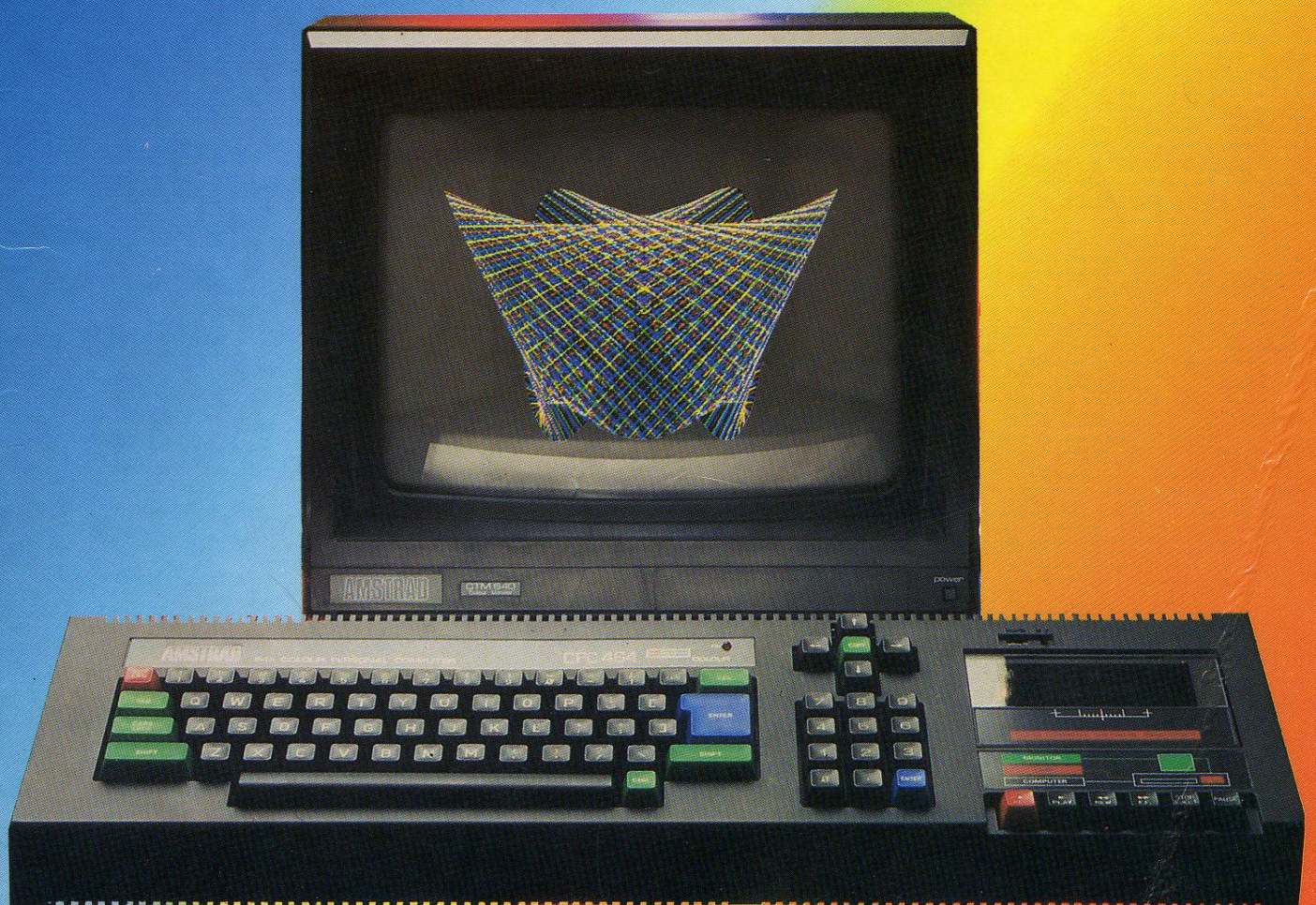


# AMSTRAD CPC 464



## guia del usuario

Escaneado por CPC Maniaco - PDF por Litos.NET



# CPC472 COLOUR PERSONAL COMPUTER 72K

Traducido y Adaptado por ***indescorp***

1ª Edición 1984

# **AMSOF**

A division of

# **AMSTRAD**

**CONSUMER ELECTRONICS PLC**

© Copyright 1984 AMSOFT, AMSTRAD Consumer Electronics plc and Locomotive Software Limited

Neither the whole or any part of the information contained herein, or the product described in this manual may be adapted or reproduced in any material form except with the prior written approval of AMSTRAD Consumer Electronics plc ('AMSTRAD').

The product described in this manual and products for use with it are subject to continuous development and improvement. All information of a technical nature and particulars of the product and its use (including the information and particulars in this manual) are given by AMSTRAD in good faith. However, it is acknowledged that there may be errors or omissions in this manual. A list of details of any amendments or revisions to this manual can be obtained by sending a stamped, self-addressed envelope to AMSOFT Technical Enquiries. We ask that all users take care to submit their reply paid user registration and guarantee cards.

AMSOFT welcome comments and suggestions relating to the product or this manual.

All correspondence should be addressed to:

**AMSOFT**  
**169 Kings Road**  
**Brentwood**  
**Essex CM14 4EF**

All maintenance and service on the product must be carried out by AMSOFT authorised dealers. Neither AMSOFT nor AMSTRAD can accept any liability whatsoever for any loss or damage caused by service or maintenance by unauthorised personnel. This guide is intended only to assist the reader in the use of the product, and therefore AMSOFT and AMSTRAD shall not be liable for any loss or damage whatsoever arising from the use of any information or particulars in, or any error or omission in, this guide or any incorrect use of the product.

Within this manual, the reference Z80 is used with acknowledgement to Zilog Inc.

First Published 1984

Published by AMSTRAD

Typeset by AMSOFT Computer Graphics

AMSTRAD is a registered trademark of AMSTRAD Consumer Electronics plc. Unauthorised use of the trademark or word AMSTRAD is strictly forbidden.

## ADDENDA

### AMSTRAD CPC472

#### INTRODUCCION

AMSTRAD CPC472 es una nueva versión del AMSTRAD CPC464. Este nuevo modelo, no ofrece diferencias relevantes con respecto al anterior, salvo la incorporación de la ROM del modelo AMSTRAD CPC664.

También ha sido necesario ampliar la memoria del equipo con 8Ks más, que han sido requeridos para la implantación de la nueva ROM.

Esta memoria adicional es, por tanto, utilizada y gestionada directamente por el procesador, para suplir las necesidades originadas por la incorporación de los nuevos comandos del Locomotive Basic, de modo que no se ha tenido que recurrir a la memoria disponible, que sigue siendo para el usuario de 64Ks.



## NUEVOS COMANDOS

La incorporación de la ROM trae consigo la disponibilidad de nuevos comandos BASIC.

Estos comandos son:

### CLEAR INPUT

CLEAR INPUT

```
10 CLS
20 PRINT "Teclee letras ahora!"
30 FOR t=1 TO 3000
40 NEXT
50 CLEAR INPUT
run
```

ORDEN: Desecha todos los caracteres tecleados antes y que todavía estén en el tampón del teclado; es decir, borra el tampón del teclado.

Para observar el efecto de esta orden, ejecute el programa ejemplo y escriba letras cuando el ordenador se lo pida. Borre luego la línea 50 y vuelva a ejecutar el programa, fijándose en la diferencia.

Palabras clave asociadas: INKEY, INKEY\$, JOY.

### COPYCHR\$

COPYCHR\$(#(expresion de numero de canal))

```
10 CLS
20 PRINT "Extremo superior izquierdo"
30 LOCATE 1,1
40 a$=COPYCHR$(#0)
50 LOCATE 1,20
60 PRINT a$
run
```

FUNCION: Lee el carácter que hay en la posición actual del cursor; la ventana tiene que ser especificada. El programa ejemplo lee el carácter que está en la posición 1,1 y lo escribe en la posición 1,20.

Si el carácter no es reconocible, esta función genera la cadena vacía.

Palabras clave asociadas: LOCATE.

## CURSOR

**CURSOR** [<conmutador del sistema>][,<conmutador del usuario>]

```
10 CURSOR 1
20 PRINT "pregunta?";
30 a$=INKEY$:IF a$=" " THEN 30
40 PRINT a$
50 CURSOR 0
run
```

**ORDEN:** El cursor puede ser activado o inhibido por dos conmutadores: el del sistema y el del usuario. Esta orden controla los dos conmutadores. Los parámetros pueden ser **0** (inhibido) o **1** (activado). En la instrucción **INKEY\$** el cursor está normalmente oculto, pero en el programa ejemplo ha sido activado por el valor **1** del parámetro <conmutador del sistema> en la línea 10.

Para que el cursor sea visible tienen que estar a **1** los dos conmutadores. El ordenador activa automáticamente el conmutador del sistema para la instrucción **INPUT**, pero lo desactiva para **INKEY\$**.

Es conveniente ocultar el cursor antes de escribir texto en la pantalla.

Se puede omitir cualquiera de los parámetros, pero no los dos. Cuando se omite uno de los parámetros, la situación del correspondiente conmutador no se modifica.

Palabras clave asociadas: **LOCATE**.

## DEC\$

**DEC\$**(<expresion numerica>,<plantilla de formato>)

```
PRINT DEC$(10↑7,"££#####,.##")
£10,000,000.00
```

**FUNCION:** Produce una representación en forma de cadena decimal de la <expresión numérica> utilizando la <plantilla de formato> especificada.

La plantilla de formato sólo puede contener los caracteres siguientes:

+ - £ \$ \* # , . ↑



La utilización de estos 'especificadores de formato' se describe en **PRINT USING**.

Palabras clave asociadas: **BIN\$, HEX\$, PRINT USING, STR\$**.

## DERR

DERR

```
LOAD "xyz.abc"
```

```
XYZ .ABC not found
```

```
Ready
```

```
PRINT DERR
```

```
146
```

**FUNCION:** Toma como valor el último código de error detectado por el sistema de archivo en disco. Se puede utilizar el valor de **DERR** para determinar qué error se ha producido. Véase en el capítulo 'Para su referencia...' la lista de los mensajes de error.

Palabras clave asociadas: **ERL, ERR, ERROR, ON ERROR GOTO, RESUME**.

## FILL

FILL <tinta>

```
10 MODE 1:mar=2:MOVE 0,200,3
20 DRAWR 100,-100:DRAWR 220,0
30 MOVER 10,0:DRAWR 220,0
40 DRAWR 100,100:MOVE 0,0
50 FILL mar
run
```

**ORDEN:** Rellena una región de la pantalla gráfica. Los límites de la región son rectas dibujadas con la tinta de la pluma de gráficos actual o con la tinta utilizada para rellenar (margen de 0 a 15).

El rellenado se realiza partiendo de la posición actual del cursor gráfico. Si esta posición está en un límite, no se produce rellenado alguno.

Palabras clave asociadas: **GRAPHICS PEN.**

## **FRAME**

**FRAME**

```
10 MODE 0
20 PRINT "sin FRAME"
30 TAG
40 MOVE 0,200
50 FOR x=0 TO 500 STEP 4
60 IF f=1 THEN FRAME
70 MOVE x,200
80 PRINT " ";CHR$(143);
90 NEXT
100 IF f=1 THEN RUN
110 CLS
120 TAGOFF
130 PRINT "con FRAME"
140 f=1
150 GOTO 30
run
```

**ORDEN:** Sincroniza la escritura de caracteres gráficos en la pantalla con el retorno del haz en el monitor. El efecto global es hacer que el movimiento de símbolos por la pantalla sea más suave, sin parpadeo ni discontinuidad.

Palabras clave asociadas: **TAG, TAGOFF.**

## **GRAPHICS PAPER**

**GRAPHICS PAPER** <tinta>

```
10 MODE 0
20 MASK 15
30 GRAPHICS PAPER 3
40 DRAW 640,0
run
```



**ORDEN:** Especifica la <tinta> del papel de la pantalla de gráficos, es decir, el color del fondo sobre el que se dibujan los gráficos. Cuando se dibujan líneas continuas, el papel gráfico se hace invisible. En el programa ejemplo, la orden **MASK** hace que la recta que se dibuja sea discontinua y que, por lo tanto, el papel gráfico se pueda ver debajo de ella.

La tinta del papel gráfico (margen de 0 a 15) se utiliza también como fondo de los caracteres que se escriben mediante **TAG** y para reteñir la pantalla gráfica cuando no se especifica otra tinta tras la instrucción **CLG**.

Palabras clave asociadas: **CLG, GRAPHICS PEN, INK, MASK, TAG, TAGOFF.**

## GRAPHICS PEN

**GRAPHICS PEN** [<tinta>][, <modo de fondo>]

```
10 MODE 0
20 GRAPHICS PEN 15
30 MOVE 200,0
40 DRAW 200,400
50 MOVE 639,0
60 FILL 15
run
```

**ORDEN:** Especifica la <tinta> con la que en lo sucesivo se van a dibujar rectas y puntos. El <modo de fondo> puede ser:

- 0 Fondo opaco
- 1 Fondo transparente

(El fondo transparente afecta al papel gráfico de los caracteres escritos con **TAG** y a los espacios que quedan entre los trazos de las líneas discontinuas).

Se puede omitir cualquiera de los dos parámetros, pero no ambos. Si se omite un parámetro, la característica controlada por él permanece inalterada.

Palabras clave asociadas: **GRAPHICS PAPER, INK, MASK, TAG, TAGOFF.**

## MASK

**MASK** [<expresion entera>][,<situacion del primer punto>]

```
10 CLS:TAG
20 MASK 1:MOVE 0,250:DRAWR 240,0
30 PRINT "(binario 00000001 en la mascara)";
40 MASK 3:MOVE 0,200:DRAWR 240,0
50 PRINT "(binario 00000011 en la mascara)";
60 MASK 7:MOVE 0,150:DRAWR 240,0
70 PRINT "(binario 00000111 en la mascara)";
80 MASK 15:MOVE 0,100:DRAWR 240,0
90 PRINT "(binario 00001111 en la mascara)";
run
```

**ORDEN:** Define una 'máscara' o plantilla que se puede utilizar en el dibujo de rectas. La <expresión entera> (margen de 0 a 255) determina la situación de los diversos bits de cada grupo de 8 pixels adyacentes: 0 para pixel apagado, 1 para pixel iluminado.

La <situación del primer punto> determina si el primer punto de la recta debe ser iluminado (1) o no (0).

Se puede omitir cualquiera de los dos parámetros, pero no ambos. La situación de la característica correspondiente al parámetro omitido permanece inalterada.

Palabras clave asociadas: **DRAW, DRAWR, GRAPHICS PAPER, GRAPHICS PEN.**

## ON BREAK CONT

**ON BREAK CONT**

```
10 ON BREAK CONT
20 PRINT "El programa CONTinua aunque usted intente
   interrumpirlo con [ESC]";PRINT
30 FOR t=1 TO 1000:NEXT:GOTO 20
run
```



**ORDEN:** Inhibe la posibilidad de interrumpir el programa pulsando la tecla **ESC**, pues intercepta la interrupción y continúa la ejecución. Esta orden se debe utilizar con precaución, pues el programa no podrá ser detenido más que reiniciando la máquina; grabe el programa antes de ejecutarlo.

Dentro de un programa, esta orden se anula con **ON BREAK STOP**.

Palabras clave asociadas: **ON BREAK GOSUB, ON BREAK STOP**.

## **Aclaraciones sobre la Guía del Usuario del CPC464**

La Informática ha recorrido un camino muy largo en un período muy corto de tiempo. De todos los avances tecnológicos del siglo veinte, indudablemente el más impresionante ha correspondido al campo de los ordenadores.

El desarrollo de estas máquinas y de los programas que las controlan ha avanzado mucho más rápidamente de lo que suponen gran parte de los usuarios, por lo que intentar mostrar a quienes poseen un CPC464 todas las posibilidades y toda la capacidad de su sistema operativo BASIC, y de los circuitos que configuran al equipo, requeriría varios miles de páginas.

Así, esta Guía de Usuario es una introducción concisa al CPC464 y a su programación. Será complementada por muchos cursos y publicaciones didácticas más específicos y detallados.

Aquellos usuarios que ya se han familiarizado con otros dialectos de BASIC lograrán -muy fácilmente y en forma rápida- dominar la estructura AMSTRAD BASIC; y los que comiencen con esta potente máquina, apreciarán la naturaleza directa y sin ambigüedades de la terminología empleada que ha sido especialmente pensada para evitar las peculiaridades muchas veces presentes en interpretaciones caprichosas del BASIC. De la misma forma, nos hemos preocupado de introducir distintos aspectos -fundamentales para este lenguaje en el futuro- cuando haya disponibles, computadoras de bajo coste que ofrezcan análogas posibilidades a las que el CPC464 posee ya y ahora.

### **La Guía de Usuario está dividida en tres secciones**

La primera la llamamos "FUNDAMENTOS PARA PRINCIPIANTES", y ha sido escrita principalmente para presentarle los conceptos y la terminología específica -y en constante evolución de los ordenadores. Si previamente no ha tenido un ordenador propio, ni tampoco la posibilidad de usar habitualmente alguno, y si ni siquiera ha podido disfrutar produciendo un pequeño y simple programa, entonces le aconsejamos que inicie la lectura de esta Guía de Usuario por los FUNDAMENTOS que le comentamos en esta primera sección.

Aquéllos que posean experiencia previa en este campo, pueden pasar directamente al Capítulo Primero de la segunda sección de esta Guía de Usuario. No sólo nos hemos preocupado de reiterar los temas esenciales y necesarios para la instalación y puesta en marcha de la máquina; sino que nos hemos concentrado en presentar los rasgos específicos del sistema CPC464 de forma clara y comprensible, y haciendo mínimas hipótesis sobre el grado de experiencia alcanzado por el lector.

Cada uno de los capítulos primordiales ha sido escrito para proporcionar una rápida visión sobre las muchas facilidades existentes en el CPC464. Algunos puntos fundamentales se resaltan y destacan repetidamente, para que así muchos de los nuevos usuarios logren su deseo de "zambullirse" inmediatamente en las distintas facetas de sonidos y gráficos, sin más que una muy breve -pero imprescindible- introducción al teclado y a los aspectos más metódicos del aprendizaje del lenguaje BASIC.

El Curso de Entrenamiento "Guía al BASIC" de AMSTRAD, está planteado para darte completa y exhaustiva ayuda para el aprendizaje de todas las facultades de tu CPC464 y de sus ilimitadas posibilidades como combinación de "tutor" en tus estudios, consola para tus juegos, y ordenador "puro". Nosotros te aconsejamos firmemente que consigas aprender sólidamente el uso del CPC464, y para ello invierte en un ejemplar de la "Guía al BASIC" de AMSTRAD, -por supuesto, si es que no lo tienes ya en tu poder.

Finalmente, la extensa sección de Apéndices, te proveerá un amplio panorama de conceptos informáticos, así como de aquellos puntos de referencia que son específicos de esta potente máquina.

Nosotros te deseamos el mejor de los éxitos -no habrías podido elegir un mejor bien para tu dinero; ni por supuesto, un ordenador con mejores facultades para desarrollar tu comprensión y dominio sobre todos los aspectos de la informática. No existe un camino mejor para aprender informática que practicar sobre un ordenador, y el CPC464 en particular -te lo garantizamos- llegará a ser tu **"más servicial amigo"**.

