto es debido a que el registro "a" puede ser utilizado de muy distintas maneras.

--El registro "bc", que procede de la palabra Bytecounter, se utiliza muy a menudo para trabajos de cálculo. El registro "bc" es el llamado registro doble o pareja de registros, ya que tanto el registro "b" como el registro "c" pueden contactarse individualmente. Al contrario del registro "a", el cual sólo puede admitir un Byte, el registro "bc" puede recibir dos Bytes y números hasta 65535.

--En tercer lugar existe la pareja de registros "hl", también denominada pareja de registros High-Low. Aquí equivale lo mismo como para la pareja de registros "bc". Cada registro puede ser contactado individualmente pero se utiliza con preferencia como registro de 16 Bits (2 Bytes), ya que el CPU ofrece este registro precisamente para el almacenaje de direcciones que como ya se sabe están representadas con 2 Bytes (Highbyte y Lowbyte).

--Por último y como pareja de registros importante deseamos mencionar la pareja de registros "de". Todo conocedor del lenguaje máquina preguntará qué ha ocurrido con los registros "ix" e "iy" o con el indicador de bloques e incluso con el contador de programas. Este capítulo no ha sido hecho para los conocedores, sino para todos aquellos que son principiantes en el campo del lenguaje máquina. Ahora pasemos al registro "de". Este puede ser aplicado individualmente como la pareja de registros "bc" o "hl". Es un registro auxiliar para poder almacenar cifras de 16 Bits, que no tendrían sentido en otros pares de registros o bien cuando éstos se encuentran ocupados; dicho de otra manera, porque alguna que otra instrucción requiere tres pares de registros.

Si observamos la configuración del CPU nos preguntaremos cómo se obtiene un valor determinado de la memoria en el registro o bien de un registro a otro. Esto se consigue mediante la sencilla instrucción "ld" (Load). Es la instrucción más importante en el juego de instrucciones Z 80. A continuación un ejemplo:

Si desea cargar el acumulador (registro a) con el valor &FF (decimal 255), deberá escribir el valor ld a,n, debiendo introducir para "n" el valor &FF. La "n" está para una entrada directa, que únicamente tendrá la longitud de un Byte. Si tenemos "nn" ésto significará que se trata asímismo de una entrada directa pero de dos Bytes y codificada mediante High- y Lowbyte. Si hay paréntesis en ambas "n" significará, que el registro estará cargado con el contenido del espacio de memoria "nn".

Ejemplo: ld hl,(&A000) significa que la pareja de registros hl está cargada con el contenido del espacio de memoria &A000.

Si por ejemplo desea cargar el registro "c" con el contenido del acumulador, deberá escribir ld c,a. Aquí se perderá el contenido del registro "c", pero no el del acumulador.

Para programar en lenguaje máquina, se requiere un campo de memoria seguro que no pueda ser solapado por un programa Basic. La siguiente secuencia de instrucciones creará un campo libre en la memoria Basic en el que todavía podrá almacenar las siguientes rutinas:

Introduzca directamente:

```
MEMORY &1FFF:PRINT HIMEM:NEW
```

En el caso de que hubiese programado caracteres autodefinidos podrá borrarlos desconectando brevemente antes de que utilice esta secuencia de instrucciones, de lo contrario podría tener complicaciones.

Así tendrá un campo libre de más de 34 K-Bytes de longitud para sus programas máquina, pero únicamente de 7,5 K-Bytes para Basic.
Ahora podrá entrar los siguientes datos: POKE &2000,&3E:POKE &2001,&FF:POKE &2002,&C9
El POKE determinará la entrada de las instrucciones en los espacios de memoria.
Los tres valores después del POKE son las direcciones en las que se escribirán las instrucciones. Son similares a los números de línea del Basic. El valor &3E está destinado para la instrucción anteriormente aclarada, o sea, ld a,n. El valor &FF es el valor que se fijará para "n". El valor &C9 está destinado para RET, lo que significa RETURN, o sea que viene a ser un salto de retroceso de la rutina al Basic.

Inicie el programa con CALL (&2000)
Como ya sabrán, mediante CALL se llama a la rutina del lenguaje máquina.

Una vez haya iniciado la rutina, el ordenador contestará después de un momento con "READY", no pudiéndose ver lo que ha ocurrido. Dicho de otra manera, no puede apreciarse ninguna modificación. Sin embargo con este programa habrán cargado el acumulador con &FF, lo que no indica sobre el ordenador una acción directa.

Experimente ahora lo siguiente: ld bc,&A1F1. Ahora introduzca un POKE &2000,&1 (ld bc,nn), POKE &2001,&F1 (Lowbyte del número), POKE &2002,&A1 (Highbyte del número), POKE &2003,&C9 (RET).

Vuelva a iniciar con CALL &2000. Nuevamente no podrá apreciarse ninguna modificación y sin embargo habrán cargado el par de registros "bc" con &A1F1. Secundariamente habrán utilizado una regla básica de la programación, o sea, que después de una instrucción en un operando de 16 Bits se insertará primero el Lowbyte y después el Highbyte.

Pasemos ahora a realizar algo con más sentido y es escribir un programa de sumas para cifras de un Byte.

Digamos que el primer sumando se llamará ADD1, el segundo ADD2 y el resultado RES. Dejemos el ADD1 en el espacio de memoria &3000, el ADD2 en el &3002 y el resultado en el &3004. Tendremos que encontrar una posibilidad de poder sumar ambos valores. En primer lugar llevaremos el primer sumando (ADD1) al acumulador y después la dirección del segundo (ADD2) al registro "hl". Esto se realizará mediante las instrucciones ld a,(&3000), ld hl,&3002. No cargaremos de inmediato el registro "hl" con ADD2 con el fin de poderle demostrar las posibilidades de programación.

Necesitamos todavía una instrucción más para poder sumar ambos sumandos. Esta es add a,(hl). Esta instrucción determina que el contenido del espacio de memoria, que está direccionado por el registro "hl" haciendo que el contenido del espacio de memoria se encuentre en el ADD2, sea añadido al contenido del acumulador. Finalmente tendremos que almacenar el resultado en el espacio de memoria &3004. Esto podrá realizarse mediante la instrucción ld (&3004),a. Al final estará todavía el RET para poder regresar al Basic.
Así pues, habremos realizado el programa mútuamente sin haber hecho comentario alguno. Detrás de las instrucciones de ensamblaje estarán los valores hexadecimales que Vd. deberá cargar en la memoria particular.

```
ld a,(&3000)        &3A &00 &30
ld hl,&3002         &21 &02 &30
add a,(hl)          &86
ld (&3004),a        &32 &04 &30
RET                 &C9
```

Para poder entrar este programa en Basic, deberá utilizar la siguiente rutina de Basic:

```
10 FOR N=&2000 TO &200A
20 INPUT VALOR
30 POKE N,VALOR
40 NEXT
```

Después de iniciar esta rutina deberá entrar los valores indicados anteriormente. Tan sólo deberá entrar dos cifras (hasta máximo 127) en los espacios de memoria &3000 y &3002 con POKE &3000,ADD1:POKE &3002,ADD2.

Inicie el programa máquina con CALL &2000.
Después del "READY" podrá saber el resultado (RES) mediante PEEK (&3004).

Las cifras sólo podrán llegar a 127 para que el resultado no sea mayor que 255, ya que de otro modo no cabría en un Byte.

Pasemos ahora a ver el proceso de una suma con el código binario mediante el siguiente ejemplo:
ADD1 es 127 y ADD2 es 64

```
ADD1=  01111111
ADD2=  01000000

RES =  10111111  ó decimal 191
```

Enseguida podrán ver que la suma de números binarios no es difícil. Si un Bit está ya fijado y el Bit correspondiente del segundo sumando no lo está, entonces en el resultado se fijará este Bit pero invertido. Si los dos Bits correspondientes se han fijado, entonces no se fijará el Bit en el sumando pero se creará una suma anterior (CARRY). Esta se

valorará en el Bit izquierdo (véanse Bits 6 y 7). Si la suma anterior se realiza en el Bit de la izquierda, o sea el Bit 7, entonces no se valorará en una operación de ocho Bits, pero sí en una suma de 16 Bits. De todo ello puede deducirse que únicamente pueden utilizarse cifras hasta 127, ya que de otro modo aparecería la suma anterior.

Una suma de dieciseis Bits es en principio lo mismo que una de ocho Bits. Para ello deberemos nombrar más espacios de memoria y entrar dos nuevas instrucciones.

En primer lugar nombraremos los espacios de memoria para los sumandos. Para el primer sumando (ADD1) tenemos a disposición los espacios de memoria &3001, más tarde denominado ADR1, y &3000, más tarde denominado ADR1-1; para el segundo sumando (ADD2) estarán disponibles los espacios de memoria &3003 (ADR2) y &3002 (ADR2-1). El resultado se almacenará en &3005 (RES) y &3004 (RES-1).

Ahora pasemos al programa:
En primer lugar se sumarán los dos Lowbytes de los números y el resultado será almacenado en RES. En el caso de aparecer una suma anterior se fijará el Carryflag (carácter de suma anterior). Este será valorado en la segunda parte del programa.

En esta segunda parte se sumarán los Highbytes y se valorará la posible suma anterior. A continuación exponemos el programa escrito simbólicamente:

```
ld a,(ADR1)     Cargar el Lowbyte del ADD1 en el
                acumulador.
ld hl,ADR2      Cargar la dirección del Lowbyte
                del ADD2 después de hl.
add a,(hl)      Sumar los Lowbytes.
ld (RES),a      Almacenar el Lowbyte del resultado
                después de RES.
```

```
ld a,(ADR1-1)   Cargar el Highbyte del ADD1 en el
                acumulador.
dec hl          Restar uno de hl.
adc a,(hl)      Sumar los Highbytes y la suma anterior.
ld (RES-1),a    Almacenar el Highbyte del resultado
                después de RES-1.
```

La primera nueva instrucción es dec hl. Dicha instrucción realiza lo mismo que ld hl,ADR-1, ya que restando uno de "hl" encontraremos en "hl" el ADR1-1 en lugar de ADR1. La ventaja de dec hl es que la instrucción sólo tiene un Byte y por lo tanto es mucho más elegante y rápida.

La segunda nueva instrucción es adc a,(hl). Dicha instrucción realiza en principio lo mismo que add a,(hl), pero en adc a,(hl) también se tendrá en cuenta la suma anterior de la primera parte.

Y ahora pasemos a la parte práctica: A continuación tendremos el programa con los correspondientes valores y detrás los valores hexadecimales que Vd. deberá entrar.

```
ld a,(&3001)      &3A &1 &30
ld hl,&3003       &21 &3 &30
add a,(hl)        &86
ld (&3005),a      &32 &5 &30
ld a,(&3000)      &3A 00 &30
dec hl            &2B
adc a,(hl)        &8E
ld (&3004),a      &32 &4 &30
RET               &C9
```

Para poder entrar este programa necesitará de nuevo el bucle, pero deberá modificar la línea 10 en 10 FOR N=&2000 TO &2012.

Ahora deberá entrar el primer sumando en los espacios de

memoria &3000 y &3001, es decir el Highbyte en &3000 y el Lowbyte en &3001; lo mismo deberá realizarse con el segundo sumando en &3002 y &3003. Supongamos que el primer sumando tiene el valor de &5F4C y el segundo &8B5A; ahora escriba lo siguiente:

```
POKE &3000,&5F:POKE &3001,&4C:POKE &3002,&8B:POKE &3003,&5A
```

Inicie ahora el programa máquina con CALL(&2000).

Después del "READY" deberá entrar:
PRINT PEEK(&3004)*256+PEEK(&3005).

Ahora obtendremos el resultado de la suma en la pantalla y según nuestras cifras es 60070. Este resultado puede asimismo ser controlado sumando 24396 (&5F4C) y 35674 (&8B5A).

Al sumar dos números de 16 Bits, cuyos Lowbytes juntos son mayores que &FF, se crea una suma anterior. Esta suma se almacenará en el Carryflag. Este Carryflag es un Bit del registro "f", el cual no ha sido comentado al principio de todo. Este registro "f" contiene todas las informaciones importantes que requiere el CPU, por ejemplo si se ha tenido una suma anterior o bien si el registro "b" es igual a cero. Si se tiene una suma anterior se fijará en el registro "f" el Carryflag. Esta suma anterior se calculará en el Bit cero del Highbyte del resultado. Para poder demostrar esto exponemos a continuación un ejemplo:

```
      Highbyte          Lowbyte

ADD1=00010000 (&10)   10000000 (&80)

ADD2=00001000 (&08)   10000001 (&81)
```

RES =00011000 (&18) 00000001 (&01)

Como podrá comprobar, en un cálculo sin suma anterior se han perdido 256. Si se desea tener en cuenta el Carryflag fijado deberá añadirse al Highbyte un uno. Del resultado falso &1801 se obtendrá el resultado correcto de &1901.

Así pues es necesario tener en cuenta en un cálculo de 16 Bits el Carryflag correspondiente. Esto lo hemos hecho en nuestro programa con la instrucción adc a,(hl), para que el resultado, siempre y cuando se tenga una suma anterior, no sea 256 más pequeño.

Ahora pasemos a algo más sencillo. Aquí deseamos fijar puntos en la pantalla. Esto es posible mediante la instrucción PLOT X,Y. También se puede conseguir cargando la memoria particular de la pantalla. Esta empieza en &C000 y termina en &FFFF. Escriba Vd. el siguiente programa:

```
10 MODE 2
20 FOR N= &C000 TO &FFFF
30 POKE N,&FF
40 NEXT
```

Después del inicio del programa podrá ver como se llena la pantalla y no de línea en línea sino llenando primero una línea, después dejando siete en blanco, para volver a marcar la octava línea. Una línea es aquí tan elevada como un punto de trazo. Todo ésto depende de la configuración de los caracteres.

Para poder comprender completamente el siguiente programa máquina deberá tener bien presente, que un bucle de este tipo podrá ser creado de la siguiente manera:

```
10 N=N+1
20 IF N=&FFFF THEN END
```

Debe de haber también una posibilidad de escribir el bucle referenciado en lenguaje máquina. Esta ha sido realizada en parte a continuación, ya que sólo se llenarán las tres primeras líneas. El motivo para ello es que no deseamos agobiarlos con demasiadas nuevas instrucciones.

Aquí deberemos cargar un espacio de memoria con &FF. Esto se conseguirá nuevamente mediante una instrucción "ld". A continuación cargaremos el próximo espacio de memoria. Si no deseamos cargar todos los espacios de memoria con una instrucción de carga propia, deberemos escribir un bucle. Para ello se ofrece el registro "b", ya que así puede controlarse fácilmente.

Las siguientes variables son las que se utilizarán:

Longitud	Longitud del campo a ser rellenado
ADR1	Dirección de inicio para ser rellenada.

En primer lugar deberemos definir el bucle. Esto se conseguirá mediante la instrucción ld b, longitud.. Después cargaremos el acumulador con el valor &FF. A continuación cargaremos el "hl" con ADR1, lo que en lenguaje máquina quiere decir que pondremos un indicador en la dirección de inicio. Finalmente cargaremos la dirección denominada mediante el registro "hl" con el contenido del acumulador, es decir &FF. Ahora aumentaremos el "hl" en uno para que el registro indique la siguiente dirección. Del registro "b" restaremos un uno para poder señalar al ordenador que sólo pueden fijarse tantos puntos como el registro "b" indica. Las operaciones de incremento se consiguen mediante la instrucción inc hl, y las de disminución con la instrucción dec b. La próxima instrucción deberá ser un salto condicionado. Un salto condicionado corresponde a un "IF... THEN GOTO..." del Basic. Así pues cuando se haya cumplido con una condición, se realizará un salto. Este salto deberá efectuarse con la instrucción ld (hl),a, ya que de otro modo se cargaría únicamente el primer espacio de memoria. La condición para nuestro salto deberá llamarse "NO CERO",

ya que deberán efectuarse tantos saltos hasta que el registro "b" no sea igual a cero. La instrucción para nuestra condición se llama jp nz, dirección; la nz equivale a "not zero" (no cero). Por último escribiremos todavía un RET, para poder volver al Basic. El programa entero tiene el siguiente aspecto:

```
ld b,&C0        &6 &C0
ld a,0          &3E &FF
ld hl,&C000     &21 &0 &C0
ld (hl),a       &77
inc hl          &23
dec b           &5
jp nz,&2007     &C2 &7 &20
RET             &C9
```

Indique los valores en un bucle tal como lo hemos descrito anteriormente. Dicho bucle deberá ir de &2000 a &200D.

Antes de iniciar el programa conmute el ordenador con MODE 2 a la modalidad de caracteres 80. Aunque se encuentre ya en la modalidad de caracteres 80, conmútelo de todas maneras para evitar tener complicaciones.

Inicie ahora con CALL &2000 y se marcarán tres líneas.

La dirección de salto pasará a la dirección en la que esté el ld (hl),a. El proceso técnico de todo ello no es del todo sencillo. Aquí tendremos que introducir el registro PC (contador de programa). Dicho registro PC contiene la dirección de la instrucción que deberá ser ejecutada. Si en este registro PC se inserta una nueva dirección, el ordenador tratará la instrucción en la dirección de entrada. Una nueva posibilidad de calcular el salto es descontar respectivamente aumentar el valor del contador del programa que se necesitará para poder conseguir la dirección correspondiente.

Este tipo de salto se llama relativo, ya que no se carga en el contador PC la dirección real absoluta, sino que se consigue mediante el descuento o aumento de un valor. El salto relativo se expresará a través de la instrucción "jr e", siendo la "e" para el factor que se añadirá, respectivamente descontará del contador del programa. Sin embargo, el cálculo de un salto de este tipo es algo más difícil de lo que se ha explicado aquí. Tal y como ya se ha dicho, este capítulo únicamente pretende representar una introducción y para ello se aclararán únicamente algunos conceptos básicos.

CARACTERES + INSTRUCCIONES

A continuación encontrarán una tabla con los caracteres y Mnemónicos así como los códigos hexadecimales y decimales de su CPC.

Código	Carácter	Hex	Ensambl. Z80	Según CBH	Según EDH
0	Cero	00	nop	rlc b	
1	SOH	01	ld bc,NN	rlc c	
2	STX	02	ld (bc),a	rlc d	
3	ETX	03	inc bc	rlc e	
4	EOT	04	inc b	rlc h	
5	ENQ	05	dec b	rlc l	
6	ACK	06	ld b,N	rlc (hl)	
7	BEL	07	rlca	rlc a	
8	BS	08	ex af,af"	rrc b	
9	HT	09	add hl,bc	rrc c	
10	LF	0A	ld a,(bc)	rrc d	
11	VT	0B	dec bc	rrc e	
12	FF	0C	inc c	rrc h	
13	CR	0D	dec c	rrc l	
14	SO	0E	ld c,N	rrc (hl)	
15	SI	0F	rrca	rrc a	
16	DLE	10	djnz DIS	rl b	
17	DC 1	11	ld de,NN	rl c	
18	DC 2	12	ld (de),a	rl d	
19	DC 3	13	inc de	rl e	
20	DC 4	14	inc d	rl h	
21	NAK	15	dec d	rl l	
22	SYN	16	ld d,N	rl (hl)	
23	ETB	17	rla	rl a	
24	CAN	18	jr DIS	rr b	
25	EM	19	add hl,de	rr c	
26	SUB	1A	ld a,(de)	rr d	
27	ESC	1B	dec de	rr e	
28	FS	1C	inc e	rr h	

29	GS	1D	dec e	rr l	
30	RS	1E	ld e,N	rr (hl)	
31	US	1F	rra	rr a	
32	SP	20	jr nz,DIS	sla b	
33	!	21	ld hl,NN	sla c	
34	"	22	ld(NN),hl	sla d	
35	#	23	inc hl	sla e	
36	$	24	inc h	sla h	
37	%	25	dec h	sla l	
38	&	26	ld,h,N	sla (hl)	
39	'	27	daa	sla a	
40	(	28	jr z,DIS	sra b	
41	)	29	add hl,hl	sra c	
42	*	2A	ld hl,(NN)	sra d	
43	+	2B	dec hl	sra e	
44		2C	inc l	sra h	
45	-	2D	dec l	sra l	
46	.	2E	ld l,N	sra (hl)	
47	/	2F	cpl	sra a	
48	0	30	jr nc,DIS		
49	1	31	ld sp,NN		
50	2	32	ld (NN),a		
51	3	33	inc sp		
52	4	34	inc (hl)		
53	5	35	dec (hl)		
54	6	36	ld (hl),N		
55	7	37	scf		
56	8	38	jr c,DIS	srl b	
57	9	39	add hl,sp	srl c	
58	:	3A	ld a,(NN)	srl d	
59	;	3B	dec sp	srl e	
60	<	3C	inc a	srl h	
61	=	3D	dec a	srl l	
62	>	3E	ld a,N	srl (hl)	
63	?	3F	ccf	srl a	
64	@	40	ld b,b	bit 0,b	in b,(c)

65	A	41	ld b,c	bit 0,c	out(c),b
66	B	42	ld b,d	bit 0,d	sbc hl,bc
67	C	43	ld b,e	bit 0,e	ld(NN),bc
68	D	44	ld b,h	bit 0,h	neg
69	E	45	ld b,l	bit 0,l	rent
70	F	46	ld b,(hl)	bit 0,(hl)	im 0
71	G	47	ld b,a	bit 0,a	
72	H	48	ld c,b	bit 1,b	inc c,(c)
73	I	49	ld c,c	bit 1,c	out (c),c
74	J	4A	ld c,d	bit 1,d	adc hl,bc
75	K	4B	ld c,e	bit 1,e	ld bc,(NN)
76	L	4C	ld c,h	bit 1,h	
77	M	4D	ld c.l	bit 1,l	reti
78	N	4E	ld c,(hl)	bit 1,(hl)	
79	O	4F	ld c,a	bit 1,a	
80	P	50	ld d,b	bit 2,b	in d,(d)
81	Q	51	ld d,c	bit 2,c	out (c),d
82	R	52	ld d,d	bit 2,d	sbc hl,de
83	S	53	ld d,e	bit 2,e	ld (NN),de
84	T	54	ld d,h	bit 2,h	
85	U	55	ld d,l	bit 2,l	
86	V	56	ld d,(hl)	bit 2,(hl)	im 1
87	W	57	ld d,a	bit 2,a	ld a,i
88	X	58	ld e,b	bit 2,b	in e,(c)
89	Y	59	ld e,c	bit 3,c	out (c),e
90	Z	5A	ld e,d	bit 3,d	adc hl,de
91	[	5B	ld e,e	bit 3,e	ld de,(NN)
92	\	5C	ld e,h	bit 3,h	
93	]	5D	ld e,l	bit 3,l	
94	↑	5E	ld e,(hl)	bit 3,(hl)	im 2
95	-	5F	ld e,a	bit 3,a	
96		60	ld h,b	bit 4,b	in h,(c)
97	a	61	ld h,c	bit 4,c	out (c),h
98	b	62	ld h,d	bit 4,d	sbc hl,hl
99	c	63	ld h,e	bit 4,e	ld (NN),hl
100	d	64	ld h,h	bit 4,h	

101	e	65	ld h,l	bit 4,l	
102	f	66	ld h,(hl)	bit 4,(hl)	
103	g	67	ld h,a	bit 4,a	rrd
104	h	68	ld l,b	bit 5,b	in l,(c)
105	i	69	ld l,c	bit 5,c	out (c),l
106	j	6A	ld l,d	bit 5,d	adc hl,hl
107	k	6B	ld l,e	bit 5,e	ld de,(NN)
108	l	6C	ld l,h	bit 5,h	
109	m	6D	ld l,l	bit 5,l	
110	n	6E	ld l,(hl)	bit 5,(hl)	
111	o	6F	ld l,a	bit 5,a	rld
112	p	70	ld (hl),b	bit 6,b	
113	q	71	ld (hl),c	bit 6,c	sbc hl,sp
114	r	72	ld (hl),d	bit 6,d	ld (NN),sp
115	s	73	ld (hl),e	bit 6,e	
116	t	74	ld (hl),h	bit 6,h	
117	u	75	ld (hl),l	bit 6,l	
118	v	76	alto	bit 6,(hl)	
119	w	77	ld (hl),a	bit 6,a	
120	x	78	ld a,b	bit 7,b	in a,(c)
121	y	79	ld a,c	bit 7,c	out (c),a
122	z	7A	ld a,d	bit 7,d	adc hl,sp
123		7B	ld a,e	bit 7,e	ld sp,(NN)
124	G	7C	ld a,h	bit 7,h	
125	R	7D	ld a,l	bit 7,l	
126	A	7E	ld a,(hl)	bit 7,(hl)	
127	P	7F	ld a,a	bit 7,a	
128	H	80	add a,b	res 0,b	
129	I	81	add a,c	res 0,c	
130	C	82	add a,d	res 0,d	
131		83	add a,e	res 0,e	
132	-	84	add a,h	res 0,h	
133		85	add a,l	res 0,l	
134	Z	86	add a,(hl)	res 0,(hl)	
135	E	87	add a,a	res 0,a	
136	I	88	adc a,b	res 1,b	

137	C	89	adc a,c	res 1,c	
138	H	8A	adc a,d	res 1,d	
139	E	8B	adc a,e	res 1,e	
140	N	8C	adc a,h	res 1,h	
141		8D	adc a,l	res 1,l	
142	+	8E	adc a,(hl)	res 1,(hl)	
143		8F	adc a,a	res 1,a	
144	T	90	sub b	res 2,b	
145	O	91	sub c	res 2,c	
146	K	92	sub d	res 2,d	
147	E	93	sub e	res 2,e	
148	N	94	sub h	res 2,h	
149	S	95	sub l	res 2,l	
150		96	sub (hl)	res 2,(hl)	
151		97	sub a	res 2,a	
152		98	sbc a,b	res 3;b	
153		99	sbc a,c	res 3,c	
154		9A	sbc a,d	res 3,d	
155		9B	sbc a,e	res 3,e	
156		9C	sbc a,h	res 3,h	
157		9D	sbc a,l	res 3,l	
158		9E	sbc a,(hl)	res 3,(hl)	
159		9F	sbc a,a	res 3,a	
160		A0	and b	res 4,b	
161		A1	and c	res 4,c	cpi
162		A2	and d	res 4,d	ini
163		A3	and e	res 4,e	outi
164		A4	and h	res 4,h	
165		A5	and l	res 4,l	
166		A6	and (hl)	res 4,(hl)	
167		A7	and a	res 4,a	
168		A8	xor b	res 5,b	ldd
169		A9	xor c	res 5,c	cpd
170		AA	xor d	res 5,d	ind
171		AB	xor e	res 5,e	outd
172		AC	xor h	res 5,h	

173	AD	xor l	res 5,l	
174	AE	xor (hl)	res 5,(hl)	
175	AF	xor a	res 5,a	
176	B0	or b	res 6,b	ldir
177	B1	or c	res 6,c	cpir
178	B2	or d	res 6,d	inir
179	B3	or e	res 6,e	otir
180	B4	or h	res 6,h	
181	B5	or l	res 6,l	
182	B6	or (hl)	res 6,(hl)	
183	B7	or a	res 6,a	
184	B8	cp b	res 7,b	lddr
185	B9	cp c	res 7,c	cpdr
186	BA	cp d	res 7,d	indr
187	BB	cp e	res 7,e	otdr
188	BC	cp h	res 7,h	
189	BD	cp l	res 7,l	
190	BE	cp (hl)	res 7,(hl)	
191	BF	cp a	res 7,a	
192	C0	ret nz	set 0,b	
193	C1	pop bc	set 0,c	
194	C2	jp nz,NN	set 0,d	
195	C3	jp NN	set 0,e	
196	C4	call nz,NN	set 0,h	
197	C5	push bc	set 0,l	
198	C6	add a,N	set 0,(hl)	
199	C7	rst 0	set 0,a	
200	C8	ret z	set 1,b	
201	C9	ret	set 1,c	
202	CA	jp z,NN	set 1,d	
203	CB		set 1,e	
204	CC	call z,NN	set 1,h	
205	CD	call NN	set 1,l	
206	CE	adc a,N	set 1,(hl)	
207	CF	rst 8	set 1,a	
208	D0	ret nc	set 2,b	

209	D1	pop de	set 2,c
210	D2	jp nc,NN	set 2,d
211	D3	out N,a	set 2,e
212	D4	call nc,NN	set 2,h
213	D5	push de	set 2,l
214	D6	sub N	set 2,(hl)
215	D7	rst 16	set 2,a
216	D8	ret c	set 3,b
217	D9	exx	set 3,c
218	DA	jp c,NN	set 3,d
219	DB	in a,N	set 3,e
220	DC	call c,NN	set 3,h
221	DD		set 3,l
222	DE	sbc a,N	set 3,(hl)
223	DF	rst 24	set 3,a
224	E0	ret po	set 4,b
225	E1	pop hl	set 4,c
226	E2	jp po,NN	set 4,d
227	E3	ex (sp),hl	set 4,e
228	E4	call po,NN	set 4,h
229	E5	push hl	set 4,l
230	E6	and N	set 4,(hl)
231	E7	rst 32	set 4,a
232	E8	ret pe	set 5,b
233	E9	jp (hl)	set 5,c
234	EA	jp pe,NN	set 5,d
235	EB	ex de,hl	set 5,e
236	EC	call pe,NN	set 5,h
237	ED		set 5,l
238	EE	xor N	set 5,(hl)
239	EF	rst 40	set 5,a
240	F0	ret p	set 6,b
241	F1	pop af	set 6,c
242	F2	jp p,NN	set 6,d
243	F3	di	set 6,e
244	F4	call p,NN	set 6,h

245	F5	push af	set 6,l
246	F6	or N	set 6,(hl)
247	F7	rst 48	set 6,a
248	F8	ret m	set 7,b
249	F9	ld sp,hl	set 7,c
250	FA	jp m,NN	set 7,d
251	FB	ei	set 7,e
252	FC	call m,NN	set 7,h
253	FD		set 7,l
254	FE	cp N	set 7,(hl)
255	FF	rst	set 7,a

4.4 LA MEMORIA DEL CPC 464 Y 6128

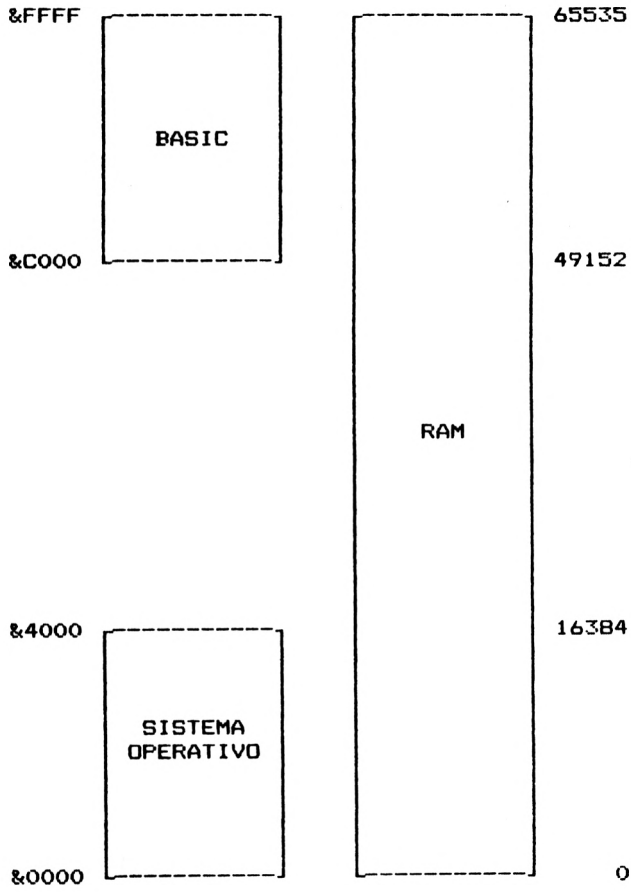
El que quiera profundizar más en su ordenador y aprovechar al máximo las posibilidades de este instrumento, deberá conocer más a fondo su configuración interna.