# IMPORTANTE

Cuando leas esta Guía de Usuario, deberás tener presente que usamos diferentes estilos de letra que indican diferentes formas de hacer referencias a los programas; que hay teclas que se encuentran en el ordenador pero que no exponen ningún carácter en la pantalla al ser pulsadas; y "descripciones generales" asociadas con palabras de programación, pero que no se teclean como parte de la instrucción donde aparecen.

1. Conecta el cable de alimentación siempre a una clavija de 3 patillas y dadas en la sección titulada **INSTALACION**.
2. Nunca conectes el Teclado, el Monitor o el Alimentador/Modulador a ninguna parte de equipo o red eléctrica diferente a la descrita en esta Guía. El no cumplimiento de esta advertencia podría producir un severo daño al equipo y -consecuentemente- invalidación de la garantía.
3. Mantén los ceniceros, bebidas, etc. lo suficientemente alejados del teclado de tu ordenador, del Alimentador/Modulador y del Monitor. Si penetrase líquido al interior de cualquiera de estas unidades, podría provocar daños irreparables. Ante un percance de esta naturaleza, consulta a personal adecuadamente cualificado.
4. No bloques ni cubras las ranuras de ventilación situadas en la parte superior y trasera del Teclado, del Monitor y del Alimentador/Modulador.
5. Toda interrupción en el abastecimiento de energía eléctrica producirá la pérdida de toda la información almacenada en la memoria de lectura y escritura del CPC464. Si deseas guardar algún programa, consulta el Capítulo 2 después de haber completado la sección de Fundamentos.
6. Se recomienda el uso de cintas magnéticas específicamente diseñadas para usar con ordenadores. Sin embargo, es perfectamente aceptable el empleo de cintas del tipo audio, de buena calidad y producidas por fábricas de prestigio, siempre que no sean Cr02 o cintas de "metal", y que no tengan una duración superior a 90 minutos (C-90).  
Para ayudarte a localizar más fácilmente los programas grabados, nos permitimos sugerirte que uses cintas C-12 (6 minutos por cara).
7. Considera que aquellas cintas que contienen programas de otros tipos de ordenador no pueden ser empleadas en el CPC464, salvo que hayan sido oficialmente declaradas "compatibles" por nosotros.



8. La tecla de Grabación (Record) no actuará si la lengüeta de seguridad (provista para impedir borrados accidentales) ha sido cortada. Por favor, no fuerces esta tecla en este caso, pues dañarías el mecanismo de grabación. Si deseas volver a grabar esas cintas en que se ha quitado la lengüeta de seguridad -situada en la parte posterior de la cinta- podrás hacerlo cubriendo el orificio resultante con cinta adhesiva.
9. Recuerda que has de asegurar que la parte inicial de la cinta (normalmente de distinto color) se encuentra ya enrollada en el carrete receptor antes de comenzar la grabación de tu programa.
10. Procura, con especial atención, no operar ni conservar ninguna de las unidades expuestas a la luz solar directa; en zonas con temperaturas excesivamente altas o excesivamente bajas; en áreas de mucha humedad o mucho polvo, o sujetas a fuertes vibraciones. Tampoco, nunca almacenes las cintas en áreas con intensos campos magnéticos, como los existentes en las inmediaciones de altavoces y grandes motores eléctricos.
11. El cuidado general de tus cintas y la limpieza regularmente de tu equipo de cassettes, te mantendrá fuera de problemas y de la pérdida de programas.
12. No existen piezas o partes dentro de la unidad que deban ser ajustadas ni engrasadas. No intentes manipular en el interior de tu equipo, que posee personal especializado.
13. Ni la totalidad, ni ninguna parte de la informática contenida en esta publicación, ni los programas o productos descritos en este manual, podrás ser adaptados o reproducidos en forma material alguna.

# Contenido

## **SOBRE ESTA GUIA DE USUARIO**

### **FUNDAMENTOS PARA PRINCIPANTES**

---

Una breve introducción para los que comienzan

F1 Instalación

F2 Familiarización con el Teclado

F3 Gráficos, Modos y Sonidos

### **1 INICIACION**

---

Conectando el Ordenador

Puesta en marcha

Lo primordial del teclado

Mostrando el repertorio de caracteres

Editando y corrigiendo

### **2 GRABADORA/REPRODUCTORA DE DATOS EN CINTA**

---

Cargando y Grabando con el cassette

La cinta de Bienvenida

### **3 LO PRIMORDIAL DEL BASIC**

---

Una Introducción a los Principios del BASIC CPC464

Sintáxis del BASIC-AMSTRAD

Variables, Operadores

Ejercicios Simples en BASIC

Teclas Definibles por el usuario

PRINT y Formas de exposición

### **4 VARIABLES, OPERADORES Y DATOS**

---

Formato de la Imagen

Datos y Tablas

Dimensionando

Ubicación

## **5 LO PRIMORDIAL DE LOS GRAFICOS**

---

Los principios en los Gráficos a Color del AMSTRAD CPC464  
INK, PEN, PAPER  
MODES, PIXELS, ORIGINS, WINDOWS  
Simples Rutinas para Gráficos  
Caracteres Definibles por el usuario

## **6 LO PRIMORDIAL DE LOS SONIDOS**

---

El Sonido en el CPC464  
Tonos y Envoltentes de Volumen  
"Colas" de Sonidos  
Efectos

## **7 IMPRESORAS Y JUEGOS**

---

Usando los "Joystick" de juegos  
Comando JOY  
Acoplamiento paralelo de una impresora

## **8 CONCISA GUIA DE REFERENCIA PARA EL "AMSTRAD-BASIC"**

---

Un conciso resumen del lenguaje "BASIC" y palabras clave usadas para la Programación del CPC464; Enunciadas en Orden Alfabético.

## **9 INFORMACION ADICIONAL PARA PROGRAMACION**

---

La organización interna de los programas residentes  
Interrupciones y su Significación  
Caracteres de control  
La relación entre las subrutinas en Código-Máquina y los Comandos "BASIC"

## **10 FACETAS DE LAS INTERRUPCIONES**

---

Características para "Tiempo Real"  
AFTER, EVERY, REMAIN

## APENDICES

---

- I    Una guía acerca de lo que un principiante puede y no puede esperar que su Ordenador haga  
      Glosario de Términos usados en Informática
- II    "Bits" y "Bytes" - Binario y hexadecimal: Una breve lección.
- III    Códigos ASCII y el Repertorio de Caracteres  
      Definiciones y retículos de los caracteres  
      Códigos en teclado, y de palabras reservadas
- IV    Introducción y Descripción General para Usuarios Expertos
- V    El "Bus" de Ampliación  
      Conexiones de entrada y salida (INPUT/OUTPUT)
- VI    Organizadores y Elaboradores de Textos en Pantalla
- VII    Compositor Musical  
      Períodos de Notas y Tonos
- VIII    Palabras Reservadas, y Códigos y Mensajes de Error



## Fundamentos 1:

## INSTALACION

Instrucciones Iniciales sobre Desembalado, Interconexiones y Puesta en marcha del Sistema CPC464.

El Ordenador Personal a Color CPC464 puede ser instalado eligiendo entre:

- 1.1 Monitor de Pantalla Verde AMSTRAD GT64
- 1.2 Monitor Polícromo AMSTRAD CTM640
- 1.3 Alimentador/Modulador AMSTRAD MP1 y un receptor doméstico de Televisión (UHF)

Por favor, consulta la sección correspondiente para preparar y conectar correctamente el sistema de ordenador, procediendo conforme a las instrucciones de operación.

### 1.1 Monitor de Pantalla Verde AMSTRAD GT64

Extraiga el Monitor de su caja y conecte una clavija de dos patillas a los cables de alimentación, conforme a las instrucciones que siguen.

#### IMPORTANTE

Los hilos conductores de este cable de alimentación, responden al siguiente código:

Azul: Neutro Marrón: Activo
--------------------------------

Si se usa una clavija de 13 Amps (BS1363) se debe insertar un fusible de 5 Amps. El fusible de 13 amperios provisto en estas clavijas NO DEBE USARSE. Si se usa cualquier otra clase de clavija, debe seguir insertándose un fusible de 5 Amps en el circuito de alimentación, sea en la clavija, en el enchufe o en el tablero de distribución. Como los colores de los hilos en el cable de alimentación de los equipos puede que no concuerden con los colores o marcas de las patillas de la clavija a emplear, procede de la siguiente forma:

El hilo de color AZUL se conecta al terminal marcado con "N" o de color NEGRO.

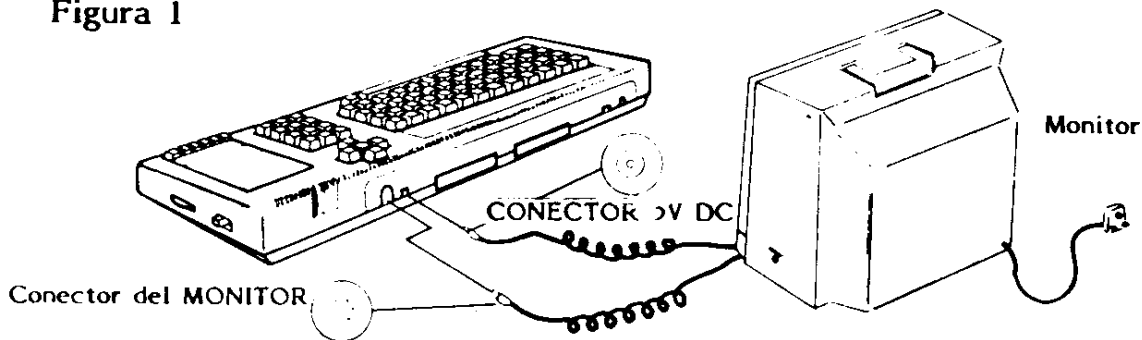
El hilo de color MARRON se conecta al terminal marcado con "L" o de color ROJO.

### ADVERTENCIA

Desconecta el enchufe de red, la clavija del cable de alimentación siempre que no estés usando el monitor. No se requiere hacer ninguna conexión en el interior del monitor, por lo tanto no trates de obtener acceso a las partes internas del Monitor.

El ordenador deberá ser colocado enfrente del Monitor en una mesa apropiada y que se encuentre cercana a algún enchufe de red con suministro de corriente. Como se muestra en la Fig. 1, conecta el cable con clavija más grande (6 pines DIN) del Monitor al enchufe hembra marcado **MONITOR** en la parte posterior del ordenador. Conecta el cable del Monitor con enchufe más pequeño (corriente continua 5Volt) al enchufe marcado **5V D.C.** en la parte posterior del ordenador.

**Figura 1**



Asegúrate que el botón **POWER** del Monitor se encuentre en la posición **OFF** (apagado). Conecta la clavija de red del Monitor al enchufe de (240 volts) corriente alterna.

Ahora enciende el monitor, y posteriormente el ordenador usando el interruptor deslizante marcado **POWER** (energía) en el extremo derecho del ordenador.

El testigo luminoso rojo **ON** situado en la parte superior central del teclado del ordenador deberá encenderse, y el Monitor mostrar en la pantalla el siguiente mensaje:

**Amstrad 64K Microcomputer (v1)**

**©1984 Amstrad Consumer Electronics plc  
and Locomotive Software Ltd.**

**BASIC 1.0**

**Ready**

Cursor

Para evitar esfuerzos innecesarios de visión, ajusta el mando del brillo **BRIGHTNESS** hasta que la pantalla muestre una imagen con brillo idóneo para la vista, sin resplandecer ni presentar las letras borrosas o sin nitidez.

También deberás ajustar el mando de contraste **CONTRAST** a la mínima posición que le permita una visión cómoda.

El control de sincronismo vertical del GT64 está marcado como **V-HOLD**, y deberás ajustarlo de manera que el cuadro de la imagen se mantenga estable y correctamente en el medio de la pantalla.

### 1.2 El Monitor a Color AMSTRAD CTM640

Saca el monitor de su caja y conecta su cable de alimentación eléctrica a una clavija industrial, conforme a las instrucciones que se posea.

#### IMPORTANTE

Los hilos conductores de este cable de alimentación, responden al siguiente código:

Azul: Neutro Marrón: Activo
--------------------------------

Si se usa una clavija de 13 Amps (BS1363) se debe insertar un fusible de 5 Amps. El fusible de 13 amperios provisto en estas clavijas **NO DEBE USARSE**. Si se usa cualquier otra clase de clavija, debe seguir insertándose un fusible de 5 Amps en el circuito de alimentación, sea en la clavija, en el enchufe o en el tablero de distribución. Como los colores de los hilos en el cable de alimentación de los equipos puede que no concuerden con los colores o marcas de las patillas de la clavija a emplear, procede de la siguiente forma:

El hilo de color **AZUL** se conecta al terminal marcado con "N" o de color **NEGRO**.

El hilo de color **MARRON** se conecta al terminal marcado con "L" o de color **ROJO**.

**ADVERTENCIA**

Desconecte el enchufe de la red cuando el Monitor se encuentre fuera de uso. No se requieren conexiones en la parte interna del Monitor, por lo tanto no intente tener acceso a las partes internas de su equipo.

El ordenador deberá ser colocado enfrente del Monitor en una mesa apropiada y que se encuentre cercana a algún enchufe de red con suministro de corriente. Como se muestra en la Fig. 1, conecta el cable con la clavija más grande (6 pines DIN) del Monitor al enchufe hembra marcado **MONITOR** en la parte posterior del ordenador. Conecta el cable del Monitor con enchufe más pequeño (corriente continua 5Volt) al enchufe marcado **5V D.C.** en la parte posterior del ordenador.

Asegúrate que el botón **POWER** del Monitor se encuentre en la posición **OFF** (apagado). Conecta la clavija de red del Monitor al enchufe de (240 volts) corriente alterna.

Ahora enciende el monitor, y posteriormente el ordenador usando el interruptor deslizante marcado **POWER** (energía) en el extremo derecho del ordenador.

El testigo luminoso rojo **ON** situado en la parte superior central del teclado del ordenador deberá encenderse, y el Monitor mostrar en la pantalla el siguiente mensaje:

**Amstrad 64K Microcomputer (v1)**

**©1984 Amstrad Consumer Electronics plc  
and Locomotive Software Ltd.**

**BASIC 1.0**

**Ready**

  
cursor

Para evitar esfuerzos innecesarios de la visión, ajusta el mando de brillo situado en un lateral del monitor y marcado **BRIGHTNESS** hasta que la pantalla muestre una imagen con luminosidad idónea y sin resplandor ni letras borrosas o faltas de nitidez.



### 1.3 Modulador/Alimentador AMSTRAD MPI y un receptor de televisión a color (UHF) doméstico

El MPI es un accesorio conveniente y que desearás comprar, si habitualmente operas tu ordenador CPC464 con el Monitor de Pantalla Verde GT64. El MPI te permitirá usar el ordenador con un Televisor a Color Doméstico, y por tanto gozar con todas las facilidades de color que te ofrece tu ordenador CPC464.

Extrae de su caja el Modulador/Alimentador MPI y conecta una clavija de corriente de alimentación al cable del Modulador/Alimentador conforme a las instrucciones siguientes:

#### IMPORTANTE

Los hilos conductores de este cable de alimentación, responden al siguiente código:

Azul: Neutro Marrón: Activo
--------------------------------

Si se usa una clavija de 13 Amps (BS1363) se debe insertar un fusible de 5 Amps. El fusible de 13 amperios provisto en estas clavijas **NO DEBE USARSE**. Si se usa cualquier otra clase de clavija, debe seguir insertándose un fusible de 5 Amps en el circuito de alimentación, sea en la clavija, en el enchufe o en el tablero de distribución. Como los colores de los hilos en el cable de alimentación de los equipos puede que no concuerden con los colores o marcas de las patillas de la clavija a emplear, procede de la siguiente forma:

El hilo de color AZUL se conecta al terminal marcado con "N" o de color NEGRO.

El hilo de color MARRON se conecta al terminal marcado con "L" o de color ROJO.

#### ADVERTENCIA

Desconecte el enchufe de la red principal cuando esta unidad se encuentre fuera de uso.

No se requieren conexiones internas de ningún tipo, por lo tanto no intente tener acceso al interior de esta unidad de su equipo.

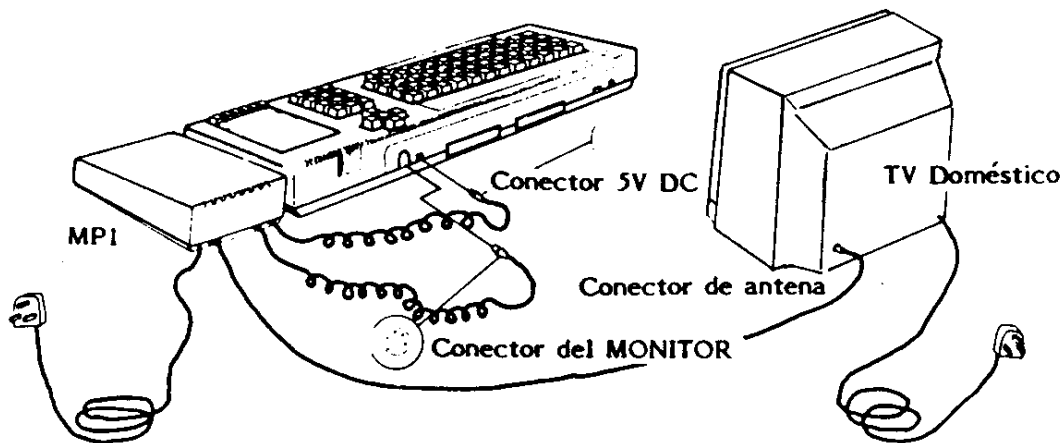


Figura 2: El MPI Modulador y Alimentador

El Modulador/Alimentador MPI deberá ser colocado a la derecha del ordenador en una mesa apropiada, que se encuentre cerca del Televisor y a un enchufe del suministro eléctrico. Como se muestra en la figura 2, conecta al cable del MPI la clavija más grande (6 pines DIN) al enchufe marcado **MONITOR** en la parte posterior del ordenador. Conecta el cable del MPI con la clavija más pequeña (corriente continua 5 voltios) al enchufe marcado **5V DC**, de la parte posterior del ordenador. Inserta el conector del cable de antena del MPI al conector de entrada de antena del Televisor.

Comprueba que el interruptor **POWER** del ordenador y situado en el costado derecho, se encuentre en la posición **OFF** (apagado); ahora conecta la clavija de alimentación MPI al enchufe de energía eléctrica (240 volts corriente alterna).

Ahora reduce el control de volumen del Televisor al mínimo; enciende el Televisor y luego el ordenador usando el interruptor deslizante situado en el costado derecho del ordenador.

El testigo luminoso rojo situado en la parte superior central del teclado del ordenador deberá encenderse, y ya puedes sintonizar el televisor hasta recibir la señal de UHF emitida por el ordenador.

Si empleas un televisor con selección de canales mediante pulsadores, presiona el correspondiente a algún canal de reserva o que no usas habitualmente. Ajusta el control de sintonía conforme a las instrucciones de fábrica para ese televisor (la señal corresponde aproximadamente al canal 36). Debes lograr una imagen con el saludo de bienvenida:

## FUNDAMENTOS

**Amstrad 64K Microcomputer (v1)**

**©1984 Amstrad Consumer Electronics plc  
and Locomotive Software Ltd.**

**BASIC 1.0**



### CURSOR

Sintoniza el televisor exactamente hasta el punto en que obtengas la imagen más nítida. Las letras deberán ser de color amarillo-oro sobre un fondo azul intenso.

Si el televisor posee un selector de canales del tipo rotativo, gira la perilla de sintonía hasta obtener la imagen indicada más arriba y hasta que ésta permanezca perfectamente estable. (Otra vez, aproximadamente el canal 36).

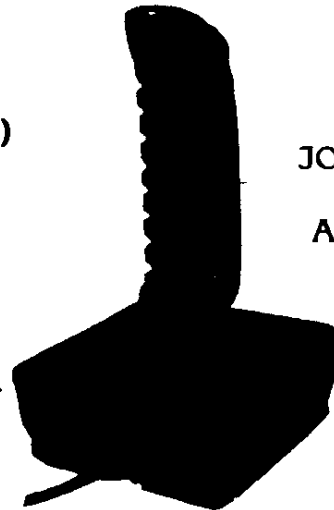
### ADVERTENCIA GENERAL

Para aquellos países que usen enchufes y clavijas de suministro eléctrico con tres tomas y terminales, los cables de alimentación del Monitor, Ordenador y Alimentador/Modulador están dotados de un tercer hilo conductor con cubierta de color verde para ser conectado a la tercera patilla de la clavija que debe corresponder a la "tierra" de la red de energía eléctrica. En ningún caso se conectará este cable a ninguno de los dos terminales cuando se empleen clavijas de dos terminales.

### 1.4 JOYSTICK

El joystick AMSOFT modelo JY1 constituye un artilugio que desearás tener si usas el ordenador CPC464 con programas de juegos que incorporen la facilidad de control de movimientos y disparos dentro del juego. El JY1 puede ser conectado a la parte posterior del ordenador usando el conector hembra de 9 vías marcado **USER PORTS (I/O)**. El ordenador AMSTRAD CPC464 puede operar simultáneamente con dos joystick de ese estilo. El segundo de ellos deberá ser conectado al conector hembra, provisto en la peana del primer joystick.

CLAVIJA DE JOYSTICK (9 vías)

JOYSTICK  
JY1  
AMSOFTSEGUNDO CONECTOR  
JY1 Joystick de control

### 1.5 Cinta de Bienvenida

Envuelta en un sobre de polietileno en la caja de tu ordenador, habrás encontrado una cassette conteniendo la "Cinta de Bienvenida". Levanta la cubierta de la ductora de cintas en cassette presionando la tecla marcada **STOP/EJECT** para parar e insertar la cassette con la cinta de bienvenida en los carriles de soporte, tal y como se muestra en la figura 3, -asegurándote de que presentas la cara 1, marcada hacia arriba.

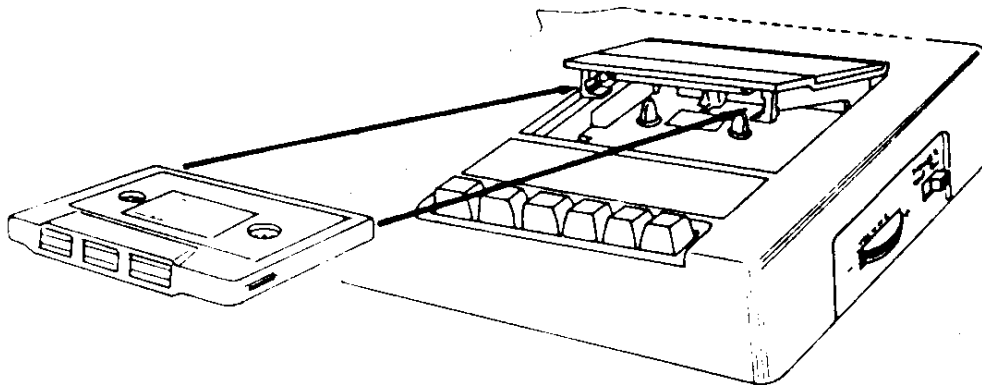


Figura 3: La forma correcta de insertar un cassette dentro de la ductora de cinta magnética.

Cierra la cubierta hasta que suene el pestillo que indica un encaje correcto, presiona la tecla de rebobinado, marcada **REW** hasta que te cerciores que la cinta está debidamente colocada en su principio. Tan pronto como la cinta se detenga, oprimes la tecla **STOP/EJECT** y repones el valor inicial en el contador de revoluciones (contadora 000) presionando el botón de restauración del contador, marcado **COUNTER RESET**.

Pulsa la tecla verde de control del teclado, marcada **CTRL** y **AL MISMO TIEMPO** la pequeña tecla azul, marcada **ENTER** en la parte inferior derecha del teclado numérico separado y situado inmediatamente al lado de la unidad ductora de cinta.



La pantalla responderá con el mensaje:

**RUN"**

Press PLAY then any key:

Ahora oprimos la tecla de reproducción, marcada **PLAY** y situada en la parte delantera de la ductora de cinta, hasta que se quede firmemente trabada en su posición inferior; luego pulsas cualquier letra, número, o cualquiera de las dos teclas **ENTER**, o incluso y más cómodo, la barra espaciadora. La cinta comenzará a correr, y después de un corto período, aparecerá en la pantalla el mensaje:

Loading **WELCOME 1** block 1

La cinta tardará cerca de 5 minutos en ser cargada y podrás observar que todo marcha correctamente a través de la pantalla, pues el número de "bloque" cambiará de 1 a 2, a 3, etc. hasta que la cinta se detenga. En este momento, el programa **Welcome** ya cargado en la memoria del ordenador, comenzará a ser ejecutado por la máquina. Simplemente acomódate en un sillón y observa. El programa "rula" continuamente, así que cuando hayas terminado de observarlo, presiona la tecla de escape, marcada **ESC** dos veces seguidas. Eso hace que se detenga el programa, y ya puedes ahora sacar la cinta, darla la vuelta y ponerla por la cara 2.

Después de cambiar la cinta a la cara 2, una vez más, recuerda presionar la tecla **REW** en la ductora de cinta, para asegurarte de que la cinta está colocada al principio del recorrido.

Pulsa la tecla **CTRL** y AL MISMO TIEMPO, la pequeña tecla **ENTER** en la parte inferior derecha del teclado numérico. La pantalla responderá como antes, con el mensaje:

**RUN"**

Press PLAY then any key:

para indicar que está lista para cargar el programa grabado en esa cara de la cinta.

Ahora oprime la tecla **PLAY** hasta que se enganche abajo, y luego pulsas cualquiera de las teclas. Todo es igual que antes. La cinta comenzará a correr y después de un corto período verás el mensaje

Loading **WELCOME 2** block 1

que te indica que está "cargando" el programa de bienvenida, (el bloque 1 de esa cara).

Sigue las instrucciones dadas en pantalla, y el programa te invitará a que participes escribiendo diversas instrucciones, más complejas a medida que va progresando el programa y tú.

## 1.6 CARGANDO OTRAS CASSETTES DE PROGRAMAS

LA CINTA DE BIENVENIDA SOLO PUEDE SER CARGADA Y SUS PROGRAMAS "EJECUTADOS" en la forma descrita en la sección previa (1.5). Los programas en BASIC -sin protecciones especiales- podrán ser cargados con los siguientes métodos alternativos. Rebobina la cinta que has insertado en la unidad ductora de cinta, presionando la tecla **REW** hasta que la cinta se detenga; que será el momento en que deberás oprimir la tecla **STOP/EJECT**.

Restaura las condiciones iniciales en el sistema (los angloparlantes llaman **RESET** a esta operación), pulsando sucesivamente las siguientes teclas: **CTRL**, **SHIFT** y **ESC** - pero manteniendo presionadas las dos primeras hasta que pulses la tecla **ESC**. La pantalla se limpiará y en pantalla reaparecerá el mismo mensaje de salutación que aparece nada más poner en marcha el ordenador.

La palabra **ENTER** en las siguientes instrucciones, indica que debes pulsar cualquiera de las dos teclas **ENTER** - no significa que debas escribir, letra por letra la palabra **ENTER**. El símbolo " se obtiene también oprimiendo cualquier tecla **SHIFT** simultáneamente con la tecla **2** de la fila superior del teclado.

Escribe:

Load "" **ENTER**

El ordenador te pide:

Press **PLAY** then any key:

Ahora presionas la tecla **PLAY** en la fila de teclas de la parte delantera del "almacenador" de datos hasta que se enganche firmemente y luego cualquiera de las del teclado, incluso la barra espaciadora.

La cinta comenzará a pasar y después de un corto período, verás aparecer en la pantalla este mensaje.

Loading <nombre del programa> block 1

Los números de bloque continuarán aumentando hasta que la cinta haya terminado de cargarse, momento en que aparecerá:

Ready

...como mensaje escrito en la pantalla, para indicarte que está dispuesta a recibir tus órdenes.

## FUNDAMENTOS

Alternativamente, puedes especificar el nombre del programa que desees cargar. Para hacerlo, escribes por teclado

Load " título " ENTER

El ordenador te pedirá:

Press PLAY then any key:

Ahora presionas la tecla **PLAY** para que la ductora de cinta comience a trasvasar a la memoria el programa grabado en la cinta, y luego pulsas alguna tecla con cualquier letra, número, cualquier tecla **ENTER**, o incluso la barra espaciadora.

La cinta comenzará a moverse. Si el programa que has pedido al ordenador que cargue no se encuentra al comienzo de la cinta, el ordenador lo buscará a lo largo de toda ella hasta encontrar uno cuyo título sea exactamente igual al que le has escrito en ese comando. Por tanto, deberás ser muy cuidadoso al escribir el título del programa en forma correcta. Si mientras el ordenador busca ese título a lo largo de la cinta se encuentra con otros programas con título distinto al que le has mandado cargar, te mostrará el siguiente mensaje en pantalla:

Found <otro título> block 1

que indica que ha encontrado un programa con ese título distinto.

El ordenador no cargará ninguno de esos programas, pero persistirá en la búsqueda del que le has mandado, a lo largo de toda la cinta hasta que encuentre el que posee un título exacto al que has escrito, o hasta que pulses la tecla **ESC** que le obliga a terminar la búsqueda en cinta.

Cuando ya ha encontrado el programa, verás aparecer en la pantalla el mensaje:

Loading <título> block 1

Para comunicarte que ha comenzado la operación de carga del programa.

Los números de bloque continuarán aumentando sucesivamente hasta que la cinta haya terminado la carga del programa, momento en el que mostrará en pantalla el mensaje:

Ready

para indicar que está dispuesta a cumplimentar otro comando.

En ese momento, puedes escribirle:

run ENTER

...y el programa que acaba de cargarse, comenzará a ser ejecutado. Si ya existía en memoria un programa, será automáticamente borrado cuando se produzca la carga de otro procedente de la cinta.

Para que el programa sea ejecutado directamente, inmediatamente después de que haya sido cargado por el ordenador, puedes teclear:

RUN "título" ENTER

para obligarle no sólo a que lo cargue, sino a que lo ejecute inmediatamente después. El ordenador te responderá con:

Press PLAY then any key:

que es un mensaje ya conocido que te solicita que presiones la tecla **PLAY** del equipo cassette y luego cualquiera de las letras, números, del teclado, o bien cualquiera de las teclas **ENTER**, o incluso la barra espaciadora. El ordenador comenzará a buscar el programa que le has pedido, y cuando lo encuentre lo trasvasará de la cinta a la memoria y lo ejecutará sin que tengas que añadir ningún otro comando por el teclado. Podrás detener la operación en cualquier momento, pulsando la tecla **ESC**, como ya conoces.

### 1.7 Cargando programas previamente grabados en cinta

Las instrucciones dadas hasta ahora te permitirán cargar cualquiera de los muchos títulos de programa disponibles para el ordenador CPC464.

Sin embargo, consulta siempre por favor, las instrucciones específicas de carga que vienen impresas en cada uno de los cassettes de programas.

### 1.8 Guardando programas en cinta

Todo programa puede ser guardado en cinta, mediante el comando BASIC SAVE, para ser usado posteriormente. Inserta una cinta en el equipo cassette, en la forma que te hemos indicado, y cierra la cubierta. (La lengüeta de protección que impide la grabación de las cintas, y que está situada en la parte posterior del propio cassette que contiene la cinta, no debe haber sido quitada).

## FUNDAMENTOS

Oprime la tecla **REW** para rebobinar la cinta y colocarla al principio, recordando pulsar la tecla **STOP/EJECT** cuando se detenga la cinta. Luego tecleas simplemente:

```
save "<título del programa>" ENTER
```

que es el comando necesario para que el ordenador trasvase a la cinta el programa que tiene en memoria, y quede en la cinta con el título que has reseñado en la instrucción.

El ordenador te responderá con el mensaje:

```
Press REC and PLAY then any key:
```

Para indicarte que en este caso han de pulsarse las teclas de grabación y de marcha al mismo tiempo, y de manera que ambas se queden enganchadas firmemente en su posición más baja; y que luego pulses cualquiera de las letras, números, teclas **ENTER**, o incluso barra espaciadora para que comience la operación.

Después de haberlo hecho, el ordenador te mostrará en pantalla el mensaje:

```
Saving <título del programa> block 1
```

para indicar que ha comenzado la operación de guardar en la cinta el programa que tenía en memoria y con el título que le has dado.

Cuando haya terminado de grabar el programa, la cinta se detendrá y verás aparecer en pantalla la palabra: Ready. En ese momento puedes pulsar la tecla **STOP/EJECT** de tu equipo cassette, porque tu programa ya se encuentra depositado en la cinta.

Ten en cuenta que no podrás almacenar programas en cinta que hayan sido previamente grabados por las empresas distribuidoras con programas de juegos o de cualquier tipo, cargándolos previamente en tu ordenador y aunque las cintas que vayas a usar estén en blanco.

Todos esos programas se encuentran protegidos para impedir que se saquen copias sin autorización.

## Fundamentos 2:

## FAMILIARIZACION CON EL TECLADO

Explicaremos ahora las funciones de algunas de las teclas del ordenador. Aquellos que posean experiencia previa con equipos de proceso de datos pueden saltarse esta sección.

### ENTER

Existen dos teclas marcadas **ENTER** en tu ordenador. Cada una de estas teclas "envían adentro" la información que has tecleado, indicando al ordenador que ya has terminado de darle la instrucción y que puede pasar a tramitarla. Después de pulsar la tecla **ENTER**, el cursor se coloca al principio de la siguiente línea de la pantalla. Todas y cada una de las instrucciones y comandos que le des al ordenador, deben ser terminadas pulsando la tecla **ENTER**, para que ingrese ese comando o instrucción y pueda ser cumplimentada. De ahora en adelante, sólo mostraremos la pulsación de esa tecla **ENTER** para destacar que debe ser pulsada después de cada instrucción o línea de programa.

### DEL

Esta tecla se usa para borrar o "delectar" el carácter situado inmediatamente a la izquierda del cursor; es decir, para suprimir una letra, número, o símbolo que sea innecesario. Escribe por ejemplo abcd y verás que la última letra tecleada, la d está situada inmediatamente a la izquierda del cursor. Si decidieras que esa letra d ya no era necesaria, te bastaría pulsar la tecla **DEL** una sola vez para comprobar que el cursor retrocede a la izquierda, suprimiendo de la pantalla dicha letra. Si mantienes pulsada la tecla **DEL** el tiempo suficiente, también irían sucesivamente borrándose de la pantalla con retroceso del cursor en cada caso, las letras cba por ese orden.

### SHIFT

Existen dos teclas de turno o cambio, marcadas con la palabra **SHIFT**. Pulsando cualquiera de ellas y manteniéndola pulsada mientras que al mismo tiempo se oprime una tecla correspondiente a una letra, lo que aparece en la pantalla es esa letra pero en mayúsculas. Escribe por ejemplo la letra a y luego vuelve a pulsar la tecla a simultáneamente con la pulsación de la tecla **SHIFT**. Verás que en pantalla aparece lo siguiente:

aA



Escribe ahora unos pocos espacios en blanco, manteniendo pulsada la barra espaciadora. Ensaya ahora lo mismo usando las teclas marcadas con números que están situadas en la fila superior del teclado; escribe por ejemplo el número 2 simplemente pulsando la tecla marcada con 2, y luego pulsa esa misma tecla pero simultáneamente mantienes pulsada la tecla **SHIFT**. Comprobarás que ahora aparece en pantalla el símbolo de las comillas, que está situado por encima del 2 dentro de la misma tecla. Por tanto en pantalla tendrás:

2"

Puedes ahora comprobar lo que pasa cuando se mantiene pulsada la tecla **SHIFT** si se pulsa cualquier otra tecla marcada con un carácter. Haz experimentos, tecleando cualquiera de esas teclas de caracteres y pulsando unas veces simultáneamente la tecla **SHIFT** y otras veces sin pulsarla. Cuando se pulsa, comprobarás que es el "turno" de las mayúsculas o de los caracteres situados en la parte superior de cada tecla.

### CAPS LOCK

Pulsando esta tecla una sola vez, se logra que las teclas correspondientes a las letras que hay en el teclado pasen a enviar siempre al ordenador el símbolo correspondiente a las mayúsculas. Decimos que el teclado está "trabado" o enclavado en mayúsculas. A partir del momento en que ha pulsado la tecla **CAPS LOCK**, ten en cuenta que todas las letras serán mayúsculas, pero para conseguir con las otras teclas el signo que figura en la parte superior, tendrás que seguir pulsando simultáneamente la tecla **SHIFT**.

Veamos un ejemplo. Pulsa la tecla **CAPS LOCK** y luego sucesivamente las marcadas con los símbolos:

abc def123456

Observarás que en pantalla lo que aparece es:

ABCDEF123456

que indica que todas las teclas de letras envían a la pantalla y al ordenador esas mismas letras pero en mayúscula; mientras que las teclas con números no envían los símbolos marcados en la parte superior de la tecla.

Pero si ahora vuelves a pulsar sucesivamente las teclas:

abc def123456

pero manteniendo pulsada simultáneamente y cada vez la tecla **SHIFT**, lo que obtienes en pantalla es:

ABCDEF ! " # \$ % &

tal y como te hemos comentado.

Si quisieras volver al modo anterior con las letras en minúscula, sólo tienes que pulsar otra vez la tecla **CAPS LOCK**. Esta tecla actúa como un "basculador", cambiando de modo cada vez que se pulsa.

Si deseas teclear letras en mayúscula y los símbolos situados en la parte superior de las otras teclas, sin tener que mantener constantemente pulsada la tecla **SHIFT**, también lo puedes lograr pulsando simultáneamente la tecla **CTRL** y la tecla **CAPS LOCK**. Después de ello, si pulsas sucesivamente las teclas marcadas:

abcdefghijklmnopqrstuvwxyz

obtendrías en pantalla:

ABCDEFGHIJKLMNOPQRSTUVWXYZ

Y sigue siendo posible teclear números en esta situación, incitada por la pulsación simultánea de **CTRL** y **CAPS LOCK**, si empleas las teclas numéricas colocadas en el teclado que hay a la derecha, separado del teclado principal.