El CPC 464 y 6128 tienen como procesador un Z80. Esto es un procesador de 8 Bits que tiene 16 líneas de dirección. Por estas 16 líneas de dirección podrá direccionar 2^16 = 65536 células de memoria, siendo cada una de ellas de 8 Bits de ancho y por lo tanto podrá contener 2^8 = 256 valores distintos. Es posible que ya sepa, que su CPC contiene 64K-Bytes RAM y 32K-Bytes ROM. ¿ Cómo podrán conectarse los 96K-Bytes del procesador pudiéndose únicamente direccionar 64K-Bytes, tal como hemos visto anteriormente ?

Para ello nos valdremos de un truco. Aquí ocuparemos doblemente el campo de direccionamiento, una vez por el RAM y 'pasando por encima' por el ROM. Mediante un interruptor el procesador podrá elegir si desea tomar datos del ROM o del RAM. Al almacenar en la memoria, automáticamente siempre se elige el RAM, ya que el ROM no se deja grabar.

El CPC 464 y 6128 contienen un módulo ROM, que tiene los 32K-Bytes completos. Los 16K-Bytes inferiores representan el sistema funcional que el procesador contactará desde la dirección 0 hasta &3FFF (16383). Los 16K-Bytes superiores contienen el interpretador Basic y pueden ser contactados de &C000 hasta &FFFF (49152 - 65535). Esto puede representarse gráficamente de la siguiente manera:

OCUPACION DE MEMORIA



4.4.1 LA OCUPACION DE MEMORIA POR EL BASIC Y EL SISTEMA OPERATIVO

La mayor parte de la memoria RAM se utiliza para el almacenaje del programa y de las variables BASIC. Normalmente este campo va desde &170 (368) hasta &AB7F (43903), es decir que es de un tamaño de 43536 Bytes.

Si partiendo del BASIC lee Vd. la memoria con PEEK obtendrá siempre el contenido del RAM. Así pues no es tan fácil seleccionar el sistema operativo o el BASIC. Sin embargo el programa monitor presentado más adelante ofrece esta posibilidad. En primer lugar vayamos a conocer la disposición de la memoria RAM.

Los 64 Bytes inferiores de la memoria son una copia del contenido ROM del mismo campo de direccionamiento. Si conoce Vd. las instrucciones del procesador Z80, sabrá que este campo contiene las rutinas Restart que pueden ser llamadas mediante la instrucción de un Byte. Estos ocho vectores RST se utilizan en el CPC 464 y 6128 para llamar rutinas del ROM y del RAM.En este proceso eligen automáticamente la selección de memoria adecuada. Para que estas rutinas puedan también ser utilizadas cuando el sistema operativo ROM esté conectado así como cuando se seleccione el RAM inferior, el RAM tendrá en este caso una copia de estas rutinas. Ya que las instrucciones RST son ampliamente utilizadas por el sistema operativo y por el BASIC, trataremos algunas de ellas brevemente. Una aplicación de la RST 3 podremos encontrarla en nuestro monitor, el cual nos permitirá seleccionar la memoria ROM.

4.5 LAS INSTRUCCIONES RST

RST 0 Dirección &0000

A través de este vector se produce un Reset completo del ordenador tal y como ocurre al conectar el aparato o al presionar simultáneamente CTRL SHIFT ESC.

```
0000  01 89 7F      LD  BC,&7F89
0003  ED 49         OUT (C),C
0005  C3 80 05      JP  &0580
```

Mediante la instrucción OUT se selecciona el ROM para elaborar en el sistema operativo la rutina RESET a partir de la dirección &0580.

Desde el BASIC podrá llamar el RESET con CALL 0 y desde el lenguaje máquina con CALL 0, JP 0 o bien RST 0.

RST 1 Dirección &0008

Esta instrucción sirve para llamar una rutina del sistema operativo o bien del RAM situado por debajo de éste. Directamente detrás de la instrucción RST deberá estar la dirección de la rutina a ser llamada. Ya que para el campo de 0 hasta &3FFF bastan 14 Bits de direccionamiento, se utilizarán los dos Bits superiores para la selección del ROM o del RAM:

```
Bit 14 = 0    Sistema operativo seleccionado
       = 1    RAM seleccionado
Bit 15 = 0    BASIC-ROM seleccionados
       = 1    RAM seleccionado
```

Una llamada de la rutina del sistema operativo &0826 podría representarse de la siguiente manera:

```
RST 1
DW  &0826 + &8000
```

Mediante el Bit 15 fijado se habrá seleccionado el RAM en el campo de &C000 hasta &FFFF, mientras que con el Bit 14 borrado podrá contactarse el sistema operativo.

El código de la dirección 10 consta de un salto a &B982.

RST 2 Dirección &0010

Esta instrucción de Restart sirve para llamar una rutina en un ROM de expansión. Después de la instrucción RST 2 deberá estar la dirección de la rutina &C000, es decir, la dirección relativa que hace referencia al inicio del ROM. Los dos Bits superiores se utilizarán para la selección de cuatro ROMS distintos. En la dirección &0010 habrá un salto a &BA16.

RST 3 Dirección &0018

Con la ayuda de esta instrucción RST podrá llamar una rutina en cualquier parte del ROM o del RAM. Para ello deberá estar detrás de la instrucción RDT 3 la dirección de un bloque de parámetros que consta de tres Bytes. Estos dos primeros Bytes contienen la dirección de la rutina que deberá llamarse y el tercer Byte tendrá consecuentemente el estado deseado del ROM/RAM. De este modo se contactará mediante los valores de 0 a 251 el ROM adicional correspondiente. Los cuatro valores restantes tendrán la siguiente

función:

Valor	&0000-&3FFF	&C000-&FFFF
252	Sistema operativo	BASIC
253	Sistema operativo	RAM
254	RAM	BASIC
255	RAM	RAM

En la dirección &0018 hay un salto hacia &B9BF

RST 4 Dirección &0020

Con la ayuda de esta instrucción RST podrá leer de un programa máquina el contenido del RAM, independientemente del estado del ROM elegido. La instrucción RST 4 sustituye a la instrucción

LD A,(HL)

HL deberá contener la dirección de la célula de memoria a ser leída. En la dirección &0020 hay un salto hacia &BACB.

RST 5 Dirección &0028

Mediante esta instrucción RST podrá saltarse a una rutina del sistema operativo. La dirección deberá seguir inmediatamente después de la instrucción RST 5. En la dirección &0028 hay un salto hacia &BA2E.

RST 6 Dirección &0030

Esta instrucción RST no se utiliza por el sistema operativo y está a libre disposición del usuario; por ejemplo puede utilizarse en un Debugger (programa de eliminación de errores) para fijar un Breakpoint.

RST 7 Dirección &0038

Ya que el Z80 se activa en el CPC 464 y 6128 en la modalidad de interrupción 1, cuando aparece una interrupción entra en función el RST 7. La interrupción se trata a través de esta instrucción. En la dirección &0038 hay un salto hacia &B939.

4.6 MINI-MONITOR

A continuación les presentaremos un programa monitor en el que podrán indicar, modificar e incluso almacenar sobre Cassette todos los contenidos de la memoria. Como particularidad existe la posibilidad de poder servirse del sistema operativo y del BASIC-ROM. Esto se efectúa a través de las rutinas RST ya comentadas. Después del listado seguirá la descripción del programa correspondiente.

```
100 MEMORY &AAFF: INK 0,1: INK 1,24: INK 2,23: INK 3,15,24:
    MODE 1
110 n=6 : FOR i=0 TO n: READ cmd$(i): NEXT: DATA a,m,s,t,
    g,r,x
120 GOSUB 830
130 PEN 2: PRINT TAB (10) "M i n i - M o n i t o r"
    CHR$(13)TAB(10);: PEN 1: PRINT CHR$(22)CHR$(1)
    "________________________": PRINT: PRINT
140  PRINT:  PRINT "Instrucciones:";CHR$(13)CHR$(22)CHR$(1)
    "_____________"CHR$(22)CHR$(0):PRINT
150 PRINT " m = Indicar campo de memoria "
160 PRINT " a = Modificar contenido de memoria "
170 PRINT " s = Almacenar campo de memoria "
180 PRINT " t = Entrar texto ASCII "
190 PRINT " g = Realizar programa máquina "
200 PRINT " r = Seleccionar ROM/RAM "
210 PRINT " x = Regreso a BASIC "
220 PRINT
230 PRINT: PRINT "Emitir sobre  ";
240 PEN 3: PRINT "P";: PEN 1: PRINT "antalla ":PRINT TAB(9)
     "o ";: PEN 3: PRINT "I";: PEN 1: PRINT "mpresora ? ";
250 WHILE a$<>"b" AND a$<>"d": a$=INKEY$: WEND
260 INK 3,24: rom=0
```

```
270 PRINT:IF a$="d" THEN 1=16: c=8: GOTO 310
280 PRINT: PRINT: PRINT "Modalidad de pantalla ? "
290 WHILE a$<>"1" AND a$<>"2": a$=INKEY$: WEND
300 a=VAL(a$): 1=a*8: c=0: IF a<>1 THEN MODE a
310 PRINT: PRINT "Instrucción ?";
320 a$=INKEY: FOR i=0 TO n: IF a$=cmd$(i) THEN PRINT: ON
    i+1 GOSUB 490,540,590,700,750,780,940: GOTO 310
330 NEXT: GOTO 320
340 '
350 IF a>b THEN RETURN
360 PRINT#c, HEX$(a,4)"  ";
370 FOR i=1 TO 1: IF rom THEN GOSUB 890: ELSE x=PEEK(a)
380 PRINT#c, HEX$(x,2)+" ";: a=a+1
390 NEXT: PRINT#c, " ";: a=a-1
400 IF c=0 THEN 450
410 FOR i=1 TO 1: IF rom THEN GOSUB 890: ELSE x=PEEK(a)
420 x=x AND &7F
430 IF x<32 OR x=127 THEN x=46
440 PRINT#c, CHR$(x);: a=a+1: NEXT: PRINT#c: GOTO 350
450 IF 1=8 THEN PEN 2
460 FOR i=1 TO 1: IF rom THEN GOSUB 890: ELSE x=PEEK (a)
470 PRINT  CHR$(1)CHR$(x);: a=a+1:  NEXT: PRINT: PEN 1:
    GOTO 350
480 '
490 INPUT "Dirección ";x$: GOSUB 560: a=x
500 PRINT HEX$(a,4) ": ";
510 INPUT  x$:  a$=LEFT$(x$,1):  IF  a$<"0"  OR  a$>"9" AND
    a$<"a" OR a$>"f" THEN RETURN
520 GOSUB 560: POKE a,x: a=a+1: GOTO 500
530 '
540 GOSUB 650: GOSUB 350: RETURN
550 '
560 x=VAL("&"+x$): IF x<0 THEN x=x+2^16
570 RETURN
```

```
580 '
590 PRINT "Almacenar: ";
600 GOSUB 650
610 INPUT "Nombre ";a$
620 SAVE a$,b,a,b-a
630 RETURN
640 '
650 INPUT "de - hasta ";a$,b$
660 x$=a$: GOSUB 560: a=x
670 x$=b$: GOSUB 560: b=x
680 RETURN
690 '
700 INPUT "Dirección ";x$: GOSUB 560
710 LINE INPUT "Entrar texto: ";a$
720 IF  LEN(a$)>0 THEN  FOR i=1 TO LEN(a$): POKE x-1+i,
    ASC(MID$(a$,i,1)): NEXT
730 RETURN
740 '
750 INPUT "Dirección ";x$
760 GOSUB 560: CALL x: RETURN
770 '
780 PRINT: INK 3,15,24
790 PRINT  "Select: ";:  PEN 3:  IF rom  THEN PRINT "ROM";:
    ELSE PRINT "RAM";
800 PEN  1: a$=INKEY$:  IF a$=" " THEN rom=1-rom:
    PRINT CHR$(13);: GOTO 790
810 IF a$=CHR$(13) THEN PRINT: INK 3,24: RETURN: ELSE 800
820 '
830 'Cargar programa máquina
840 DATA &DF,&04,&AB,&C9
850 DATA &07,&AB,&FC
860 DATA &3A,&00,&00,&32,&0E,&AB,&C9
870 FOR i=0 TO &D: READ a: POKE &AB00+i,a: NEXT: RETURN
880 '
```

```
890 'Leer ROM
900 ah=INT(a/256): al=a-ah*256
910 POKE &AB08,al: POKE &AB09,ah
920 CALL &AB00
930 x=PEEK (&AB0E): RETURN
940 '
950 CLS: END
```

Descripción del programa para el Mini-Monitor

100 Bajar el límite superior de la memoria para BASIC determinación de los colores y elegir la modalidad de los 40 carácters.

110 Lectura de las instrucciones en la variable cmd$.

120 Cargar el programa máquina.

130-140 Emisión del título. Para el subrayado se utilizará la modalidad de transparencias.

150-220 Se indicarán las instrucciones y las letras correspondientes para llamar a las instrucciones pertinentes.

230-260 Fijación del medio de emisión para la instrucción M. Entrando la 'B' o la 'D' puede elegirse la pantalla o bien la impresora. Las letras correspondientes se representarán de modo reluciente (PEN 3).

270-300 Independientemente de la entrada de datos se fijará en la impresión el número de Bytes por línea (1) en 16 y el número de canal (C) en 8. En el caso de elegir la emisión de pantalla podría seleccionarse la modalidad entre 40 y 80 carácters
(1 ó 2).

310-330 Esto es el bucle principal donde se espera la entrada de una instrucción. El carácter introducido se comparará con los comandos y en caso de concordar se llamará al subprograma correspondiente.

350-390 Dump hexadecimal. Después de emitir la dirección se dará 1 (8 ó 16) Byte como número hexadecimal de dos posiciones. Si es leído por el RAM podría realizarse mediante PEEK. Si se representa el contenido del ROM entonces se leerá el ROM por medio de GOSUB 890.

400-470 Dump ASCII. Al emitir los caracteres correspondientes del ASCII se distinguirá en primer lugar si deberá efectuarse a través de la impresora o bien de la pantalla. Si se emite mediante la impresora se borrará el Bit superior del carácter. Si se trata de un carácter de control o bien de un código para DEL, se tomará como reemplazo el código ASCII para el punto. Si la emisión se hace sobre pantalla se conmutará en la modalidad de 40 caracteres al segundo color y todos los caracteres, incluyendo los de control (mediante CHR$(1)), se emitirán como caracteres representativos.

490-520 Esta rutina sirve para la modificación de contenidos de memoria. Después de que haya entrado una dirección, podrá insertar el nuevo contenido. Automáticamente después se entrará la siguiente dirección y se esperará a la próxima entrada de datos. Esta modalidad podrá finalizarla, entrando un carácter hexadecimal no válido, por ejemplo un punto.

540 Estas dos llamadas del subprograma tratan la instrucción M. Con el GOSUB 650 se entrarán las direcciones de inicio y de fin del campo a ser determinado, mientras que la segunda llamada se hará cargo del indicador.

560-570 Esta rutina transforma una secuencia de números hexadecimales entrados en un valor numérico, ahorrándole así la anteposición de '&'.

590-630 Esta rutina sirve para poder almacenar un campo RAM sobre Cassette. Para ello deberá indicar los límites de este campo así como su nombre, bajo el cual deberá realizarse.

650-680 Esta rutina sirve para poder entrar un campo de memoria. Los límites se darán en las variables "a" y "b".

700-730 Esta rutina se utiliza para almacenar en la memoria una cadena de textos. En la línea 720 se cargará en la memoria particular el texto a modo de caracteres.

750-760 Estas dos líneas de programa le facilitarán el poder realizar un programa máquina en la memoria RAM, cuya dirección deberá entrar.

780-810 Esta rutina sirve para la selección del RAM y del ROM. El estado momentáneo elegido se indicará de modo reluciente. Presionando la tecla espaciadora podrá conmutarse del RAM al ROM y viceversa. Si ya ha elegido la memoria deseada, podrá volver nuevamente a la inserción de instrucciones con la simple presión de la tecla ENTER.

830-870 Aquí se ha almacenado el programa máquina para leer el ROM en DATA-Statements, que serán escritos en la línea 870 en la posición correcta de la memoria.

890-930 Aquí se realiza la propia lectura de un Byte del ROM. La dirección del Byte a ser leído deberá darse en la variable "a". Después de desglosarla en Highbyte y Lowbyte se insertará en la rutina de máquina. Cuando se llame a la rutina con CALL, ésta depositará el resultado en la célula de memoria &AB0E, de donde podrá ser leída mediante PEEK.

950 Después de entrar una X se borrará la pantalla finalizando así el programa.

Para poder seleccionar el ROM del CPC 464 y 6128 se utilizará una rutina del sistema operativo. A través del vector 3 del Restart, el cual corresponde a la llamada de un subprograma de la dirección &18, podrá llamarse una rutina tanto del RAM como del ROM. Para ello deberá haber directamente detrás de la instrucción RST 3 la dirección de un vector de 3 Bytes para el llamado FAR CALL. Este vector contiene en primer lugar la dirección de la rutina que deberá ser realizada y en el siguiente Byte se tendrá la información sobre la posible elección del ROM o del RAM.

```
AB00                        ORG  &AB00
AB00 DF                     RST  3
AB01 04 AB                  DW   TABLE
AB03 C9                     RET
                    ;
AB04 07 AB          TABLE   DW   READROM
AB06 FC                     DB   &FC ; Seleccionar Rom
                    ;
AB07 3A 00 00 READROM LD    A,0 ; Leer Byte
AB0A 32 0E AB         LD    BYTE,A ; y almacenar
AB0D C9               RET
                    ;
AB0E                BYTE    DS    1   ;Espacio para Byte leído
```

La tabla contiene los valores correspondientes para la selección del ROM y del RAM.

Valor	Memoria seleccionada
&00 - &FB	ROM de expansión
&FC	Sistema operativo + BASIC
&FD	BASIC
&FE	Sistema operativo
&FF	RAM

Con un valor de &FC podrán elegirse tanto el sistema operativo inferior como el BASIC superior del ROM. Para la misma rutina que está en el RAM ésto no tiene importancia, ya que en el campo de direccionamiento del &4000 hasta el &BFFF sólo existe el RAM. La selección del Byte a ser leido se realiza almacenando en memoria particular el Low y Highbyte detrás de la instrucción LD de la dirección READROM antes de llamar a la rutina.

Como ejemplo para una emisión de monitor encontrarán un Dump hexadecimal al principio de este programa como ejemplo de un contenido del RAM. El segundo ejemplo reproduce el inicio del ROM BASIC.

```
0160  00 00 00 00 00 00 00 00  ........
0168  00 00 00 00 00 00 00 00  ........
0170  31 00 64 00 AA 20 1C FF  1.d.* ..
0178  AA 01 20 A2 20 0E 2C 0F  *. " .,.
0180  01 20 A2 20 0F 2C 19 18  . " .,..
0188  01 20 A2 20 10 2C 19 17  . " .,..
0190  01 20 A2 20 11 2C 19 0F  . " .,..
0198  2C 19 18 01 20 AD 20 0F  ,... - .
01A0  00 41 00 6E 00 0D 05 00  .A.n....
01A8  EE EF 14 20 01 20 9E 20  no. . .
01B0  0D 0E 00 E9 EF 0E 20 EC  ...io. l
01B8  20 0D 05 00 EE 01 20 C3   ...n. C
```

```
01C0  20 03 09 00 63 6D E4 28   ...cmd(
01C8  0D 0E 00 E9 29 01 20 B0   ...i). 0
01D0  01 20 8C 20 61 2C 6D 2C   . . a,m,
01D8  73 2C 74 2C 67 2C 72 2C   s,t,g,r,
01E0  78 00 0A 00 78 00 9F 20   x...x..
01E8  1D 06 0B 00 6B 00 82 00   ....k...
01F0  BB 20 10 01 20 BF 20 EA   ; .. ? j
01F8  20 28 19 0A 29 20 22 4D    (..) "M
0200  20 69 20 6E 20 69 20 2D    i n i -
```

Inicio del ROM BASIC

```
C000  80 01 00 00 4C C0 31 00   ....L@1.
C008  C0 CD CB BC CD C4 F4 DA    MK<MDtZ
C010  00 00 21 00 AC 36 00 06   ..!.,6..
C018  1B 23 36 C9 10 FB 21 3F   .#6I.;!?
C020  C0 CD 37 C3 AF 32 00 AC   @M7C/2.,
C028  CD CB DD CD 84 CA CD 97   MKäM.JM.
C030  BD CD D3 C0 CD 3E C1 11   =MS@M>A.
C038  F0 00 CD 06 F7 18 25 20   p.M.w.%
C040  42 41 53 49 43 20 31 2E   BASIC 1.
C048  30 0A 0A 00 42 41 53 49   0...BASI
C050  C3 00 CD E1 CE C0 31 00   C.MaN@1.
C058  C0 CD 9A E7 CD 63 E1 CD   @M.gMcaM
C060  43 CA 38 54 CD 01 AC 31   CJ8TM.,1
C068  00 C0 CD 62 C1 CD D6 DD   .@MbAMVä
C070  DC B6 BC CD 48 BB CD 86   ö6<MH;M.
C078  C3 3A 45 AE B7 C4 3E C1   C:E.7D>A
```

C A P I T U L O 5

ARCHIVO DE LINEAS BASIC, VARIABLES Y TOKENS

5.1 ARCHIVO DE UNA LINEA BASIC

Para poderse realizar este capítulo necesitamos el programa del monitor del lenguaje máquina del capítulo 4. En el caso de que Vd. todavía no lo hubiese escrito, le aconsejamos hacerlo para evitar que aparezcan dificultades de comprensión.

En este capítulo se tocará un tema que podría denominarse como miembro de unión entre el lenguaje máquina y Basic. Aquí se trata del archivo de una línea Basic y de su estructuración. Además comentaremos el archivo de variables en una línea Basic.

Tal como ya sabrán del capítulo 4, sus CPC 464 y 6128 entienden únicamente el código binario. Mediante el sistema operativo y el interpretador Basic (traductor) se transforman todas las instrucciones Basic en dichos códigos binarios. Si Vd. escribe la línea Basic 1 PRINT "DATA BECKER", su CPC no entenderá nada sin el interpretador Basic. Sin embargo, el ordenador tampoco lo entiende todo con el interpretador. Vd. ya lo habrá experimentado al principio de sus pruebas de programación por los numerosos mensajes de "SYNTAX ERROR".

Y ahora pasemos a la parte práctica: Complete su programa del monitor en lenguaje máquina con la línea anteriormente indicada de 1 PRINT "DATA BECKER". Inicie pues el programa monitor y elija el sistema decimal y la correspondiente modalidad de lectura. Como dirección de inicio indique el 368 y como dirección final el 386. La expresión siguiente la hemos escrito nuevamente para evitar que surjan malos entendidos. En la columna de la derecha hemos añadido además un breve comentario, a pesar de que todo será oportunamente aclarado.

Dirección:	Valor:	Comentario:
368	20	Lowbyte de la long. de línea
369	0	Highbyte de la long. de lín.
370	1	Lowbyte del núm. de línea
371	0	Highbyte del núm. de línea
372	191	TOKEN para PRINT
373	32	Valor ASCII para espacio en blanco
374	34	Valor ASCII para carácter de exclamación
375	68	Valor ASCII para "D"
376	65	Valor ASCII para "A"
377	84	Valor ASCII para "T"
378	65	Valor ASCII para "A"
379	32	Valor ASCII para espacio en blanco
380	66	Valor ASCII para "B"
381	69	Valor ASCII para "E"
382	67	Valor ASCII para "C"
383	75	Valor ASCII para "K"
384	69	Valor ASCII para "E"
385	82	Valor ASCII para "R"
386	34	Valor ASCII para carácter de exclamación

A continuación aclararemos los valores de arriba a abajo. Los conceptos Lowbyte y Highbyte deberían tenerlos todavía en memoria del capítulo 2; si no fuera así, les detallamos a continuación la formula para el cálculo del propio número que se expresa mediante Highbyte y Lowbyte:

```
HIGHBYTE*256+LOWBYTE
```

Como longitud de línea resulta el número 20. Si cuenta las direcciones impresas anteriormente obtendrá tan sólo la cifra 19. Esto es debido a que para su CPC la longitud de

línea no es interesante, ya que únicamente necesita este valor para poder calcular la dirección de inicio de la siguiente línea. El número de línea consta nuevamente de Highbyte y Lowbyte; si no fuere así sólo obtendríamos números de líneas hasta el 255. El número de línea es 1 y ésto puede Vd. deducirlo del Highbyte y Lowbyte. El próximo valor es el llamado Token. Esta palabra y su significado se aclara detenidamente en el siguiente capítulo 5.2, ya que es uno de los conceptos más importantes en relación con la estructura de líneas Basic.

Los valores siguientes son los de ASCII para la composición tipográfica "DATA BECKER".

Tal y como está estructurada esta línea, así será en principio cada línea del programa monitor o de cualquier otro programa. Existe únicamente una excepción notable y es una línea Basic en la que se define una variable. En cuanto a esta excepción pasaremos a comentarla más adelante.

5.2 TOKENS

El concepto TOKEN procede, como casi toda palabra técnica del mundo del ordenador, del inglés y significa "carácter". El Token es un valor que está asignado a una instrucción Basic, exactamente igual como los valores binarios de las instrucciones del lenguaje máquina. Este se creó con el fin de ahorrar espacio de memoria. El Token para el Print tiene, por ejemplo, únicamente un Byte. Si Vd. cuenta ahora las letras obtendrá cinco valores, es decir que normalmente se necesitarían cinco Bytes para PRINT. Además, el ordenador puede distinguir mucho mejor la instrucción de cualquier variable que tenga un nombre similar. Por este motivo no está permitido definir una variable con el nombre de PRINT, ya que el ordenador interpretaría este conjunto de letras como una instrucción y aplicaría el correspondiente Token.

Hemos recopilado una lista de todos los Tokens, que estarán representados mediante un valor. Esta lista no está del todo completa, ya que los Tokens que se distinguen con dos Bytes faltan.

A continuación un buen consejo para la protección del programa:
Como primera línea de su programa escriba una línea REM en la que inserte entre caracteres de exclamación su correspondiente "Copyright". Dicha línea sería aproximadamente según sigue:

```
1 REM "COPYRIGHT BY DATA BECKER"
```

Como segunda línea aplique una observación importante sin la cual no podría funcionar el programa. Una observación de este tipo podría ser por ejemplo un DIM o bien una definición de función. Aplique pues en medio del programa una línea con el siguiente contenido.

```
POKE 372,191
```

Esto induce a que de la primera línea REM se transforme en una línea PRINT. Si falta la línea REM, la segunda línea

con la instrucción importante quedará destruida. Este pequeño truco evita que las personas ajenas al tema saquen la línea "COPYRIGHT".

Instruc.	Valor	Instruc.	Valor	Instruc.	Valor
After	128	Mid	172	Tron	211
Auto	129	Mode	173	Wait	212
Border	130	Move	174	Wend	213
Call	131	Mover	175	While	214
Cat	132	Next	176	Width	215
Chain	133	New	177	Window	216
Chain Merge	133&171	On	178	Write	217
Clear	134	On Break	179	Zone	218
Clg	135	On Error goto	180	Di	219
Close in	136	On sq	181	Ei	220
Close out	137	Open in	182	Erl	227
Cls	138	Open out	183	Fn	228
Cont	139	Origin	184	Spc	229
Data	140	Out	185	Step	230
Def fn	141	Paper	186	Swap	231
Def int	142	Pen	187	Tab	232
Def read	143	Plot	188	Then	235
Def Str$	144	Plotr	189	To	236
Deg	145	Poke	190	Using	237
Delete	146	Print	191	>	238
Dim	147	Chr$(39)	192	=	239
Draw	148	Rad	193	>=	240
Drawr	149	Randomize	194	<	241
Edit	150	Read	195	<>	242
Else	151	Release	196	<=	243
End	152	Rem	197	+	244
Ent	153	Renum	198	-	245
Env	154	Restore	199	*	246
Erase	155	Resume	200	/	247
Error	156	Return	201	↑	248
Every	157	Run	202	\	249

For	158	Save	203	And	250
Gosub	159	Sound	204	Mod	251
Goto	160	Speed	205	Or	252
If	161	Stop	206	Xor	253
Ink	162	Symbol	207	Not	254
Input	163	Tag	208	Abs	255
Key	164	Tag off	209		
Key Def	164&141	Troff	210		
Let	165				
Line Input	166&163				
List	167				
Load	168				
Locate	169				
Memory	170				
Merge	171				

Los valores Token 221 hasta 226 así como 232 y 233 no están utilizados pero crean un efecto realmente interesante. Como primera línea escriba 1 REM y memorice el segundo número de línea. A continuación escriba POKE 372,226. Si después intenta listar el programa con LIST, aparecerá un "SYNTAX ERROR". Si inicia el programa únicamente con "RUN", su CPC indicará contínuamente un "SYNTAX ERROR IN 1", que Vd. únicamente podrá interrumpir mediante un RESET (presión simultánea de las teclas "CTRL", "SHIFT" y "ESC"), pero su programa se perderá. Por ésto inicie Vd. el programa con GOTO, segundo número de línea, que suponemos que Vd. habrá memorizado según se ha comentado anteriormente. Un principiante iniciará el programa con "RUN" y así lo destruirá. Si Vd. desea eliminar nuevamente esta protección del programa, deberá escribir POKE 372,197.