Manteniendo pulsada la tecla **CTRL** y volviendo a pulsar simultáneamente la tecla **CAPS LOCK**, volverás de nuevo al modo en que estabas previamente: sea con las mayúsculas trabadas, o en minúsculas. Por supuesto, que si has vuelto al modo de mayúsculas trabadas, el efecto basculador de la tecla **CAPS LOCK** hará que a la siguiente pulsación de esa tecla, vuelvas al modo minúsculas.

### CLR

Esta tecla se usa para borrar ("influido por el inglés, diríamos CLAREAR") para borrar el carácter situado en la misma posición que el cursor.

Teclea por ejemplo ABCDEFGH. El cursor terminará situado inmediatamente a la derecha de la última letra escrita, la H. Ahora pulsa la tecla de desplazamiento a izquierda del cursor (+), y hazlo cuatro veces seguidas. El cursor se moverá cuatro posiciones hacia la izquierda, colocándose encima mismo de la letra E.

Observa que la letra E todavía es visible dentro del cursor. Si ahora pulsas la tecla **CLR** una sola vez, notarás que desaparece la letra E y que las letras FGH se desplazan un espacio hacia la izquierda, con lo que ahora es la letra F la que aparece dentro del cursor. Si pulsas ahora una segunda vez la tecla **CLR** desaparecerá esa letra F y manteniéndola pulsada, verás cómo se van borrando sucesivamente las letras y desplazándose hacia la izquierda.

**ESC**

Esta tecla se usa para ESCAPAR de la función que el ordenador esté desarrollando en ese momento. Pulsando una sola vez la tecla **ESC**, sólo se consigue que el ordenador realice una pausa en el desarrollo de la función vigente, y que la continúe tan pronto como sea pulsada cualquier otra tecla distinta de **ESC**.

Si en esa situación pulsas **ESC**, habrán sido dos pulsaciones sucesivas de la tecla **ESC**, lo que tiene como efecto que el ordenador termine y deje definitivamente la función que estaba desarrollando en ese momento. El ordenador pasará entonces al modo normal en que espera que le dictes algún otro comando o instrucción.

Pulsa ahora por dos veces consecutivas la tecla **ESC**.

**IMPORTANTE**

Cuando llegues al borde derecho de la imagen, porque hayas tecleado más de 40 caracteres consecutivos, el siguiente carácter que teclees aparecerá automáticamente en la siguiente línea de la pantalla y situado en el extremo izquierdo de esa línea. Eso significa que **NO DEBES** pulsar **ENTER** cuando estés cerca del final de una línea, tal y como se acostumbra a hacer en las máquinas de escribir, sino que debes continuar tecleando, que automáticamente el ordenador pasará a la línea siguiente cuando lo necesite.

Si no lo hicieras, y pulsaras intempestivamente la tecla **ENTER**, el ordenador mostraría su desacuerdo -habitualmente mediante el mensaje: **Syntax error**, ya sea en el momento en que tecleas o posteriormente cuando pases ese programa.

**Syntax Error**

Si aparece en pantalla el mensaje: **Syntax error**, es señal que el ordenador no entiende el comando o instrucción que le acabas de teclear. Por ejemplo, si escribes:

```
printt ENTER
```

Aparecerá en pantalla el citado mensaje de error sintáctico.

```
syntax error
```

Ese mensaje aparece porque el ordenador no comprende la palabra "printt" (debido a la doble t).

Si cometes una equivocación en una línea de un programa, de la misma clase que antes, como por ejemplo:

```
10 printt "abc" ENTER
```

no aparecerá el mensaje de error sintáctico hasta que esa instrucción vaya a ser cumplimentada por el ordenador en el momento de ejecutar el programa.

Lo puedes comprobar, tecleando ahora:

run **ENTER**

Verás que en pantalla aparece el mensaje:

```
Syntax error in 10  
10 printt "abc"
```

que te indica un error sintáctico como antes, pero señalándote además el número de línea en que ha detectado el error, y mostrando en pantalla toda esa línea del programa, juntamente con el cursor de "edición" situado sobre el primer carácter de esa línea, para indicarte que espera la corrección.

Pulsa en esta situación la tecla de avance del cursor a derecha, marcado (→) hasta que el cursor se sitúe encima de una de las t de dicha palabra printt. Basta pulsar la tecla **CLR** para que desaparezca esa t que sobra. Ahora no olvides que hasta que pulses la tecla **ENTER**, el ordenador no se da por enterado que has corregido esa línea.

Después de hacerlo, ya puedes repetir el comando run **ENTER** para comprobar que ya el ordenador acepta esa instrucción, la interpreta y cumplimenta, y por tanto, expone en pantalla las letras abc.

## Una introducción a las palabras clave del BASIC AMSTRAD

En el capítulo 8, encontrarás una descripción ordenada de todas las palabras claves aceptadas en el lenguaje "BASIC AMSTRAD". En esta sección te presentamos algunas de esas palabras clave que se usan más frecuentemente.

### CLS

Teclea esa palabra cls y verás cómo se LIMPIA la pantalla. La palabra **CLS** la puedes escribir como todas las demás palabras reservadas usando letras mayúsculas o minúsculas, y siempre pulsando para terminar todo comando la tecla **ENTER**. Notarás que después de quedarse la pantalla en limpio, aparece la palabra habitual Ready, con el cursor ■ situado en la esquina superior izquierda de la pantalla, indicándote que está dispuesto a recibir nuevas órdenes.

**PRINT**

Esta palabra clave se usa habitualmente para hacer que el ordenador EXPONGA en pantalla lo que tú deseas, sean palabras, datos o figuras. Teclea por ejemplo, el siguiente comando:

```
print "hola"  ENTER
```

Y verás que en la pantalla aparece la palabra:

```
hola
```

porque el ordenador ha entendido que debía emitir hacia la pantalla lo que estaba incluido entre comillas.

Las comillas (") se usan para indicar al ordenador que lo que debe considerar como dato, y en este caso exponerlo en pantalla, es lo que está incluido dentro de las comillas. En este caso fue la palabra `hola`, y cumplimentó ese comando tan pronto como pulsaste la tecla **ENTER**. Escribe ahora `cls` **ENTER** para dejar limpia la pantalla.

**RUN**

El ejemplo previo nos ha mostrado cómo cumplimenta el ordenador un comando de forma inmediata. Para que no ejecute inmediatamente lo mandado, debemos preceder nuestro mandato de un número de línea cualquiera. En ese caso, el ordenador no cumplimenta inmediatamente lo mandado, sino que lo considera como una INSTRUCCION que debe, de momento, almacenar en su memoria y que más tarde le diremos qué hacer con ella. Ese número caracteriza a cada instrucción y le sirve además al ordenador para colocarlas en su memoria según el orden ascendente de esos números. Un programa es simplemente una serie de instrucciones almacenada en la memoria del ordenador, y no se ejecuta hasta que le mandamos que lo ejecute, usando el comando **RUN**.

Escribe por ejemplo la instrucción:

```
10 print "hola"  ENTER
```

y pulsas para terminarla la tecla **ENTER**.

Observa que cuando pulsaste **ENTER**, el ordenador no escribió en pantalla la palabra `hola`, porque lo que tecleaste lo interpretó como una instrucción y lo tiene simplemente almacenado en su memoria. Es un programa de una sola línea. Si ahora le das, es decir, le tecleas el comando:

```
run  ENTER
```

verás cómo aparece en pantalla la palabra `hola`, como resultado de haber ejecutado ese programa.

## FUNDAMENTOS

Te queremos hacer notar que la palabra **PRINT** se usa continuamente en los programas, por lo que puedes abreviarla pulsando simplemente la tecla de interrogación, situada en la parte superior de la tecla marcada con el símbolo **/**.

Por ejemplo, puedes escribir abreviadamente:

```
10 ? "hola" ENTER
```

### LIST

Después de que el programa haya sido almacenado en la memoria del ordenador, es posible comprobar toda la serie de instrucciones que lo forman, sacando un LISTADO del programa. Escribe por ejemplo:

```
list ENTER
```

y en la pantalla verás la única línea de nuestro programa:

```
10 PRINT "hola"
```

Observa que la palabra **PRINT** está ahora en mayúsculas, a pesar de haberla escrito en minúsculas. Eso significa que el ordenador no solamente ha aceptado esa palabra como una palabra clave en **BASIC**, sino que la ha transformado y la ha escrito en mayúsculas.

Da el comando **CLS** para limpiar la pantalla, y ten en cuenta que sólo es la pantalla la que se ha quedado en limpio mediante el comando **CLS**; el programa continúa estando en memoria, y lo puedes comprobar volviendo a pedir al ordenador que lo liste.

### GOTO

La palabra clave **GOTO** le dice al ordenador que VAYA desde la línea que se encuentra hasta la línea cuyo número se reseña en esa instrucción. Se usa (cada vez menos) para dar un salto progresivo o regresivo en el número de instrucción que se ejecutará a continuación.

Si por ejemplo tecleas:

```
10 print "hola" ENTER  
20 goto 10 ENTER
```

y luego le pides que ejecute ese programa, mediante el comando:

```
run ENTER
```

verás que el ordenador escribe continua y sucesivamente en la pantalla la palabra **hola**, colocándolas una detrás de otra y en el extremo izquierdo de la imagen. Para detener este programa, puedes hacerlo como ya te hemos mencionado, pulsando la tecla **ESC** una sola vez. Para que vuelva a comenzar de nuevo, te basta pulsar cualquier tecla distinta de **ESC**. Para pararlo nuevamente de forma definitiva, y poder escribir otras instrucciones nuevas, pulsa dos veces la tecla **ESC**.

Teclea ahora:

```
cls ENTER
```

para dejar la pantalla en limpio.

Para verla palabra hola escrita continuamente en cada línea, una vez detrás de otra y rellenando por tanto la totalidad de la pantalla, puedes modificar el programa anterior y usar el signo punto y coma (;) después de las comillas de cierre. Es decir, debes escribir:

```
10 print "hola"; ENTER
20 goto 10 ENTER
run ENTER
```

Observa que el signo punto y coma (;) es interpretado por el ordenador como que le pides que escriba el siguiente carácter inmediatamente detrás del anterior. ESCAPA de este programa pulsando dos veces la tecla **ESC**. Ahora cambia nuevamente la línea 10, colocando esta vez el signo coma (,) en lugar del punto y coma anterior, e inmediatamente detrás de las comillas de cierre. Es decir, debes escribir:

```
10 print "hola", ENTER
run ENTER
```

Verás que el signo de coma en una instrucción **PRINT**, es interpretado por el ordenador como que debe escribir el siguiente carácter (o serie de caracteres) 13 posiciones más allá de donde haya escrito el primero de la serie de caracteres anterior. Esta facilidad se aprovecha para exponer en pantalla los datos tabulados en columnas. Observa sin embargo, que si el número de caracteres que compone una serie dada, excede de 12, el siguiente carácter sería desplazado hacia delante otras 13 columnas, situándose por tanto en la columna número 26.

Otra vez, escapa de ese programa pulsando la tecla **ESC** dos veces consecutivas. Pero ahora, para que la memoria del ordenador quede completamente "en blanco" mantén pulsadas sucesiva y simultáneamente las teclas **SHIFT**, **CTRL** y **ESC** en ese mismo orden, con lo que el ordenador se restaurará a sí mismo a las condiciones iniciales de funcionamiento.

## INPUT

Esta instrucción se usa para hacer saber al ordenador que al cumplimentarla debe esperar a que se le IMPONGA un dato por teclado. Por ejemplo, cuando en el programa se hacen preguntas por pantalla y el usuario le responde por teclado. Escribe por ejemplo, el siguiente programa:

```
10 input "que edad tienes"; edad ENTER
20 print "no aparentas tener "; edad " años" ENTER
run ENTER
```



## FUNDAMENTOS

En la pantalla, cuando rules el programa, aparecerá la pregunta:

**¿qué edad tienes?**

Contestale adecuadamente, que será el dato **IMPUESTO**, y lo terminas pulsando **ENTER**. Si tu edad fuera de 18 años, el ordenador sacaría por pantalla el mensaje:

**no aparentas tener 18 años.**

Este ejemplo nos demuestra el uso de la instrucción **INPUT**, y de cómo el dato tecleado se **IMPONE** como el valor de una variable numérica, en este caso es la que tiene por nombre **edad**, tal y como está reseñada al final de la línea 10. Observa que a partir de ahora, el ordenador asociará a la variable con nombre **edad** el número que le tecleaste en el momento de sacar en pantalla la pregunta correspondiente, y que ese mismo dato impuesto es el que expone posteriormente cuando cumplimenta la instrucción **PRINT** de la línea 20 donde figura esa variable numérica llamada **edad**. Aunque hemos usado como nombre de variable la palabra **edad** en este ejemplo, podríamos igualmente haber usado en forma más breve otro nombre para la variable, como por ejemplo la letra **b** simplemente.

Restaura el ordenador para borrar el programa que tienes en memoria (teclas **CTRL**, **SHIFT** y **ESC**). Si hubieras querido que el dato impuesto estuviera formado por una serie cualquiera de caracteres cualesquiera (letras, números o signos), deberías usar el signo dólar (\$) al final del nombre de la variable, para indicarle al ordenador que le vas a imponer por teclado un "LITERO" en lugar de un "NUMERO" (el ordenador tomará ese dato "**literalmente**" tal como se lo escribas).

Teclea el siguiente programa: (y observa que en la línea 20 debes colocar un espacio en blanco después de la **a** de **hola** y antes de la **y** de **yo**).

```
10 input "cómo te llamas"; nombre$ ENTER
20 print "hola "; nombre$ " yo me llamo Alvaro" ENTER
run ENTER
```

Luego verás en pantalla la pregunta:

**¿cómo te llamas?**

Teclea tu nombre en ese momento, dato que será **IMPUESTO** como valor de la variable que has llamado **nombre\$** en esa instrucción 20. Si tu nombre es Pepe, verás que aparece en la pantalla la frase:

**hola Pepe yo me llamo Alvaro**

Aunque en el ejemplo anterior hemos usado **nombre\$** para designar esa variable **LITERAL** (también llamada de cadena o sarta, o ristra) podríamos haber usado simplemente una letra como por ejemplo **a\$**. Es el signo dólar al final lo que distingue a las variables numéricas de las variables literales.

Vamos ahora a combinar estos dos ejemplos en un solo programa. Restaura el ordenador a sus condiciones iniciales pulsando **CTRL, SHIFT y ESC**, y luego tecleas la siguiente instrucción:

```
5 cls ENTER
10 input "como te llamas"; a$ ENTER
20 input "que edad tienes"; b ENTER
30 print "te felicito ";a$ " no aparentas tener";
  b " años" ENTER
run ENTER
```

En este programa usamos dos variables, **a\$** para el nombre que será un lítero, y **b** para la edad, que será un número. En la pantalla aparecerá primero:

¿como te llamas?

En este momento, puedes teclear tu nombre (v.g. Pepe) y luego pulsar **ENTER**. Con eso, el ordenador pasará a la siguiente instrucción por lo que en pantalla aparecerá:

¿que edad tienes?

Teclea ahora tu edad (v.g. 18) y luego pulsas **ENTER**. Con esto, el ordenador pasará a la siguiente instrucción, por lo que en pantalla aparecerá:

te felicito Pepe no aparentas tener 18 años

y con esto se termina la ejecución de este sencillo programa.

## EDICION DE PROGRAMAS

Si cualquiera de las líneas del programa ha sido tipografiada incorrectamente, provocará un error sintáctico u otra clase de error, y será necesario revisar y corregir (editar es la palabra que usamos) esa línea en lugar de teclearla de nuevo. Para mostrarte la edición de programas, puedes teclear el programa anterior pero en forma incorrecta.

Ensaya por ejemplo con lo siguiente:

```
5 class
10 input"cbmo te llamas"; a$ ENTER
20 input"què eda tienes"; b ENTER
30 print"debo felicitarte "; a$ ";" que no aparentas
  tener "; b ";" años " ENTER
```

Hay tres equivocaciones en el programa anterior:

En la línea 5 pusimos **c l s s** en lugar de **c l s**.

En la línea 10 escribimos **hedad** en lugar de **edad**.

En la línea 30 olvidamos colocar un blanco entre el final de la palabra **advertirte** y las correspondientes comillas de cierre.

Hay tres métodos para editar un programa. El primero es simplemente volver a teclear correctamente la línea. Cuando se teclea una línea con el mismo número de una ya existente, el ordenador interpreta que simplemente queremos sustituir la que ya existe por la que ahora tecleamos. En segundo lugar, está el método propiamente del ordenador en modo edición, y finalmente podemos usar el método de copiar lo que aparece en pantalla mediante movimientos del cursor, como ya veremos.

### Método de Edición

Para corregir la equivocación de la línea 5, teclea el comando:

**edit 5 ENTER**

El ordenador entra en el modo de edición, y nos muestra dicha línea 5 con el cursor situado directamente encima del 5. Para quitar la **s** extra de la palabra **clss**, desplaza el cursor hacia la derecha mediante la tecla de avance de cursor marcada (→) hasta que se sitúe sobre cualquiera de las eses, y luego pulsas la tecla **CLR**. Verás que la **s** desaparece de la pantalla (y de la memoria). Pulsa **ENTER** para indicar que la operación ha terminado, y lista el programa para ver que esa instrucción ha sido corregida. Es decir, teclea

**list ENTER**

y verás que la línea 5 aparece ya corregida (también diríamos editada).

### Método de Copia moviendo el cursor

Para corregir las equivocaciones de las líneas 10 y 30, puedes hacerlo manteniendo pulsada la tecla **SHIFT** y pulsando la tecla de subida del cursor, marcada (↑), hasta que el "cursor que copia" se sitúa en el principio mismo de la línea 10. Verás que el cursor principal no se ha desplazado, sino que tienes ahora dos cursores en la pantalla. Pulsa ahora la tecla de COPIAR marcada con la palabra **COPY**, hasta que el cursor de copiado se sitúe entre el espacio en blanco entre **edad** y las comillas de cierre. Observarás que la línea 20 se estaba también escribiendo en la última línea de la pantalla y que el cursor principal se detiene en la columna homóloga a la del cursor de copia. Teclea ahora la letra **d** que falta, y observa cómo sólo aparece en la última línea. Y además que el cursor principal sí se desplaza una posición a la derecha, pero el cursor de copiar permanece donde estaba.

Pulsa ahora la tecla **COPY** hasta que copies la totalidad de esa línea 20; luego pulsas **ENTER** para indicar que la operación ya VALE, y esa nueva línea 10 corregida sustituye en la memoria a la anterior. El cursor de copia desaparece y el cursor principal se coloca por sí mismo bajo la nueva línea 10.

Para corregir la segunda equivocación, mantén pulsada la tecla **SHIFT** y pulsa la tecla de subida del cursor, marcada (**↑**) hasta que el cursor de copiar aparezca situado al principio mismo de la línea 30. Pulsa la tecla **COPY** hasta que sitúe el cursor de copiar sobre las comillas de cierre que van inmediatamente detrás de la palabra **advertirte**. Ahora simplemente pulsa la barra espaciadora una vez. Con ello se insertará un espacio, como puedes comprobar en la línea que se está escribiendo en la parte inferior. Luego continúa copiando con la tecla **COPY** hasta que completes toda la línea 30. Pulsa la tecla **ENTER** para decir que vale, y ya puedes listar el programa y comprobar que todo está corregido en la memoria del ordenador, lo que puedes hacer simplemente tecleando **LIST ENTER**.