A continuación una breve observación: El programa que lleve esta protección de programa, no deberá tener ningún GOTO 1 o bien GOSUB 1, ya que también se destruiría.

5.3 ARCHIVO DE VARIABLES

Tal como se ha comentado en el capítulo 5.1, la gran excepción de las líneas Basic es aquélla en la que se define una variable. Nosotros intentaremos aclararle este tema sumamente problemático del modo más fácil posible. Para ello hemos elegido cinco casos representativos de variables.

Antes de que Vd. continúe leyendo, desearíamos rogarle que cargue el programa del monitor en lenguaje máquina del capítulo 4 para que Vd. pueda seguir por completo todas nuestras explicaciones.

Como primer ejemplo introduzca la línea 1 C=100.

Inicie el programa monitor, elija Vd. nuevamente el sistema decimal y la modalidad de lectura y como dirección inicial indique el 368 y para la dirección final el 379. Aquí habremos expuesto la expresión del monitor, acompañada de breves comentarios:

Dirección	Valor	Comentario
368	12	Lowbyte de la long. de línea
369	0	Highbyte de la long. de línea
370	1	Lowbyte del núm. de línea
371	0	Highbyte del núm de línea
372	13	Indica variable de número
373	5	Long. del nombre de variable+4
374	0	Cero de separación
375	227	Valor ASCII nom. de vari. +128
376	239	Valor Token para "="
377	25	Tamaño de la variable
378	100	Valor de la variable
379	0	Cero de separación

La explicación de los valores de las primeras cuatro direcciones podrá encontrarla en el capítulo 5.1. Ahora pasemos al valor 13 de la dirección 372. Normalmente está el Token para la instrucción. Aquí en este caso no es así; aquí se indica que la variable C es una variable numérica. Si hubiésemos utilizado una variable en cadena aparecería en este lugar un "3".

La longitud del nombre de la variable en la dirección 373 está codificada del siguiente modo:

Valor = longitud del nombre de la variable + 4. El cero siguiente está únicamente como separación y no tiene ningún otro significado; lo mismo equivale para el cero en el 379. Ahora pasemos al nombre de la variable. Este se descifrará de la siguiente manera: Al valor ASCII de la última letra del nombre de la variable se le sumará el número 128. Todas las demás letras del nombre de la variable se almacenarán con su valor ASCII. El valor 239 de la dirección 366 es el valor Token para el carácter de igualdad. Aquí no se tomará el valor ASCII para tener bien claro que el carácter "=" no pertenece al nombre de la variable. En este lugar finaliza el nombre de la variable. El próximo valor (25) es el valor codificado para el tamaño de la variable. Este indica al ordenador que el valor de la variable cabe en un Byte, o sea, que no es mayor que 255 y que tampoco tiene posiciones detrás de la coma. El valor de la variable es una cifra íntegra (número entero). Los distintos valores para el tamaño de la variable han sido recopilados en una tabla que expondremos más adelante. Y ahora pasemos al "100" de la dirección 378: Esto es el valor de la variable, ya que habíamos fijado C=100. Ahora llegamos al segundo ejemplo y por lo tanto deberán entrar la siguiente línea:

```
1 C = 1000
```

Como dirección de inicio de lectura deberán entrar el 372 y como dirección final el 380. De esta manera empezaremos a

leer después del número de la línea, ya que ésta se ha mantenido igual. A continuación exponemos nuevamente la lista con breves comentarios:

Dirección	Valor	Comentario
372	13	Indica, que hay una variable numérica
373	5	Longitud del nom. variable + 4
374	0	Cero de separación
375	227	Valor ASCII del nombre + 128
376	239	Token para "="
377	26	Tamaño de la variable
378	232	Lowbyte del val. de la variable
379	3	Highbyte del val. de la varia.
380	0	Cero de separación

Como podrán apreciar, hasta la dirección 376 no se ha modificado nada pero en la dirección 377 encontraremos ahora un 26. Este 26 indica que el valor de la variable es mayor que 255 y menor que 65535. En todo caso, deberá ser siempre una cifra íntegra. El Highbyte y el Lowbyte del valor de la variable dan por resultado según el sistema de cálculo conocido la cifra 1000, siendo el último cero de separación también conocido.

Ahora se plantea la cuestión de cómo el CPC archivará las variables que tengan un valor superior al 65535 o bien cómo archivará asimismo números con posiciones después de las comas. Para ambos se efectuará del mismo modo.

Introduzcan ahora la siguiente línea:

```
1 C = 100000
```

Como dirección inicial deberán entrar el 372 y como dirección final el 383.

El listado terndrá el siguiente aspecto:

Dirección	Valor	Comentario
372	13	Indica, que hay una variable numérica
373	5	Longitud del nom. de la variab.
374	0	Cero de separación
375	227	Valor ASCII del nombre de la variable +128
376	239	Token para "="
377	31	Tamaño de la variable
378	0	VAR 1
379	0	VAR 2
380	80	VAR 3
381	67	VAR 4
382	145	VAR 5
383	0	Cero de separación

Hasta la dirección 376 todo es conocido. El 31 de la dirección 377 indica que el valor de la variable es, o bien mayor que 65535, o que no es una cifra íntegra. Los próximos 5 valores están compuestos de un modo correcto, siendo el valor de la variable. La fórmula para calcular el valor de la variable es muy compleja y no poco fácil de entender. No tema si Vd. no la comprende.

```
Valor = (2 ^ (VAR 5 - 145)) * (65536 + (VAR 2 - 128) +
        (VAR 3 * 2) + (VAR 4 *512) + (VAR 1 - 32800))
```

C A P I T U L O 6 RUTINAS DE UTILIDAD

6.1 EL JOYSTICK COMO RATON

Ya hemos comentado el punto de la ergonomía, es decir, el amable servidor de un programa.

Para poder crear un programa agradable para el usuario se han desarrollado últimamente muchos conceptos y ordenadores.

Un factor importante en todos estos métodos es el llamado "ratón". Este es una pequeña caja con una bola rodante y una tecla. Vd. podrá mover esta caja sobre la bola rodante de una base. Análogamente a ésto se mueve sobre la pantalla del ordenador conexionado un símbolo que generalmente es una pequeña flecha. Mediante esta flecha podrá Vd. indicar campos determinados sobre la pantalla en los que se han especificado determinadas funciones. Si presiona Vd. el botón sobre el ratón, el ordenador ejecutará la función correspondiente. Así pues, para estas "manipulaciones" no necesitará Vd. el teclado. Después de una cierta práctica podrá Vd. moverse mucho más rápidamente con el ratón que con el teclado. A no ser que Vd. hubiese aprendido a escribir con los diez dedos.

Para el CPC todavía no se tiene a disposición el ratón al caso. Sin embargo, todo lo dicho puede conseguirse también con el Joystick. Este ya fué introducido en nuestro programa de caracteres. Por este motivo deberá saber ya lo positivo que puede ser este aparato adicional de dirección.

Para poderle aclarar mejor el proceso de trabajo le hemos escrito una pequeña rutina que puede emitir algunos puntos de menú (en nuestro ejemplo un procesamiento de textos). Después, y con un Joystick debidamente conexionado (Amstrad o el compatible Atari), podrá mover un pequeño punto sobre la pantalla. Posicione Vd. este pequeño punto sobre el carácter

("O") detrás del punto del menú que Vd. elegirá y active la tecla de "fuego". Después el programa se ramifica en la rutina correspondiente.

A continuación le facilitaremos la rutina y después la aclaración así como un par de posibilidades de mejora, para que Vd. pueda aplicarlas en sus programas.

¡ Al copiar este programa tenga presente de entrar el número correcto de espacios libres en las líneas 30, 50, 70, 90, 110, 130, 150 y 170 !

```
10 CLS : MODE 2
20 LOCATE 1,1
30 PRINT "Crear texto       o"
40 LOCATE 40,1
50 PRINT "Ver texto         o"
60 LOCATE 1,8
70 PRINT "Modificar texto   o"
80 LOCATE 40,8
90 PRINT "Borrar texto      o"
100 LOCATE 1,17
110 PRINT "Almacenar texto  o"
120 LOCATE 40,17
130 PRINT "Cargar texto     o"
140 LOCATE 1,25
150 PRINT "Imprimir texto   o"
160 LOCATE 40,25
170 PRINT "Finalizar prog.  o"
180 a=JOY(0)
190 PLOT x*8,400-y*16,0
200 IF a=1 THEN y=y-1
210 IF y<1 THEN y=25
220 IF a=2 THEN y=y+1
230 IF y>25 THEN y=1
240 IF a=4 THEN x=x-1
250 IF x<1 THEN x=80
```

```
260 IF a=8 THEN x=x+1
270 IF x>80 THEN x=1
280 PLOT x*8,400-y*16,1
290 IF a=16 THEN GOTO 300 ELSE GOTO 180
300 IF x=18 AND y=1 THEN GOTO 400
310 IF x=58 AND y=1 THEN GOTO 440
320 IF x=18 AND y=8 THEN GOTO 480
330 IF x=58 AND y=8 THEN GOTO 520
340 IF x=18 AND y=17 THEN GOTO 560
350 IF x=58 AND y=17 THEN GOTO 600
360 IF x=18 AND y=25 THEN GOTO 640
370 IF x=58 AND y=25 THEN GOTO 680
380 GOTO 180
390 END
400 CLS
410 PRINT "Crear texto"
420 INPUT a$: IF a$<>"w" GOTO 420
430 GOTO 10
440 CLS
450 PRINT "Ver texto"
460 INPUT a$: IF a$<>"w" GOTO 460
470 GOTO 10
480 CLS
490 PRINT "Modificar texto"
500 INPUT a$: IF a$<>"w" GOTO 500
510 GOTO 10
```

Aclaración del programa:

10-170	Emisión del cuadro del programa con las partes individuales a ser elegidas.
180	Un valor del Joystickport será leido en la variable.
190	Se borrará el pequeño cursor.
200-270	Esta es la rutina para modificar el pequeño

cursor según el movimiento del Joystick. Esto se efectuará al mover correspondientemente hacia arriba o hacia abajo las variables para ambas dimensiones de la pantalla (dirección x/y). Seguidamente deberá incorporarse la cuestión sobre si el valor está fuera de la pantalla para que más tarde no aparezca un IMPROPER ARGUMENT.

280 Aplicación del pequeño cursor de gráficos.

290 Si se hubiese presionado el botón de "fuego", el programa saltaría a la parte correspondiente; de otro modo se cuestionarría de nuevo el Joystickport.

300-370 Aquí saltará el programa si se ha presionado el botón de "fuego". Línea por línea se controlará en qué posición estaba el cursor en el momento de presionar el botón de "fuego".

380 Salto de retroceso para cuestionar el Port en el caso de que el botón de fuego se hubiese presionado en un lugar en el que no hubiese ningún "0".

400 A partir de aquí podrá incorporar las rutinas de su programa. Aquí hemos escrito únicamente un par de pequeños ejemplos. Tenga aquí presente que los números de las líneas concuerden exactamente con éstas.

Después de que hayan iniciado el programa podrán mover mediante el Joystick un pequeño cursor. En este programa sólo le será posible elegir los tres primeros puntos, ya que para los otros no se ha encuadrado ningún tipo de rutina. El programa deberá únicamente resaltar el principio del mismo.

Una agradable misión será mejorar la rutina en dos posiciones. En primer lugar el cursor debería ser mayor. Esto podrá conseguirlo si trabaja con la instrucción LOCATE en lugar de hacerlo con la instrucción PLOT. La segunda mejora

debería consistir en configurar el punto en donde se presiona el botón de "fuego" algo mayor. De este modo sería más fácil de localizar. Para ello deberá destinar un mayor espacio en las líneas 300-370 que aquél que hemos reservado. Esto podrá realizarlo de manera más adecuada mediante combinaciones lógicas.

6.2 CONTROL DEL MOTOR DE CASSETTES

Aquí quisiéramos facilitarle todavía dos direcciones con las que podrá dirigir el motor del Cassette del programa.

Estas son: Arranque del motor del Cassette: &BC6E
Paro del motor del Cassette: &BC71

Si desea Vd. que el motor empiece a funcionar, entonces deberá entrar únicamente CALL &BC6E.

6.3 PROTECTOR DE COPIAS SIMPLIFICADO

Un nuevo punto que queremos tratar a continuación es la protección de programas contra el copiado no permitido.

Lamentablemente, ésto se ha convertido en una mala costumbre muy extendida que perjudica a los autores de los programas con respecto a sus ganancias (de las que, en definitiva, tienen que vivir). Además, esto no es un quebranto de derechos de importancia (no es un delito de caballeros como el aparcar indebidamente).

Lamentablemente los llamados copiadores "piratas" no dejan asustarse por esta realidad. Por este motivo se desarrollan los mecanismos de protección de copias, que deberán evitar la reproducción no permitida de programas.

¡ Su CPC ya tiene incorporada dicha protección de copias ! Esta se esconde sencillamente bajo la instrucción SAVE.

¿ Cómo deberá tratarse esta protección de copias ? Si almacena un programa que deberá ser protegido, deberá añadirle únicamente al nombre del programa una coma y una "P" (para protected = protegido).

Un programa que se haya almacenado de este modo deberá cargarse únicamente con "RUN". Luego se iniciará automáticamente. Podrá interrumpir el programa, pero entonces "desaparecerá". No es posible listar este programa, ni siquiera almacenarlo o algo similar.

6.4 ALMACENAMIENTO DE UN CAMPO DE MEMORIA

Una nueva ayuda útil está asímismo algo apartada del manual. Esta ayuda es una instrucción para poder almacenar determinados campos de memoria. Esta instrucción podrá utilizarla, por ejemplo, en un ensamblador para archivar determinados campos sobre Cassette (esto ya ha sido realozado con nuestro monitor).

Una nueva posibilidad extraordinaria de esta instrucción es poder almacenar toda la pantalla (o partes de la misma). Esta propiedad ha sido aprovechada en el editor de gráficos. Todos estos campos podrán volverse a cargar y así dejar aparecer, relativamente de prisa, imágenes fantásticas sobre su monitor sin tener que trabajar con las instrucciones lentas de los gráficos del CPC. Para la pantalla observe detenidamente el campo de memoria en el que ésta se encuentra almacenada. Ambas direcciones ya han sido comunicadas en una de las partes de los capítulos anteriores.

6.5 OCUPACION UTIL DEL TECLADO

Su CPC tiene la posibilidad de modificar con dos sencillas instrucciones la ocupación del teclado. Estas son KEY y KEY DEF. A continuación le expondremos un programa que modifica la ocupación del teclado. Por lo demás, el programa se explica por si sólo.

```
10 MODE 2
20 LOCATE 22,5:PRINT "K E Y     M A N A G E R "
30 LOCATE 22,6:PRINT "======================="
40 PRINT :PRINT :PRINT :PRINT :PRINT
50 PRINT  "El siguiente programa ocupa el bloque de decima-
 les de su teclado  con  algunas  instrucciones auxiliares,
 así como  algunas teclas  con caracteres especiales alema-
 nes"
60 PRINT :PRINT "Pulsar una tecla por favor"
70 CALL &BB18
80 CLS
90 PRINT :PRINT :PRINT
100 KEY 137,d$
120 PRINT "Tecla 0    : LIST"
130 PRINT "Tecla 1    : RUN"
140 PRINT "Tecla 2    : LOAD"
150 PRINT "Tecla 3    : MODE"
160 PRINT "Tecla 4    : SAVE"
170 PRINT "Tecla 5    : EDIT"
180 PRINT :PRINT :PRINT
190 PRINT "Además  tiene  la  posibilidad  de  ocupar cinco
 teclas más  con una  cadencia de  señales de 15 caracteres
 cada
 una"
200 INPUT "Si desea Vd. tenerlas, introduzca 'd'";d$
210 IF d$="d" THEN GOTO 280
220 CLS
230 PRINT "Además este programa pone  a su  disposición los
 caracteres especiales  alemanes e intercambia las teclas y
 e  z."
240 PRINT:PRINT:PRINT:PRINT
250 PRINT "Pulsar una tecla por favor"
```

```
260 CALL &BB18
270 GOTO 350
280 CLS
290 INPUT "Ocupación de la tecla 6";a$
300 INPUT "Ocupación de la tecla 7";b$
310 INPUT "Ocupación de la tecla 8";c$
320 INPUT "Ocupación de la tecla 9";d$
330 INPUT "Ocupación de la tecla '.'";e$
340 GOTO 220
350 KEY 128,"list"+CHR$(13)
360 KEY 129,"run"+CHR$(13)
370 KEY 130,"load"+CHR$(13)
380 KEY 131,"mode"
390 KEY 132,"save"+CHR$(13)
400 KEY 133,"edit"
410 KEY 134,a$
420 KEY 135,b$
430 KEY 136,c$
440 KEY 137,d$
450 KEY 138,e$
460 SYMBOL AFTER 32
470 SYMBOL 59,0,20,0,31,34,66,66,127
480 SYMBOL 58,0,36,0,60,66,66,66,60
490 SYMBOL 60,0,0,36,0,66,66,66,60
500 SYMBOL 62,129,60,66,66,126,66,66,66
510 SYMBOL 64,129,60,66,66,66,66,66,66
520 SYMBOL 36,129,0,66,66,66,66,66,60
530 SYMBOL 39,60,68,68,120,68,124,64,64
540 SYMBOL 122,0,36,36,36,28,4,56
550 SYMBOL 90,0,65,34,20,8,8,8,8
560 SYMBOL 121,0,0,15,1,1,2,4,15
570 SYMBOL 89,64,2,4,8,8,16,16,31
```

6.6 DIRECCIONES DE SALTO

En las páginas siguientes encontrará algunas direcciones útiles que Vd. podrá insertar en sus programas. Para poder utilizar estas rutinas del sistema, no es necesario que Vd. sea un progrmador de lenguaje máquina. Aquí es suficiente, si Vd. se lee por completo el capítulo 4 con la introducción en el lenguaje máquina. Para el lector que se interese, se describe brevemente la función y los registros de entrada y salida requeridos. Es evidente, que se indicarán también determinadas particularidades de las rutinas (mientras éstas sean conocidas). A continuación se especificará de nuevo el cargador, de Basic mediante el cual pueden llamarse todas las funciones del Basic.

```
10 REM Cargador Basic
20 MEMORY &2FFF : d = 0
30 INPUT "Valor ";v
40 IF v = 0 THEN GOTO 80
50 POKE &3000+d,v
60 d = d + 1
70 GOTO 30
80 CALL &3000
```

El cargador de Basic mencionado anteriormente ha sido ligeramente modificado comparado con aquél que se menciona en el capítulo 4.3. Aquí se trata de un "bucle sinfín" con un criterio de interrupción. En la línea 40 se examinará si el valor entrado es igual a 0. Si es éste el caso, entonces se llamará de inmediato al subprograma creado con el cargador. Si alguna vez se le presenta el caso de tener que entrar un 0 como valor de dato, entonces modifique el valor de la línea 40 y se originrá la interrupción de la entrada de valores.

Si desea intercalar algunas funciones en un programa, se aconseja archivar los valores del cargador del Basic en las líneas DATA con las que se configurará el subprograma correspondiente. Los valores de las variables pueden ser trasladados mediante una instrucción Poke en el espacio adecuado de la memoria.

6.6.1 COLOREADO DEL CAMPO

Dirección de salto: BC44 (hexadecimal)
48196 (decimal)

Con esta rutina pueden colorearse campos de la pantalla. El tamaño depende de la modalidad operativa elegida y de los puntos limítrofes de libre elección.

Registros de transmisión:

a	Valor del color codificado (véase también el capítulo 2.10 sobre la memoria de la pantalla)
h	Columna izquierda del campo a ser rellenado
d	Columna derecha del campo
l	Línea superior del campo
e	Línea inferior del campo

Tengan presente de no salirse de los límites del campo de la pantalla, ya que de otro modo podría "estrellarse" su CPC. El ángulo superior izquierdo de la pantalla tiene el valor físico de 0,0.

Registro de admisión----

Valores hexadecimales para el cargador de Basic: 3E, (valor para el color del carácter/muestra Bit), 21, (arriba), (izquierda), 11, (abajo), (derecha), CD, 44, BC, C9, 0

Los valores para arriba, la derecha, etc. son libremente elegibles y corresponden a las líneas o cifras de las columnas.

Formato del ensamblador:

```
ld    a,N
ld    hl,NN
ld    de,NN
call  44,BC
ret
```

6.6.2 CAMBIAR EL BANCO DE LA PANTALLA

Dirección de salto: BC06 (hexadecimal)
48134 (decimal)

Con esta rutina puede ser trasladada la memoria de la pantalla a uno de los cuatro bancos disponibles (0-3). Aquí tendrá más sentido utilizar únicamente el banco 1 o bien el banco estándar 3, ya que en las otras dos posiciones podrían tenerse conflictos con el sistema operativo del CPC.

Registro de transmisión: a En el acumulador se entrará el Byte significativo (Higher-Byte) de la dirección inicial del nuevo campo de la pantalla.

Tengan presente de utilizar únicamente los valores 00, 40, 80 y C0, ya que de otro modo podrían tenerse repercusiones no deseadas.

Registro de admisión----

Valores hexadecimales para el cargador de Basic: 3E, (valor inicial nuevo), CD, 06, BC, C9, 0

Observen las indicaciones del nuevo valor inicial, que podrán encontrar en el registro de transmisión.

Formato del ensamblador:
```
ld   a,N
call 06,BC
ret
```

6.6.3 ATENDER A LA TECLA

Dirección de salto: BB06 (hexadecimal)
47878 (decimal)

El arranque de esta rutina ocasiona una interrupción del proceso del programa. Aquí se esperará hasta presionar cualquier tecla.

Registro de transmisión----

Registro de admisión: a Del acumulador podrá leerse el código de la tecla presionada.

Valores hexadecimales para el cargador de Basic ---

Esta función podrá llamarse en cualquier lugar del programa mediante CALL &BB06.

Formato del ensamblador: call 06,BB

6.6.4 SCROLLING

Dirección de salto: BC4D (hexadecimal)
48205 (decimal)

Al llamar esta rutina será posible desplazar la pantalla completa en una línea hacia arriba o hacia abajo. El valor del color (PAPER) de la nueva línea insertada podrá ser libremente determinado dentro de las posibilidades del MODE elegido.

Registro de transmisión: b Mediante el registro b se concretará la dirección de desplazamiento. Si el contenido tiene el valor 0 entonces se moverán pantallas hacia abajo. Cualquier otro valor ocasionará un desplazamiento hacia arriba.

a En el acumulador se indicará el valor del color codificado para la nueva línea que deberá aparecer.

Registro de admisión----

Valores hexadecimales para el cargador de Basic: (hacia arriba) 6, 1, 3M, (código del color de la nueva línea a ser intercalada), CD, 4D, BC, C9, 0

Valores hexadecimales para el cargador de Basic: (hacia abajo) 6, 0, 3E, (código del color de la nueva línea a ser intercalada), CD, 4D, BC, C9, 0

Formato del ensamblador:

```
ld   b,N
ld   a,N
call 4D,BC
ret
```

6.6.5 SCROLLING PARCIAL DE LA PANTALLA

Dirección de salto: BC50 (hexadecimal)
48208 (decimal)

Llamando a esta rutina estará Vd. en condiciones de poder desplazar campos parciales (ventanas). La dirección de desplazamiento y el código del color de las nuevas líneas corresponderán con los registros del Scrolling para la pantalla completa.

Registro de transmisión:
- a Valor del color codificado
- b Dirección de desplazamiento
- h Columna izquierda de la ventana a ser desplazada
- d Columna derecha de la ventana
- l Línea superior de la ventana
- e Línea inferior de la ventana

Tengan presente de no sobrepasar los límites de la pantalla, ya que de otro modo podríamos tener complicaciones. El ángulo izquierdo superior de la pantalla tiene el valor físico de 0,0.

Registro de admisión----

Valores hexadecimales para el cargador de Basic: 6, (dirección), 3E, (código del color), 21, (arriba), (izquierda), 11, (abajo), (derecho), CD, 50, BC, C9, 0

Los límites del campo elegidos no dependen de ninguna ventana previamente definida.

Formato del ensamblador:

```
ld   b,N
ld   a,N
ld   hl,NN
ld   de,NN
call 50,BC
ret
```

6.6.6 SELECCION DE MODALIDAD

Dirección de salto: BC0E (hexadecimal)
48142 (decimal)

Mediante esta rutina le será posible poder modificar la modalidad de la pantalla en el programa máquina.

Registro de transmisión: a El acumulador contiene el valor de la nueva modalidad elegida.

El valor del acumulador deberá estar únicamente entre 0 y 2.

Registro de admisión----

Valores hexadecimales para el cargador de Basic: 3E, (nueva modalidad), CD, E, BC, C9, 0

Formato del ensamblador:
```
ld   a,N
call 0E,BC
ret
```

6.6.7 INVERSION DE CARACTERES

Dirección de salto: BC4A (hexadecimal)
48202 (decimal)

Con esta rutina puede invertirse un carácter, que ha sido fijado mediante sus coordenadas físicas (0,0 ángulo superior izquierdo) sobre 2 máscaras de muestras Bit. La inversión puede conseguirse a través de un EXCLUSIVE OR doble lógico.

Registro de transmisión: b Máscara de inversión 1 (color 1)
c Máscara de inversión 2 (color 2)
h Columna
l Línea

Deberá tenerse en cuenta que la posición dependerá de la modalidad que se habrá elegido.

Registro de admisión----

Valores hexadecimales para el cargador de Basic: 6, (máscara 1), E, (máscara 2), 21, (columna), (línea), CD, 4A, BC, C9, 0

Ejemplo: Modalidad 1, color de los caracteres amarillo, color de fondo azul. El carácter de la posición a ser invertida sobre la pantalla es un "O". Observemos la inversión de las dos líneas superiores de las muestras Bit. Como máscara de inversión 1 elegiremos "1000 0000", la máscara 2 tendrá este aspecto: "0000 1111".

```
Situación de partida .........0111 0000  1100 0000
Después de la inversión 1 ....1111 0000  0100 0000
Después de la inversión 2 ....1111 1111  0100 1111
Color del punto de imagen ....RRRR RRRR  HHRR HHHH
                              R = rojo
                              H = azul claro
```

```
Formato del ensamblador: ld   b,N
                         ld   c,N
                         ld   hl,NN
                         call 4A,BC
                         ret
```

6.6.8 DESPLAZAMIENTO HORIZONTAL DE LA PANTALLA

Dirección de salto: BC05 (hexadecimal)
48233 (decimal)

Esta rutina le facilitará el desplazamiento horizontal de la pantalla (Scrolling derecha e izquierda). Con ello se unirá la columna que sale por la izquierda al lado derecho de la pantalla (wrap arround).

Registro de transmisión: h Higher Byte del valor de desplazamiento
l Lower Byte del valor de desplazamiento

Deberá tenerse en cuenta que el valor de desplazamiento deberá poderse dividir fundamentalmente por 2.

Registro de admisión----

Valores hexadecimales para el cargador de Basic: 21, (Lower Byte del valor de desplazamiento), (Higher Byte del valor de desplazamiento), CD, 5, BC, C9, 0

Formato del ensamblador:
```
ld   hl,NN
call 05 BC
ret
```

6.6.9 POSICIONAMIENTO DEL CURSOR

Dirección de salto: BB75 (hexadecimal)
47989 (decimal)

Esta rutina corresponde en su posición a la instrucción de Basic "LOCATE".

Registro de transmisión: h Nueva columna del cursor
l Línea del cursor

No se efectuará la verificación sobre si los valores fijados se encuentran dentro de la pantalla.

Registro de admisión----

Valores hexadecimales para el cargador de Basic: 21, (linea), (columna), CD, 75, BB, C9, 0

Formato del ensamblador:
```
ld   hl,NN
call 75 BB
ret
```

6.6.10 POSICIONAMIENTO DE COLUMNAS DEL CURSOR

Dirección de salto: BB6F (hexadecimal)
47983 (decimal)

Sobre esta rutina es posible el posicionamiento del cursor dentro de la línea en la que se encuentra.

Registro de transmisión: a Nuevo valor de columnas

No se efectuará el control plausible sobre el valor fijado de las columnas.

Registro de admisión----

Valores hexadecimales para el cargador de Basic: 3E, (columna), CD, 6F, BB, C9, 0

Formato del ensamblador:
```
ld   a,N
call 6F BB
ret
```

6.6.11 POSICIONAMIENTO DE LINEAS DEL CURSOR

Dirección de salto: BB72 (hexadecimal)
47986 (decimal)

Sobre esta rutina es posible el posicionamiento del cursor dentro de la columna en la que se encuentra.

Registro de transmisión: a Nuevo valor de líneas

No se efectuará el control de si la columna que se ha fijado de nuevo está realmente disponible.

Registro de admisión ---

Valores hexadecimales para el cargador de Basic: 3E, (nueva línea), CD, 72, BB, C9, 0

Formato del ensamblador:
```
ld   a,N
call 72,BB
ret
```

6.6.12 PREGUNTAS AL JOYSTICK

Dirección de salto: BB24 (hexadecimal)
47908 (decimal)

Aquí se trata de la rutina de máquina con la que se consulta a los conmutadores cerrados de ambos Joysticks conexionados entre si.

Registro de transmisión ---

Registro de admisión: a Después de la consulta se encontrará el código de los contactos cerrados en el acumulador.
h Tiene la misma ocupación que el acumulador.
l El estado del 2º Joystick será intercalado en el registro 1.