Restaura el ordenador pulsando las teclas **CTRL**, **SHIFT** y **ESC**.

## IF THEN

Vamos ahora a ampliar el programa anterior con el uso de instrucciones que sólo SI se cumple una determinada condición, son PUES cumplimentadas por el ordenador.

Teclea lo siguiente, observando que hemos añadido dos símbolos, a saber, el signo MENOR QUE, marcado **<** y situado al lado de la tecla **M**, y el signo MAYOR QUE, marcado **>** y situado a continuación de la tecla anteriormente citada.

```
5 cls ENTER
10 input "cómo te llamas"; a$ ENTER
20 input "qué edad tienes"; edad ENTER
30 if edad < 13 then 60 ENTER
40 if edad < 20 then 70 ENTER
50 if edad > 19 then 80 ENTER
60 print "aun "; a$; " no eres un adolescente con
tus "; edad; " años":end ENTER
70 print "ya "; a$; " eres un adolescente a tus";
edad; " años":end ENTER
80 print "comprenderás "; a$; " que ya no eres
un adolescente con tus "; edad; " años" ENTER
```

## FUNDAMENTOS

Para comprobar que lo has tecleado correctamente, pide al ordenador que liste el programa y luego teclea:

```
list ENTER
```

Luego vete contestando las preguntas sugeridas por el ordenador y viendo lo que sucede.

Analicemos ahora los efectos que los comandos **IF...THEN** tienen en un programa. Hemos añadido también la palabra **end** al final de las líneas 60 y 70. Obviamente, este comando **end** se emplea para que el ordenador **TERMINE** de ejecutar el programa. Si no estuviera ese comando **end** en la línea 60, el programa continuaría rulando y llevando a cabo las instrucciones reseñadas en las líneas 70 y 80.

De igual manera, si no estuviera el comando **end** en la línea 70, el programa continuaría pasando y cumplimentando las instrucciones de la línea 80. El signo dos puntos (:) antes de la palabra **end** separa este comando del comando anterior. Los dos puntos se usan para poder incluir dos o más instrucciones dentro de una misma línea del programa; se tiene lo que llamamos línea multi-instrucción. También hemos añadido la línea 5 para limpiar la pantalla al inicio del programa. Haremos eso mismo a partir de ahora para que los resultados aparezcan más estéticos.

Otras palabras asociadas con la instrucción **IF...THEN** son **ELSE**, **OR** y **GOTO**. El uso de esas palabras dentro de esta clase de instrucciones se te hará más obvio a medida que progrese en el estudio de esta Guía.

Restaura el ordenador pulsando las teclas **CTRL**, **SHIFT** y **ESC**.

### FOR...TO...NEXT

Vamos ahora a mostrar el uso de los comandos **FOR** y **NEXT**, que nos permiten obligar al ordenador que repita la serie de instrucciones incluidas entre la instrucción **FOR...TO** y la instrucción **NEXT**. Para este ejemplo mostraremos la manera en que podemos hacer que el ordenador exponga la tabla de multiplicar por el número 12 (1x12, 2x12, 3x12, etc.). Es decir, que repita el producto, **CON (FOR)** uno de los datos, variando desde el valor inicial 1, **HASTA (TO)** que ese dato llegue a tener como valor final 12. Observa que el producto se simboliza en **BASIC** mediante el signo asterisco (\*). Teclea lo siguiente:

```
5 cls ENTER
10 for a = 1 to 20 ENTER
20 print a" * 12 ="a*12 ENTER
30 next a ENTER
run ENTER
```

Observarás que las columnas aparecen desordenadamente en pantalla, así que teclea ahora:

```
5 cls ENTER
10 for a = 1 to 9 ENTER
20 print a" * 12 ="a*12 ENTER
30 next a ENTER
40 for a = 10 to 20 ENTER
50 print a"* 12 ="a*12 ENTER
60 next a ENTER
run ENTER
```

Experimenta con este programa para hallar las tablas de multiplicar usando otros números. Por ejemplo, si quieres ver la tabla del 17, sustituye el número 12 de las líneas 20 y 50 por el número 17. Finalmente, restaura como siempre el ordenador con las teclas **CTRL**, **SHIFT** y **ESC**.

Observa que es posible especificar el valor por el que se incrementa la variable que controla este bucle cuando va a efectuar OTRA (**NEXT**) ronda, usando la palabra reservada para definir el paso (**STEP**). Para más información, estudia la descripción de los bucles **FOR...NEXT** en el capítulo 8.

## ARITMETICA SIMPLE

Tu ordenador CPC464 puedes usarlo también como un calculador de sobremesa.

Para comprender esta forma de operar, realiza los siguientes ejemplos. Como ya sabes, podemos usar el signo de interrogación (?) en lugar de teclear la palabra **PRINT**, y así abreviar. Las respuestas se expondrán en pantalla tan pronto como pulses la tecla **ENTER**.

## ADICION

Usa la tecla **SHIFT** y la marcada con ; para conseguir el signo más

Teclea

```
?3+3 ENTER
6
```

(observa que no tienes que teclear el signo =)

Teclea:

```
?8+4 ENTER
12
```

## -- FUNDAMENTOS

### SUSTRACCION

Directamente consigues el signo menos, sin teclear **SHIFT**).

Teclea:

?4-3 ENTER  
1

Teclea

?8-4 ENTER  
4

### MULTIPLICACION

Teclea **SHIFT** y **\***, para conseguir el **\*** que simboliza en BASIC la multiplicación.

Teclea:

?3\*3 ENTER  
9

Teclea:

?8\*4 ENTER  
32

### DIVISION

Directamente consigues la **/** que simboliza la división.

Teclea:

?3/3 ENTER  
1

Teclea:

?8/4 ENTER  
2

### RAIZ CUADRADA

Para hallar la raíz cuadrada de un número, teclea la función **SQR (X)**, siendo el argumento de la función, la X, el número cuya raíz cuadrada quieres saber.

Teclea:

?sqr(16) ENTER  
4

lo que significa  $\sqrt{16}$ .

Teclea:

?sqr(100) ENTER  
10



## EXPONENCIACION

Usa directamente el signo de "elevado a", marcado con  $\uparrow$ . (Está debajo de la  $\pm$ ).

La potencia de un número es el resultado de multiplicar ese número tantas veces como indica el exponente. Por ejemplo  $3^2$ ,  $3^3$ ,  $3^7$ . Teclea:

?3 $\uparrow$ 3 ENTER  
27

esto significa  $3^3$

Teclea:

?8 $\uparrow$ 4 ENTER  
4096

eso significa  $8^4$

## RAIZ CUBICA

Fácilmente puedes calcular raíces cúbicas, si recuerdas que son equivalentes a elevar el número a la potencia  $1/3$ .

Para hallar la raíz cúbica de 27 ( $\sqrt[3]{27}$ ). Teclea:

?27 $\uparrow$ (1/3) ENTER  
3

Para hallar la raíz cúbica de 125, teclea:

?125 $\uparrow$ (1/3) ENTER  
5

## CALCULOS COMBINADOS (+, -, \*, /)

Los cálculos combinados son bien interpretados por el ordenador, siempre y cuando se calculen de acuerdo con las prioridades estipuladas para cada operación.

La primera prioridad corresponde a la multiplicación, luego la división, luego la adición, y finalmente la sustracción. Esas prioridades se aplican únicamente cuando en la expresión figuran sólo estas cuatro operaciones. Las prioridades serán ampliamente explicadas más adelante e incluirán otras clases de operación.

Si el cálculo fue:

$$3+7-2*7/4$$

Puedes pensar que podría calcularse en la forma siguiente:

$$\begin{aligned} &3+7-2 * 7 / 4 \\ = &8 * 7 / 4 \\ = &56 / 4 \\ = &14 \end{aligned}$$

## FUNDAMENTOS

Pero de hecho, el ordenador lo calcula como:

```
3+7-2*7/4
=3+7-14/4
=3+7-3.5
=10-3.5
=6.5
```

Compruébalo tecleando esta expresión tal y como está escrita.

Teclea:

```
?3+7-2*7/4  ENTER
6.5
```

Puedes cambiar la manera en que el ordenador calcula una expresión si empleas paréntesis. El ordenador tratará primero las operaciones incluidas dentro de los paréntesis de acuerdo con las prioridades establecidas, y luego pasará a las que están fuera de los paréntesis. Ensáyalo tecleando la expresión anterior y colocando paréntesis, tal y como el siguiente ejemplo:

```
? (3+7-2)*7/4  ENTER
14
```

## MAS SOBRE POTENCIACION

Si quieres usar en tus cálculos números muy grandes o muy pequeños, es conveniente que uses la denominada **notación científica**. En esta notación se usa la letra E para indicar exponentes de los números con relación a la base 10. Puedes usar la e minúscula o la E mayúscula.

Por ejemplo, 300 es lo mismo que  $3 \times 10^2$ ; por lo que en notación científica diríamos 3E2. Similarmente, 0,03 (la coma decimal es el punto decimal de los angloparlantes) es lo mismo que  $3 \cdot 10^{-2}$ . Ensayá con los siguientes ejemplos:

1.  $30 \times 10$

Puedes teclear en el ordenador:

```
?30*10  ENTER
300
```

o puedes teclear:

```
?3e1*1e1  ENTER
300
```

2.  $3000 \times 1000$

Teclea:

```
?3e3*1e3  ENTER
300000
```

3. 3000x0,001

Teclea:

?3e3\*1e-3 ENTER  
3

## Fundamentos 3:

## GRAFICOS, MODOS Y SONIDOS

El Ordenador Personal a Color AMSTRAD CPC464 dispone de tres **modos** de operación en lo referente a la presentación de imágenes en pantalla: Modo 0, Modo 1 y Modo 2.

Cada vez que se pone en marcha el ordenador, automáticamente adopta el Modo 1.

Para comprender estos tres modos diferentes, enciende el ordenador y pulsa la tecla marcada con **1**. Mantenla pulsada hasta que rellenes dos líneas de unos. Si contaras la cantidad de unos que hay en cada línea, verás que son 40. Eso significa que en el modo 1, el "prescrito" como normal, hay 40 columnas en cada línea de texto. Pulsa **ENTER** -y obviamente obtendrás un mensaje de error sintáctico- pero no te preocupes, es una manera rápida y simple de volver a conseguir el aviso **Ready**, que te dice que el ordenador está dispuesto para recibir tus órdenes.

Teclea ahora: **mode 0 ENTER**

Verás que los caracteres en la pantalla se ensanchan. Pulsa como antes la tecla con el **1** y mantenla pulsada hasta que rellenes dos líneas completas de unos. Si cuentas el número de unos en cada línea, verás que son 20. Eso significa que en el modo 0, hay 20 columnas por cada línea. Pulsa de nuevo **ENTER**.

Ahora teclea: **mode 2 ENTER**

Verás que éste es el modo en que los caracteres son más estrechos, y que si tecleas filas de unos como antes, contarás 80 columnas por cada línea. Eso significa que el modo 2 corresponde a una imagen de 80 columnas. Para resumir:

Modo 0 = 20 columnas  
Modo 1 = 40 columnas  
Modo 2 = 80 columnas

Finalmente, pulsas otra vez la tecla **ENTER**.

### COLORES

Hay una gama de 27 colores. Obviamente, en un monitor de fósforo verde (GT64) aparecen como diferentes matices de verde. Si has comprado el monitor GT64, puedes adquirir el modulador/alimentador MPI AMSTRAD con el fin de aprovechar las posibilidades de color en tu ordenador empleando tu televisor cassette.

En el Modo 0, hasta 16 de los 27 colores disponibles pueden aparecer en la pantalla en cualquier momento.

En el Modo 1, hasta 4 de los 27 colores pueden aparecer en pantalla en cualquier momento.

En el Modo 2, hasta 2 de los 27 colores pueden aparecer en pantalla en cualquier momento.

También podrás cambiar el borde de la pantalla (**BORDER**), el área en que aparecen los caracteres (**PAPER**) o la **PLUMA** con que el ordenador pintará los caracteres (**PEN**), y estas tres facetas, todas independientemente una de otra.

Los 27 colores disponibles se enumeran en la tabla 1, cada uno con el número que sirve de referencia a la **TINTA** que se usa tanto para el papel como para la pluma.

**TABLA MAESTRA DE COLORES**

Nº Tinta	Color Tinta	Nº Tinta	Color Tinta
0	Negro	14	Azul Pastel
1	Azul	15	Naranja
2	Azul Brillante	16	Rosa
3	Rojo	17	Magenta Pastel
4	Magenta	18	Verde Brillante
5	Malva	19	Verde Marino
6	Rojo Brillante	20	Ciano Brillante
7	Púrpura	21	Verde Lima
8	Magenta Brillante	22	Verde Pastel
9	Verde	23	Ciano Pastel
10	Ciano	24	Amarillo Brillante
11	Azul Cielo	25	Amarillo Pastel
12	Amarillo	26	Blanco Brillante
13	Blanco		

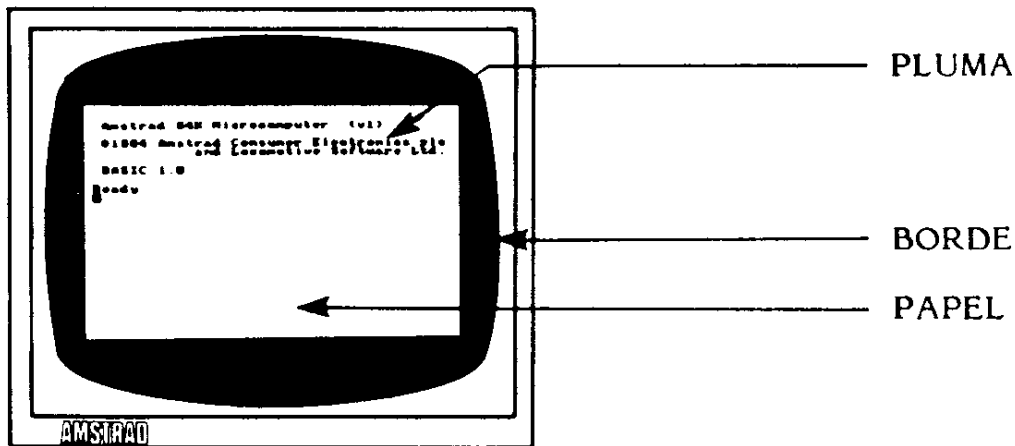
**Tabla uno:** Los números de tinta (**INK**) y los colores correspondientes.

Como mencionamos anteriormente, cada vez que se pone en marcha el ordenador adopta el modo 1 para las imágenes en pantalla. Los colores del **BORDE**, **PAPEL**, y **PLUMA** en este momento son:

Borde: Número de **TINTA**: 1 (azul)

Papel (el fondo): Número de **TINTA**: 1 (azul)

Pluma (los símbolos): Número de **TINTA**: 24 (amarillo brillante)



El BORDE es el área que rodea al PAPEL. (Observa que al poner en marcha el ordenador, tanto el BORDE como el PAPEL son azules). El PAPEL es el área de la pantalla rodeada por el BORDE, donde aparecen los caracteres. La PLUMA es la que usa el ordenador "para pintar los caracteres", y tiene plumas con diferente tinta.

Para explicar esto más detalladamente, es por lo que hemos asociado los comandos PEN y PAPER usados para las imágenes del monitor a realmente una PLUMA de escribir y a una hoja de papel para notas. A medida que el color de la TINTA con que se impregna la PLUMA se cambia, se cambia también el color con que los caracteres aparecen en pantalla. A medida que se cambia el color de la TINTA empleada para "entintar" el papel, así cambiará el color con que el fondo aparece en la pantalla.

Para cambiar el color del BORDE que rodea al papel:

```
border 0 ENTER
```

Con eso verás que el BORDE cambia de color, pasando del azul al negro. Si consultas la Tabla 1, verás que el color con número 0 corresponde al negro. El BORDE puede cambiarse a cualquiera de los colores de esa tabla mediante el comando **BORDER**, seguido del número de color que se desea.

Ahora teclea:

```
cls ENTER
```

para limpiar la pantalla.

Para ver cómo cambia el color del PAPEL, teclea:

```
paper 2 ENTER
```

Verás cómo con eso, el color del fondo que enmarca la palabra **Ready** ha cambiado a un color ciano brillante. Y no es exactamente el azul brillante que esperabas!!

Teclea ahora:

```
cls ENTER
```

para que se limpie la pantalla y quede completamente con el nuevo color de PAPEL.

Para ver cómo cambia el color de la PLUMA, teclea:

**PEN 3 ENTER**

Verás que el color de los caracteres ha cambiado y que la palabra **Ready** aparece con sus letras escritas en un rojo brillante, y no en un simple rojo.

Teclea ahora:

**cls ENTER**

Para que comprendas más fácilmente lo que ha sucedido aquí, debes también consultar la Tabla 2. Cuando se pone en marcha el ordenador, usa como número de PAPEL el número 0. Y si miras en la Tabla 2, en la primera columna que corresponde al número de PAPEL o de PLUMA, encontrarás el número 0. Si ahora sigues la línea correspondiente al 0, te encontrarás en la columna del modo 1, que figura como número de color el 1.

Si ahora pasas a consultar la tabla maestra de colores (Tabla1) verás que al número 1 le corresponde tinta azul, que es el color del PAPEL que observarás al poner en marcha el ordenador, como ya te hemos dicho.

En los ejemplos anteriores, cambiamos el papel con la instrucción **PAPER 2**. Si en la columna izquierda de la tabla 2, buscas el número 2 y luego sigues esa línea hasta la columna correspondiente al modo 1, verás que tienes como número de color el 20. Y volviendo a la tabla 1, verás que el color con número 20 es el ciano brillante que mencionamos.

## COLOR DE LA TINTA

Nº Papel/Pluma	Modo 0	Modo 1	Modo 2
0	1	1	1
1	24	24	24
2	20	20	1
3	6	6	24
4	26	1	1
5	0	24	24
6	2	20	1
7	8	6	24
8	10	1	1
9	12	24	24
10	14	20	1
11	16	6	24
12	18	1	1
13	22	24	24
14	Parpadeo 1,24	20	1
15	Parpadeo 16,11	6	24

Tabla dos: Referencias PAPEL/PLUMA/MODO/TINTA





La PLUMA usada cuando el ordenador se pone en marcha es la número 1. Consultando la Tabla 2 y buscando en la columna de modo 1 la que corresponde a una PLUMA con número 1, verás que es el número de color 24. Si consultas ahora la Tabla 1, verás que el número de color 24 es amarillo brillante, que es el color con que aparecen los caracteres (pintados por la PLUMA) cuando se pone en marcha el ordenador, como ya mencionamos. En los ejemplos, cambiamos el número de pluma mediante el comando **PEN 3**. Si consultamos la Tabla 2, veremos que para esa pluma número 3 y estando en el modo 1, le corresponde como número de color el 6. Y en la Tabla 1 vemos que el color 6 es rojo brillante. Que era lo que habíamos comentado.