Un Bit fijado corresponde a un contacto cerrado. Las siguientes posiciones de conmutadores están destinadas a los distintos Bits:

Bit 0 Arriba
Bit 1 Abajo
Bit 2 Izquierda
Bit 3 Derecha
Bit 4 Fuego 2 (Botón de fuego estándar)
Bit 5 Fuego 1
Bit 6 No utilizado
Bit 7 No utilizado

Valores hexadecimales para el cargador de Basic: CD, 24, BB, 32, (Lower Byte fuego 2), (Higher Byte fuego 2), 7D, 32, (Lower Byte fuego 1), (Higher Byte fuego 1), C9, 0

Después de haber llamado a esta rutina podrá consultar el estado de ambos Joysticks mediante dos instrucciones PEEK.

Formato del ensamblador:

```
call 24,BB
ld   (NN),a
ld   a,l
ld   (NN),a
ret
```

6.6.13 CONEXION Y DESCONEXION DEL INVERS

Dirección de salto: BB9C (hexadecimal)
48028 (decimal)

A través de esta rutina se intercambiarán entre sí los colores actuales del PAPER y del PEN. Con la primera llamada se conectará el "Invers" y con la próxima volverá a desconectarse. En Basic podrá conseguirse esta función con la expresión: Print Chr$(24).

Registro de transmisión ---

Registro de admisión ---

Valores hexadecimales para el cargador de Basic ---

Una consulta será posible a través de CALL &BB9C.

Formato del ensamblador: call 9C,BB

6.7 RANDOMIZE - SOBRE LA PISTA DEL AZAR

Su CPC dispone de dos instrucciones con las que Vd. podrá dirigir el llamado azar. Esto es, en principio, la función RND. Con dicha función se dará el próximo valor de una serie de números al azar. Pruebe Vd. el siguiente ejemplo y podrá comprobar lo que el azar lleva consigo. En primer lugar deberá hacer retroceder su ordenador mediante CTRL + Shift + ESC, para que así tengamos las mismas predisposiciones.

```
10 REM Azar?
20 FOR t = 1 to 5
30 PRINT RND
40 NEXT t
```

Compare sus cifras con las que encontrará a continuación. Podrá constatar que no existe ninguna diferencia.

```
0.271940658
0.528612386
0.021330127
0.175138616
0.657773343
```

Esto es debido a que la secuencia de números al azar retrocede nuevamente con cada inicialización del ordenador. Imagínese que escribe un juego o lo compra del que, después de breve tiempo, ya puede prever las acciones que su contrario realizará. Este juego se volvería rápidamente muy aburrido.
Por esto, existe otra instrucción que se denomina RANDOMIZE xx. Para xx deberá entrar un valor de números. La secuencia de números al azar empieza, entonces, con un valor que dependerá del valor de inicio indicado al RANDOMIZE. A

continuación, una nueva prueba. Ponga su CPC en su estado de conexión. Seguidamente entre el programa "azar?". Aquí también tendremos en cuenta la función RANDOMIZE. Por este motivo deberá añadir una nueva línea en su programa:

```
15 RANDOMIZE 33
```

Inicie con "RUN" y anótese las cifras al azar conseguidas de este modo. Aquí habremos conseguido, en cierto modo, un éxito parcial, ya que la nueva secuencia de números al azar no tiene ningún parecido con la que aquí imprimiremos.
¿ Habremos eliminado así el problema de llamar siempre los mismos números después de un RESET ? ¡ No !

Si deseamos crear un auténtico número al azar deberemos conectar un factor que no podamos ver. Para ello se ofrece aquí el indicador de tiempo de su ordenador. Aparte de los cuatro indicadores de tiempo con los que Vd. activa los Interrupts, tiene además un reloj a tiempo real a su disposición. En este reloj se indicará el tiempo que ha transcurrido desde que ha conectado el aparato. Y mejor todavía, dicho tiempo transcurrido se medirá en 3/100 de segundo. Aquí pondremos mediante la función RANDOMIZE el valor de inicio RND sobre una cifra que vendrá determinada por el tiempo transcurrido. El programa muestra que indicaremos a continuación deberá ser probado varias veces. Vd. podrá constatar que prácticamente no existen repeticiones. Si alguna vez obtuviese una línea idéntica, entonces podría realmente hablarse de una casualidad. A continuación exponemos la función base para el auténtico azar:

```
10 REM Azar
20 RANDOMIZE TIME
30 FOR t = 1 to 5
40 PRINT RND
50 NEXT t
```

La pieza clave es la línea 20 donde la función RANDOMIZE

se fijará con el tiempo. La variable TIME está reservada en su ordenador para esta función.

RND suministrará siempre un número en el campo mayor que 0 y menor que 1. Si desea crear números enteros en campos previamente indicados deberá modificarse este valor base. Supongamos como ejemplo que desea crear números al azar en el campo entre el 2 y el 12. El valor 12 será el límite superior del campo y el 2 el valor inferior del límite. El valor superior del límite menos el inferior más 1 dará por resultado el número de los posibles valores que se puedan obtenerse. En nuestro caso sería: 12 - 2 + 1 = 11. El número al azar suministrado por RND se multiplicará por el factor de cantidad conseguido de esta manera. Teniendo en cuenta que RND no facilitará nunca el valor 1 deberemos buscar nuestros números al azar modificados entre el 0 y el 10. El último paso hará que el campo sea empujado hacia el valor de inicio más pequeño. Aquí se sumará el límite del campo inferior a la cifra obtenida. A continuación exponemos la fórmula:

x = INT (RND * (límite superior - límite inferior + 1) + límite inferior

Mediante la función INT se extraerá finalmente la parte numérica completa de las cifras al azar obtenidas de este modo.

6.8 EL CPC COMO MAQUINA DE CALCULAR

En los principios de la creación de ordenadores la única meta de los constructores era obtener un ordenador que pudiese solucionar cálculos difíciles mucho más rápido que el hombre. Esta propiedad la tiene hoy en día todo ordenador a pesar de que especialmente en los ordenadores personales pase a ser algo secundario. Para todos aquellos lectores entre Vds que utilizan principalmente el CPC para cálculos deseamos facilitarles algunas sugerencias para que puedan realizar dichos cálculos más fácilmente.

En primer lugar simplificaremos la utilización del CPC como "calculadora de bolsillo" mediante un pequeño programa. Como ya habrán podido comprobar, es bastante molesto tener que trabajar con la tecla SHIFT cuando se realizan operaciones sencillas, como en el caso de la adición. Por este motivo nuestro programa modificará la disposición de los caracteres de cálculo sobre el teclado para evitar tener que presionar ya sea el SHIFT o bien varias otras teclas de una vez. Esto se obtendrá mediante la función KEY DEF.

Si a Vd. no le importa trabajar con la línea superior de números en lugar de hacerlo con el bloque de cifras, entonces podrá copiar tranquilamente la subrutina a partir del 200, ya que ésta genera en el bloque de cifras que están conexionadas fuertemente con las funciones (SIN(X); COS(X)) de una calculadora de bolsillo normal.

La utilización del programa es muy sencilla. Después de iniciarlo su CPC contestará con un aviso de "CAMBIOS REALIZADOS". Todas las teclas con caracteres matemáticos habrán sido consecuentemente modificadas para su fácil utilización. Además la tecla para la "P" habrá sido ocupada con el "?" con el fin de evitar la entrada de la instrucción PRINT, respectivamente el presionado de la tecla para la división en relación con el SHIFT.

La tabla siguiente demuestra detalladamente los cambios que han sido realizados.

Caracteres matemáticos	Tecla	Valor de la tecla
*	:	29
+	;	28
(		17
)		19
&		26
SIN(	0	128
COS(	1	129
TAN(	2	130
ATN(	3	131
ABS(	4	132
INT(	5	133
HEX$(	6	134
BIN$(	7	135
STR$(	8	136
?	p	27

Una buena ayuda de servicio sería el hacerse pequeños adhesivos con los caracteres mencionados anteriormente y pegarlos sobre las teclas correspondientes para no tener que buscar continuamente la tecla deseada.

```
10 REM Calculadora
20 MODE 2
30 KEY DEF 29,1,42
40 KEY DEF 28,1,43
50 KEY DEF 17,1,40
60 KEY DEF 19,1,41
70 KEY DEF 26,1,38
80 KEY DEF 27,1,63
90 GOSUB 200
```

```
100 PRINT"Cambio realizado"
110 END
200 KEY 128,"sin("
210 KEY 129,"cos("
220 KEY 130,"tan("
230 KEY 131,"atn("
240 KEY 132,"abs("
250 KEY 133,"int("
260 KEY 134,"hex$("
270 KEY 135,"bin$("
280 KEY 136,"str$("
290 RETURN
```

Aclaraciones del listado:

10	Título
20	Ajustar la modalidad de caracteres 80
30-80	Cambiar las teclas a los caracteres matemáticos mediante la instrucción KEY DEF
90	Salto a la subrutina a partir de 200; si se deja la subrutina dicha línea se eliminará
100	Emisión del aviso de final de programa
110	Final de programa
200-280	Modificación del bloque de números en teclas de función mediante la instrucción KEY
290	Salto de retroceso a la parte principal

Si la selección de las funciones no le gusta pueden modificarse en las líneas 200 - 290. Asímismo, puede modificarse la ocupación de las teclas por caracteres matemáticos en las líneas 30 - 80; para ello deberá modificar los valores de las teclas según sus necesidades. Los valores de las teclas podrá encontrarlos en el apéndice del manual de su CPC.

6.8.1 EXACTITUD DE CALCULO

Bajo exactitud de cálculo de un ordenador se entiende el número de posiciones después de la coma que coincide con el valor estándar del cálculo. Asi por ejemplo un ordenador que tiene el PI como 3.1416 no posee una exactitud de cálculo elevada, ya que los valores acordados, como en el caso del PI o bien de la raiz de un número, no han sido definidos con la suficiente exactitud.

Su CPC "únicamente" podrá almacenar correctamente números hasta nueve posiciones y media detrás o bien delante de la coma, es decir que cuando se trata de números mayores la inexactitud será mayor que 1. Esto vamos a demostrarlo en un ejemplo:

Introduzca: PRINT 5E9+1-5E9

Seguramente Vd. esperará que como resultado aparezca un uno, pero no es así. La inexactitud del cálculo en este caso es de uno, ya que como resultado encontraremos un dos.

El número íntegro mayor que puede almacenarse correctamente es el 2^32-1, no expresado exponencialmente:

4294967295

Este hecho depende del almacenaje de las variables tal y como hemos comentado en el capítulo 5.

Con el fin de poder aumentar la exactitud de cálculo existe la posibilidad de poder localizar mediante fórmulas los valores requeridos. Sin embargo en dichas fórmulas no deberá aparecer ningún tipo de valor fijo, como raíces o el PI, ya que éstos dañan únicamente la exactitud a la que aludíamos.

Una de estas fórmulas ha sido modificada en un programa, de modo que Vd. tan sólo tendrá que copiarlo. Dicho programa podrá servirle de ejemplo para aplicar sus propias fórmulas en programas sin tener que utilizar la instrucción DEF FN. En algunos casos dicha instrucción no puede ser utilizada.

La fórmula calcula el EXP(X) del siguiente modo:

EXP(X)=1+X/1!+X^2/2!+X^3/3!....

En nuestro programa existe la posibilidad de poder calcular la línea superior hasta el factorial 31; después se habrá conseguido el límite de 1E34.

Ya que el CPC no posee la función del factorial, hemos simulado la misma en la subrutina a partir del 100.

El manejo del programa es muy sencillo; únicamente debe darse el valor X deseado y se obtendrá después de un corto tiempo de cálculo el EXP(X).

```
10 REM exp(x)
20 INPUT"x-Valor";x
30 FOR n=1 TO 31
40 x1=X^N
50 GOSUB 100
60 erg1=x1/fa
70 erg2=erg2+erg1
80 NEXT
90 PRINT erg2+1
95 END
100 fa=1
110 FOR m=1 TO n
120 fa=fa*m
130 NEXT m
140 RETURN
```

Aclaraciones del listado:

10 Título
20 Introducción del valor X
30 Apertura del bucle que determina el número de factoriales

```
40      Formación de las potencias X
50      Salto a la subrutina a partir del 100
60      Elaboración de las factoriales
70      Sumar los resultados
80      Cerrar el bucle
90      Emisión del resultado
95      Finalizar el programa
100     Retirar la factorial
110     Apertura del bucle para la factorial
120     Fórmula para calcular la factorial
130     Cerrar el bucle
140     Salto de retroceso a la parte principal
```

Una fórmula de este tipo como la del EXP(X) no puede calcularse con exactitud mediante la instrucción DEF FN, ya que las 31 partes no pueden ser introducidas en la instrucción DEF FN.

Un nuevo ejemplo para la aplicación de fórmulas es el cálculo de las posiciones cero. Las posiciones cero son los puntos en los que el gráfico de una función corta el eje X. Son puntos muy importantes para analizar los gráficos de una función que nunca podrán deducirse con exactitud de un diseño.

Como ejemplo hemos tomado el cálculo de las posiciones cero de una comparación en forma de cuadrado. Esta tiene la forma:

```
F(X)=X^2+P*X+Q
```

La fórmula para el cálculo de las posiciones cero es:

```
X1= -P/2+SQR(P^2/4*Q); X2= -P/2-SQR(P^2/4*Q)
```

Para el manejo debe decirse que en primer lugar deberá entrarse el valor para P y después el valor para Q. Después, el programa indicará el valor para las posiciones cero (X1,X2).

```
10 INPUT p
20 INPUT q
30 x1=-p/2+SQR(p*p/4-q)
40 x1=-p/2-SQR(p*p/4-q)
50 IF x1*x2<>q THEN 80
60 PRINT x1,x2
70 END
80 PRINT"Ninguna posición de ceros !"
90 END
```

Aclaraciones del listado:

```
10      Introducir P
20      Introducir Q
30      Calcular la primera posición de ceros
40      Calcular la segunda posición de ceros
50      Controlar las  posiciones cero  mediante el conjunto
        de Vieta
60      Emisión de las posiciones cero
70      Final del programa
80      Emisión por no tener posiciones de cero
```

Si el programa se interrumpe mediante un aviso de error, entonces no se tendrá ninguna posición de cero.

6.8.2 VELOCIDAD DE CALCULO

Un distintivo muy importante para poder juzgar un ordenador es su tiempo de cálculo. El usuario podrá así valorar adecuadamente las propiedades del ordenador.
La velocidad de cálculo se localiza mediante un programa de ensayo que haya funcionado ya en varios ordenadores. De este modo se consiguen valores comparativos entre los distintos ordenadores y podrá decidirse cuál será el adecuado para la aplicación deseada.

Existen varios tipos de trabajo que pueden plantearse a un ordenador con el fin de poder localizar la velocidad de cálculo. Todos los trabajos estándar se realizan bajo el concepto de test BENCHMARK. Un test de este tipo ha sido también utilizado en este capítulo. Se llama "Colador del Eratóstenes" y es un método muy utilizado para la determinación de diferencias de velocidad entre compiladores.

A la velocidad de cálculo pertenece también la elaboración de cuestionarios, bucles y administración de campos de datos. Todos estos trabajos son elaborados por el "Colador de Eratóstenes". Con el fin de efectuar comparaciones, hemos dejado calcular estos trabajos por dos ordenadores. Los resultados obtenidos fueron los siguientes:

Apple-II-Europlus con MBASIC 5.2 necesitó 391.8 seg.
VC-20 con tarjeta de 64K necesitó 394.8 seg.

Para que Vd. mismo pueda comprobar el tiempo de su CPC, le hemos especificado a continuación el listado al caso. Ejecute el programa, anote el tiempo y compare si su CPC puede aceptar dicha comparación.

```
10 REM Eratóstenes
20 DEFINT a-z
30 DIM f1(1000):i=1000:PRINT"Inicio"
40 FOR j=1 TO 10
50 co=0
60 FOR i=1 TO 1:f1(i)=1:NEXT i
70 FOR i=1 TO 1
80 IF f1(I)<>1 THEN 160
90 prim=i+i+3
100 k=i+prim
110 IF k>1 THEN 150
120 f1(k)=0
130 k=k+prim
140 GOTO 110
150 co=co+1
160 NEXT i
170 NEXT j
180 PRINT co;"Números primos"
190 END
```

Una aclaración detallada sobre el listado no deseamos darla, ya que sería muy compleja y difícil de exponer.

Después de este test podrán comprobar que su CPC puede ser comparado con otros ordenadores con respecto al campo de la velocidad.

Y para finalizar, un pequeño consejo: su CPC posee un gran número de caracteres matemáticos especiales. Sin embargo, éstos pueden únicamente ser conseguidos mediante el PRINT CHR$(X), es decir, que no son alcanzables directamente desde el teclado. Aquí no sería muy difícil crear un programa, de modo que dichos caracteres especiales pudiesen cumplir su propia función. Un ejemplo para ello sería el número PI.

6.9 MOVIMIENTOS DE LA PANTALLA

En algunos dialectos del Basic existe la instrucción SCROLL que permite desplazar la pantalla una línea hacia arriba o hacia abajo. Aquí hemos aprovechado una rutina del ROM del CPC para simular dicha instrucción.

La instrucción SCROLL puede ser utilizada, por ejemplo, en los juegos o bien al hacer listados de tablas.

La manera más normal de mover la pantalla es intercalando un carácter en el ángulo inferior derecho para que la pantalla se desplace hacia arriba. Sin embargo, falta aquí la posibilidad de poder mover la pantalla hacia abajo. Normalmente se requiere para ello una rutina de lenguaje máquina bastante larga. En el CPC existe la posibilidad de poder utilizar rutinas del ROM.

La rutina para el desplazamiento de la pantalla se encuentra en el ROM a partir de &BC4D (véase también capítulo 6.6 sobre las direcciones de salto). Dicha rutina mueve la pantalla en 8 puntos de imagen (un carácter). La dirección del desplazamiento depende del valor del registro B. Si el registro B es igual a cero el desplazamiento será hacia abajo. En todos los otros valores (que no sean igual a cero) se moverá la pantalla hacia arriba.

Existen dos posibilidades de contactar la rutina ROM. Por un lado podrá escribirse una rutina corta de máquina en la que se modifique el valor para el registro B mediante un POKE, o bien, podrán escribirse desde un buen principio dos rutinas, o sea, una para arriba y otra para abajo.

Nosotros nos hemos decidido por la última posibilidad, ya que este método es menos problemático y no requiere ningún tipo de modificación por su parte.

Lo que debemos tener claro es lo que la rutina de la máquina deberá conseguir.

En primer lugar deberá cargarse el registro B con un valor.

Para ello tenemos la instrucción LD B,N. Esta tiene el código 06. El valor que le seguirá depende del desplazamiento que la pantalla deberá realizar; para el desplazamiento hacia abajo será el cero y para el despazamiento hacia arriba el 255. En principio, será ésta la única diferencia de ambas rutinas en cuanto a las direcciones del despazamiento.

El siguiente paso será la llamada de la rutina ROM. Esto se realizará mediante la instrucción CALL NN. El código para ello es el 205. A éste deberá seguirle la dirección que será &BC4D. Dicha dirección deberá estar codificada en Highbyte y Lowbyte. Al código 205 le seguirá el valor 77 (Lowbyte 4D) y luego el valor 188 (Highbyte BC).

Por último deberá efectuarse un salto de retroceso hacia el Basic; ésto se realizará mediante un RET, cuyo código es el 201.

Ambas rutinas de máquina serán según sigue:

Para el desplazamiento hacia arriba:

```
LD B,FF                 06;255
CALL &BC4D              205;77;188
RET                     201
```

Para el desplazamiento hacia abajo:

```
LD B,0                  06;0
CALL &BC4D              205;77;188
RET                     201
```

Para crear las rutinas de máquina el programa Basic las almacena según el 43880 (hacia arriba) o bien según el 43886 (hacia abajo). Una vez terminado el programa bórrelo Vd. mediante DELETE-140 y después escriba su propio programa que será utilizado por las rutinas.

```
10 REM Hacia arriba
20 MEMORY 43879
30 DATA 6,255,205,77,188,201
40 FOR adr=43880 TO 43885
50 READ valor
60 POKE adr,valor
70 NEXT
80 REM Hacia abajo
90 DATA 6,0,205,77,188,201
100 FOR adr=43886 TO 43891
110 READ valor
120 POKE adr,valor
130 NEXT adr
140 END
```

Aclaración del listado:

10	Título
20	Limitación de la memoria Basic
30	Datos para la rutina de desplazamiento hacia arriba
40	Apertura de un bucle para la introducción de la rutina del lenguaje máquina
50	Lectura de los valores
60	Introducción de los valores en las posiciones de la memoria
70	Cerrar el bucle
80	Título
90	Datos para la rutina de desplazamiento hacia abajo
100-130	Igual que 40-70
140	Final del programa

Para utilizar las rutinas deberá mencionarse que éstas únicamente podrán ser contactadas mediante CALL del Basic (CALL 43880 hacia arriba y CALL 43886 hacia abajo).

Un pequeño programa podrá demostrarle las posibilidades del SCROLL. Escribiremos un texto en la pantalla y con ambas rutinas lo desplazaremos cinco líneas hacia arriba y cinco líneas hacia abajo respectivamente.

La velocidad del desplazamiemto podrá determinarla mediante una presión de teclas. Si activa una tecla se moverá el texto; de lo contrario, el programa se esperará hasta que vuelva a presionar otra tecla.

```
10 REM SCROLL Demo
20 MODE 1
30 LOCATE 15,12:PRINT"SCROLLDEMO"
35 REM HACIA ARRIBA
40 FOR n=1 TO 5
50 IF INKEY$="" THEN 50
60 CALL 43880
70 NEXT n
75 REM Hacia arriba
80 FOR n=1 TO 5
90 IF INKEY$="" THEN 90
100 CALL 43886
110 NEXT n
120 END
```

Aclaración del listado:

```
10        Título
20        Inserción de la modalidad de la pantalla
30        Emisión del texto
35        Título
40        Apertura del bucle para el número de desplazamien-
          tos
50        Preguntas al teclado
60        Llamar a la rutina SCROLL (hacia arriba)
70        Cerrar el bucle
75        Título
80        Apertura del bucle para el número de desplazamien-
          tos
90        Preguntas al teclado
100       LLamar a la rutina SCROLL (hacia abajo)
110       Cerrar el bucle
120       Final del programa
```

Tenga presente que el Demo no funcionará si no ha puesto el programa en marcha para la creación de las rutinas del lenguaje máquina.

Para simplificar la llamada de las rutinas podría, por ejemplo, definir dos variables y asignarles los valores para la llamada al caso. Las variables podrían ser:

HACIA ARRIBA = 43880 HACIA ABAJO = 43886

De este modo podrá trabajar más cómodamente con dichas rutinas.

Al utilizar las rutinas deberá tenerse en cuenta que el valor del color de la línea que se ha añadido arriba, respectivamente abajo, esté en el registro A. Partiendo de este hecho será posible crear otras funciones mediante la rutina SCROLL.

Hemos escrito un pequeño programa que trabaja con las rutinas SCROLL y que permite modificar la dirección, la longitud del desplazamiento y el color de la línea añadida mediante una entrada de datos.

Para poder crear estas funciones adicionales deberemos ampliar la rutina del lenguaje máquina en algunas instrucciones. La dirección del desplazamiento se almacena siempre en el registro B; sin embargo también utilizamos dicho registro como contador del número de líneas a ser añadidas. El registro A contiene el valor del color de las líneas añadidas.

Ya que el contenido de los registros se pierde al llamar la rutina ROM, deberemos poner los contenidos en el bloque para volverlos a buscar después de efectuada la llamada. Esto lo conseguiremos mediante las instrucciones PUSH (empujar) y POP (buscar).

La rutina de lenguaje máquina ampliada será según sigue:

Dirección	Instrucción	Código	Aclaración
43870	LD B,N	06	El número de líneas (N)
43871		N	se cargará después de B
43872	LD A,M	62	El valor del color (M)
43873		M	se cargará después de A
43874	PUSH BC	197	La B se pondrá en el bloque
43875	PUSH AF	245	La A se pondrá en el bloque
43876	LD B,R	06	La B se cargará con el
43877		R	valor de la dirección
43878	CALL &BC4D	205	Llamada de la rutina ROM
43879		77	Lowbyte de la dirección
43880		188	Highbyte de la dirección
43881	POP AF	241	La A se tomará del bloque
43882	POP BC	193	La B se buscará del bloque
43883	DJNZ,e	16	Seguirá más tarde
43884		245	
43885	RET	201	Salto de retroceso al Basic

Para poder dar la aclaración de la instrucción DJNZ,e deberemos seguir con algunas explicaciones. En primer lugar veamos el significado de las palabras: Disminuir, salta a "e", si no es cero.

El proceso del programa requiere un salto a la dirección 43874 para que el número de líneas no esté únicamente limitado a uno. El registro B sirve, como ya se ha indicado, para trabajos de cálculo. Para cada pasada del programa deberá reducirse en uno. Esto viene motivado por la reducción. Si después el registro B no es igual a cero, el programa saltará a la dirección 43874.

Aquí se cuestiona sin embargo cómo pasaremos de la dirección 43874 al valor 245.

Esto se explica por tener que retroceder el programa en nueve Bytes. A estos nueve Bytes se le añadirán dos Bytes, ya que el contador del programa apuntará ya a la siguiente instrucción. De este modo se obtendrá un número de 11. Si se desea efectuar un salto de retroceso deberá restarse de 256 la anchura del salto. En nuestro caso se obtiene el resultado de 245 y esto será el valor que deberá entrarse después del código de instrucción.

Si el registro B es igual a cero, el programa regresará al Basic.

```
10 REM SCROLLDEMO
20 MEMORY 43869
30 ADR=43869: SCROLL=43870
40 DATA 6,10,62,0,197,245,6,255,205,77
50 DATA 188,241,193,16,245,201
60 FOR CO=1 TO 16
70 READ VALOR
80 POKE ADR+CO,VALOR
90 NEXT CO
100 MODE 1
110 INPUT"Color";m
120 INPUT"Número de líneas";n
130 INPUT"Hacia arriba(1) o hacia abajo";r
140 IF r=1 THEN r=255 ELSE r=0
150 POKE 43873,m
160 POKE 43871,n
170 POKE 43877,r
180 CALL scroll
190 GOTO 100
```

Aclaración del listado:

10	Título
20	Limitación de la memoria Basic
30	Fijar la dirección de inicio del programa
40-50	Datos para la rutina de máquina
60	Apertura del bucle para poder leer los datos
70	Lectura de los códigos del lenguaje máquina
80	Inscripción de los valores en las posiciones de memoria
90	Cerrar el bucle
100	Ajustar la modalidad de la pantalla
110	Entrada del valor de los colores
120	Entrada del número de líneas
130	Decidir si se moverá hacia arriba o hacia abajo
140	Elaboración de la última entrada de datos
150	Inscripción de los valores del color
160	Inscripción del número de líneas
170	Inscripción de la dirección del desplazamiento
180	Llamar la rutina del lenguaje máquina
190	Salto de retroceso al 100

Para utilizar el programa deberá decirse que las entradas de datos deberán estar en los siguientes campos:

Valor del color = 0 a 255
Número de líneas = 1 a 25

Si desea Vd. incorporar la rutina de máquina en uno de sus programas, necesitará las siguientes direcciones:

43873	El valor del color deberá inscribirse en este espacio de memoria.
43871	El número de líneas deberá inscribirse en este espacio de memoria.
43877	La dirección de desplazamiento deberá inscribirse en este espacio de memoria.
43870	Dirección de inicio para la rutina del lenguaje máquina.

6.10 CLASIFICACION DE DATOS

Un trabajo que más tarde o temprano deberá realizar cualquier programador es la clasificación de datos. Aquí puede tratarse de datos numéricos o de textos, como por ejemplo, el conjunto de datos de un fichero de direcciones.

Como criterio del orden para dicha clasificación podrá tomarse, en el caso de cifras, su tamaño, los textos se clasificarán según el código ASCII utilizando la debida codificación. El código se elegirá de manera que, por ejemplo, la 'A' sea menor que 'B' y ésta a su vez menor que 'C'. Ante las letras mayúsculas se han destinado otros caracteres especiales tal como ocurre para los números; las letras minúsculas seguirán a las mayúsculas. Si escribimos una rutina de clasificación no deberemos preocuparnos, ya que el interpretador BASIC permite comparar directamente dos Strings teniendo en cuenta los criterios que acabamos de exponer.

Si deseamos clasificar datos podemos elegir el método dentro de un gran número de algoritmos. Estos algoritmos se distinguen por un lado por su complejidad y por el otro por su eficacia. Al elegir un método adecuado, el número de datos a ser clasificados juega un papel muy importante. Si Vd. tiene únicamente 10 ó 50 registros de datos a ser clasificados, entonces le bastarán los métodos más sencillos. Sin embargo, si tiene que clasificar varios centenares o incluso millares de registros de datos entonces los métodos simples ya no pueden tenerse en cuenta, ya que el tiempo de clasificación se prolongaría en varias horas. Para poderle facilitar una visión más amplia deseamos presentarle el método sencillo y el más rápido, o sea, el Bubble-Sort y el Quick-Sort.

En el Bubble-Sort se comparan dos elementos contiguos y si es necesario se intercambian. En el BASIC ésto puede programarse con dos bucles encajados.

```
100 FOR i=1 TO N: fl=0
110 FOR j=n TO i step -1
120  IF  a$(j-1)>a$(j) THEN h$=a$(j):a$(j)=a$(j-1):a$(j-1)=
                           h$:fl=1
130 NEXT j: IF fl=0 THEN RETURN
140 NEXT i: RETURN
```

El programa se llamará mediante el GOSUB 100 como subprograma. Los datos a ser clasificados deberán constar como Stringarray en a$. El tamaño del campo (el índice máximo) deberá entrarse en la variable "n", por ejemplo:

```
10 n=50: DIM a$(n)
20 'Leer los datos
30 GOSUB 100 'Clasificar datos
```

Un programa de clasificación tan sencillo es adecuado cuando únicamente deben clasificarse pocos datos o bien cuando éstos ya han sido ordenados previamente y por ejemplo se les debe incorporar un nuevo registro.

El programa de clasificación que se conoce bajo el nombre de Quick-Sort es el método adecuado para clasificar mayores cantidades de datos. Aquí se clasifica asimismo un campo String, cuyo límite de índice deberá constar bajo "n".

El programa genera como demostración un número de palabras al azar indicado por Vd., las muestra, las clasifica y las muestra de nuevo clasificadas correlativamente, indicando el tiempo invertido para ello.

```
10 DEFINT b-z
20 INPUT "Número de palabras ";n: DIM d$(n),o(20),u(20)
30 FOR i=1 TO n: b=5+10*RND
40 FOR j=1 TO b: d$(i)=d$(i)+CHR$(RND*25+65):NEXT: NEXT
50 GOSUB 80: a=TIME: GOSUB 100: a=(TIME-a)/300
60 PRINT: GOSUB 80: PRINT: PRINT a "Seg.": END
70 '
80 FOR i=1 TO n: PRINT d$(i): NEXT: RETURN
90 '
100 s=1: o(1)=1: u(1)=n
110 l=o(s): r=u(s): s=s-1
120 i=l: j=r: h$=d$((l+r)/2)
130 WHILE d$(i)<h$ AND i<r: i=i+1: WEND
140 WHILE d$(j)>h$ AND j>l: j=j-1: WEND
150 IF i<=j THEN d$(i):d$(i)=d$(j):d$(j)=d$:i=i+1:j=j-1
160 IF i<=j THEN 130
170 IF r-i<=j-1 THEN 200
180 IF l<j THEN s=s+1: o(s)=l: u(s)=j
190 l=i: GOTO 220
200 IF i<r THEN s=s+1: o(s)=i: u(s)=r
210 r=j
220 IF r>l THEN 120
230 IF s>0 THEN 110
240 RETURN
```

Si se inicia el programa con distintos tamaños de campos, obtendrá los siguientes resultados:

n	Seg.
50	3.8
100	7.9
200	18.6
500	110

El tiempo requerido para 500 registros es relativamente elevado y ésto tiene algo que ver con otro fenómeno. Aquí se trata del llamado Garbage Collection que entra en acción cuando se trabaja con muchos Strings. Si Vd. ha realizado el último ejemplo con los 500 Strings, entonces deberá entrar:

```
? FRE("")
```

Aquí deberá esperar aproximadamente 29 segundos. Este tiempo lo necesita el interpretador de BASIC para poder ordenar su memoria String. Cada vez que se modifica el contenido de una variable String, el interpretador escribirá el nuevo contenido en la memoria libre mientras que el antiguo contenido del String seguirá constando en la memoria. Esto funcionará hasta que toda la memoria haya sido completada con dichos Strings. La mayor parte de estos Strings ya no se necesitará, pudiéndose entonces ser borrada. Para ello se examinará toda la memoria y todos los Strings que ya no se necesiten, se borrarán y se solaparán con los Strings válidos, de modo que se habrá creado un nuevo espacio para la entrada de nuevos Strings. Cuantos más Strings válidos se tengan en la memoria, tanto más tiempo necesitará este método; en el caso de los 500 Strings fueron 29 segundos. Cuando se efectúe la clasificación, donde continuamente se copian Strings, y el espacio de memoria ya no sea suficiente, deberá intercalarse una Garbage Collection la cual aumenta todavía el tiempo de clasificación. A efectos de comparación indicamos los tiempos requeridos en el método del Bubble-Sort:

n	Seg.
50	12.1
100	49.0
200	238.2

6.11 TRANSMISION DE DATOS A OTROS ORDENADORES

Si Vd. desea transmitir datos desde su CPC 464 y 6128 a otros ordenadores, normalmente no es factible mediante cinta o Diskette ya que los formatos de registro no son compatibles. Sin efectuar un desvío a través de un sistema de almacenamiento masivo no sería posible realizarlo con la mayoría de los ordenadores. El ordenador al que le desean transferir los datos del CPC deberá tener para ello una posición de intersección paralela que se utilizará como entrada. Todos los ordenadores Commodore disponen, por ejemplo, de una de estas posiciones de intersección.

Su CPC tiene asimismo una posición de intersección paralela que únicamente puede utilizarse para la emisión de datos. Aquí se trata de la posición de intersección de la impresora a la que normalmente se conecta una impresora Centronic con posición de intersección. El protocolo de transmisión de una de estas posiciones de intersección es muy sencillo y puede realizarse sobre el ordenador de recepción mediante un pequeño programa. Para poder comunicar ambos ordenadores se requiere únicamente un cable que conecte la conexión de la impresora del CPC con el Userport de un Commodore. Ya que la impresora del CPC ya está asistida, la posición de intersección podrá ser conectada a través del canal B. Este método se utilizó por ejemplo para poder tomar los listados de programa impresos en este libro en el tratamiento de textos de un ordenador Commodore. Tal y como hemos dicho que la posición de intersección del CPC está asistida, nos limitaremos a darle a continuación como ejemplo un pequeño programa máquina para el Commodore 64. Aquí se tomarán los datos del CPC, los convertirá en códigos del Commodore y los escribirá en un fichero secuencial, por ejemplo sobre Cassette, Diskette o bien sobre pantalla. Si Vd. ha activado este programa en un C64 podrá transferir desde el

CPC con un

```
LIST #8
```

un programa al C64. Cualquier texto o carácter podrá ser transferido con la instrucción PRINT #8, por ejemplo:

```
PRINT #8, "Una noticia del CPC 464 y 6128"
```

El programa máquina del C64 se realizó de tal manera que la transmisión del CPC 464 y 6128 puede finalizarse mandando un CHR$(0).

Aquí deberá tenerse en cuenta otra especialidad del CPC 464 y 6128. Una posición de intersección del Centronic trabaja normalmente con 8 Bits de datos. El CPC 464 y 6128 únicamente pone a disposición 7 Bits en la posición de intersección; el Bit de datos superior está constantemente fijado sobre masa. Al efectuarse transmisiones de textos ASCII ésto no es de gran importancia, ya que el código ASCII es un código de 7 Bits. Otra cosa sería si Vd. desea hacer por ejemplo una Hardcopy de la pantalla de gráficos sobre la impresora. La mayoría de impresoras trabajan con un gráfico de 8 agujas, de modo que una Hardcopy de gráficos iría acompañada de muchos problemas. Ahora pasemos a la transmisión del ordenador. A continuación exponemos el programa máquina para el C64.

```
30:   DD00          CIA       =   $DD00
40:   DD00          PORTA     =   CIA      ;Port para 'Busy'
50:   DD01          PORTB     =   CIA+1    ;Port para Datos
60:   DD0D          CR        =   CIA+13   ;Registro de control
70:                 ;
80:   C000                    *=  $C000
90:   C000 AD 00 DD START     LDA PORTA
100:  C003 29 FB              AND #%11111011   ;Busy low
```

```
110:  C005 8D 00 DD        STA PORTA
120:  C008 A9 10           LDA #%10000
130:  C00A 2C 0D DD WAIT   BIT CR
140:  C00D F0 FB           BEQ WAIT      ; Esperar a Strobe
150:  C00F AD 00 DD        LDA PORTA
160:  C012 09 04           ORA #%100     ; Busy hi
170:  C014 8D 00 DD        STA PORTA
180:  C017 AD 01 DD        LDA PORTB     ; Leer datos
180:  C01A F0 2A           BEQ FIN       ; Cero,entonces fin
180:  C01C C9 0A           CMP #10       ; Ignorar Line Feed
180:  C01E F0 E0           BEQ START
181:  C020 C9 41           CMP #"A"
181:  C022 90 12           BCC END       ; Transformar en
182:  C024 C9 5B           CMP #"Z"+1
182:  C026 B0 04           BCS TEST      ; CBM-ASCII
183:  C028 09 80           ORA #$80
183:  C02A 30 0A           BMI END
184:  C02C C9 61    TEST   CMP #$61
184:  C02E 90 06           BCC END
185:  C030 C9 7B           CMP #$7B
185:  C032 B0 02           BCS END
186:  C034 29 5F           AND #$5F
189:  C036 48       END    PHA           ; Memorizar datos
189:  C037 A2 01           LDX #1
189:  C039 20 C9 FF        JSR $FFC9     ; Emisión canal 1
190:  C03C 68              PLA
190:  C03D 20 D2 FF        JSR $FFD2     ; Emitir carácter
195:  C040 20 CC FF        JSR $FFCC     ; Emis. en Default
200:  C043 4C 00 00        JMP START     ; y al inicio
210:  C046 60       FIN    RTS           ; finalizar
```

Del Commodore 64 se abrirá primero un canal de datos con el número lógico de uno, por ejemplo:

```
OPEN 1,1,1,"DATOS" para Cassette o

OPEN 1,8,2,"DATOS,S,W" para Diskette respectivamente

OPEN 1,3
```

en el caso que sólo tenga que emitirse sobre pantalla. Entonces podrá iniciarse el programa con SYS 49152. Así pues, Vd. podrá mandar, por ejemplo, un listado de programa del CPC al C64 mediante un LIST #8. La transmisión podrá finalizarla entrando un

```
PRINT #8, CHR$(0);
```

en el CPC 464 y 6128. El C64 volverá a incorporarse con un 'READY' pudiendo así cerrar el fichero con CLOSE 1 en el que se insertaron los datos. Para finalizar les indicaremos la ocupación Pin para el cable de conexión entre ambos ordenadores:

CPC 464 y 6128 Impresora Port Pin	Significado	C64 User-Port Pin
1	STROBE	B
2	D0	C
3	D1	D
4	D2	E
5	D3	F
6	D4	H
7	D5	J
8	D6	K
9	D7	L
11	BUSY	M
12	GND	N

6.12 IMPRESION DEL FONDO

¿ Cuántas veces ha estado sentado delante de su ordenador, nervioso, con los dedos sobre el teclado y esperando a que su impresora terminase ?

Es especialmente desagradable cuando se realiza el llamado proceso paquete, es decir, entrar datos, tratarlos técnicamente según el programa para después dejarlos imprimir.

En tales casos deberá esperar con la entrada de nuevos datos hasta que la impresora haya finalizado con su trabajo.

A continuación deseamos presentarle una pequeña rutina que le facilitará poder continuar con el programa sin tener que esperar a la emisión de datos.

La aplicación es realmente sencilla:
Añada a su programa las líneas 65000 hasta 65240. A continuación deberá asignar únicamente los datos a ser emitidos por las variables pr$ y realizar un GOSUB 65000.

El tampón de impresión se llenará con pr$ y su programa continuará de inmediato, a no ser que el tampón esté ya lleno.

Por este motivo podrá adaptar su tamaño a sus necesidades. La instrucción EVERY indica al ordenador de controlar cíclicamente si hay algo en el tampón. En caso afirmativo se emitirá una línea, etc.

Atención: El final de su programa deberá tener el mismo aspecto que las líneas 220 y 240 para asegurar que el

tampón haya sido completamente impreso.

Las líneas 100 hasta 200 sirven como demostración. Cuando se trate del 'proceso real' deberá modificar en la linea 65200 el 7 en 8.

A continuación exponemos el pequeño programa:

```
100 CLS:WINDOW #7,21,40,1,25
140 FOR i=0 TO 130
160 pr$=HEX$(i,6)
180 GOSUB 65000
200 NEXT
210 PRINT"Finalizar"
220 ON bfl GOTO 220
240 END
65000 REM fill buffer
65020 ON fl GOTO 65060
65040 DIM pr$(999):fl=1:pfc=1
65050 EVERY 30,0 GOSUB 65160
65060 IF pc=pfc THEN 65060
65080 DI:pr$(fc)=pr$:bfl=1
65100 fc=fc+1:IF fc>999 THEN fc=0
65120 pfc=fc+1:IF pfc>999 THEN pfc=0
65140 RETURN
65160 REM print buffer
65180 IF pc=fc THEN bfl=0:RETURN
65200 PRINT#7,pr$(pc)
65220 pc=pc+1:IF pc>999 THEN pc=0
65240 RETURN
```

C A P I T U L O 7 PROGRAMAS DE APLICACION

7.1 INTRODUCCION

Cuando se escribió este libro, todavía no se disponía, a nuestro entender, de ningún Software (racional) para el CPC 464 y 6128. Ya que un ordenador se "abre a la vida" mediante programas, les facilitaremos a continuación algunos de ellos.

Hemos dado especial importancia al DATA del CPC, ya que un programa de administración de ficheros es el programa más importante, más universal y más comercial de todos. Su nuevo ordenador, con el correspondiente monitor y los 80 caracteres, se adaptan de modo ideal para uno de estos programas.

Los siguientes programas están todos preparados para su correcto funcionamiento. Sin embargo, ya que se ha mencionado que sólo con la práctica se convierte uno en maestro, nos hemos propuesto trazar todos los registros técnicos del programa. Tanto el Basic como el buen manual invitan a uno a programar directamente. Por este motivo al comentar los programas damos siempre consejos para sus mejoras. Así pues podrá elaborar su programa, mejorarlo, hacerlo más rápido y más elegante, etc.

Para todo ello le deseamos mucha suerte.
EL PROBAR VA POR ENCIMA DEL ESTUDIAR.

7.2 PROGRAMA DE TRATAMIENTO DE FICHEROS

¿ Qué es un programa de tratamiento de ficheros ? Un tratamiento de ficheros es universal; en contraposición, una administradión de direcciones o de almacenamiento es especial. Estas se utilizan únicamente para campos determinados. Un tratamiento de ficheros puede ser adaptado.

Esto se consigue mediante el concepto de un tratamiento de ficheros. Dicho tratamiento consiste de REGISTROS DE DATOS y de CAMPOS. Para poder comentar ambos conceptos es mejor coger el ya conocido complejo del fichero. Un fichero consta de una cantidad determinada de fichas sobre las que consta ya un determinado y conocido número de inserciones. Todas las fichas forman nuestro fichero; la ficha corresponde a nuestro REGISTRO DE DATOS y una única inserción corresponde al conocido CAMPO.

En un tratamiento de ficheros puede determinarse libremente el número de registro de datos y de campos. A cada campo, y según su deseo, puede asignarle un nombre. Así pues, si desea crear un fichero de direcciones debería definir el primer campo como APELLIDO - NOMBRE, el segundo como CALLE, el tercero como LUGAR, etc. Así tendría, pues, preparado su tratamiento de ficheros como fichero de direcciones. Ahora comprenderá porque damos tanta importancia al tratamiento de ficheros; con algo de fantasía podría Vd. incluso utilizar el programa para el tratamiento de textos.

Ya que el siguiente programa es algo complejo y largo, lo hemos dividido en módulos (es decir en partes individuales y con sentido propio). Si Vd. copia el programa introduzca todas las partes directa y correlativamente.

P A R T E 1 - M E N U -

En el menú se le ofrecerán las funciones del programa que están a disposición. Conseguirá la parte del programa entrando el número que le corresponde y luego presionando la tecla ENTER; el programa se bifurcará a dicha parte. Si Vd. ha realizado una parte determinada, el programa retrocederá de nuevo al menú, de modo que Vd. podrá iniciar cualquier otro trabajo.

```
10 REM*****************************************************************
**********************
20 REM********************* C  O  P  Y  R  I  G  H  T ********
**********************
30 REM********************* C P C 464 Y 6128  T E A M ********
**********************
40 REM********************* D A T A   B E C K E R  ********
**********************
50 REM*********************        1 9 8 4           ********
**********************
60 REM*****************************************************************
**********************
70 REM    T R A T A M I E N T O   D E   F I C H E R O S
  V E R S I O N  1 . 0 (6.6.)
80 REM*****************************************************************
**********************
90 MODE 2
100 LOCATE 20,3: PRINT"C     P     C     D    A    T    A"
110 LOCATE 20,5: PRINT STRING$(&24,"=")
120 LOCATE 12,7: PRINT"COPYRIGHT   1 9 8 4   by   D A T A
B E C K E R"
130 LOCATE 15,10: PRINT"                  M  E  N  U  "
140 LOCATE 15,11: PRINT"                  ------------"
150 LOCATE 15,13: PRINT"Crear FICHERO            - 1 -"
160 LOCATE 15,14: PRINT"Entrar FICHERO           - 2 -"
170 LOCATE 15,15: PRINT"Cuidar FICHERO           - 3 -"
180 LOCATE 15,16: PRINT"Almacenar FICHERO        - 4 -"
190 LOCATE 15,17: PRINT"Cargar FICHERO           - 5 -"
200 LOCATE 15,18: PRINT"Buscar                   - 6 -"
```

```
210 LOCATE 15,19: PRINT"Imprimir FICHERO          - 7 -"
220 LOCATE 15,20: PRINT"Finalizar                 - 8 -"
230 LOCATE 15,21: PRINT"Borrar                    - 9 -"
240 PRINT: PRINT: INPUT"Su ELECCION por favor (1-9)   >ente
r<";a
250 ON a GOSUB 280,630,1010,1440,1570,1700,1900,2020,2070
260 GOTO 90
```

Aclaraciones sobre el programa:

10-80	Aquí se encuentra el Copyright así como una indicación sobre la versión del programa.
90	Aquí se pasa de una instrucción de Basic del CPC a la modalidad de caracteres 80.
100-230	En estas líneas se emitirá el cuadro del menú con la ayuda de la instrucción Locate.
240	Aquí se requiere que Vd. elija la parte del programa deseada.
250	Esta es una rutina corta para poder saltar a las distintas partes.
260	Si se efectúa un salto de retroceso de una parte del programa mediante el RETURN, esta línea es la que se ocupará del salto al principio del menú.

Variables utilizadas:

a	Contiene el número de la parte del programa deseada.

Utilización de una parte del programa:

La utilización del menú es muy sencilla. Vd. deberá entrar únicamente el número deseado de la parte del programa y presionar después la tecla ENTER. Todo lo demás lo realizará el programa para Vd.

P A R T E 2 - C R E A C I O N D E F I C H E R O S -

En esta parte de DATOS del CPC deberá definir el fichero. Aquí deberán efectuarse dos pasos. En primer lugar deberá indicar el tamaño que tendrá el fichero y en segundo lugar cómo se denominarán los campos del fichero al caso. Después de la definición tendrá Vd. todavía la posibilidad de realizar modificaciones en el caso de que Vd. se hubiese equivocado.

```
280 REM c r e a r   f i c h e r o
290 CLS
300 PRINT STRING$(&50,"-")
310 LOCATE 15,3: PRINT"C R E A R   F I C H E R O"
320 PRINT STRING$(&50,"-")
330 LOCAIE 1,8: INPUT"Cuántos registros de datos deberá
tener el fichero";registro de datos
340 PRINT: INPUT"Cuántos campos deberá tener un registro de
datos";campos
350 DIM denominaciónd$ (registro de datos)
360 DIM denominaciónf$ (campos)
370 DIM contenido$ (campos,registro de datos)
380 REM definición de los campos
390 PRINT"Definición de los campos de un registro de datos"
400 hregistro de datos=registro de datos
410 hcampos=campos
420 PRINT"---------------------------------------"
430 PRINT: PRINT"(Vd. tiene";campos;"definición de campos)"
440 PRINT: PRINT
450 FOR campos = 1 TO hcampos
460  PRINT"Denominación  del ";campos;"ten campo": INPUT"";
denominaciónf$(campos)
470 NEXT campos
480 PRINT: INPUT"Desea efectuar modificaciones";
modificaciones$
490 IF LEFT$(modificaciones$,1)="n" THEN GOTO 90
500 CLS
510 PRINT"Modificación de denominaciones de un campo"
520 PRINT"---------------------------------------"
530 FOR campos = 1 TO campos -1
```

```
540 PRINT"Denominación";campos;":",denominaciónf$(campos)
550 NEXT
560 PRINT: PRINT
570 INPUT"Qué campo desea modificar";campos
580 PRINT"Denominación del antiguo campo "denominaciónf$
(campos)
590 INPUT"Nueva denominación del campo ";denominaciónf$
(campos)
600 INPUT"Desea realizar otra modificación";modificaciones$
610 IF LEFT$(modificaciones$,1)="n" THEN GOTO 90  ELSE GOTO
560
620 RETURN
```

Aclaraciones sobre el programa:

290-320 En cabezamiento de la parte del programa.
330+340 Entrada del tamaño del fichero.
350+360 Las variables "denominadas" se dimensionarán con el tamaño deseado.
400+410 El número de campos y de registros de datos se salvarán en variables auxiliares (campos h, registros de datos h), ya que las antiguas variables pueden recibir otros valores al utilizarse en bucles.
380-440 Encabezamiento de la parte del subprograma DEFINICION DE LOS CAMPOS.
450-470 Bucle para leer las denominaciones de los campos.
480 Entrada sobre si deben efectuarse modificaciones,
490 si "no" entonces salto de retroceso al menú.
500-520 Encabezamiento de la parte del subprograma MODIFICAR.
530-550 Bucle para la emisión de las denominaciones de los campos.
560-620 Rutina para la modificación de una denominación; si no se requiere ninguna otra modificación, se realizará un salto de retroceso al menú.

Variables utilizadas:

Registros de datos Contiene el número de

	registros de datos del fichero.
Campos	Contiene el número de campos del fichero.
Registros de hdatos	Contiene el número permanente de registros de datos.
hcampos	Contiene el número permanente de campos.
Denominaciónd$(registros	de datos) Contiene la denominación de los registros de datos.
Denominaciónf$(campos)	Contiene la denominación de campos.
Modificaciones$	Contiene la decisión si/no.

Posibilidades de corrección:

Tal como está planteada la definición del fichero, el número de registros de datos y de campos ya está concretado, es decir que es estático. En el caso que su fichero se haga mayor que lo que estaba planificado, necesitará un concepto dinámico, es decir, que el fichero deberá crecer a medida que se agranda. Ya que ésto es una modificación de principio, ésta debería realizarse por personas avanzadas en la materia. El principiante debería intentar de escribir un par de líneas en las que se evite cualquier error de entrada o bien en las que se utilizara la tecnología WINDOW al efectuarse la entrada de datos. Sobre el particular vea una vez más el manual. Mediante la utilización de WINDOWS puede mejorarse decisivamente el "OUTFIT" así como el factor de Ergonomía.

Utilización de una parte del programa:

La utilización es también aquí relativamente fácil. El programa está dirigido mediante diálogos, es decir que Vd. realiza con el programa un juego de "preguntas y respuestas". En primer lugar deberá entrar los registros de datos (por ejemplo cuando se trata de un fichero de direcciones indicar la cantidad de personas que desea recopilar) y luego el número de campos (cantidad de inserciones que Vd.

requiere para una persona). A continuación introduzca la denominación de los campos (nombre, calle, lugar,...).

Una vez haya concluido este proceso podrá elegir, si lo desea, volver al menú (=entrar no) o bien si desea efectuar modificaciones (=entrar si). Si desea realizar modificaciones, el programa le dará en primer lugar la denominación de los campos. Vd. deberá indicar el número así como la nueva denominación. Una vez se haya concluido este proceso, podrá realizar ya sea una modificación o bien finalizar definitivamente el programa retrocediendo al menú.

P A R T E 3 - E N T R A R F I C H E R O -

La siguiente parte del programa tiene la misión de llenar el fichero definido con los correspondientes datos. Así pues podrá almacenar nombres, calles, lugares, etc. en el fichero. Además existe la posibilidad de modificar directamente después campos individuales. Al principio de esta parte tendrá que determinar el nombre del fichero, como por ejemplo, "fichero de direcciones", "administración del almacén", "todas mis amigas" o bien lo que desee.

```
630 CLS
640 campos = hcampos
650 registro de datos = registro de hdatos
660 PRINT STRING$(&50,"-")
670 LOCATE 15,3: PRINT"E N T R A R  F I C H E R O"
680 PRINT STRING$(&50,"-")
690 LOCATE 1,10: PRINT"(Su fichero tiene";registro de
datos;"Registro de datos y";campos;"Campos)"
700 PRINT: PRINT: INPUT"Cómo deberá llamarse el fichero";
nombre del fichero$
710 CLS
720 REM ver linea 365
730 FOR registro de datos = 1 TO registro de hdatos
740 campos = hcampos
750 FOR campos = 1 TO hcampos
760 WINDOW SWAP 0
770 WINDOW 1,80,1,8
780 PRINT STRING$(&50,"-")
790 PRINT"Nombre del fichero :";nombre del fichero$
800 PRINT"Número mayor de registro de datos :"registro de
datosh,"Número de campos :";hcampos
810 PRINT"Número del registro de datos actual :";registro
de datos
820 PRINT"Número del campo actual                    :";campos
830 PRINT STRING$(&50,"-")
840 WINDOW 1,80,9,25
850 CLS
860 PRINT denominaciónf$(campos);: INPUT"";cantenido$
(campos,registro de datos)
```

```
870 NEXT campos
880 NEXT registro de datos
890 LOCATE 1,10
900 INPUT"Desea realizar modificaciones";modificaciones$
910 IF modificaciones$="n" GOTO 90
920 MODE 2
930 PRINT"Modificaciones de los contenidos de un campo"
940 PRINT"------------------------------"
950 PRINT: INPUT"Qué campo desea modificar";campos
960 INPUT"En qué registro de datos";registro de datos
970 PRINT"Contenido del antiguo campo :";contenido$(campos,
registro de datos)
980 INPUT"Nuevo contenido del campo";contenido$(campos,
registro de datos)
990 INPUT"Desea realizar otra modificación";modificaciones$
1000 IF modificaciones$="n" GOTO 90 ELSE GOTO 920
```

Aclaraciones sobre el programa:

630-690	Encabezamiento de la parte del programa.
650+660	A las variables "registro de datos" y "campos" se les asignarán los antiguos valores en el caso de haberse modificado.
700	Aquí se entrará el nombre del fichero.
720	La variable "contenido$" se dimensionará; ésta recibirá los datos del fichero.
730+750	Se abrirán los bucles para leer los datos.
770-830	Encabezamiento de la parte de entrada de datos; ésta informa sobre los campos, registro de datos, etc. Tecnología Window.
840-880	Aquí se leerán los datos de su fichero.
890-910	Decisión sobre si se deben efectuar modificaciones; si "no", salto de retroceso al menú.
920-1000	Proceso de modificación con posibilidad de seguir modificando.

Variables utilizadas:
---

registro de datos	Conocido
campos	Conocidos
registro de hdatos	Conocido
hcampos	Conocido
nombre del fichero $	Contiene el nombre del fichero
contenido$ (campos,registro de datos)	Contiene todos los datos del fichero, Array bidimensional
modificaciones$	Conocidas

Posibilidades de corrección:

En principio existen dos posibilidades de corrección. La primera es tener en cuenta que al entrar el registro de datos no se borren los campos entrados previamente.

La segunda posibilidad es de saltar a la rutina de modificación después de cada registro de dato.

Ambas modificaciones son fáciles y pueden ser realizadas por cualquier principiante. Observen aquí la utilización de los Windows. Dicho método puede extenderse a cualquier parte del programa incluso a cualquier otro programa.

Utilización de una parte del programa:

Después de haber elegido la parte de ENTRAR se le exigirá indicar el nombre del fichero. El nombre del fichero puede

ser, por ejemplo, el tipo (fichero de direcciones, administración del almacén) o bien el tema (todas mis amigas, socios).

Ahora llegamos al CUADRO DE ENTRADAS. Aquí se le informará continuamente en las líneas superiores sobre el nombre, número mayor de campos y registro de datos así como sobre el número actual de campos y el número de registro de datos. En la parte central deberá entrar consecutivamente (campo por campo) sus datos. En este proceso se irán anotando los números de campos y de registros de datos progresivamente.

Si Vd. ha entrado sus datos tendrá la posibilidad de modificar los campos. Todo lo que tendrá que hacer se lo dirá el programa durante el diálogo. Si Vd. desea hacer una nueva modificación, introduzca en la pregunta correspondiente una "s", de lo contrario la parte del programa regresará de nuevo al menú.

P A R T E 4 - MANTENIMIENTO DEL FICHERO -

La siguiente parte del programa tiene la misión de mantener siempre su fichero "a la orden del día". Muy a menudo ocurre que deben modificarse datos. Por ejemplo cuando un socio se separa de Vd. o bien cuando un conocido se casa y por consiguiente se le asigna otro apellido. A veces también es necesario borrar todo un registro de datos e incluso varios campos. Esta misión la realiza la parte del programa número 3. A esta actividad se la denomina mantener (o modificar) un fichero.

Con el fin de configurar dicho mantenimiento del modo más práctico posible hemos insertado dos posibilidades: O bien introduce Vd. directamente la posición en la que debe modificarse el dato al caso o "busca" (mirando todos los registros de datos manualmente) en el fichero y decide entonces si desea realizar las modificaciones.

```
1010 CLS
1020 PRINT STRING$(&50,"_")
1030 LOCATE 15,3: PRINT"              MANTENIMIENTO DEL
FICHERO "
1040 PRINT STRING$(&50,"_")
1050 PRINT: PRINT"Modificación directa del contenido de un
campo    - 1 -"
1060 PRINT"Buscar  con la  consiguiente posterior modifica-
ción     - 2 -"
1070 PRINT: INPUT"Elija Vd. por favor ";a
1080 IF a=2 GOTO 1170
1090 CLS
1100 PRINT"Modificación directa del contenido de un campo"
1110  PRINT: PRINT: PRINT: INPUT"Qué campo  desea  Vd.
modificar ";campos
1120 INPUT"En qué registro de datos ";registro de datos
1130 PRINT: PRINT: PRINT"Contenido del antiguo campo :";
contenido$(campos, registro de datos)
1140 INPUT"Nuevo contenido del campo";contenido$(campos,
registro de datos)
1150 PRINT:  PRINT: INPUT"Desea  Vd. realizar una modifica-
ción ";modificaciones$
```

```
1160 IF modificaciones$="s" GOTO 1090 ELSE RETURN
1170 CLS
1180 campos = hcampos
1190 registro de datos = registro de hdatos
1200 FOR registro de datos = 1 TO registro de hdatos
1210 WINDOW 1,80,1,8
1220 PRINT STRING$(&50,"-")
1230 PRINT"Fichéro actual :"nombre del fichero$
1240 PRINT"Registro de datos actual :"registro de datos
1250 PRINT STRING$(&50,"-")
1260 GOSUB1290
1270 INPUT"Desea  realizar modificaciones ";modificaciones$
: IF LEFT$(modificaciones$,1)="n" THEN GOTO 1420
1280 GOTO 1350
1290 WINDOW 1,80,9,20
1300 CLS
1310 FOR campos = 1 TO hcampos
1320 PRINT denominaciónf$(campos);":";contenido$(campos,
registro de datos)
1330 NEXT campos
1340 RETURN
1350 WINDOW 1,80,21,25
1360 CLS
1370 PRINT STRING$(&50,"-")
1380 INPUT"Qué campo desea modificar ";campos
1390 INPUT"Nuevo contenido del campo ";contenido$(campos,
registro de datos)
1400 INPUT"Desea realizar una modificación";modificaciones$
1410 IF LEFT$(modificaciones$,1)="s" THEN GOTO 1360
1420 NEXT registro de datos
1430 RETURN
```

Aclaraciones del programa:

1010-1060	Cuadro del título del encabezamiento de la parte del programa.
1070	Selección de la parte del subprograma.
1080	Salto a la parte del subprogrma correspondiente.
1090-1140	Proceso de modificación indicando el número del registro de datos y del número del campo del dato a ser modificado.
1150+1160	Decisión sobre si desea modificar algún dato; si "no" salto de retroceso al menú, o salto al principio de la parte del subprograma.
1170-1250	Encabezamiento de las demás partes del subprograma.
1200	Apertura del bucle para la indicación del registro de datos.
1260	Salto a la rutina que indica los campos.
1290-1330	Rutina que emite mediante un bucle la denominación así como el contenido de los campos del registro de datos.
1340	Salto de retroceso al 1270.
1270	Decisión sobre si debe modificarse algo; si "no" pasar al siguiente registro de datos; si "sí", salto al 1350.
1350-1400	Rutina de modificación.
1400+1410	Decisión sobre si debe modificarse un campo en el registro de datos; sino pasar al próximo registro de datos.