En este momento, estamos pues usando un PAPEL número 2, y una PLUMA número 3. Podemos continuar cambiando estos colores y podemos hacerlo usando el comando **INK**. Este comando de TINTA, tiene dos números. El primero es el número de la PLUMA o del PAPEL que queremos cambiar; el segundo es el número que queremos asignar a la PLUMA o PAPEL respectivamente. Consulta la Tabla 1 para ver los números de los colores, como ejemplo te mostraremos cómo cambiar el color de papel con número 2 a negro, y el color de la pluma con número 3 a blanco brillante. En la Tabla 1, verás que el número de color para el negro es el 0, y el número de color para el blanco brillante es el 26.

Por lo tanto, puedes teclear ahora:

**ink 2,0 ENTER**

(Como explicamos, 2 es el número del PAPEL vigente, y 0 es el número de color del negro).

Ahora teclea:

**ink 3,26 ENTER**

(3 es el número de PLUMA vigente y 26 es el número de color blanco brillante).

Ahora restaura completamente el ordenador a sus condiciones iniciales, pulsando las teclas **CTRL, SHIFT y ESC**.

Como explicamos anteriormente, cada vez que se pone en marcha la máquina, o se restaura a las condiciones iniciales usando las teclas **CTRL, SHIFT y ESC**, el PAPEL tiene por número de 0 y la PLUMA tiene por número el 1. El color de un PAPEL con número 1, es azul y el de una PLUMA con número 24, es amarillo brillante. Eso se teclearía como **ink 0,1** para el PAPEL, e **ink 1,24** para la PLUMA. Para cambiar inmediatamente estos colores a caracteres pintados en blanco (PLUMA) sobre un fondo negro (PAPEL), teclea:

**ink 0,0 ENTER**

## FUNDAMENTOS

Y luego teclea:

```
ink 1,26 ENTER
```

### COLORES PARPADEANDO

Es posible hacer que los colores de los caracteres alternen entre uno y otro, logrando el parpadeo o destello. Eso puede lograrse añadiendo un número extra de color dentro del comando **INK** relativo a la PLUMA. Por ejemplo, para ver los caracteres alternativamente en blanco brillante y rojo brillante, teclea:

```
ink 1,26,6 ENTER
```

En este caso, 1 es el número de la PLUMA, mientras que 26 es el nuevo color blanco brillante y 6 es el otro color empleado rojo brillante.

También es posible hacer que el color del papel situado detrás de los caracteres sea el que alternativamente cambie de un color a otro. Eso puede lograrse como antes, añadiendo un número extra de color al comando **INK** tecleado para el PAPEL.

Por ejemplo, para ver que el PAPEL cambia alternativamente entre verde y amarillo brillante, por detrás de los caracteres, teclea:

```
ink 0,9,24 ENTER
```

En este caso, 0 es el número del PAPEL, mientras que 9 es el nuevo color verde y 24 el color alternativo amarillo brillante.

Restaura ahora el ordenador usando **CTRL, SHIFT y ESC**.

Observa que el modo 0, en la Tabla 2, figuran dos números de tinta o de papel (números 14 y 15) que están prescritos como colores parpadeantes. En otras palabras, no es necesario añadir un número extra en el comando de tinta, cuando se usan esos números 14 ó 15.

Teclea lo siguiente:

```
mode 0 ENTER
```

```
pen 15 ENTER
```

y en la pantalla, verás que la palabra **Ready** es alternativamente pintada en azul cielo y rosa.

Ahora teclea:

```
paper 14 ENTER
```

```
cls ENTER
```

Verás que además de que parpadeen los caracteres de la palabra **Ready** entre azul cielo y rosa, el fondo de la imagen, el PAPEL, también cambia alternativamente de color entre azul y amarillo brillante.

Y encima además, es posible cambiar estos colores parpadeando prescritos en el ordenador, si le tecleas un nuevo comando **INK** tanto para el **PAPEL** como para la **PLUMA**. Para cambiar el color de la **PLUMA** a un parpadeo entre negro y blanco brillante, teclea:

```
ink 15,0,26 ENTER
```

En este caso, 15 es el número de **PLUMA**, mientras que 0 es el número de color del negro, y 26 es el número de color blanco brillante.

Finalmente, también es posible hacer que el **BORDE** parpadee, cambiando entre dos colores, si añadimos un número extra de color dentro del propio comando **BORDER**. Teclea:

```
border 6,9 ENTER
```

Y verás que todo el área del **BORDE** está cambiando entre rojo brillante y verde.

Ahora restaura el ordenador mediante **CTRL, SHIFT** y **ESC**, y prepárate que todavía hay más cosas.

## PROGRAMA DE DEMOSTRACION

Para mayor demostración de los colores disponibles, teclea y ejecuta el siguiente programa:

```
10 mode 0: ink 0,2:ink 1,24: paper 0 ENTER
20 pen 1: for b=0 to 26: border b ENTER
30 locate 3,12:print"COLOR DEL MARGEN";B ENTER
40 sound 4,(40-b) ENTER
50 for t=1 to 600:next t:next b:cls ENTER
60 for p=0 to 15:paper p:pen 5:print "FONDO";
  p=print ENTER
70 for n=0 to 15:pen n:print "TRAZO";n ENTER
80 sound 1,(n*20+p) ENTER
90 for t=1 to 100:next t:next n ENTER
100 for t=1 to 1000:next t:cls:next p ENTER
110 cls:paper 0: pen 1:locate 7,12: print
  "FIN": for t=1 to 2000: next t ENTER
120 mode 1: border 1:ink 0,1:ink 1,24:paper 0:
  pen 1 ENTER

run ENTER
```

## FUNDAMENTOS

Hemos incluido además, algunos sonidos en el programa. Más adelante los explicaremos en la sección pertinente.

### GRAFICOS

De ahora en adelante, no te pediremos que pulses la tecla **ENTER** después de cada línea. Simplemente supondremos que ya se ha convertido en un hábito automático en tí.

Hay cierto número de caracteres en la memoria del ordenador que corresponden a los símbolos (letras, cifras, signos, etc.) que usamos para las comunicaciones con el ordenador. Para obtener cualquiera de esos símbolos, usamos la palabra clave **chr\$(X)**, siendo la X incluida dentro de los paréntesis el número que corresponde a ese símbolo, que debe estar dentro de la gama 32 a 255.

Pulsa **CTRL, SHIFT y ESC** para restaurar el ordenador y luego teclea:  
`print chr$(250)`

En la pantalla verás el símbolo que corresponde al carácter con número 250, y en este caso, es un hombre en postura de caminar hacia la derecha.

Para ver todos los símbolos que pueden aparecer en pantalla, junto con su número asociado de carácter, teclea el siguiente programa:

```
10 for n=32 to 255: print n;chr$(n)
20 next n

run
```

Como referencia, la gama de caracteres junto con los números asociados y los símbolos que representan, aparece en el Apéndice III al final de este libro.

### UBICANDO EL CURSOR

Existe un comando que se usa para ubicar el cursor de texto en una parte específica de la pantalla. A menos que se **UBIQUE** el cursor en un sitio determinado, mediante el comando **LOCATE**, el cursor de caracteres comienza en la esquina superior izquierda de la pantalla, que corresponde a las coordenadas x (eje horizontal) e y (eje vertical) de valores 1,1. En el modo de imagen 1, sabemos que hay 40 columnas y 25 líneas dentro del área de exposición de caracteres. Para situar un determinado carácter en el centro de la línea superior de la pantalla, estando en el modo 1, usaríamos la pareja 20,1 como coordenadas x,y.

Para ver esto, teclea: (recuerda que después de cada línea has de pulsar **ENTER**).

```
mode 1
```

...limpia la pantalla, y el cursor a su posición de base.

```
10 locate 20,1
20 print chr$ (250)
run
```

Simplemente para comprobar que está en la línea superior, puedes teclear:

```
border 0
```

Con esto, el BORDE se habrá puesto en negro, y verás más fácilmente que el hombrecillo está situado en le medio de la línea superior de la pantalla.

En el modo 0, sólo hay 20 columnas, pero las mismas 25 líneas. Si ahora tecleas:

```
mode 0
run
```

Verás que el hombre aparece ahora en la esquina superior derecha de la pantalla. Eso sucede porque la coordenada x con valor 20, es la última columna si está en el modo 0.

En el modo 2, hay 80 columnas y 25 líneas. Usando este mismo programa, probablemente serás capaz de predecir dónde aparecerá el hombrecillo. Teclea pues:

```
mode 2
run
```

y ahora regresa al modo 1, simplemente tecleando:

```
mode 1
```

Ahora experimenta por tí mismo, modificando la ubicación del cursor y el número de la función carácter para situar diferentes símbolos en cualquier parte de la pantalla. Meramente como ejemplo, teclea:

```
locate 20,12:print chr$ (240)
```

Verás una flecha en el centro de la pantalla. Observa que en este comando:

- 20 es el valor de la coordenada horizontal (x) en la gama 1 a 40.
- 12 es el valor de la coordenada vertical (y) en la gama 1 a 25.
- 240 es el número de carácter cuyo símbolo se expone en pantalla (en la gama 32 a 255).

Para conseguir que el símbolo asociado al carácter 250 sea repetido a lo ancho de la pantalla, teclea el siguiente programa:

```
5 cls
10 for x = 1 to 39
20 locate x,20
40 print chr$ (250)
50 next x
60 goto 5
run
```

## FUNDAMENTOS

Pulsa la tecla **ESC** dos veces para escapar de este programa que nunca termina.

Con el fin de quitar de la pantalla el carácter anterior, antes de mostrar otro carácter, puedes teclear:

```
40 print " "; chr$(250)
```

(Al escribir esta línea 40, automáticamente sustituye a la línea que previamente tenía como número el 40).

Ejecuta ahora el programa para comprobarlo.

```
run
```

Para mejorar el movimiento del carácter a lo ancho de la pantalla, añade la siguiente línea:

```
30 call &bd19
```

Este programa puede mejorarse adicionalmente, perfeccionando el movimiento al añadirle algunos bucles de retardo y usando un símbolo diferente para la vuelta.

Teclea:

```
list
```

Y ahora añade las siguientes líneas a este programa:

```
60 for n = 1 to 300 : next n
65 for x = 39 to 1 step -1
70 locate x, 20
75 call &bd19
80 print chr$(251); " "
85 next x
90 for n = 1 to 300:next n
95 goto 10
run
```

Intenta este programa interesante a pesar de su brevedad. Hemos añadido algunos otros comandos que explicaremos en capítulos subsiguientes. Por ahora, simplemente teclea:

```
new
```

```
10 mode 1
20 locate 21,14:print chr$(244)
30 tag
40 for x=0 to 624 step 2
50 mover -16,0
60 if x<308 or x>340 then y=196:goto 90
70 if x<324 then y=x-104:goto 85
```

```

80 y=536-x
85 sound 1,0,20,7
90 move ox, oy:print " ";:ox=x:oy=y
100 move x,y
110 if (x mod 4) = 0 then print chr$(250) ;
    else print chr$(251) ;
120 for n=1 to 4: call &bd19:next n
130 next x
140 tagoff
150 goto 20
run

```

## PLOT

A diferencia del comando de ubicación del cursor, el comando de PINTAR se usa para determinar la posición del cursor interno usado para gráficos, que usa otros ejes de coordenadas, y con diferentes medidas. De hecho, determina la **resolución** en pantalla, y lo que se pinta es un diminuto punto que solemos llamar MOTA (o bien pixel como abreviatura de elemento pictórico).

Observa que el cursor gráfico es invisible, y es también independiente del cursor de texto o caracteres.

Puedes pintar 640 motas dentro de una línea horizontal, y 400 motas dentro de cada línea vertical. Usando las coordenadas x, y, se toma como origen la esquina inferior izquierda de la pantalla, que por tanto tiene coordenadas 0,0. A diferencia del comando de ubicación del cursor usado para caracteres, las coordenadas usadas para las motas son independientes del modo 0, 1, 2 en que esté el ordenador.

Para experimentar con esto, restaura primero el ordenador usando **CTRL, SHIFT y ESC**, y luego teclea, recordando pulsar **ENTER** al final de cada línea, el siguiente comando:

```
plot 320,200
```

Y verás que aparece en el centro de la pantalla un punto diminuto: la MOTA.

Cambia ahora el modo, tecleando:

```
mode 0
plot 320,200
```

Verás que la mota continúa todavía en el centro de la pantalla, pero ahora es más grande en su dimensión horizontal.

Cambia el modo de nuevo y teclea el mismo comando para ver el efecto en el modo 2. Por tanto teclea:

```
mode 2
plot 320,200
```

La mota continúa en el centro, pero ahora es mucho más pequeña.

Pinta diferentes motas en diversos sitios de la pantalla y usando diversos modos, con el fin de acostumbrarte a este comando. Cuando hayas concluido, regresa al modo 1 y limpia la pantalla, simplemente con el comando:

```
mode 1
```

### DRAW

Restaura primero el ordenador usando **CTRL, SHIFT y ESC**. Este comando **DRAW**, nos permite DIBUJAR una línea a partir de la corriente posición del cursor de gráficos. Para ver esto con más detalle, dibuja un rectángulo en la pantalla usando el programa del ejemplo. En él se comienza por situar el cursor gráfico mediante el comando de PINTE. Luego se traza una línea desde esa posición del cursor, hacia la esquina superior izquierda, luego desde allí hacia la esquina superior derecha, etc.

Teclea:

```
5 cls
10 plot 10,10
20 draw 10,390
30 draw 630,390
40 draw 630,10
50 draw 10,10
60 goto 10
run
```

Pulsa dos veces **ESC** para "interrumpir" la ejecución de este programa sin fin.

Ahora añade al programa las siguientes líneas que dibujan un segundo rectángulo dentro del primero. Teclea:

```
60 plot 20,20
70 draw 20,380
80 draw 620,380
90 draw 620,20
100 draw 20,20
200 goto 10
run
```

Pulsa dos veces la tecla **ESC** para "rumpir" la ejecución de este programa.

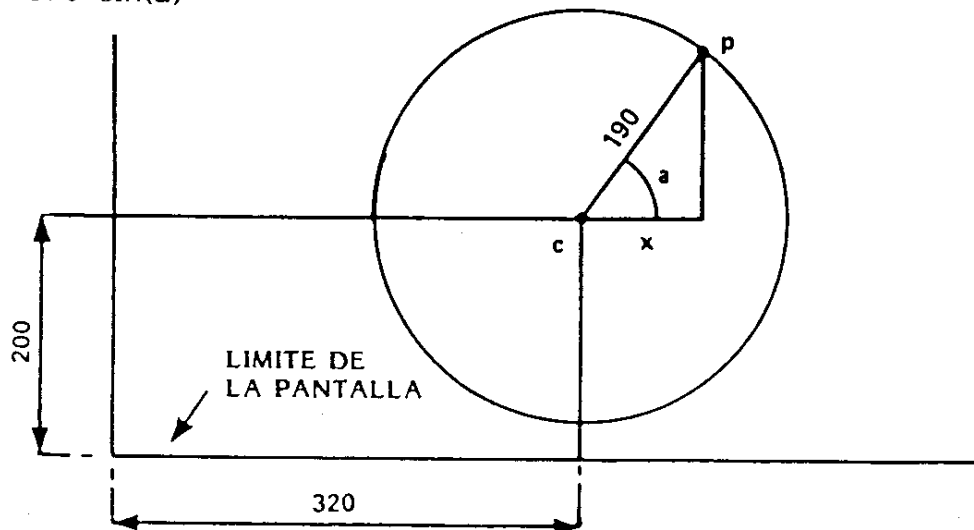


## CIRCULOS

Los círculos pueden ser pintados (mota a mota) o dibujados (a trazos o líneas). Un método para formar un círculo, es pintar las coordenadas  $x$ ,  $y$  de cada uno de los puntos de la circunferencia. Consulta la figura inferior y verás cómo el punto  $p$  de la circunferencia puede pintarse usando como coordenadas  $x$  e  $y$  las dadas por las ecuaciones:

$$x = 190 * \cos(a)$$

$$y = 190 * \sin(a)$$



Vamos ahora a comenzar a usar la palabra clave **NEW** antes de teclear un programa, para que deje como NUEVA la memoria del ordenador. Es decir, con este comando se le dice al ordenador que vacíe completamente la memoria en forma similar a lo que hacemos con la restauración de condiciones iniciales (**CTRL, SHIFT y ESC**). Sin embargo, difiere de la restauración en que no se limpia la pantalla, sino solamente la memoria. Es útil si queremos mantener en pantalla el programa viejo para que nos sirva de referencia al escribir un NUEVO programa.

Volvamos ahora a ver cómo pintamos un círculo. En el programa anterior, los puntos pintados lo eran con respecto a la esquina inferior izquierda de la pantalla. Si ahora queremos situar el círculo en el centro de la pantalla, tendríamos que pintar el centro del círculo en las coordenadas 320, 200; y luego pintar todos los puntos del círculo con relación a ese centro, añadiéndole las coordenadas  $x$ ,  $y$  como función del ángulo.

Por lo tanto, para pintar el círculo, sería algo como esto:

```
new
5 cls
10 for a=1 to 360
15 deg
20 plot 320,200
30 plot 320+190*cos(a),200+190*sin(a)
40 next
run
```

## FUNDAMENTOS

El radio del círculo puede reducirse disminuyendo la cifra de 190 que aparece en el programa (el 190 se refiere a número de motas).

Para ver el efecto de las medidas angulares que usamos, (entre radianes o grados), suprime la línea 15 del programa anterior que fija esa medida en grados. Simplemente teclea 15 y **RETURN**, y ejecuta de nuevo el programa.

Para ver un círculo sólido (y no simplemente la circunferencia) dibujado mediante líneas trazadas desde el centro, edita la línea 30 sustituyendo la palabra clave **plot** con la palabra clave **draw**. La línea 30 será pues:

```
draw 320+190*cos(a), 200+190*sin(a)
```

Ensayá esto también con y sin la línea 15 que anteriormente suprimimos.

Observarás que en la línea 40 de este programa hemos usado **next** en lugar de **next a**.

Es aconsejable teclear simplemente **next** en los programas. El ordenador determinará a qué bucle e instrucción **FOR** está asociado esa instrucción **NEXT**. Sólo en programas donde hay numerosos bucles **FOR...NEXT**, puede ser aconsejable añadir la variable de control correspondiente después de cada instrucción **NEXT** con el fin de que puedas identificar más fácilmente los bucles al examinar el programa.

## ORIGEN

En el programa anterior, usamos el comando **PLOT** para designar el centro del círculo, y luego añadíamos las coordenadas x, y de los puntos de la circunferencia relativas a ese centro. En lugar de efectuar esta suma, podemos usar el comando que fija el **ORIGEN** de coordenadas. Si con él señalamos el centro de un círculo, simplemente después sólo tenemos que precisar las coordenadas x, y de todos los puntos de la circunferencia (en pasos de un grado) con respecto a este nuevo origen. Para verlo, teclea:

```
new
5 cls
10 for a = 1 to 360
15 deg
20 origin 320,200
30 plot 190*cos(a),190*sin(a)
40 next
run
```

En este programa, también puedes alterar la línea 15 para quitar los grados, y/o la línea 30 para dibujar un círculo en lugar de pintar una circunferencia.