Variables utilizadas:
---

a	Contiene el valor de la parte del subprograma a la que debe saltar.
campos	Conocidos.

```
registro de datos      Conocido.
hcampos                Conocidos.
registro de hdatos     Conocido.
contenido$(campos,registro de datos) Conocido.
modificaciones$        Conocidas
nombre del fichero$    Conocido
```

Posibilidades de modificación:

Las modificaciones son únicamente necesarias en la segunda parte. Momentáneamente allí no es posible regresar al menú mediante una activación de teclas; primeramente deberá buscar en todos los registros de datos. Ya que ésto puede llegar a ser muy engorroso en un fichero grande hasta que haya llegado a localizar el registro de datos a ser modificado, debería aquí buscarse otra ayuda. Esto puede conseguirlo mediante una rutina corta que examine si ha sido activada la tecla determinada para retroceder al menú como está previsto (por ejemplo :&). Si realmente se ha activado esta tecla podrá conseguirse un salto de retroceso al menú mediante la simple instrucción de GOTO.

Utilización de una parte del programa:

En primer lugar tendrá que tomar la decisión sobre si desea realizar directamente la modificación o bien si desea hojearlo todo primero. Una modificación directa la podrá hacer tan sólo si conoce el número del campo y del registro de datos. De lo contrario deberá elegir la parte número 2.

Si ha escogido la parte número 1 deberá introducir primero el número del campo y después el número del registro de datos. A partir de aquí el programa le dará el antiguo contenido del campo y tendrá así la posibilidad de poderlo modificar. Después podrá elegir si desea todavía modificar un campo. Si la respuesta es "no" el programa saltará de nuevo al menú.

Si ha elegido la parte número dos el programa le indicará los campos del primer registro de datos. Después tendrá la posibilidad de poderlos modificar. Si desea realizar la modificación deberá introducir únicamente el número del campo y el nuevo contenido del mismo. Ahora o bien puede modificar un campo o mirarse el próximo registro de datos.

P A R T E 5 - ALMACENAR EL FICHERO -

La siguiente parte del programa tiene la misión de almacenar sobre cinta los ficheros creados por Vd. mediante los datos del CPC, para que no los pierda al desconectar el ordenador.

```
1440 CLS
1450 PRINT STRING$(&50,"_")
1460 PRINT"        A L M A C E N A R   E L   F I C H E R O"
1470 PRINT STRING$(&50,"_")
1480 LOCATE 5,10: PRINT"Su fichero se está almacenando,
espere por favor."
1490 OPENOUT nombre del fichero$
1500 FOR registro de datos = 1 TO registro de hdatos
1510 FOR campos = 1 TO hcampos
1520 PRINT#9,contenido$(campos,registro de datos)
1530 NEXT campos
1540 NEXT registro de datos
1550 CLOSEOUT
1560 RETURN
```

Aclaraciones sobre el programa:

1440-1470	Encabezamiento de la parte del programa.
1480	Comunicación para el usuario.
1490	Apertura de un fichero sobre cinta bajo el nombre del fichero que se encuentra en el ordenador.
1500-1540	Bucle para poder escribir los datos sobre cinta.
1550	Cerrar el fichero.
1560	Salto de retoceso al menú.

Variables utilizadas:
---

nombre del fichero$	Conocido
registro de datos	Conocido
campos	Conocidos
registro de hdatos	Conocido
hcampos	Conocidos
contenido$(campos,registro de datos)	Conocido

Posibilidades de modificación:

En principio no se realizan modificaciones. Si se requiere es posible modificar el "Outfit" del programa (es decir el aspecto o la Ergonomía de la pantalla).

Utilización de una parte del programa:

Después de elegir la parte del programa le aparecerá después de un breve espacio la petición de activar las teclas PLAY y RECORD de su grabadora. Si ha realizado ya este proceso deberá activar únicamente cualquier tecla. El resto lo solucionará el programa para Vd.

P A R T E 6 - CARGAR EL FICHERO -

La parte siguiente tiene la misión, de volver a cargar ficheros que Vd. haya creado y después almacenado. Para ello será necesario conocer el número de campos y de registros de datos del fichero. Estos parámetros podrá ajustarlos en la parte del programa número 1.

```
1570 CLS
1580 PRINT STRING$($50,"_")
1590 PRINT"                C A R G A R          F I C H E
R O"
1600 PRINT STRING$(&50,"_")
1610 LOCATE 5,10: PRINT"Su fichero se está cargando, espere
por favor."
1620 OPENIN "
1630 FOR registro de datos=1 TO registro de hdatos
1640 FOR campos = 1 TO hcampos
1650 INPUT#9,contenido$(campos,registro de datos)
1660 NEXT campos
1670 NEXT registro de datos
1680 CLOSEIN
1690 RETURN
```

Aclaración de la parte del programa:

1570-1600	Emisión del encabezamiento de la parte del programa.
1610	Comunicación para el usuario.
1620	Apertura de un fichero para la lectura del Cassette.
1630-1670	Bucle para la lectura del fichero del Cassette.
1680	Cerrar el fichero.
1690	Salto de retroceso al menú.

Variables utilizadas:
---

registro de datos	Conocido
campos	Conocidos
registro de hdatos	Conocido
hcampos	Conocido
contenido$(campos, registro de datos)	Conocido

Modificaciones:

Como en la parte del programa ALMACENAR FICHERO, las modificaciones serán posibles en el campo óptico. Además sería factible modificar la rutina de lectura sin conocer el número de campos y del registro de datos.

Utilización de una parte del programa:

La utilización es muy sencilla. Después de que haya elegido esta parte rebobine el Cassette delante del fichero que desea cargar. Después active la tecla PLAY a través del programa y acto seguido cualquier otra tecla. La parte del programa leerá automáticamente el fichero de la cinta.

Después se realizará un salto de retroceso al menú.

P A R T E 7 - B U S C A R -

La siguiente parte del programa le brinda la posibilidad de buscar en su fichero un dato determinado, como por ejemplo, un nombre, un número de teléfono o similares. Esto se realiza de tal manera que deberá entrar el dato a ser buscado y la cantidad de posiciones significativas, es decir, que sean importantes. Un pequeño número de posiciones significativas tiene la ventaja que el concepto se encuentra mucho más de prisa. En todo caso se ha aplicado una rutina de búsqueda relativamente rápida (para listados no clasificados). El número no deberá elegirse demasiado pequeño.

Si por ejemplo Vd. ha elegido únicamente 3 posiciones, para el ordenador "Zipi" y "Zape" son las mismas personas, ya que las tres primeras posiciones concuerdan. Si elige el número de las posiciones significativas con cuatro, ésto no ocurriría.

```
1700 CLS
1710 PRINT STRING$(&50,"-")
1720 PRINT"                                        B  U  S  C  A  R"
1730 PRINT STRING$(&50,"-")
1740 LOCATE  5,10: INPUT"Como  se llama el concepto buscado
";concepto$
1750 LOCATE 5,11:  INPUT"Cuantas  posiciones significativas
";posiciones
1760 PRINT: PRINT: PRINT"Por favor espere, está buscando"
1770 PRINT"=============================================="
1780 busca$ = LEFT$(concepto$,posiciones)
1790 FOR registro de datos = 1 TO registro de hdatos
1800 FOR campos = 1 TO hcampos
1810 ayuda$=contenido$(campos, registro de datos)
1820 find$=LEFT$(ayuda$,posiciones)
1830 IF busca$ = find$ THEN GOTO 1880
1840 NEXT campos
1850 NEXT registro de datos
```

```
1860 INPUT"Concepto no encontrado, buscar otro concepto";
selección$
1870  IF  LEFT$(selección$,1)  =  "n" THEN RETURN ELSE GOTO
1700
1880  PRINT"El  concepto  buscado";concepto$;"  está  en el
campo";campos;"  en  el  registro  de  datos  ";registro de
datos: INPUT"Buscar otra vez ?"; selección$
1890 IF LEFT$(selección$,1) =  "N"  THEN  RETURN  ELSE GOTO
1700
```

Aclaraciones del programa:

1700-1730	Encabezamiento de la parte del programa.
1740-1750	Lectura de las informaciones sobre el concepto a ser buscado.
1760-1770	Información para el usuario.
1780	A la variable busca$ se le asignarán las posiciones significativas del concepto a ser buscado.
1790-1800	Se abrirá el bucle para el proceso de búsqueda.
1810	A la variable ayuda$ se le asignarán correlativamente los contenidos del fichero mediante el bucle.
1820	A la variable find$ se le asignarán correlativamente las posiciones significativas de los contenidos del fichero.
1830	Si busca$=find$ entonces se habrá encontrado el el concepto a ser buscado.
1840+1850	Final de ambos bucles.
1860+1870	Si el programa no se bifurca durante el bucle, el concepto no se habrá encontrado; en esta línea se emitirá el correspondiente aviso. Existe la posibilidad de buscar un nuevo concepto, de lo contrario se hará un salto de retroceso al menú.

1880+1890 Aquí se bifurcará cuando el concepto haya sido encontrado. Se emitirá el concepto así como el número del registro de datos y del campo; además existe la posibilidad de volver a buscar. De lo contrario se efectuará un salto de retroceso al menú.

Variables utilizadas:
---

concepto$	Contiene el concepto a ser buscado.
posiciones	Contiene el número de las posiciones significativas.
busca$	Contiene las posiciones significativas del concepto a ser buscado.
ayuda$	Variable auxiliar a la que se le asignarán correlativamente los contenidos del fichero.
find$	Contiene las posiciones significativas de los datos del fichero con los que se comparará el busca$.
registro de datos	Conocido.
campos	Conocidos.
registro de hdatos	Conocido.
hcampos	Conocidos.
selección$	Conocida.

Posibilidades de modificación:

En la misma rutina de búsqueda no será necesario efectuar modificación alguna. Esta ha sido óptimamente sintonizada en el CPC y es muy rápida para una rutina de Basic con cantidades de datos relativamente grandes. La única posibilidad de modificación que podría tenerse sería no sólo indicar el número del registro de datos y del campo sinó también el registro de datos total con los correspondientes campos. Esto no debería representar ningún problema

para el principiante, ya que el número del registro de datos y la cantidad de campos son conocidos en un mismo registro (variable camposh).Así pues esta modificación podría realizarse perfectamente con un único bucle de emisión.

Utilización de la parte del programa:

Después de seleccionar el BUSCAR deberá entrar en primer lugar el concepto a ser buscado. A continuación se le exigirá indicar el número de las posiciones significativas. El significado de éstas lo ha conocido ya en la introducción de estaparte. En el caso de que Vd. ya no se acuerde de ello, vuelva a leerlo de nuevo.

Si ya se ha encontrado el concepto, el programa le indicará el número del registro de datos así como el del campo. A continuación deberá decidirse si desea seguir buscando algún dato (tecla "s"), de lo contrario el programa retrocederá al menú.

Si el concepto no se ha encontrado, el programa se lo comunicará igualmente. Ahora podrá volver al menú o bien introducir un proceso de búsqueda. Vd. podría llevarlo a cabo con el mismo concepto pero con posiciones menos significativas, ya que muy a menudo ocurre que uno se equivoca no reconocerá tales errores. Para ello existen las llamadas "rutinas de tolerancia", que por ejemplo emiten con un 90 % de conformidad un concepto. Rutinas de este tipo se encuentran tan sólo en tratamientos de ficheros para máquinas comerciales de poco valor.

P A R T E 8 - IMPRIMIR FICHERO -

La siguiente parte del programa le brinda la posibilidad de imprimir mediante la conexión de una impresora, los ficheros creados con los datos de su CPC.

Malas lenguas dicen que con los ordenadores se ha iniciado el siglo del papel; la realidad es que la mayoría de usuarios de ordenadores desean verlo todo "negro sobre blanco". Esto no es únicamente una ascensión a lo anticuado, sino que también tiene su lado positivo. Si por ejemplo se pierde su fichero (que no puede volver a cargarlo o bien lo ha solapado), al menos dispondrá de todos los datos sobre papel.

```
1900 CLS
1910 PRINT STRING$(&50,"_")
1920 PRINT"       I m p r i m i r     F I C H E R O
 "
1930 PRINT STRING$(&50,"_")
1940 LOCATE 5,10: PRINT"El fichero será imprimido. Sírvase
esperar."
1950 FOR registro de datos = 1 TO registro de hdatos
1960 FOR campos = 1 TO hcampos
1970 PREINT#8,contenido$(campos,registro de datos)
1980 NEXT campos
1990 NEXT registro de datos
2000 PRINT: INPUT"Una impresión más";selección$
2010 IF selección$="s" GOTO 1950 ELSE RETURN
```

Explicación del programa:

1900-1930	Encabezamiento de la parte del programa.
1940	Información para el usuario.
1950-1990	Rutina para la impresión mediante un bucle, una instrucción PRINT y la dirección de la impresora (8).

2000-2010 Posibilidad de volver a realizar una impresión; si se ha presionado "n" se efectuará un salto de retroceso al menú.

Variables utilizadas:
---

registro de datos	Conocido
campos	Conocidos
registro de hdatos	Conocido
hcampos	Conocidos
selección$	Conocida

Posibilidades de modificación:

La rutina actual para la impresión es relativamente sencilla; es la llamada edición no formateada. Los distintos campos del fichero se emiten correlativamente.

Al contrario de lo expuesto existe una edición formateada. Una de estas ediciones podría por ejemplo emitir el número del registro de datos y de campos, la descripción del campo, etc; además podría dejar apartados entre los registros de datos y muchas cosas más.

Todas estas ampliaciones pueden realizarse en su CPC mediante la fabulosa instrucción de PRINT USING. Léase este apartado detenidamente en el manual; no es precisamente muy sencillo pero puede ahorrar mucho trabajo en las funciones String (right$, ,left$...).

Utilización de la parte del programa:

La utilización de esta parte del programa es muy sencilla. Después de elegir esta parte se imprimirá el fichero. Unicamente necesitará elegir al final de la impresión si desea imprimir sobre papel el fichero al caso. Evidentemente podrá asimismo elegir para la impresión de sus datos el tipo de escritura que desee. Si así lo requiere, antes de trabajar con este programa deberá emitir los códigos de dirección correspondientes a su impresora.

La compresión de caracteres la obtendrá Vd. a través de la instrucción:

```
PRINT #8;CHR$(15)
```

Caracteres ampliados los podrá obtener mediante la instrucción:

```
PRINT #8;CHR$(14)
```

P A R T E 9 - FINALIZAR EL PROGRAMA -

La siguiente parte del programa tiene la misión de interrumpir el programa. Si no desea seguir trabajando con los DATOS de su CPC, elija esta parte. El programa se finaliza por sí solo.

```
2020 CLS
2030 PRINT STRING$($50,"_")
2040 PRINT"              F I N A L I Z A R    E L
  P R O G R A M A"
2050 PRINT STRING$(&50,"_")
2060 LOCATE 5,10: INPUT"Desea realmente finalizar el
programa";b$: IF b$="s" THEN END ELSE RETURN
```

Explicaciones del programa:

2020-2050	Emisión del encabezamiento de la parte del programa.
2060	Decisión sobre si se desea realmente finalizar el programa; si "no" se realizará un salto de retroceso al menú.

Variables utilizadas:

b$	Contiene la decisión si/no sobre si debe finalizarse el progrma.

Posibilidades de modificación:

Las posibilidades de modificación son únicamente de naturaleza óptica, ya que esta parte no presenta en cuanto a la técnica del programa exigencia alguna.

Utilización de la parte del programa:

Después de efectuar la selección sólo tendrá que decidirse sobre si realmente desea finalizar el programa. Esta pregunta se incorporó, ya que puede ocurrir que Vd. por equivocación elija esta parte del programa y todos los datos que Vd. no haya almacenado o bien impreso se pierdan.

P A R T E 10 - B O R R A R -

La última parte del programa le brinda la posibilidad de borrar datos. Esto se hace necesario por ejemplo, cuando un conocido ha cambiado de dirección o bien a muerto, cuando Vd. no puede continuar con un artículo o algo parecido. Aquí se recomienda ser prudente con el borrado ya que las informaciones pueden ser borradas rápidamente pero difíciles de volver a conseguir.

```
2070 CLS
2080 PRINT STRING$(&50,"_")
2090 PRINT"                                        B O R R A R"
2100 PRINT STRING$(&50,"_")
2110 LOCATE 10,10: PRINT"Borrarlo todo       - 1 -"
2120 LOCATE 10,11: PRINT"Borrar el registro de datos - 2 -"
2130 LOCATE 10,14: INPUT"Por favor elija ";selección$
2140 IF selección$="2" THEN GOTO 2210
2150 CLS
2160 PRINT"Borrarlo todo"
2170 PRINT"------------"
2180 LOCATE 5,10: INPUT" Desea realmente borrarlo todo";
selección$
2190 IF selección$="s" THEN CLEAR
2200 GOTO 90
2210 CLS
2220 PRINT"Borrar el registro de datos"
2230 PRINT"-----------------"
2240 LOCATE 5,10: INPUT"Que registro de datos desea borrar
";registro de datos
2250 FOR campos = 1 TO hcampos
2260 contenido$(campos,registro de datos)="      "
2270 NEXT campos
2280 LOCATE 5,15: INPUT"Desea borrar todavia algún registro
de datos ";selección$
2290 IF selección$="s" GOTO 2210 ELSE RETURN
```

Aclaración del programa:

2070-2120 Encabezamiento de la parte del programa.
2130+2140 Decisión sobre una de las dos partes del programa.
2150-2200 Rutina para borrar todos los datos mediante la instrucción Basic CLEAR; en primer lugar se preguntará si realmente desea borrar todos los datos.
2220-2230 Encabezamiento de la parte del subprograma.
2240 Introducción de la indicación sobre el registro de datos que debe ser borrado.
2250 Apertura de un bucle para los campos.
2260 Borrado de los datos mediante la asignación de un String vacío.
2270 Final del bucle.
2280-2290 Posibilidad de elección si todavía debe borrarse un registro de datos; si "no" se efectuará un salto de retroceso al menú.

Variables utilizadas:
---

selección$	Conocida
registro de datos	Conocido
campos	Conocidos
hcampos	Conocidos
contenido$(campos,registro de datos)	Conocido

Posibilidades de modificación:

En la parte del subprograma borrarlo TODO hay modificaciones únicamente de naturaleza óptica. En la parte de borrar el registro de datos podría realizarse una ampliación. Esta

sería selectivamente tan sólo para borrar el campo de un registro de datos. Dicha modificación podría efectuarse preguntando el número del campo y después destinando un String vacío a la variable contenido$, fijada con el número del campo y el número del registro de datos.

Utilización de la parte del programa:

Después de haber elegido la parte, deberá decidirse sobre si desea borrar todos los datos o bien únicamente un registro de datos.

Si realmente se ha decidido en borrarlo TODO, el programa volverá a preguntar si Vd. está seguro de ello. Si no desea borrar todos los datos, entonces entre una "n"; de lo contrario introduzca una "s".

Si se ha decidido por la segunda parte, deberá introducir el número del registro de datos que desea borrar. El borrado lo realizará el progrma para Vd. Si el registro de datos se ha borrado, tendrá la posibilidad de borrar un nuevo registro de datos. Si no es ésto lo que Vd. desea, introduzca una "n" y el programa retrocederá al menú.

P A R T E 11 - E P I L O G O -

Hasta ahora ha trabajado Vd. algunas páginas de este libro y habrá entrado aproximadamente 9K de programa

Aquí le hemos brindado un tratamiento completo de ficheros además de toda la documentación que se necesita.

Con estas informaciones debería serle posible comprender completamente el programa tanto en lo que se refiere a la utilización como a la programación del programa. Hemos intentado mantener la estructura lo más sencilla posible, para que Vd. pueda ir realizando lo hasta ahora expuesto. Estudie tanto el programa sus correspondientes aclaraciones con la mayor exactitud. De este modo podrá llegar a aprender mucho sobre programación.

Si Vd. llegase a conseguir la incorporación de las posibilidades de modificación del programa, tendrá, no sólo un tratamiento de ficheros de primera clase, sino también una gran cantidad de conocimientos sobre programación.

Trabaje con el programa que sólo así podrá aportarle ventajas.

7.3 TRATAMIENTO DE TEXTOS

INTRODUCCION

La expresión "tratamiento de textos" se ha arraigado desde hace algunos años en nuestro lenguaje de tal manera que, actualmente, tendríamos problemas en el momento de describir algunos dictámenes si no existiese la expresión antes citada.

¿ Qué hay detrás de este fenómeno, detrás de este objeto, que actualmente ya no podemos prescindir de él ? En el propio sentido de la palabra significa tratar textos. La entonación deberá marcarse claramente sobre la palabra "tratamiento", ya que para escribir cartas existían y existen máquinas de escribir.

El tratamiento de textos es más que la propia elaboración de una simple carta. Esto empieza asegurando el texto creado en una memoria de datos para posteriormente, y según las necesidades, volverlo a imprimir; aquí podemos llegar hasta los códigos de dirección con los que se puede mezclar cualquier texto dentro de una impresión. Todo ésto son ventajas de las que nunca podría disfrutar con las máquinas de escribir actuales. ¿ Desearía Vd. quizás, después de escribir 5 letras, cambiar los caracteres de su máquina eléctrica, porque en su texto existen palabras que deberían resaltar del conjunto ?

Un programa de tratamiento de textos es hoy en día, además del tratamiento de ficheros, el medio más usual cuando se trata de aplicar el ordenador personal con sentido propio y racionalmente.

Por este motivo, deseamos llevarlo en este libro un poco por detrás de la fachada de un tratamiento de textos. Quizás se asombre Vd., cuando vea todo lo que deberá tenerse en

cuenta en uno de estos programas. Un buen tratamiento de textos se distingue actualmente por la buena guía del usuario y la elevada seguridad contra errores de manipulación.

Es evidente que dentro del marco de este libro no podemos ofrecer un tratamiento de textos con la calidad de un TEXTOMAT o bien WORD STAR, ya que dichos programas han sido escritos para lenguaje máquina y únicamente el listado de uno de ellos superaría en su longitud al número de páginas de este libro. Además debería añadirse aquí el mismo número de páginas con el fin de aclarar y documentar ampliamente un programa Ensamblador.

Con ello hemos tomado ya una decisión básica, es decir, escribir el tratamiento de textos en Basic. De todos modos, no quisiera encubrir que un programa escrito en Basic tiene una desventaja fundamental: Es lento, según sea el problema de Basic que deba solucionarse; incluso podría llegar a ser muy lento.

Sin embargo, creo estar en lo cierto que uno de estos programas no estará destinado para ser utilizado en una oficina. Lo que únicamente pretendemos, es facilitarle una visión y un inicio en el mundo fascinate del tratamiento de textos.

TODO LO QUE SE NECESITA

La pregunta sobre el lenguaje de programación que se utilizará para tratamiento de textos ha sido ya contestada en el último apartado.

Una nueva e importante decisión deberá asimismo ser tomada, ya que ésta influenciará decisivamente la estructura de nuestro programa. Aquí se trata de cómo queremos archivar nuestros textos, cartas o cualquier otro tipo de documento que pueda estar en la memoria del CPC.

Me parece oirle estar diciendo ¿ cómo y dónde está el problema ? Veamos nuevamente las posibilidades del almacenamiento de datos que se nos ofrecen.

1.) Para cada línea de texto que deseamos recopilar y procesar necesitaremos una variable, o mejor dicho, una variable String. Ya sabemos que esta solución es más que suficiente y ésto cualquier profano en la materia lo comprenderá; sabemos también que después cada variable deberá ser contactada con su correspondiente denominación (por ejemplo variable1$). Una impresión debería por consiguiente programarse según sigue:

```
1234 PRINT#8,variable1$
1235 PRINT#8,variable2$
1235 PRINT#8,variable3$
```

Esta posibilidad no es ninguna solución para nuestro problema.

2.) Lo que después nos queda más cerca es coger una variable del campo que, según la línea del texto en la que nos encontramos, será indexada con un índice. La idea no está mal, pero si lo observamos detenidamente encontraremos también aquí fallos de importancia. En un campo de esta

tabla unidimensional pueden recopilarse hasta 256 letras. Los problemas aparecerán cuando dentro de un campo debe añadirse algo y dicha inserción repercute además sobre la línea siguiente.

Aquí debería crearse una rutina muy amplia que disgregase los distintos campos del texto y finalmente los volviese a recopilar de forma adecuada. Todo ésto sería muy costoso en tiempo y en ocupación de espacio de memoria; por este motivo hemos preferido archivar rápidamente esta posibilidad que planteábamos.

3.) Mucho más fácil de administrar es la matriz bidimensional. Esta pueden imaginársela como el tablero de un ajedrez, donde en cada campo se archivará exactamente una letra. Incluso las inserciones de texto o bien el borrado de los mismos es relativamente fácil de manejar. ¿ Qué nos impide aquí aceptar esta buena oferta ?

En realidad nada, sólo el hecho de tener que reservar para cada letra un campo separado de nuestro tablero de ajedrez. Cada uno de estos campos queda administrado por nuestro interpretador de Basic que, si lo observamos detenidamente, necesitará para cada letra algunos Bytes para la administración interna. La relación podríamos calcularla aquí de aproximadamente 5 a 1.

Dicho en otras palabras, para un texto de 33 líneas el interpretador de Basic nos "roba" 10000 (¡) Bytes para poder administrar debidamente las citadas líneas.

4.) ¿ Qué nos queda todavía ? Tomemos sencillamente un tratamiento de textos profesional como ejemplo. Aquí se escribirá cada letra que introduzcamos en una posición de memoria directa, ya que al tratarse de lenguaje máquina no queda otra posibilidad. Deberemos tener en cuenta que el

campo del ordenador que necesitamos para nuestro texto ha sido declarado como campo tabú para cualquier utilización del interpretador, del programa o bien por variables normales. Sin embargo existe una instrucción cómoda en el CPC 464 y 6128 en los que se nos reserva un espacio libre.

```
1000 REM ***********************************
1010 REM ***          text me              ***
1020 REM ***********************************
1030  GOSUB  4030: REM estructuración de las diéresis
alemanas
1040 ON ERROR GOTO 4170
1050 MEMORY 32700: MODE 2: rmargen=69
1060 storelen=&1000
1070 lmargen=10: omargen=5: umargen=66
1080 paplen=72: papnum=0: página=1
1090 FOR t = 32701 TO 32715 : READ a : POKE t,a : NEXT t
1100 FOR t = 32721 TO 32735 : READ a : POKE t,a : NEXT t
1110 CLS: línea=1: columna=1 : screen=1
1120 store=32768: lostore=store
1130 loline=1: histore=lostore
```

En estas primeras líneas se habrán archivado casi todas las informaciones importantes que más adelante serán requeridas.

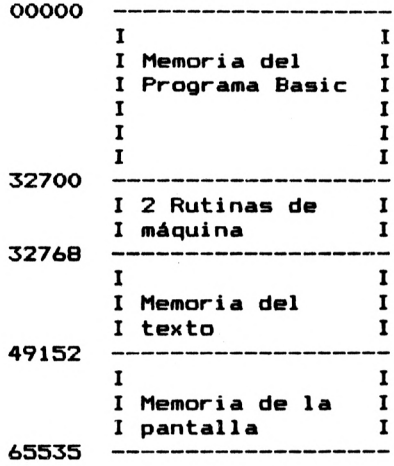
De la línea 1030 se saltará a un subprograma el cual creará las diéresis alemanas sobre el "Schneider". Si más adelante desea imprimir un texto con diéresis, para la impresora Schneider (NLQ 401) no se necesitará ningún tipo de adaptación. En otro tipo de impresoras seguramente se necesitaría la adaptación adecuada. Una propiedad de su CPC es que cuando haya llamado la instrucción SYMBOL AFTER no podrá volver de nuevo sobre esta instrucción, ya que se emitiría un "Improper Argument Error".

Si más tarde vuelve a trabajar con el programa y, después de haberlo iniciado ya una vez, lo llama de nuevo, deberá saltarse el subprograma para la creación de las diéresis alemanas. Esto es muy fácil; para ello deberá iniciar el programa con

RUN 1040

en lugar de la instrucción normal de RUN.

En la línea 1050 limitaremos la memoria Basic mediante la instrucción MEMORY. Después de realizada la limitación, la división de la memoria habrá quedado según sigue:



Después de realizar la limitación tendremos a disposición para nuestro texto máximo 16KB. Este campo libre deberá ser borrado, ya que no sabemos lo que había anteriormente en el mismo. La rutina del borrado la encontrará de las líneas 1230 a 1270.

Ya que esta rutina es una de las partes más lentas del programa, podrá disminuir o ampliar a través de la variable (storelen) definida en la línea 1060 el campo a ser borrado. En el presente programa se preparará un campo desde $1000 = 4096 caracteres para el texto.