Para pintar cuatro pequeños círculos en pantalla, teclea el siguiente programa:

```
new
5 cls
10 for a=1 to 360
15 deg
20 origin 196,282
30 plot 50*cos(a),50*sin(a)
40 origin 442,282
50 plot 50*cos(a),50*sin(a)
60 origin 196,116
70 plot 50*cos(a),50*sin(a)
80 origin 442,116
90 plot 50*cos(a),50*sin(a)
100 next
run
```

De nuevo, también puedes experimentar con la línea 15 y modificar las líneas 30, 50, 70 y 90 para usar el comando DIBUJE en lugar del comando PINTE.

## GOSUB RETURN

Si hay una serie de instrucciones dentro de un programa que efectúan una tarea determinada que se realiza un cierto número de veces en el programa de forma "rutinaria", pueden considerarse estas instrucciones como una **subrutina**, y hacer que el ordenador VAYA a efectuar esa tarea, mediante el comando **GOSUB** seguido del número de línea en que comienza la subrutina.

El final de dichas subrutinas, se marca tecleando la instrucción que hace que el ordenador VUELVA (**RETURN**) al punto en que se desvió. Es decir, cuando el ordenador encuentra la instrucción **RETURN**, volverá a la instrucción que va inmediatamente detrás de aquélla en que se produjo el desvío mediante el comando **GOSUB**.

En el programa anterior, la instrucción **plot 50\*cos(a), 50\*sin(a)** se repetía cuatro veces. Esta instrucción puede ser considerada y separada como una subrutina, hacia la que se desvía el ordenador cada vez que se necesita usando la instrucción **GOSUB**. Observalo en el siguiente programa:

```
new
5 cls
10 for a=1 to 360
15 deg
20 origin 196,282
30 gosub 120
40 origin 442,282
```

## FUNDAMENTOS

```
50 gosub 120
60 origin 196,116
70 gosub 120
80 origin 442,116
90 gosub 120
100 next
110 end
120 plot 50*cos(a),50*sin(a)
130 return
run
```

Observa también que en la línea 110 usamos la instrucción **END**; porque de lo contrario, el programa obviamente continuaría después de la instrucción 100 y llevaría a cabo la 120, que sólo se necesita cumplimentar cuando la subrutina sea citada mediante **GOSUB**.

Para concluir esta sección, ensaya con el siguiente programa que incorpora un montón de los comandos de programación y de las palabras clave que a estas alturas ya debes comprender. Teclea:

```
new
10 mode 0:border 6:paper 0:ink 0,0
20 gosub 160:for x=1 to 19:locate x,3
30 pen 15:print " ";chr$(238)
40 for t=1 to 50:next t:sound 2,(x+100)
50 next x:gisub 160:for b=3 to 22
60 locate 20,b:pen 7:print chr$(252)
70 cls:gosub 160:next b
80 sound 2,0,100,15,0,0,1
90 gosub 160:border 16,24:locate 20,25
100 pen 14:print chr$(253);
110 for t = 1 to 1000:next t
120 border 6:gosub 160:for f=3 to 24
130 locate 10,(25-f):pen 2
140 print chr$(144):cls:gosub 160
150 sound 7,(100-f),5:next f:goto 10
160 locate 10,25: pen 12
170 print chr$(239):return
```

## SONIDOS

Los efectos sonoros son producidos por un altavoz incorporado dentro del propio ordenador. Si estás usando el modulador/alimentador MP1 y un televisor doméstico, sigue manteniendo en el mínimo el control de volumen del televisor.

El nivel de sonido del ordenador, puede ajustarse usando el mando de volumen, marcado **VOLUME** en el costado derecho del ordenador. El sonido puede también enviarse hacia el enchufe de entrada de un sistema estéreo, usando el enchufe de entrada/salida, marcado **I/O** en la parte izquierda del panel posterior del ordenador. Eso te permitiría escuchar el sonido generado por el ordenador a través de los altavoces o auriculares de tu equipo de alta fidelidad.

El comando **SOUND**, tiene 7 parámetros para el SONIDO. Los primeros 2 de ellos, deben usarse siempre, el resto es opcional. El comando se teclea en la forma:

**SONIDO** estado del canal, período del tono, duración, volumen, envolvente de volumen, envolvente de tono, período del ruido.

En los ejemplos siguientes, teclearemos un 1 como estado del canal; en otras palabras, el número de referencia.

### Período del Tono

Consulta el Apéndice VII y verás que la nota DO media, tiene un período del tono de 478.

Teclea:

`new`

`10 sound 1, 478`

`run`

Oirás una breve nota que es la "media c" y dura 0,2 segundos.

### Duración

Cuando no se especifica duración del sonido generado, sonará durante 0,2 segundos, la unidad de duración es de 0,01 segundos. Para hacer que una nota dure 1 segundo, usaríamos por lo tanto 100 en este parámetro. Para que dure 2 segundos, usaríamos 200, etc. Teclea:

`10 sound 1, 478, 200`

`run`

Y ahora oirás durante 2 segundos la nota media c.

### Volumen

Este número especifica el volumen con que arranca una nota. El número debe estar en la gama 0 a 7. Pero sin embargo, si se especifica un envolvente de volumen, la gama se amplía de 0 a 15. Si no se utiliza en el comando ningún número, se presupone 4 que es el valor prescrito para las omisiones. Teclea:

`10 sound 1, 478, 200, 3`

Y observa el volumen de este sonido. Luego teclea lo mismo pero usando un número de volumen mayor:

`10 sound 1, 478, 200, 7`

## FUNDAMENTOS

Y comprobarás que es mucho más alto que el anterior.

### Envolvente de Volumen

El comando para el envolvente de volumen usa la palabra reservada **env**. Normalmente tiene 4 parámetros: los últimos 3 aparecen en cualquiera de las cinco secciones de envolvente que hay disponibles como máximo. Aquí solamente usamos uno de ellos. Las explicaciones adicionales aparecen en el capítulo 6.

**env** número del envolvente, número de pasos, amplitud (tamaño) del paso, tiempo del paso.

### Número del envolvente

Es el número dado a un determinado envolvente, de modo que pueda especificarse en el comando **SOUND**. La gama de número de envolventes es de 0 a 15.

### Número de pasos

Se usa en conjunción con el tiempo del paso. Por ejemplo, puedes desear tener 10 pasos de 1 segundo cada uno. En tal caso, el número de pasos es 10. La gama de valores posibles para los números de pasos es de 0 a 127.

### Amplitud del paso

Cada paso puede variar en amplitud desde un nivel 0 hasta un nivel 15 con respecto al paso precedente. Los 15 niveles de amplitud son los mismos que se emplean en el comando **SOUND**. Sin embargo, cada paso puede ajustarse desde -128 a +127 de manera que no solamente se puede variar la amplitud subiéndola o bajándola en la forma obvia, sino que también puede variarse usando números mayores de 15 para obtener algunos efectos especiales y extraños. La gama de números empleada para las amplitudes del paso es de -128 a +127.

### Tiempo del paso

Este número especifica el tiempo transcurrido entre los pasos en unidades de 0,01 segundo (1/100 de segundo). La gama de números para el tiempo del paso es de 0 a 255. Por lo tanto, el tiempo más largo transcurrido entre los pasos es de 2,55 segundos.

Para experimentar con la envolvente de volumen, teclea el siguiente programa:

```
5 env 1, 10, 1, 100
10 sound 1, 284, 1000, 1, 1
run
```

La línea 10 especifica un sonido con un período del tono de 284 (convenio internacional a) que dura 10 segundos y con un volumen de arranque de 1, y usando la envolvente de volumen número 1; envolvente que está definida en la línea 5 y consta de 10 pasos, aumentando la amplitud en 1 a cada paso, y con un tiempo del paso de 1 segundo (100x0,01 segundos).

Cambia la línea 5 según las instrucciones que te presentamos ahora, y luego ejecuta el programa cada vez para oír el efecto obtenido al cambiar la envolvente de volumen.

5 env 1, 100, 1, 10

5 env 1, 100, 2, 10

5 env 1, 100, 4, 10

5 env 1, 50, 20, 20

5 env 1, 50, 2, 20

5 env 1, 50, 15, 30

Y finalmente ensaya con esto:

5 env 1, 50, 2, 10

Observarás que la mitad del tiempo que dura el sonido, la amplitud permanece constante. Es debido a que el número de pasos que estipulamos era 50, y el tiempo de cada paso era de 0,1 segundos. Por lo tanto, el tiempo durante el que la amplitud de la envolvente varió fue sólo de 5 segundos; mientras que la duración del sonido especificada en el comando **SOUND** de la línea 10 fue de 10 segundos.

Experimenta por tí mismo, para ver los tipos de sonido que puedes generar.

### Envolvente de Tono

El comando para la envolvente de tono es **ent**.

Tiene normalmente 4 parámetros. Los 4 parámetros aparecen en cualquiera de las 5 secciones de envolvente que hay disponibles como máximo.

Aquí solamente estamos usando una de ellas. Las explicaciones completas aparecerán en el capítulo 6.

**ent** número de envolvente, número de pasos, período de tono del paso y tiempo del paso.

### Número de envolvente

Es el número dado a la envolvente particular de forma que pueda especificarse en el comando **SOUND**. La gama de números de envolvente es de 1 a 15.

## FUNDAMENTOS

### Número de pasos

Se usa en conjunción con el tiempo del paso. Por ejemplo, puedes desear tener 10 pasos de 1 segundo cada uno. La gama de números de paso es de 0 a 239.

### Período de Tono del Paso

El período del tono de cada paso puede variar entre -128 a +127. Los valores negativos aumentan la frecuencia de las notas (haciendo las notas más agudas). El período más breve del tono es 0. Eso debes recordarlo cuando calcules envolventes del tono. La gama completa de períodos del tono se muestra en el Apéndice VII. Los números para períodos de tono del paso, debe estar en la gama de -128 a +127.

### Tiempo del Paso

Este número especifica el tiempo del paso en unidades de 0,01 segundo (1 centésima de segundo). La gama de números para el tiempo del paso es de 0 a 255. Por lo tanto, el tiempo del paso más largo es de 2,55 segundos.

Para experimentar con la envolvente del tono, teclea el siguiente programa:

```
5 ent 1, 100, 2, 2
10 sound 1, 284, 200, 15, 0, 1
run
```

La línea 10 especifica un sonido con un período de tono de 284 (convenio internacional a), que dura 2 segundos, con un volumen de arranque de 15 (el máximo), con una envolvente de volumen 0 y con una envolvente de tono designada por el número 1.

La línea 5 es la envolvente de tono designada con el número 1, y consta de 100 pasos, creciendo en 2 a cada paso el período del tono (reduciendo la frecuencia del tono, cada 0,02 segundos, que es el tiempo del paso).

Cambia ahora la línea 5 a los valores que te presentamos a continuación, y luego ejecuta el programa cada vez para apreciar el efecto de cambiar la envolvente del tono:

```
5 ent 1, 100, -2, 2
5 ent 1, 10, 4, 20
5 ent 1, 10, -4, 20
```

Ahora sustituye el comando **SOUND** y la envolvente del tono, tecleando:

```
5 ent 1, 2, 17, 70
10 sound 1, 142, 140, 15, 0, 1
15 goto 5
run
```

Ejecútalo y luego pulsa la tecla **ESC** dos veces para "interrumpir" la ejecución de este programa sin fin.



Ahora puedes emplear los comandos de sonido, envolvente del volumen, y envolvente del tono, para generar diferentes sonidos. Comienza tecleando:

```
new
5 env 1,100,1,3
10 ent 1,100,5,3
20 sound 1,284,300,1,1,1
run
```

Luego sustituye la línea 10 tecleando:

```
10 ent 1,100,-2,3
```

Ahora sustituye todas las líneas, tecleando:

```
5 env 1,100,2,2
10 ent 1,100,-2,2
20 sound 1,284,200,1,1,1
run
```

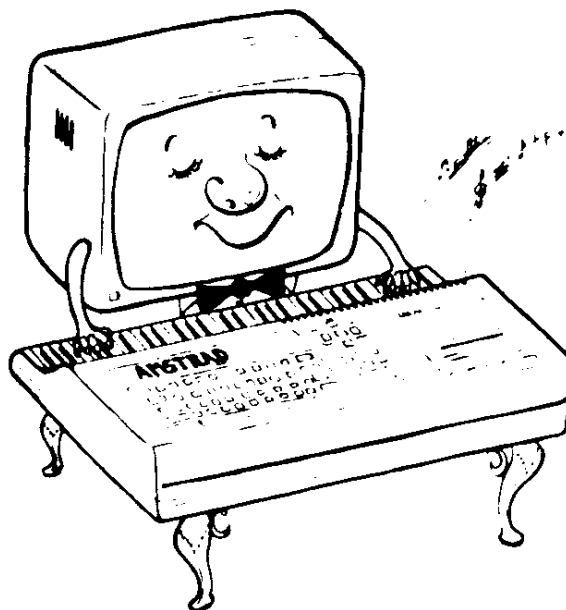
Y finalmente ensaya con las variaciones que se te ocurran a tí.

### Ruido

Se puede añadir ruido al final del comando **SOUND**. Se dispone de una gama de ruidos con números del 1 al 31. Ensáyalo, simplemente añadiendo el número del ruido al final de los parámetros del comando **SOUND** y usando todavía el comando de envolvente. Sustituye las líneas 5 y 20, tecleando ahora:

```
5 env 1,100,3,1
10 sound 1,200,100,1,1,1,5
run
```

Dé nuevo, puedes ensayar algunos sonidos poco habituales, modificando la envolvente del volumen y el comando de sonido, y añadiendo o no el ruido.



# 1. Arranque

Para aquéllos que os habéis saltado la introducción para principiantes, los detalles de instalación, puesta en marcha y familiarización con el teclado, os advertimos que si encuentras confusa la terminología que usamos, es conveniente que repases rápidamente la sección de introducción de los FUNDAMENTOS.

Temas comentados en este capítulo:

- \* Convenios de notación usados en esta Guía de Usuario
- \* Puesta en marcha
- \* Familiarización con el teclado

No importa el grado de experiencia que tengas en programación y ordenadores, cerciorate que sigues fielmente las instrucciones de puesta en marcha. Si simplemente has abierto el embalaje y no puedes esperar a comenzar adecuadamente, este capítulo te dirá todo lo que necesitas saber para satisfacer tu curiosidad inicial y pasar a sumergirte en las aplicaciones del BASIC. Esta sección está planteada para usuarios que ya tienen alguna familiaridad con los ordenadores. Los principiantes debieran comenzar con el capítulo de introducción de la sección de FUNDAMENTOS.

**IMPORTANTE - DEBES leer este párrafo:**

## Terminología

Con el objeto de aclarar las referencias en el texto con respecto a las teclas del ordenador, así como del texto que forma parte de los listados de programas, hemos adoptado los siguientes convenios, que usaremos a partir de ahora en esta Guía.

**ENTER:** Teclas a las que no está asociado un símbolo "visivo" en la pantalla, se muestran en esta forma, siempre y cuando no den lugar a confusión; en caso contrario se encierran entre corchetes "[ ]".

**QWERTYuiop?":** Teclas que sí corresponden con un símbolo "visivo" en pantalla, aparecen de esta forma, sin precisar si corresponden al símbolo superior o inferior que aparece marcado en la propia tecla.

**10 FOR N = 1 to 1000** : Texto que aparece en pantalla, o lo que los comandos que has de imponer mediante el teclado, se muestran en este estilo de letra, observa la diferencia entre el 0 y la O mayúscula.

Se supone que terminas cada línea de instrucciones o de comandos pulsando siempre la tecla **ENTER**, para indicarle que **VALE** y pase a cumplimentar lo que le hayas dicho; por lo tanto, no la repetiremos en el listado en lo que queda de Guía, aunque antes sí lo hayamos hecho.

Además, se supone que siempre ejecutarás cada programa ejemplo, después de haberlo tecleado y usando el comando **RUN**. BASIC convierte todas las palabras claves impuestas por teclado en minúsculas, a letras mayúsculas cuando se lista el programa. Los ejemplos mostrados a partir de ahora, usarán mayúsculas para las palabras clave, dado que así es como aparecen los programas al ser listados. Si tú tecleas usando letras minúsculas, será más conveniente para detectar los errores de tecleado más rápidamente, dado que las palabras clave mal deletreadas, seguirán mostrándose en minúsculas cuando se liste el programa.

### 1.1.1 ABRE LA CAJA DE LAS SORPRESAS

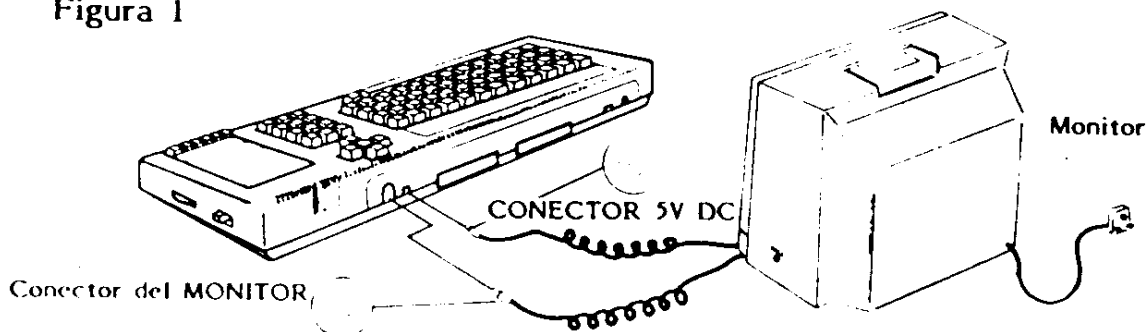
#### El Monitor a Color

##### IMPORTANTE

Por favor, consulta las instrucciones de **Puesta en Marcha** detalladas al comienzo de esta Guía de Usuario, y que describe cómo conectar el cable de alimentación de tu equipo a una clavija adecuada.

Con el ordenador conectado correctamente, tal y como se muestra en la figura 1, enciende el monitor, y luego el ordenador usando el interruptor deslizante del costado derecho.

Figura 1



Después de aproximadamente 30 segundos de "tiempo de calentamiento", aparecerá en el monitor:

**Amstrad 64K Microcomputer (v1)**

**©1984 Amstrad Consumer Electronics plc  
and Locomotive Software Ltd.**

**BASIC 1.0**

**Ready**  
  
 Cursor

Este mensaje se conoce con varios nombres, tales como "Bienvenida", "Restauración", "Despertador", "Mañanero" o "Saludo", etc. y lo importante es que indica que el ordenador está completamente RESTAURADO a sus condiciones iniciales -una condición que ocurre siempre que se pone en marcha después de haberlo apagado, y después de imponer por teclado, una secuencia correcta de tres teclas especiales que se han pulsado simultáneamente. Es decir, pulsando consecutivamente **CTRL, SHIFT y ESC**, y manteniendo pulsada la anterior hasta que las tres están pulsadas y luego soltándolas, se logra la restauración del sistema. Ensáyalo ahora, antes de teclear ninguna otra cosa.