El significado de las próximas variables es según sigue:

rmargen = margen derecho del texto para la impresión y la pantalla
lmargen = margen izquierdo del texto para la impresión y la pantalla
omargen = margen superior del texto para la impresión
umargen = margen inferior del texto para la impresión
paplen = número de las posibles líneas de impresión por página
página = número actual de la página
línea = línea actual en la que se encuentra el cursor
columna = columna actual en la que se encuentra el cursor
screen = un indicador auxiliar en el que queda retenido en qué línea de la pantalla se encuentra el cursor
stor = dirección en la que se almacena un carácter
lostore = final inferior de la memoria del texto
histore = posición más elevada de la memoria del texto que se ha contactado
loline = número de líneas de las líneas superiores del texto que aparecen en pantalla

En las líneas 1090 y 1100 se estructurarán dos programas máquina que permitirán un cambio rápido de pantalla del texto representado.

Los datos para estas rutinas los encontrará al final del programa en las líneas 4220 y 4230.

LO PRACTICO Y UTIL

Lo que hasta ahora hemos programado es correcto para el programa, pero no indica ningún tipo de progreso en la pantalla. Sin embargo, en las siguientes líneas ésto cambiará enormemente. Aquí estructuraremos el encabezamiento de la máscara que nos informará siempre sobre todos los parámetros importantes del programa (en la línea que nos encontramos, etc) y que además le dará un aspecto distinto a nuestro programa.

```
1140 REM estructura de la máscara de la pantalla
1150 REM locate position en x e y
1160 x=1: y=1: GOSUB 3310
1170 GOSUB 3330
1180 FOR y=2 TO 4: GOSUB 3350: NEXT y
1190 GOSUB 3330
1200 x=5: y=3: GOSUB 3310
1210 PRINT"TEXT-ME    1985 Data Becker        Línea:  Colum
na:        Página:"
1220 GOSUB 3370
1230 LOCATE 30,5: PRINT 32768+storelen;" - "
1240 FOR t=32768 TO 32768+storelen: POKE t,32
1250 LOCATE 40,5: PRINT t: NEXT t: POKE t,ASC("E"):POKE t+1
,ASC("N"): POKE t+2,ASC("D"): POKE t+3,ASC("E")
1260 LOCATE 30,5: PRINT STRING$(20,"*")
1270 FOR t=1 TO 3: PRINT CHR$(7): FOR i=1 TO 100: NEXT i:NE
XT t
1280 GOSUB 3420
```

Las variables x e y son dos campos auxiliares que se utilizan para el posicionamiento del cursor y que se realiza en el subprograma a partir de la línea 3310. Hasta la línea 1210 se estructurará el encabezamiento del programa sobre la pantalla. Las emisiones PRINT se encuen-

tran en los subprogramas por utilizarse más frecuentemente.

El subprograma que se encuentra a partir de la línea 3370 crea directamente debajo del encabezamiento del programa una barra que nos indicará siempre dónde nos encontramos en la anchura del formulario definido libremente. Además y para mayor orientación tiene una división central.

A partir de la línea 1230 se preparará finalmente la memoria del texto según el tamaño de la variable (storelen) para la admisión de nuestro texto. Durante el proceso de borrado se indicará en la pantalla la posición actual de la memoria que en aquel momento se está borrando y la dirección, hasta donde se preparará la memoria del texto.

Ya que este proceso puede requerir algunos minutos de tiempo, pero que por otro lado pueden aprovecharse, su CPC le avisará mediante un tono prolongado (línea 1270) sobre la terminación de los preparativos realizados.

Además, detrás del final de la memoria del texto preparada se escribirá la palabra "FIN", ya que si Vd. no tiene en cuenta este final podría tener sorpresas desagradables.

Y AHORA VAMOS A EMPEZAR

Aquí ya tenemos el corazón de nuestro programa. En la línea 1320 se examinará si hay una tecla activada. Si este no fuera el caso, el programa se esperará en esta posición hasta que hayamos presionado una tecla.

En las líneas 1340 hasta 1410 se realizará bajo activación de una tecla la correspondiente verificación sobre si se trata de un carácter normal o de una función de mando. Las funciones de mando que se filtran directamente son:

Flecha hacia la derecha.....	Desplazar el cursor hacia la derecha.
Flecha hacia la izquierda.....	Desplazar el cursor hacia la izquierda.
Flecha hacia abajo.....	Desplazar el cursor una línea más abajo.
Flecha hacia arriba.....	Desplazar el cursor una línea más arriba.
Tecla CLR.....	Emitir carácter de final de párrafo, borrado de las líneas restantes y salto al inicio de la próxima linea.
Tecla ENTER.....	Salto al inicio de la próxima línea.

Las lineas 1420 y 1430 se ocupan finalmente de que la tecla acabada de activar y siempre y cuando se trate de una letra válida, también sea archivada en nuestra memoria de textos y además aparezca sobre pantalla. La representación inversa de la próxima posición del cursor tendrá lugar en el subprograma a partir de la línea 3470.

```
1290 x=lmargen : y=7 :GOSUB 3310
1300 GOSUB 3470
1310 REM modalidad de escritura
1320 tecla$ = INKE$
1330 IF tecla$ = "" THEN GOTO 1320
1340 IF tecla$ = CHR$(243) THEN GOTO 1530
1350 IF tecla$ = CHR$(242) THEN GOTO 1560
1360 IF tecla$ = CHR$(241) THEN GOTO 1640
1370 IF tecla$ = CHR$(240) THEN GOTO 1710
1380 IF tecla$ = CHR$(16) THEN GOTO 1780
1390 IF tecla$ = CHR$(13) THEN GOTO 1840
1400 IF ASC(tecla$) < 32 THEN GOTO 2360 : REM ctrl codes
1410 IF ASC(tecla$) > 126 THEN GOTO 1910
1420 POKE store,ASC(tecla$)
1430 PRINT tecla$;
1440 IF columna = rmargen-lmargen-3 THEN PRINT CHR$(7);
1450 x=x+1: columna=columna+1: store=store+1
1460 IF store>=histore THEN histore=store
1470 IF x>rmargen THEN x=lmargen: columna=1: y=y+1: linea=
linea+1: screen=screen+1
1480 IF screen > 18 THEN screen = 18 :GOTO 3680
1490 GOSUB 3420 : REM emitir posiciones
1500 GOSUB 3310
1510 GOSUB 3470 : REM emitir caracteres
1520 GOTO 1310
```

A partir de la línea 1450 y después de la emisión de un carácter regular continuarán escribiéndose todos los caracteres necesarios y parámetros. La dirección de memoria deberá aumentarse localizándose aquí la nueva posición de columnas del cursor. Podría ser posible que hubiésemos escrito el último carácter exactamente en el ángulo inferior izquierdo de la pantalla. El contenido de la pantalla deberá subirse una línea más y la línea recientemente mezclada deberá ser organizada; finalmente deberá fijarse de nuevo el cursor sobre la columna izquierda, disponiendo

el contenido de las variables hisstore al valor actual. Seguidamente se emitirán todos los nuevos parámetros en el encabezamiento del programa.

En la linea 1520 se realizará el salto de retroceso a la modalidad de escritura.

Una pequeña, pero útil ampliación, podrá encontrarla en la línea 1440. Cada vez que alcancen la antepenúltima posición antes del final de la línea y durante la creación normal de texto, sonará una pequeña señal, lo que le recordará a las antiguas máquinas de escribir.

EL MOVIMIENTO LO ES TODO

Las partes siguientes del programa sirven para el libre movimiento del cursor del texto por la pantalla. Son las rutinas que han sido llamadas de las líneas 1340 hasta 1370. Aquí también será válido encontrar un número de errores de manipulación involuntarios así como de casos especiales. Así, por ejemplo, se examinará en la línea 1600, si al mover el cursor hacia la izquierda se ha quedado por debajo del margen izquierdo del formulario. Si éste fuera el caso, deberá ponerse el cursor en la columna derecha de la línea anterior. Antes de que ésto ocurra, deberá examinarse si en realidad existe una línea anterior (Vd. podría estar también al principio de un texto) y si ésta existe en pantalla o bien si el contenido de la pantalla deberá ser desplazado una línea más abajo.

```
1530 REM poner el cursor a la derecha
1540 GOSUB 3550
1550 GOTO 1430
1560 REM poner el cursor a la izquierda
1570 GOSUB 3550
1580 PRINT tecla$;
1590 x=x+1 : columna=columna-1 : store = store-1
1600 IF x<lmargen THEN x=rmargen: y=y-1: linea=linea-1:
columna=rmargen-lmargen+1: screen=screen-1
1610 IF linea<1 THEN linea=1: columna=1: store=32768:
x=lmargen: y=7: screen=1
1620 IF screen < 1 THEN screen = 1 : GOTO 3750
1630 GOTO 1490
1640 REM  bajar el cursor
1650 GOSUB 3550
1660 PRINT tecla$; CHR$(8);
```

```
1670 y=y+1: linea=linea+1:store=store+(rmargen-lmargen+1)
1680 IF store>=histore THEN histore=store
1690  screen=screen+1:  IF  screen>18  THEN screen=18: GOTO
3680
1700 GOTO 1490
1710 REM subir el cursor
1720 GOSUB 3550
1730 PRINT tecla$ ; CHR$(8);
1740 linea=linea-1: IF linea<1 THEN linea=1: GOTO 1490
1750 y=y-1 :store=store-(rmargen-lmargen+1)
1760 screen=screen-1:IF screen<1 THEN screen=1 : GOTO 3750
1770 GOTO 1490
```

REGRESO AL PRINCIPIO

Tal como se ha descrito en la modalidad de escritura, tendrá Vd. dos posibilidades de llegar al principio de una continuación de línea.

La tecla ENTER motiva un salto sencillo al principio de la próxima línea; la tecla CLR borrará el resto de la línea donde se había activado. Además, la tecla CLR indicará el llamado carácter de final de párrafo; ésto es una pequeña flecha que indicará sobre la pantalla hacia la izquierda. Esta flecha es un símbolo de limitación, ya que todas las añadiduras posteriores de caracteres o bien los borrados tienen siempre un efecto sobre el párrafo en el que han sido llamados.

También en estas dos partes del programa deberán examinarse los casos especiales (alcanzar el final inferior de la pantalla).

```
1780 REM dar return
1790 POKE store,13 : PRINT CHR$(242);
1800 IF x=rmargen THEN GOTO 1450
1810 FOR t=x+1 TO rmargen: store=store+1
1820 POKE store,32: PRINT" ";: NEXT t
1830 x = rmargen : GOTO 1800
1840 REM dar return sin borrar
1850 carácter$=CHR$(PEEK(store))
1860 IF carácter$=CHR$(13) THEN carácter$=CHR$(242)
1870 PRINT carácter$;
1880 IF x=rmargen THEN GOTO 1450
1890 x=x+1: store=store+1
1900 GOTO 1880
```

BORRAR E INTERCALAR

Las siguientes cuatro partes del programa se pondrán en funcionamiento cuando se haya activado una de las teclas, cuyo código sea mayor que 126. Para nuestro programa únicamente son de importancia cinco códigos del teclado:

Tecla CTRL y la flecha hacia abajo.....intercalar línea

Esta función origina el intercalado de una línea del texto a la posición actual del cursor. A continuación la pantalla deberá formarse de nuevo, para lo que se necesitarán algunos segundos.

```
1910 REM ctrl crsr down = line insert
1920 IF ASC(tecla$) <> 249 THEN GOTO 2020
1930 anchura=rmargen-lmargen+1: ayuda=(linea-1)*anchura
1940 FOR t=histore TO lostore+ayuda STEP -1
1950 POKE t+anchura,PEEK(t)
1960 NEXT t
1970 FOR t=lostore+ayuda TO lostore+ayuda+anchura-1
1980 POKE t,32: NEXT t
1990 histore=histore+anchura
2000 IF memorizar=1 THEN memorizar=0: RETURN
2010 GOTO 2250
```

La tecla CTRL y la flecha hacia arriba.....borrar línea

Esta función borra la línea en la que se encuentra el cursor del texto. Las líneas restantes se juntarán correspondientemente y el contenido de las variables histore se reducirá en la línea borrada.

```
2020 REM ctrl crsr up = line delete
2030 IF ASC(tecla$) <> 248 THEN GOTO 2120
2040 anchura=rmargen-lmargen+1: ayuda=(linea-1)*anchura
2050 FOR t=lostore+ayuda TO histore
2060 POKE t,PEEK(t+anchura)
2070 NEXT t
2080 FOR t=histore-anchura TO histore
2090 POKE t,32: NEXT t
2100 GOTO 2250
```

La tecla CTRL y la flecha hacia la derecha.....intercalar signo

Esta función intercala en la posición actual del cursor un espacio libre y desplaza todos los demás signos del apartado (hasta el signo final del apartado) en una posición hacia la derecha. Aquí tenga presente que, si el signo final del apartado es desplazado por encima del margen derecho, el inicio de la línea siguiente se borrará. Para evitar ésto deberá intercalar previamente en su texto una línea en blanco (con la CTRL, la flecha hacia abajo y función).

```
2110 REM insert one letter
2120 IF ASC(tecla$) <> 251 THEN GOTO 2190
2130 FOR t=store TO histore :signo=PEEK(t): IF signo=13
THEN GOTO 2150
2140 NEXT t: t=t-1
2150 REM
2160 FOR ayuda=t+1 TO store+1 STEP -1
2170 POKE ayuda,PEEK(ayuda-1): NEXT ayuda
2180 POKE ayuda,32 : GOTO 2240
```

La tecla CTRL y la flecha hacia la izquierda.....borrar carácter
Tecla DEL

Ambas teclas tienen el mismo efecto. El carácter bajo la posición actual del cursor se borrará. Todos los siguientes caracteres hasta el próximo signo final del párrafo se desplazarán en una posición, es decir, que se moverán hacia la izquierda.

```
2190 REM delete one letter
2200 IF ASC(tecla$) <> 250 AND ASC(tecla$) <> 127 THEN GOTO
1310
2210 FOR t=store TO histore
2220 POKE t,PEEK(t+1): IF PEEK(t)=13 THEN POKE t+1,32 :GOTO
2240
2230 NEXT t: POKE t,32
2240 anchura=rmargen-lmargen+1: ayuda=(linea-1)*anchura
```

La siguiente parte del programa se saltará por las cuatro rutinas anteriores, ya que después de cada desplazamiento deberá entrarse de nuevo la memoria de la pantalla. Este subprograma únicamente creará de nuevo la pantalla a partir de la línea actual del cursor.

Por este motivo se aconseja realizar intercalaciones o bien borrados en el margen inferior de la pantalla, ya que de este modo se crean menos líneas de pantalla.

```
2250 REM emisión de los campos parciales de la pantalla
2260 ayuday=y
2270 FOR t=screen TO 18
2280 LOCATE lmargen,ayuday
2290 FOR i=lostore+ayuda TO lostore+ayuda+anchura-1
2300 carácter$=CHR$(PEEK(i))
2310 IF carácter$=CHR$(13) THEN carácter$=CHR$(242)
2320 PRINT carácter$;: NEXT i
2330 PRINT: ayuda=ayuda+anchura
2340 ayuday=ayuday+1: NEXT t
2350 GOTO 1490
```

LAS FUNCIONES Y SUS COMETIDOS

Aquí vienen las partes del programa que Vd. podrá llamar mediante la tecla de control. Contrariamente al programa dirigido con un menú, no necesitará Vd. abandonar aquí la modalidad de escritura en la que se encontrará siempre el programa.

Algunas de estas funciones parecen ser desde su estructura un tanto complicadas, pero si Vd. emite con toda tranquilidad una función podrá comprobar que todo ésto no es tan difícil.

CTRL y la tecla X.....finalizar programa

Esta función sirve para abandonar el programa de un modo correcto. El programa le preguntará todavía, si Vd. desea realmente abandonar el "TEXT-ME". Conteste Vd. esta pregunta de seguridad mediante "n" para "no", y entonces se volverá a encontrar Vd. en la modalodad normal de escritura del programa.

```
2360 REM check para códigos crtl
2370 IF ASC(tecla$) <> 24 THEN GOTO 2410 :REM ctrl código x
para abandono
2380 LOCATE 25,5 :PRINT"   TEXT-ME abandono (s/n)    "
2390 a$=INKE$: IF a$="n" THEN LOCATE 25,5: PRINT STRING$(30
,"*"): GOTO 1490
2400 IF a$<>"s" THEN GOTO 2390 ELSE CLS : END
```

CTRL y la tecla O.....posicionar el margen superior del formulario

Esta función sirve para modificar el margen superior del formulario. Al emitirse sobre pantalla el margen superior no tiene importancia alguna. Será interesante cuando Vd. plasme sobre papel los textos que haya entrado, ya que el mismo y partiendo del margen superior de la hoja, determinará la línea a partir de la cual deberá efectuarse la impresión.

```
2410 IF ASC(tecla$) <> 15 THEN GOTO 2460 : REM ctrl código
para el margen superior
2420 LOCATE 20,5 :PRINT" es el margen superior :";omargen;:
INPUT" Valor nuevo :",omargen$
2430 IF omargen$="" OR VAL(omargen$)<1 OR VAL(omargen$)>=
umargen THEN GOTO 2450
2440 omargen=VAL(omargen$)
2450 LOCATE 20,5: PRINT STRING$(50,"*"): GOTO 1490
```

CTRL y la tecla U.....posicionar margen inferior del formulario

Correspondiendo a la función para posicionar el margen superior de la hoja, podrá determinar aquí la última línea de una hoja sobre la que todavía deberá imprimirse algún texto. Según las condiciones del programa, Vd. no podrá introducir aquí valores que sean menores que el margen superior o bien mayores que la longitud de la hoja. Ambas funciones podrán ser abandonadas con valores no modificados activando únicamente la tecla ENTER

```
2460 IF ASC(tecla$) <> 21 THEN GOTO 2510 : REM ctrl código
para el margen inferior
2470 LOCATE 20,5: PRINT"Es margen inferior:";umargen;:INPUT
"Valor nuevo :",umargen$
2480 IF umargen$="" OR VAL(umargen$)>paplen OR VAL(umargen$
)<=omargen THEN GOTO 2500
2490 umargen=VAL(umargen$)
2500 GOTO 2450
```

CTRL y la tecla P.....fijar el número de líneas del papel

Con esta función podrá modificar el número máximo posible de líneas impresas que el papel que Vd. utiliza pueda admitir. Si por ejemplo Vd. desea imprimir etiquetas, la longitud de hoja usual de las etiquetas es de 9 líneas (8 líneas a imprimir y una línea de intermedio). La previsión normal de 72 líneas corresponde al formato estándar del papel de los ordenadores. También esta función puede dejarse sin modificación alguna mediante la tecla ENTER.

```
2510 IF  ASC(tecla$) <> 16 THEN GOTO 2550 : REM ctrl código
para la longitud de la hoja
2520 LOCATE 20,5 :PRINT" La antigua longitud de  la hoja es
:";paplen;:INPUT" Valor nuevo:",paplen$
2530 IF paplen$="" OR VAL(paplen$)<umargen THEN GOTO 2450
2540 paplen=VAL(paplen$) : GOTO 2450
```

CTRL y la tecla S.....almacenado de un texto

La rutina a partir de la línea 2550 sirve para almacenar el texto creado. El programa le exigirá indicar un nombre para su texto que al mismo tiempo será almacenado con dicho nombre sobre Diskette o bien Cassette. Todos los avisos del sistema que aparezcan serán automáticamente contestados mediante el signo de exclamación antepuesto al nombre del fichero.

```
2550 IF  ASC(tecla$) <> 19 THEN GOTO 2610 : REM ctrl código
para el almacenado
2560 LOCATE 20,5: INPUT"Almacenar texto !  Nombre del texto
:",nombre del texto$
2570 IF LEN(nombre del texto$)<1 OR LEN(nombre del texto$)
>8 THEN LOCATE 47,5  :PRINT"***************" : GOTO 2560
2580 longitud=histore-lostore
2590 SAVE"!"+nombre del texto$+".txt",b,lostore,longitud
2600 GOTO 2450
```

CTRL y la tecla E.....leer un texto

Esta rutina representa la pieza opuesta de la parte del programa anterior, ya que con la misma pueden leerse nuevamente textos creados además de poder continuar con su proceso. Aquí también le pedirá el programa el nombre del fichero bajo el cual está archivado el texto a ser cargado. Después del proceso de carga, durante el cual se habrán destruido todos los demás textos que se encuentren en memoria, la pantalla será nuevamente conformada.

```
2610 IF ASC(tecla$) <> 5 THEN GOTO 2660  : REM  ctrl código
para cargar
2620 LOCATE 20,5: INPUT"Leer texto ! Nombre de texto:",
nombre del texto$
2630 IF LEN(nombre del texto$)<1 OR LEN(nombre del texto$)
>8 THEN LOCATE 47,5: PRINT"***************" : GOTO 2450
2640 LOAD"!"+nombre del texto$+".txt"
2650 GOTO 3000
```

CTRL y tecla N.....imprimir los números de las páginas

En esta parte del programa podrá concretar, si en el texto deberá imprimirse algún número de página. Después de haber llamado esta rutina podrá conmutar el interruptor para la numeración de páginas presionando cualquier tecla de letras. Esta parte del programa podrá ser abandonada con la tecla ENTER.

El número de páginas empieza en la impresión con el número 1 y se encuentra centrado bajo la última línea del texto. Una modificación del ajuste podrá realizarse en la parte del programa "imprimir".

```
2660 IF  ASC(tecla$) <> 14 THEN GOTO 2720 : REM ctrl código
para indicación de página
2670 LOCATE 30,5: PRINT" Número de página =  ";:IF papnum=0
THEN PRINT"de " ELSE PRINT"en "
2680 a$=INKEY$: IF a$="" THEN GOTO 2680
2690 IF a$=CHR$(13) THEN GOTO 2450
2700 papnum=papnum+1: IF papnum>1 THEN papnum=0
2710 GOTO 2670
```

CTRL y la tecla D.....impresión de un texto

La función D de control indica a su CPC la emisión sobre la impresora del texto que se encuentra en la memoria. La anchura del formulario corresponde al formato actual de la pantalla. Esto tiene la ventaja que durante la creación del texto pueda ver como aparecerá después su texto sobre papel. El subprograma que habrá saltado en esta rutina se encargará de empujar la hoja hasta el margen superior donde se habrá definido la primera línea de impresión.

En la línea 2880 podrá comprobar si la indicación de los números de página está activada y si la línea, en la que deberá imprimirse el número de página, se ha alcanzado. El número de páginas se encuentra archivado en la variable pagenum; para cada llamada que se haga en esta rutina dicha variable volverá a fijarse sobre el "1" (línea 2860)

```
2720 IF ASC(tecla$) <> 4 THEN GOTO 2920 : REM imprimir ctrl
d
2730 GOSUB 3670
2740 anchura=rmargen-lmargen
2750 FOR indice1 = lostore TO histore STEP anchura+1
2760 dlinea$ = STRING$(lmargen-1)," ")
2770 FOR indice2 = 0 TO anchura
2780 dlinea$ = dlinea$  + CHR$(PEEK(indice1+indice2)) :NEXT
indice2
2790 PRINT #8,dlinea$ : dlinea = dlinea+1
2800 IF dlinea = umargen THEN GOSUB 2850: GOSUB 3670
2810 NEXT indice1
2820 FOR t=dlinea+1 TO umargen : PRINT #8,CHR$(13) : NEXT t
2830 GOSUB 2850
2840 GOTO 2450
```

```
2850 REM avance de pié de página
2860 pagenum=1
2870 FOR t=umargen+1 TO paplen: d$=CHR$(13)
2880 IF papnum=1 AND t=umargen+1 THEN d$=STRING$(lmargen-2
+INT((rmargen-lmargen)/2)," ")+STR$(pagenum): pagenum=page
num+1
2890 PRINT #8,d$
2900 NEXT t
2910 RETURN
2920 IF  ASC(tecla$) <>  18 THEN  GOTO 2960 : REM ctrl para
fijar el margen derecho
```

CTRL y la tecla R.....fijar el margen derecho del formulario

Con esta función podrá Vd. fijar el margen derecho de su formulario. El formulario, y de acuerdo con la graduación que le habrá dado, aparecerá de inmediato sobre la pantalla. También esta rutina podrá ser abandonada sin modificación alguna a través de la tecla ENTER.

```
2930 LOCATE 20,5:PRINT" Antiguo margen derecho:";rmargen;:
INPUT"Valor nuevo :",rmargen$
2940 IF rmargen$="" OR VAL(rmargen$)<lmargen OR VAL(rmarge
n$)>79THEN GOTO 2450
2950 rmargen=VAL(rmargen$):GOTO 3000
2960 IF  ASC(tecla$) <>12  THEN GOTO  3010 :  REM ctrl para
fijar margen izquierdo
```

CTRL y la tecla L.....fijar el margen izquierdo del formulario

Esta función sirve, siendo equivalente a la de la graduación del margen derecho, para la modificación del margen izquierdo del formulario.

```
2970 LOCATE 20,5:PRINT" Antiguo margen izquierdo :";lmarge
n;:INPUT"Valor nuevo :",lmargen$
2980 IF lmargen$="" OR VAL(lmargen$)<1 OR VAL(lmargen$)>=r
margen THEN GOTO 2450
2990 lmargen=VAL(lmargen$)
3000 GOSUB 3590:LOCATE 20,5:PRINT STRING$(40,"*"):x=lmarge
n:y=7:GOSUB 3310:GOTO 1490
```

CTRL y la tecla H.....indicar textos auxiliares

Llamando la función H de control obtendrá sobre pantalla un listado de todas las funciones importantes de CTRL. El programa regresará a la modalidad de escritura, tan pronto haya Vd. activado cualquier tecla.

Los textos auxiliares se encuentran almacenados en un subprograma a partir de la línea 3850.

```
3010 IF  ASC(tecla$) <>  8 THEN GOTO 3060 : REM ctrl código
para texto de ayuda
3020 WINDOW #4,lmargen,rmargen,7,24
3030 CLS #4
3040 GOSUB 3820
3050 GOTO 3000
```

CTRL y la tecla C.....borrado de la memoria de textos

Mediante esta función podrá Vd. borrar los textos que se encuentran en la memoria. Ya que puede ser enojoso que ésto ocurra involuntariamente, el programa planteará primero una pregunta de seguridad para saber si Vd. desea realmente borrar la memoria.

```
3060 IF ASC(tecla$) <> 3 THEN GOTO 3120  : REM  ctrl código
para clear store
3070 LOCATE  20,5:PRINT"Borrar texto ! Está Vd. seguro ? (s
/n)"
3080 a$ = INKEY$: IF a$="" THEN GOTO 3080
3090 IF a$="n" THEN LOCATE 20,5: PRINT STRING$(40,"*"):GOTO
 1490
3100 IF a$<>"s" THEN GOTO 3080
3110 storelen=histore-lostore: GOTO 1110
```

CTRL y la tecla l.....Indicar índice

Esta función dispone sobre la pantalla una ventana, en la que se emitirá el índice de un Diskette. Un salto de retroceso a la modalidad de escritura será posible activando cualquier tecla.

```
3120 IF ASC(tecla$) <> 9 THEN GOTO 3200: REM ctrl código
para emitir el contenido
3130 WINDOW #4,1,80,7,24
3140 CLS #4
3150 CAT
3160 PRINT #4,"Pulsar una tecla por favor !"
3170 a$=INKEY$: IF a$="" THEN GOTO 3170
3180 CLS #4:WINDOW 1,80,1,25
3190 GOTO 3000
```

CTRL y la tecla A.....saltar al principio de la ventana

A través de esta rutina podrá Vd. saltar directamente al ángulo superior izquierdo de la ventana actual de la pantalla, sin que Vd. tenga que pasar con el cursor por encima de la misma.

```
3200 IF ASC (tecla$) <> 1 THEN GOTO 1310 : REM ctrl código
para ir al principio
3210 carácter$=CHR$(PEEK(store))
3220 IF carácter$=CHR$(13) THEN carácter$=CHR$(242)
3230 PRINT carácter$;
3240 IF screen=1 THEN GOTO 3280
3250 screen=screen-1: store=store-(rmargen-lmargen+1)
3260 línea=línea-1: y=y-1
3270 GOTO 3240
3280 IF columna=1 THEN GOTO 1490
3290 columna=columna-1: store=store-1
3300 x=x-1 : GOTO 3280
```

SUBPROGRAMAS

En las siguientes páginas encontrará Vd. todos los subprogramas que son necesarios para hacer funcionar el programa a pleno rendimiento. Todos los subprogramas tienen muchos REMarks. Así pues no tendrá Vd. muchas dificultades de poder llegar a comprender el sentido y la finalidad de las distintas rutinas.

Es muy posible que alguna de las funciones tenga algo que desear o también pueda ser mejorada. Para los primeros intentos de ampliación de un programa ya se ofrecen funciones sencillas de control que puedan ser incorporadas en el programa.

Para todo aquel que quiera iniciarse en el lenguaje máquina se le recomienda convertir pequeñas rutinas (por ejemplo la rutina de borrado de la memoria de texto) en lenguaje máquina, ya que este trabajo no es especialmente difícil pero sí ofrece enormes ventajas en cuanto a su rapidez.

```
3310 REM fijar cursor
3320 LOCATE x,y : RETURN
3330 REM emitir línea de estrellitas
3340 PRINT STRING$(80,"*"); : RETURN
3350 REM emitir margen de estrellitas
3360 LOCATE 1,y :PRINT"*"; : LOCATE 80,y :PRINT"*"; :RETURN

3370 REM indicar anchura del formulario
3380 LOCATE 1,6: PRINT STRING$(80," ")
3390 FOR a=lmargen TO lmargen+((rmargen-lmargen)/2) :LOCATE
a,6:PRI NT"<";:NEXT a
3400 FOR a=lmargen+((rmargen-lmargen)/2) TO rmargen :LOCATE
a,6:PRIN
T">";:NEXT a
3410 RETURN
```

```
3420 REM indicación de línea y columna
3430 LOCATE 44,3: PRINT línea
3440 LOCATE 58,3: PRINT columna
3450 LOCATE 71,3: PRINT INT(línea/(umargen-omargen))+1
3460 RETURN
3470 REM leer caracteres y emitir
3480 caracteres = PEEK(store)
3490 IF caracteres = 13 THEN caracteres = 242
3500 PRINT CHR$(24);:REM conectar revers
3510 PRINT  CHR$(caracteres) ;:REM  simular caracteres bajo
el cursor
3520 PRINT CHR$(24); : REM desconectar revers
3530 PRINT CHR$(8) ; : REM función backspace
3540 RETURN
3550 REM preparar texto para la función peek
3560 tecla$ = CHR$(PEEK(store))
3570 IF tecla$=CHR$(13) THEN tecla$=CHR$(242)
3580 RETURN
3590 REM crear una página de pantalla según ctrl r, ctrl l
y leer
3600 WINDOW #2,1,80,6,24: CLS #2: WINDOW 1,80,1,24
3610 GOSUB 3370
3620 FOR índice1 = loline TO loline+17
3630 LOCATE lmargen,indice1+6: FOR indice2=lostore+(índice
1-1)*(rmargen-lmargen+1) TO lostore+(índice1*(rmargen-lmar
gen+1)-1)
3640 caracteres$=CHR$(PEEK(índice2)): IF caracteres$=CHR$
(13)THEN caracteres$=CHR$(242)
3650 PRINT caracteres$;: NEXT índice2 : NEXT índice1
3660  linea=1  :  columna=1  :  screen=1  :  store= lostore
:RETURN
3670 dlínea=0:FOR t=1 TO omargen: PRINT #8,CHR$(13):dlínea
=dlínea+1: NEXT t: RETURN
3680 REM rutina scroll para bajar1
3690 CALL 32701
3700 y=y-1 : LOCATE lmargen,24:FOR t=lostore+(línea-1)*(rm
argen-lmargen+1) TO lostore+(línea*(rmargen-lmargen+1)-1)
3710 caracteres$=CHR$(PEEK(t))
3720 IF caracteres$=CHR$(13)THEN caracteres$=CHR$(242)
3730 PRINT caracteres$; : NEXT t
3740 GOTO 1490
```

```
3750 REM rutina scroll para subir
3760 CALL 32721
3770 y=y+1 : LOCATE lmargen,7: FOR t=lostore+(linea-1)*(rm
argen-lmargen+1) TO lostore+(linea*(rmargen-lmargen+1)-1)
3780 caracteres$=CHR$(PEEK(t))
3790 IF caracteres$=CHR$(13)THEN caracteres$=CHR$(242)
3800 PRINT caracteres$; : NEXT t
3810 GOTO 1490
3820 REM emitir textos auxiliares
3830 PRINT  #4,"* * *  Texto auxiliar para las funciones de
CTRL  * * *"
3840 PRINT #4
3850 PRINT #4,"CTRL-E = Cargar un texto"
3860 PRINT #4,"CTRL-S = Almacenar un texto"
3870 PRINT #4,"CTRL-D = Imprimir un texto"
3880 PRINT #4,"CTRL-R = Fijar el margen derecho"
3890 PRINT #4,"CTRL-L = Fijar el margen izquierdo"
3900 PRINT #4,"CTRL-O = Fijar el margen superior"
3910 PRINT #4,"CTRL-U = Fijar el margen inferior"
3920 PRINT #4,"CTRL-P = Fijar la longitud de página"
3930  PRINT  #4,"CTRL-N  =  Emisión  del  número  de página
on/off"
3940 PRINT #4,"CTRL-A = Vaya al principio de texto"
3950 PRINT #4,"CTRL-C = Borrar la memoria"
3960 PRINT #4,"CTRL-X = Finalizar el programa"
3970 PRINT #4,"CTRL-I = Indicar el índice"
3980 PRINT #4
3990 PRINT #4,"Pulsar una tecla por favor !"
4000 a$=INKEY$: IF a$="" THEN GOTO 4000
4010 CLS #4: WINDOW 1,80,1,25
4020 RETURN
```

```
4030 REM caracteres alemanes
4040 SYMBOL AFTER 90
4050 SYMBOL91,&X1011010,&X111100,&X1100110,&X1100110,&X1111
110,&X1100110,&X1100110,&X0
4060 SYMBOL92,&X10111010,&X1101100,&X11000110,&X11000110,&X
11000110,&X1101100,&X111000,&X0
4070 SYMBOL93,&X1100110,&X0,&X1100110,&X1100110,&X1100110,&
X1100110,&X111100,&X0
4080 SYMBOL123,&X1001000,&X0,&X1111000,&X1100,&X1111100,&X1
1001100,&X1110110,&X0
4090 SYMBOL124,&X100100,&X0,&X111100,&X1100110,&X1100110,&X
1100110,&X111100,&X0
4100 SYMBOL125,&X1000100,&X0,&X1100110,&X1100110,&X1100110,
&X1100110,&X111110,&X0
4110 SYMBOL126,&X111000,&X1101100,&X1101100,&X1101100,&X110
0110,&X1110110,&X1101100,&X11000000
4120 KEY DEF 22,1,124,92
4130 KEY DEF 19,1,125,93
4140 KEY DEF 17,1,123,91
4150 KEY DEF 26,1,126,96
4160 RETURN
4170 REM Rutina de error
4180 LOCATE 15,6:PRINT"  Número  de  error  :";ERR;"  en la
linea :";E RL
4190 a$=INKEY$: IF a$<>CHR$(13) THEN GOTO 4190
4200 LOCATE 20,6:PRINT STRING$(40,"*")
4210 RESUME
4220 DATA &6,&1,&3e,&0,&21,&6,&0,&11,23,79,&cd,&50,&bc,&c9,
&0
4230 DATA &6,&0,&3e,&0,&21,&6,&0,&11,23,79,&cd,&50,&bc,&c9,
&0
```

7.4 C A Z A L A B O M B A

Estando seguros que no sólo habrá trabajado fuertemente con su CPC, hemos pensado facilitarle a través de este libro uno o dos pequeños juegos. Nuestro primer juego se llama "Caza la bomba" que por su brevedad seguramente le parecerá a primera vista no tener problema alguno; de todos modos no es tan fácil de jugar con el mismo como así aparenta.

Aquí se trata de cazar con un "vagón" en el margen inferior de la pantalla las bombas que caen. Si Vd. caza la bomba obtendrá un punto; si la bomba cae al suelo se le descontará un punto. La velocidad de caída y el número de bombas podrán ser determinados por Vd. al principio del juego.

El "vagón" será movido con las dos teclas del cursor hacia la derecha e izquierda.

Al copiar el programa es importante que en la línea 180 se entren dos posiciones en blanco después de la instrucción PRINT.

La velocidad inicial más idónea es de 0.5. Con esta velocidad no tendrá Vd. dificultades en poder cazar las bombas. Sin embargo, si le gusta el peligro podría elegir una velocidad mayor al 2.

Evite abandonar el programa mediante "ESC", ya que hemos disminuido la función de repetición de las teclas y por ello podría tener dificultades al entrar una instrucción. Presionando brevemente las teclas, los caracteres aparecerán varias veces sobre la pantalla.

Para todos aquellos lectores que sean valientes les brindamos aquí un buen consejo: Modifique la línea 140 por:

```
140 D=1:ANF=INT(RND(1)*30)
```

A continuación modifique también la línea 240 por:

```
240 IF KO >30 THEN KO=30
```

Después de estas modificaciones las bombas volarán sobre una anchura horizontal más amplia, siendo así mucho más difícil de cazarlas.

Evidentemente podrá modificarse el programa en otras posiciones. Por ejemplo podría mejorarse la gráfica del fondo o bien instalarse sonido e incluso desarrollar caracteres para aquellas bombas que han estallado. Dichos caracteres podrían moverse por la pantalla mediante la instrucción LOCATE en el caso de que las bombas no fuesen cazadas.

```
10 REM caza la bomba
20 MODE 1
30 SYMBOL AFTER 57
40 SYMBOL 58,128,192,192,255,255,201,28,8
50 SYMBOL 59,1,3,3,255,255,147,58,16
60 SYMBOL 60,4,8,28,62,127,127,62,28
70 SYMBOL 61,255,227,255,255,255,255,255,255
80 INPUT"Velocidad de caida";ca
90 INPUT"Número de bombas";bo
100 SPEED KEY 1,1
110 CLS
120 ko=10
130 FOR bombas=1 TO bo
140 CLS
150 LOCATE 1,22:PRINT STRING$(&28,"=")
160 d=1:anf=INT(RND(1)*20)+1
170 LOCATE anf,d:PRINT" "
180 LOCATE ko,21:PRINT":;"
190 LOCATE anf,d+fa:PRINT"<"
200 LOCATE ko,21:PRINT"  "
210 d=d+ca
220 IF INT(d)>=20 THEN GOTO 280
230 in$=INKEY$
240 ko=ko+(1 AND in$=CHR$(243))-(1 AND in$=CHR(242))
250 IF ko<1 THEN ko=1
260 IF ko>=25 THEN ko=25
270 GOTO 170
280 IF ko=anf OR anf=ko+1 THEN tr=tr+1
290 IF ko<>anf AND ko<>anf-1 THEN tr=tr-1
300 NEXT
310 CLS
320 SPEED KEY 20,3
330 LOCATE 15,10:PRINT"Puntos:",tr
340 LOCATE 10,20:PRINT"Otra vez ?"
350 INPUT a$
360 IF a$="s" THEN RUN
370 SPEED KEY 20,3
```

7.5 HUNDIR BARCOS

A continuación encontrará el segundo juego que le hemos preparado.

No se deje asustar por el nombre; aquí no se trata de la versión 126 del conocido juego.

En nuestro "HUNDIR BARCOS" el sentido del juego es muy distinto.

En la primera parte del programa deberá introducir algunos parámetros.

En primer lugar vayamos al grado de dificultad. Este se encuentra en el campo entre 0-2. En el grado de dificultad hemos aprovechado una de las propiedades del CPC, o sea las distintas modalidades de caracteres (20-40-80). En el grado 0 hemos trabajado con la gama de caracteres 20. Aquí la gráfica se ve mucho mejor, ya que la dilatación x de un carácter es relativamente mayor. Evidentemente aquí el grado de dificultad es menor. Los demás grados de dificultad son asimismo análogos a las modalidades de los caracteres.

El segundo parámetro a ser entrado es la unidad de tiempo. Cuanto más pequeña se elija, tanto más corto será el tiempo que Vd. disponga para realizar su misión. Nosotros partimos de un valor informativo; partir de una unidad es menos posible.

El tercer parámetro es la velocidad de las bombas. Con la ayuda de este valor podrá Vd. regular la velocidad con la que se moverá la bomba por su pantalla.

La última introducción es el número de bombas con las que Vd. desea jugar.

Hasta aquí le hemos tenido ya tanto en vilo que ahora ya podremos pasar lentamente a aclararle el propio juego.

Si ya ha entrado los valores descritos anteriormente, le aparecerá el cuadro real del juego. En el mismo verá un gran barco, con un puente y un cañón en la proa; además tendrá un submarino y una bomba al lado del mismo. Su misión será guiar la bomba con la ayuda de la tecla del cursor sobre la proa del barco y así destruirlo. Para ello recibirá Vd. puntos; si Vd. no hace blanco sobre el barco después del tiempo fijado por Vd. mismo, el programa pasará a enseñarle el próximo cuadro de juego.

Al final del juego obtendrá Vd. una puntuación por su habilidad. En dicha puntuación se concretará el número de aciertos, el tiempo que tuvo a disposición, la modalidad y la velocidad de las bombas.

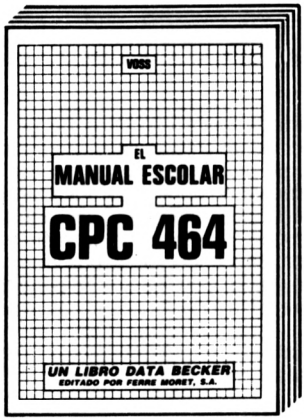
Ciertamente podrá Vd. equipar este juego según sus deseos y ello podría ser añadiendo color y tono. Esto no debería presentarle dificultad alguna si vuelve a leer los capítulos 2 y 3 de este libro.

```
10 MODE 2
15 impacto=0
20 SYMBOL AFTER 32
30 LOCATE 20,10:PRINT"H U N D I R   B A R C O S"
40 GOSUB 350
50 LOCATE 1,15: INPUT"QUE GRADO DE DIFICULTAD (0-2)";A
60  INPUT"CUANTAS UNIDADES DE TIEMPO PARA DIRIGIR LA
BOMBA";B
70 INPUT"VELOCIDAD DE LA BOMBA";VELOC
80 INPUT"CUANTAS BOMBAS";BOMBAS
85 DIM BO1(BOMBAS)
90 MODE A
95 FOR BB=1 TO BOMBAS
96 CLS
100 IF A=0 THEN F=18
110 IF A=1 THEN F=38
120 IF A=2 THEN F=78
130 YP=INT(RND(1)*24)+2
140 XP=INT(RND(1)*F)+1
150 YP2=INT(RND(1)*25)+1
160 XP2=INT(RND(1)*F)+1
170 IF XP=XP2 OR YP=YP2 THEN GOTO 130
171 XP3=XP2+2:YP3=YP2
172 B1=B*100
175 SPEED KEY 1,VELOC
176 FOR MNN=1 TO B1
180 LOCATE XP+2,YP-1:PRINT"!"
190 LOCATE XP,YP:PRINT"&%#"
200 LOCATE XP2,YP2:PRINT"üÄ"
220 LOCATE XP3,YP3:PRINT" "
230 IN$=INKEY$
240 XP3=XP3+(1 AND IN$=CHR$(243))-(1 AND IN$=CHR$(242))
250 YP3=YP3+(1 AND IN$=CHR$(241))-(1 AND IN$=CHR$(240))
260 IF XP3<1 THEN XP3=1
270 IF XP3>F+2 THEN XP3+F+2
280 IF YP3<1 THEN YP3=1
290 IF YP3>25 THEN YP3=25
300 IF YP3=25 AND XP3=F+2 THEN XP3=F+1
310 LOCATE XP3,YP3:PRINT"^"
320 IF XP3=XP AND YP3=YP THEN GOTO 440
330 NEXT MNN
```

```
335 NEXT BB
340 GOTO 3000
350 REM DEFINIR CARACTERES
360 SYMBOL 33,62,8,28,28,124,124,124,252
370 SYMBOL 35,252,252,252,255,254,252,248,240
380 SYMBOL 37,1,129,129,255,255,255,255,255
390 SYMBOL 38,126,15,15,127,63,15,3,1
400 SYMBOL 91,128,192,252,254,255,255,254,0
410 SYMBOL 93,0,1,31,63,255,255,63,0
420 SYMBOL 94,0,0,156,190,127,190,156,0
430 RETURN
440 MODE 0
445 IMPACTO=IMPACTO+1
446 LOCATE 3,12
447 BO1(BB)=((B1-MNN)/B1)*10
448 FOR FOR ZEI=1 TO 15
449 PRINT CHR$(224);
450 NEXT
451 LOCATE 3,13:PRINT CHR$(244);"I M P A C T O";CHR$(244)
452 LOCATE 3,14
453 FOR ZEI=1 TO 15
454 PRINT CHR$(244);
455 NEXT
460 SOUND 129,450,150,15,,,15
465 FOR XX=1 TO 3000
466 NEXT XX
467 MODE A
470 GOTO 335
3000 MODE 2
3005 FOR II=1 TO BOMBAS
3006 PUNTOS=PUNTOS+BO1(II)
3007 NEXT
3008 PUNTOS=PUNTOS*IMPACTO*(A+1)
3010 LOCATE 15,9:PRINT"J U E G O  F I N A L I Z A D O"
3020 LOCATE 2,11:PRINT"VD. TIENE EN ";BOMBAS;"
BOMBAS";IMPACTO;"IMPACTOS CONSEGUIDOS."
3021 LOCATE 2,13:PRINT"ESTO ES UNA PUNTUACION DE:";PUNTOS
3025 SPEED KEY 20,3
3030 INPUT"DESEA JUGAR OTRA VEZ";WAHL$
3040 IF WAHL$="S" THEN RUN
3050 END
```

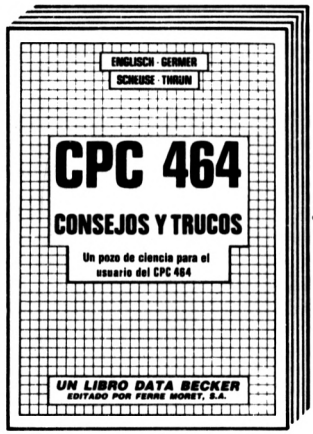
64 EN EL CAMPO DE LA TECNICA Y LA CIENCIA, 296 pág.
P.V.P. 2.800,- ptas.

Ofrece un campo fascinante y amplio de problemáticas científicas. Para esto el libro contiene muchos listados interesantes: Análisis de Fornier y síntesis, análisis de redes, exactitud de cálculo, formateado de números, cálculo del valor PH, sistemas de ecuaciones diferenciales, modelo ladrón presa, cálculo de probabilidad, medición de tiempo, integración, etc.



CPC-464 EL LIBRO DEL COLEGIO
P.V.P. 2.200,- ptas.

Escrito para alumnos de los últimos cursos de EGB y de BUP, este libro contiene muchos programas para resolver problemas y de aprendizaje, descritos de una forma muy compleja y fácil de comprender. Teorema de Pitágoras, progresiones geométricas, escritura cifrada, crecimiento exponencial, verbos irregulares, igualdades cuadráticas, movimiento pendular, estructura de moléculas, cálculo de interés y muchas cosas más.



CPC-464 CONSEJOS Y TRUCOS
P.V.P. 2.200,- ptas.

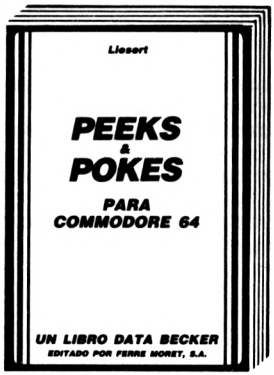
Ofrece una colección muy interesante de sugerencias, ideas y soluciones para la programación y utilización de su CPC-464: Desde la estructura del hardware, sistema de funcionamiento - Tokens Basic, dibujos con el joystick, aplicaciones de ventanas en pantalla y otros muchos interesantes programas como el procesamiento de datos, editor de sonidos, generador de caracteres, monitor de código máquina hasta listados de interesantes juegos.

ROBOTICA PARA SU COMMODORE 64, 230 pág.
P.V.P. 2.800,- ptas.

En el libro de los robots se muestran las asombrosas posibilidades que ofrece el CBM 64, para el control y la programación, presentadas con numerosas ilustraciones e intuitivos ejemplos. El punto principal: Cómo puede construirse uno mismo un robot sin grandes gastos. Además, un resumen del desarrollo histórico del robot y una amplia introducción a los fundamentos cibernéticos.
Gobierno del motor, el modelo de simulación, interruptor de pantalla, el Port-Usuario cómodo del modelo de simulación, Sensor de infrarrojos, concepto básico de un robot, realimentación unidad cibernética, Brazo prensor, Oír y ver.

MANUAL ESCOLAR PARA SU COMMODORE 64, 351 pág.
P.V.P. 2.800,- ptas.

Este libro, escrito especialmente para escolares de grado medio y superior, contiene muchos interesantes programas de aprendizaje para solucionar problemas, descritos detalladamente y de manera fácilmente comprensible. Facilitan un aprendizaje intensivo y ameno, con, entre otros, los siguientes temas: Teorema de pitágoras, progresiones geométricas, palanca mecánica, crecimiento exponencial, verbos irregulares, ecuaciones de segundo grado, movimientos de péndulo, formación de moléculas, aprendizaje de vocablos, cálculo de interés y su capitalización. Una corta repetición de los elementos BASIC más importantes y una introducción a los rasgos esenciales del análisis de problemas, entre otros, completan el conjunto.



PEEKS y POKES, 177 pág.
P.V.P. 1.600,- ptas.

Con importantes comandos PEEK y POKE se pueden hacer también desde el Basic muchas cosas, para las que se necesitarían normalmente complejas rutinas en lenguaje máquina. Este libro explica de manera sencilla el manejo de PEEKs y POKEs. Con una enorme cantidad de POKEs importantes y su posible aplicación. Para ello se explica perfectamente la estructura del Commodore 64: Sistema operativo, interpretador, página cero, apuntadores y stacks, generador de caracteres, registros de sprites, programación de interfaces, desactivación del interrupt. Además una introducción al lenguaje máquina. Muchos programas ejemplo.

TODO SOBRE EL FLOPPY 1541, 482 pág.
P.V.P. 3.200,- ptas.

La obra Standard del floppy 1541, todo sobre la programación en disquettes desde los principiantes a los profesionales, además de las informaciones fundamentales para el DOS, los comandos de sistema y mensajes de error, hay varios capítulos para la administración práctica de ficheros con el FLOPPY, amplio y documentado Listado del Dos. Además un filón de los más diversos programas y rutinas auxiliares, que hacen del libro una lectura obligada para los usuarios del Floppy.

MANTENIMIENTO Y REPARACION DEL FLOPPY 1541,
200 pág.
P.V.P. 2.800,- ptas.

Saberse apañar uno mismo, ahorra tiempo, molestias y dinero, precisamente problemas como el ajuste del floppy o reparaciones de la platina se pueden arreglar a menudo con medios sencillos. Instrucciones para eliminar la mayoría de perturbaciones, listas de piezas de recambio y una introducción a la mecánica y a la electrónica de la unidad de disco, hay también indicaciones exactas sobre herramientas y material de trabajo. Este libro hay que considerarlo en todos sus aspectos como efectivo y barato.

EL MANUAL DEL CASSETTE, 190 pág.
P.V.P. 1.600,- ptas.

Un excelente libro, que le mostrará todas las posibilidades que le ofrece su grabadora de casettes. Describe detalladamente, y de forma comprensible, todo sobre el Datassette y la grabación en cassette. Con verdaderos programas fuera de serie: Autostart, Catálogo (¡busca y carga automáticamente!), backup de y a disco, SAVE de áreas de memoria, y lo más sorprendente: un nuevo sistema operativo de cassette con el 10-20 veces más rápido FastTape. Además otras indicaciones y programas de utilidad (ajuste de cabezales, altavoz de control).

LENGUAJE MAQUINA PARA
COMMODORE 64, 1984, 201 pág.
P.V.P. 2.200,- ptas.

¡Por fin una introducción al código máquina fácilmente comprensible! Estructura y funcionamiento del procesador 6510, introducción y ejecución de programas en lenguaje máquina, manejo del ensamblador, y un atractivo muy especial: ¡un simulador de paso a paso escrito en BASIC!

LENGUAJE MAQUINA PARA AVANZADOS
CBM 64, 1984, 206 pág.
P.V.P. 2.200 ptas.

¿Ud. ha logrado iniciarse en código máquina? Entonces el «nuevo English» le enseñará cómo convertirse en un profesional. Naturalmente con muchos programas ejemplo, rutinas completas en código máquina e importantes consejos y trucos para la programación en lenguaje máquina y para el trabajo con el sistema operativo.

EL ENSAMBLADOR

Este libro ofrece al programador interesado una introducción fácilmente comprensible para los tan extendidos Assembler PROFI-ASS, SM MAE y T.EX.ASS. con consejos y trucos de gran utilidad, indicaciones y programas adicionales. Al mismo tiempo sirve de manual orientado a la práctica, con aclaraciones de conceptos importantes e instrucciones.
250 páginas. 2.200,- ptas.

ZX SPECTRUM EL MANUAL ESCOLAR
P.V.P. 2.200,- ptas.

Escrito para alumnos de los últimos cursos de EGB y de BUP, este libro contiene muchos programas para resolver problemas y de aprendizaje, descritos de una forma muy completa y fácil de comprender. Teorema de Pitágoras, progresiones geométricas, escritura cifrada, crecimiento exponencial, verbos irregulares, igualdades cudráticas, movimiento pendular, estructura de moléculas, cálculo de interés y muchas cosas más.

ZX SPECTRUM CONSEJOS Y TRUCOS, 211 pág.
P.V.P. 2.200,- ptas.

Una interesante colección se sugestivas ideas y soluciones para la programación y utilización de su ZX ESPECTRUM. Aparte de muchos peeks, pokes y USRs hay también capítulos completos para, entre otros, entrada de datos asegurado sin bloqueo de ordenador, posibilidades de conexión y utilización de microdrives y lápices ópticos, programas para la representación de diagramas de barra y de tarta, el modo de utilizar óptimamente ROM y RAM.

METODOLOGIA DE LA PROGRAMACION
P.V.P. 2.200,- ptas.

El primer libro recomendado para escuelas de enseñanza de informática y para aquellas personas que quieren aprender la programación. Cubre las especificaciones del Ministerio de Educación y Ciencia para Estudios de Informática. Realizado por un alto mando del ejército Español, un Dr. Ingeniero y Diplomado en Informática y profesor de la UNED y por un oficial técnico especialista en informática de gestión. Utilizado en todos los institutos politécnicos del ejército español. Es un seguro candidado a ediciones en lengua inglesa, alemana y francesa. Es el primer libro que introduce a la lógica del ordenador. Es un elemento de base que sirve como introducción para la programación en cualquier otro lenguaje. No se requieren conocimientos de programación ni siquiera de informática. Abarca desde los métodos de programación clásicos a los más modernos.

MSX EL MANUAL ESCOLAR
P.V.P. 2.800,- ptas.

Escrito para alumnos de los últimos cursos de EGB y de BUP, este libro contiene muchos programas para resolver problemas y de aprendizaje, descritos de una forma muy completa y fácil de comprender. Teorema de Pitágoras, progresiones geométricas, escritura cifrada, crecimiento exponencial, verbos irregulares, igualdades cuadráticas, movimiento pendular, estructura de moléculas, cálculo de interés y muchas cosas más.

MSX GRAFICOS Y SONIDOS, 250 pág.
P.V.P. 2.800,- ptas.

Las computadoras MSX no sólo ofrecen una relación precio/rendimiento sobresaliente, sino que también poseen unas cualidades gráficas y de sonido excepcionales. Este libro expone las posibilidades de los MSX de forma completa y fácil. El texto se completa con numerosos y útiles programas ejemplo.

MSX PROGRAMAS Y UTILIDADES, 1985, 194 pág.
P.V.P. 2.200,- ptas.

El libro contiene una amplia colección de importantes programas que abarcan, desde un desensamblador hasta un programa de clasificaciones deportivas. Juegos superemocionantes y aplicaciones completas. Los programas muestran además importantes consejos y trucos para la programación. Estos programas funcionan en todos los ordenadores MSX, así como en el SPECTROVIDEO 318 328. ESTRACTO DEL CONTENIDO: Volcado memoria hexadecimal. Editor gráficos. Editor de sonido. Escritura de ordenador. Lista referencia de variables. Calendario. Desensamblador. ADMINISTRACION de una colección de discos L.P. HOLLOW - JUEGO DE LAS CEREZAS. DIAGRAMAS DE BARRAS. TABLAS DEPORTIVAS.

TODO SOBRE EL NUEVO COMMODORE 128

P.V.P. 2.200,- ptas.

El Libro de Primicias del Commodore 128 no ofrece solamente un resumen completo de todas las características y rendimientos del sucesor del C-64 y con ello una importante ayuda para su adquisición. Muestra, además, todas las posibilidades del nuevo equipo en función de sus tres modos de operación.

Entre otros se describen el hardware, los modos de operación: modo 64, modo 128 y modo CP/M, las configuraciones de memoria, la disposición de la página cero, trabajos con dos pantallas, modo de 80 caracteres, Basic V 7.0: comandos de gráficos y sonidos, comandos de control, periféricos rápidos (1571) etc.

RESPUESTA COMERCIAL

F.D. Autorización 6975
(B.O. de Correos N.º 80 de 26-7-85)

HOJA PEDIDO DE LIBRERIA

NO NECESITA SELLOS

A franquear en destino

FERRE MORET, S.A.

Apartado N.º 551. F.D.
08080 BARCELONA

RESPUESTA COMERCIAL

F.D. Autorización 6975
(B.O. de Correos N.º 80 de 26-7-85)

HOJA PEDIDO DE LIBRERIA

NO NECESITA SELLOS

A franquear en destino

FERRE MORET, S.A.

Apartado N.º 551. F.D.
08080 BARCELONA

Puesta al día de datos

EDITORIAL FERRER MORET, S.A. mantiene vivo y amplía el contenido informativo de sus libros y programas, mediante el envío de un servicio de puesta al día, junto con una síntesis noticiosa de la actualidad y perspectivas de la realidad informática española.
Agradecemos cualquier sugerencia o crítica que desee formular y que nos ayude a mejorar las ediciones. Muchas gracias.

¿Qué añadiría?
.....

.....

¿Qué suprimiría?
.....

.....

Observaciones
.....

.....

Título del libro
.....

Nombre
.....

Dirección Tfno.
.....

Código Postal y Población Provincia
.....

UN SERVICIO GRATUITO

Información

FERRE MORET, S.A. cuenta con un amplio fondo de libros y Software y mantiene un servicio de información por correo sobre las novedades que edita.
Agradecemos nos indique los temas que representan para Vd. mayor interes.

Libros

☐ ATARI	☐ MSX	☐ AMSTRAD
☐ COMMODORE	☐ LENGUAJES	☐ APPLE
☐ SINCLAIR	☐ IBM	☐ SOFTWARE

Si está interesado en recibir alguno de estos servicios, rellene y envíe la tarjeta correspondiente; no necesita franqueo. Muchas gracias.

DESEO RECIBIR EL LIBRO .....

EL PROGRAMA .....

☐ Adjunto cheque ☐ Contra reembolso
.....

Nombre
.....

Dirección Tfno.
.....

Código Postal y Población Provincia
.....

UN SERVICIO GRATUITO

EL CONTENIDO:

El CPC 464 y 6128. Consejos & Trucos, ofrece una interesante colección de sugerencias, ideas y soluciones cumplimentadas para la programación y aplicación de su CPC 464 y 6128.

Extracto del contenido:

- Estructuración del Hardware
- Sistema operativo, Basic-Tokens
- Estructuración de la pantalla
- Aplicaciones de las posibilidades Window
- Un tratamiento completo de ficheros
- Un programa de tratamiento de textos bien documentado
- Editor de sonido
- Generador práctico de caracteres
- Juegos interesantes
- Monitor de lenguaje máquina y muchas otras cosas...

ESTE LIBRO HA SIDO ESCRITO POR:

Lothar English, conocido autor de distintos libros de DATA BECKER (El libro del Lenguaje Máquina para el C-64). Jörg Germer es un estudiante y su hobby es naturalmente el ordenador. Thomas Scheuse es asimismo un estudiante con muchos años de experiencia en ordenadores personales. Frank Thurn es un programador del departamento de Software de DATA BECKER.

ISBN 84-86437-17-2

AMSTRAD CPC

MÉMOIRE ÉCRITE
MEMORY ENGRAVED
MEMORIA ESCRITA

https://acpc.me/