Ajusta el control de brillo, situado en el costado derecho del monitor. El color está prefijado en fábrica, así que si deseas cambiar el color de la pantalla (lo habitual es símbolos en oro sobre un fondo azul), debes hacerlo mediante las instrucciones programables pertinentes, para lo que tendrás que saltar al capítulo "Lo primordial en Gráficos" y al capítulo 8 sobre palabras reservadas en BASIC. Si no te importa que nos anticipemos un poco, aquí hay un breve programa que te dará una de las mejores y más legibles combinaciones de colores para meter textos, usando el modo de pantalla en que la resolución es de 80 columnas por línea.

Teclea:

```
10 MODE 2
20 INK 1,0
30 INK 0,13
40 BORDER 13
```

...que también podrías teclear como una sola línea de comandos ejecutada inmediatamente.

Si el programa anterior no significa nada para tí, o no puedes ver si lo has tecleado correctamente, es señal de que necesitas repasar los FUNDAMENTOS PARA PRINCIPIANTES, sección que encontrarás al principio de esta Guía.

Comprobarás que la imagen con 80 columnas es con mucho, el modo más provechoso para desarrollar programas -puede que te guste GUARDAR el breve programa anterior (cuando ya hayas leído totalmente el capítulo 2) al principio de una cinta en blanco, para ahorrarte teclear cada vez que pones en marcha el ordenador.

**¡PRECAUCION!** Un brillo elevado en el monitor a color significa que debes ser cuidadoso para evitar los esfuerzos visuales cuando estás sentado muy cerca, o usar niveles de brillo que sean superiores a los elegidos por las condiciones luminosas ambientales. En el momento que sientas molestias en la vista, apaga y haz otra cosa, pero en primer lugar, evita siempre:

1. Trabajar con suficiente luz ambiental para poder leer este manual con facilidad. Si estás leyendo este manual con el reflejo de la pantalla, es una de las situaciones que vigorosamente no recomendamos.
2. Usa el mínimo de brillo requerido para ver lo que estás haciendo cómodamente.
3. Siéntate tan lejos de la pantalla como puedas.

Una lámpara de mesa situada a lo largo del monitor, ayudará a reducir el esfuerzo visual -siempre que esté situada por detrás de la parte frontal de la pantalla para evitar la reflexión.

### 1.1.2 El Monitor de Fósforo Verde

El monitor GT64 tiene tres controles por debajo de la parte frontal. Están previstos para poder ajustar el brillo (**BRIGHTNESS**), el contraste (**CONTRAST**) (que es la diferencia entre las partes más brillantes y las partes más oscuras de la imagen) y el ajuste de sincronismo vertical (**VHOLD**), que permite al usuario enclavar la imagen e impedir su "despliegue" vertical.

El mando **VHOLD** no requiere que lo manipules frecuentemente -y una vez que lo hayas colocado inicialmente, te puedes olvidar de él. El brillo y el contraste puedes ajustarlo de vez en cuando para adaptarlo a las condiciones luminosas de la habitación donde estás usando el CPC464.

Usando un monitor monocromo (una pantalla de un solo color que varía la intensidad de los caracteres mostrados para lograr el contraste entre los diferentes elementos de la imagen), el mensaje de "salutación" será el mismo que con el monitor a color (policromo), excepto en que el texto aparecerá en un verde brillante sobre un fondo verde mate y más tenue.

Aunque el uso extremadamente excesivo de tu ordenador con el monitor GT64 puede llevarte a cansancio visual, la imagen generalmente MAS SUAVE del monitor monocromo, hace que sea mucho más cómodo trabajar con él durante un período largo.

## CPC464: INTRODUCCION

En particular, tendrás la posibilidad de sacar el mejor partido del modo "80 columnas" (en que en cada línea horizontal de la pantalla puedes colocar 80 símbolos) dado que la definición o resolución (que indica la capacidad de mostrar un número de diminutos elementos en la pantalla próximos uno a otro pero sin que aparezcan borrosos, sino que continúen siendo nítidos) de un monitor monocromo es intrínsecamente superior a cualquiera, aunque sea el más caro, monitor policromo.

Ajusta el control de BRILLO para producir una imagen adecuadamente iluminada sin que las "MOTAS" que forman los caracteres mostrados en pantalla se vuelvan excesivamente borrosas.

Para "implantar" el modo de 80 columnas, basta que teclees el siguiente programa en tu ordenador CPC464. Con ello se te ofrece una gama de colores a elegir:

```
10 REM señalando formato consola
20 FOR n=0 TO 26
30 MODE 2
40 INK 1,n
50 INK 0,(26-n)
55 BORDER n
60 LOCATE 15,12: PRINT "Pulsa una tecla
    para cambiar el formato de la consola"
70 a$=INKEY$
80 IF a$="" GOTO 70
90 NEXT
100 GOTO 20
```

Este ejemplo nos ilustra además un punto importante que concierne a la presentación y estilo de este manual. Algunas líneas en los listados del programa sobrepasan el margen derecho establecido y las continuamos en la línea siguiente. Es obvio, pero importante, observar que no es preciso cuando teclees ese programa que incluyas los espacios en blanco adicionales que aparecen en el extremo final de cada renglón; simplemente los insertamos para "adornar" el listado y hacerlo más "didáctico". A medida que progresses, apreciarás otras muestras de las "habilidades del editor".

Con el programa anterior no se ilustran todas las posibles combinaciones de la escala de grises (colores), pero sí que sacarás una buena idea de los que están disponibles. Cuando encuentres una combinación que te guste, detén el programa pulsando por dos veces la tecla de **ESCAPE**, (con ello provocarás además un mensaje de **RUPTURA** o "brecha", en que como veremos, aparece la palabra **\*Break\***).

A partir de ahora, la Guía hará muchas referencias a detalles específicos de la opción pertinente al monitor a color.

Programas que presentan efectos interesantes en colores y gráficos, pueden aparecer casi sin poder ser apreciados en un monitor monocromo, aunque se ha procurado con gran cuidado producir una gama de colores que corresponde a niveles progresivos en la escala de grises (se describe más ampliamente en el capítulo 5).

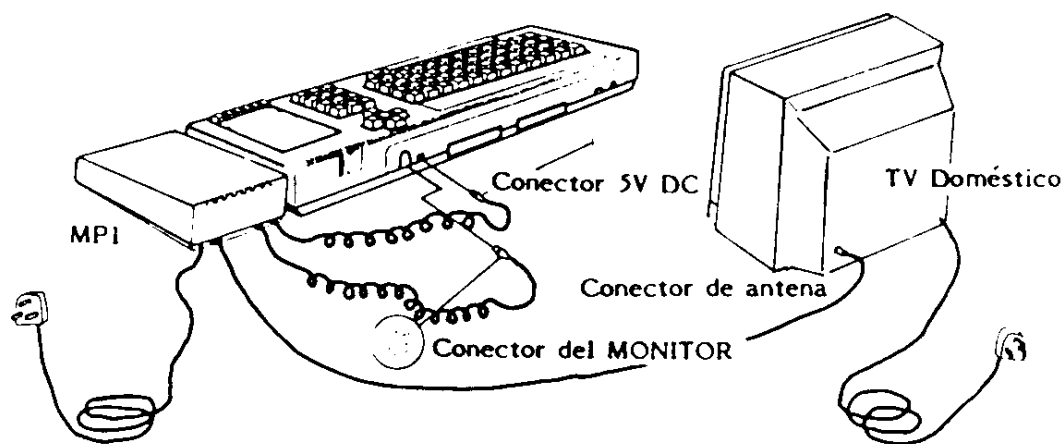
La ventaja de los monitores monocromos, radica en que producen una imagen menos crispante y fatigante y es muy conveniente al hacer el desarrollo del programa; pero es inevitable de vez en cuando que te sientas "picado por el agijón de la abejita computadora" y en esos casos hay poca esperanza de que evites lo inevitable: tener también la opción de color.

### 1.1.3 El TV Modulador/Alimentador MPI

El MPI es una unidad adicional que seguramente desearás comprar si habitualmente usas tu ordenador CPC464 con el monitor monocolor verde GT64. El MPI te faculta para usar el ordenador con tu televisor doméstico, y gozar por lo tanto de todas las enormes posibilidades que en el campo del color posee el ordenador CPC464.

#### IMPORTANTE

Consulta por favor, las instrucciones de **Puesta en Marcha** detalladas al principio de esta Guía de Usuario, que describe cómo conectar los cables de alimentación de tu equipo a una clavija adecuada.



**Figura 2:** Conexiones para el MPI, ordenador y entrada de antenna en tu TV.

El modulador/alimentador (MP1) debiera situarse en el costado derecho del ordenador y sobre una mesa apropiadamente cercana al televisor y al enchufe de suministro eléctrico. Como se muestra en la figura 2 de la página anterior.

Reduce ahora el control de volumen de tu televisor al mínimo -el CPC464 tiene su propio altavoz interno, por lo que no usa el canal de sonido del televisor y reduciendo el volumen al mínimo conseguirás que no moleste el fundido propio del televisor. Luego enciende tu televisor, y a continuación el ordenador usando el interruptor deslizante, marcado **POWER**, en el costado derecho. El pequeño testigo luminoso rojo, situado aproximadamente en el centro del teclado, marcado **ON**, deberá iluminarse para indicar que está en marcha el ordenador; con lo que ya puedes pasar a sintonizar tu televisor y recibir la señal **UHF** procedente del ordenador.

Si tienes un televisor con selección de canales por pulsadores, pulsa uno de ellos para elegir un canal de reserva o poco usado. Ajusta el control de sintonía del televisor de acuerdo con las instrucciones que haya dado el fabricante (la señal del ordenador está aproximadamente en el canal 36 si tu televisor tiene un limbo incorporado), hasta que recibas una imagen igual a esta:

**Amstrad 64K Microcomputer (v1)**

**©1984 Amstrad Consumer Electronics plc  
and Locomotive Software Ltd.**

**BASIC 1.0**

**Ready**



Sintoniza tu televisor exactamente hasta que veas la imagen más nítida. La escritura aparecerá en amarillo (oro) sobre un fondo azul fuerte, aunque variará de acuerdo con las características individuales del TV.

Si tu TV tiene un selector rotativo de programas, gira la perilla de sintonía hasta que aparezca la imagen anterior y permanezca perfectamente estable (como antes, corresponde aproximadamente al canal 36).

Como la señal pasa a través de las diversas etapas del proceso, siendo primeramente "modulada" y después "demodulada", se produce un cierto deterioro en la señal de video. Los resultados no pueden por tanto ser tan buenos como los conseguidos mediante un monitor al que directamente se inyecta la señal de video, y también depende de la calidad de tu TV; por lo que puede suceder que en el modo de texto con 80 columnas (modo 2) los resultados logrados no sean perfectos, en cuyo caso, debieras usar el modo 1 para imponer textos que probablemente te resultará más conveniente.



## 1.2 Primeros pasos

Ahora que ya estás "conectado": el alimentador debiera estar enchufado, los terminales del cable de video enganchados ya sea en el monitor, ya sea en el modulador; y ya has encendido. Tu ordenador está esperando que le mandes hacer algo.

La imagen en pantalla de saludo o de "despertar" es el único texto incorporado en la memoria del ordenador que puedes mostrar en pantalla sin que previamente tengas que imponer un comando o una instrucción mediante el teclado. Si ya estás familiarizado con el lenguaje de programación BASIC, hay grandes probabilidades que inmediatamente hayas metido un breve programa para "empezar a conoceros". El BASIC AMSTRAD te será familiar en muchos aspectos, y simplemente puedes continuar tu marcha; te mostraremos un breve programa que puedes teclear para mostrar todo el repertorio de caracteres incorporado en tu ordenador. Este REPERTORIO de símbolos, y que es el término que usamos para describir todo el surtido completo de cifras, letras, signos, y otros caracteres no visivos en pantalla, pero que también pueden ponerse en acción mediante el teclado.

Otros de los caracteres del repertorio no son directamente accesibles pulsando ninguna tecla, sino que solamente están disponibles para ser mostrados usando la instrucción **PRINT CHR\$(*<número>*)** que posteriormente te describiremos.

Esto es debido a que cada elemento almacenado en la memoria del ordenador, lo hace en unidades de información denominadas "**bicos**", entre otras cosas. Así por ejemplo, en inglés se dice "**bytes**" que por su significado primitivo "mordisco", les destaca el hecho de ser la cantidad de información manejada en cada operación elemental. HABITUALMENTE, y como puedes observar en el Apéndice II, un **bico** (byte) es simplemente cada una de las 256 combinaciones diferentes que pueden hacerse con una serie de OCHO elementos binarios (que puedes simbolizar como quieras: 1/0; alto/bajo; ping/pong; zig/zag). Pero como la gran mayoría de ordenadores tienen que usar como mínimo 8 **bits** por cada carácter almacenado, otra de las facultades del CPC464 es que aprovecha todas esas 256 posibles combinaciones, y no se queda simplemente satisfecho con las 96 combinaciones que la mayoría de los demás manejan, desperdiciando las otras 160 posibilidades.

Esa gama standard de caracteres es pues un "subconjunto" del repertorio del CPC464. Por tanto, ese surtido de 96 caracteres, también conseguido en el CPC464, es lo que se conoce a lo ancho del mundo informático como el sistema "**ASCII**", siglas derivadas de:

American	Americano
Standard	Estándar
Code for	Código para
Information	Intercambio de
Interchange	Información

...y es primordialmente un sistema que asegura que la información que se intercambia o transmite de un ordenador a otro es en una forma reconocible por ambos. Es quizá el único aspecto de informática que verdaderamente es universal, así que te aconsejamos concretamente que te familiarices con todos los aspectos del ASCII. El Apéndice III no sólo presenta el ASCII, sino que además te muestra los caracteres adicionales disponibles en el CPC464 y sus correspondientes códigos numéricos, en notación decimal, hexadecimal y binaria.

Alguno de esos otros caracteres adicionales del repertorio del CPC464, también pueden mostrarse en pantalla usando una combinación de la tecla de control, marcada **CTRL** en el teclado, y de otras teclas normales, -pero no te preocupes sobre esto todavía, hasta que no comprendas la función desempeñada por la tecla de control, porque puedes hacer más daño que beneficio si empiezas a pulsarlas al azar.

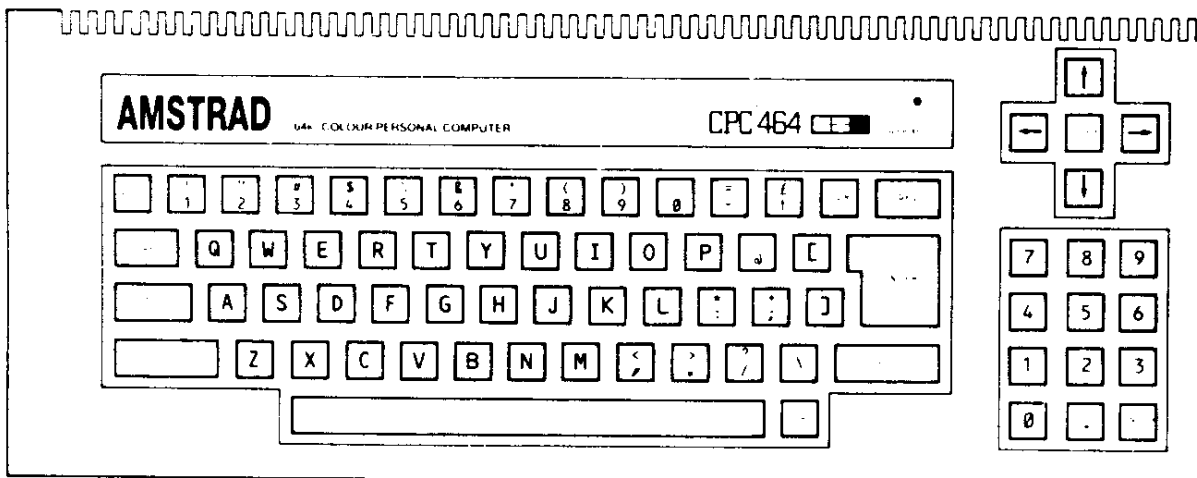
### 1.2.2

Para ver exactamente cómo todos estos caracteres aparecen en la pantalla, teclea el programa que te presentaremos en breve y así satisfacerás tu curiosidad y empezarás a practicar con el CPC464. Este programa te ayudará a adquirir confianza tanto en la simplicidad de la programación, como en el hecho de que cuando consigues el "saludo" al poner en marcha el ordenador, es muy probable que no encuentres ningún problema "circuital", y que el CPC464 está simplemente esperando ser programado y nutrido con la información correcta para empezar a trabajar.

(Si cometes un error cuando teclees ese programa, salta a la sección 1.2.7 para ver si puedes corregir el error sin necesidad de volver a meter el programa desde el principio).

Cuando teclees el programa en tu ordenador, no importará si usas letras minúsculas (**abc**), o letras mayúsculas (**ABC**), conseguidas pulsando al mismo tiempo la tecla de turno o cambio, marcada **SHIFT**. El ordenador aceptará el programa en una y en otra forma. Observa que **DEBES** delimitar las palabras clave usando espacios en blanco u otros delimitadores (coma, punto y coma, etc., según convenga) en las posiciones reseñadas, dado que el BASIC AMSTRAD también permite que uses esas palabras reservadas (completamente enunciadas en el Apéndice VIII) dentro de los nombres de las variables, cuando no provoca confusión.

El aspecto "físical" del teclado aparece en la figura 4. Y hemos mencionado la palabra "físical" dado que muchas de las teclas están disponibles para que puedan ser definidas por el usuario de forma que la función que ejerzan o el símbolo o símbolos que presentan al ser pulsadas es dependiente de lo que el usuario haya definido. Ya lo describiremos mas ampliamente en las secciones subsiguientes.



**Figura 4:** El Teclado del CPC464

Al pulsar la tecla **ENTER** se logra que el comando o instrucción que tecleas en el ordenador, y que hasta ahora ha sido simplemente "escuchado" por él y mostrado en la pantalla para comprobar que no te equivocas, tiene como efecto que interprete esa pulsación final de **ENTER** como que ya VALE y que debe dejar de escuchar y pasar a cumplimentar lo que le has mandado: inmediatamente si es un comando, o simplemente almacenarlo en su memoria si es una instrucción (lo que sabe distinguir porque la instrucción va precedida del número de línea correspondiente).

La tecla **ENTER** se menciona frecuentemente como "**RETURN**" porque proviene de cuando se usaban más las impresoras que las pantallas, e indicaba que el ordenador debía hacer un "retorno del carro" de impresión para comenzar en la línea siguiente, de forma similar a la palanca usada primitivamente en las máquinas de escribir. El término ha continuado, y se ha incrustado para siempre dentro del repertorio ASCII, donde se simboliza ese carácter no visivo mediante la abreviatura "**CR**". Teclea ahora el programa que estamos comentando:

```
10 FOR N = 32 TO 255
20 PRINT CHR$(N);
30 NEXT N
RUN
```



Y tanto las variables como las constantes son datos para el ordenador, aunque de distinta índole. Unos son **numéricos** o **NUMERALES**, y a los otros como ya veremos, los llamamos **LITERICOS**, **literales** o **alfanuméricos**.

¿Pero cómo sabe el ordenador la diferencia?

Si hubieramos declarado la letra **N** como un carácter literal, habríamos tenido que teclear **N** encerrada entre comillas.

Y el ordenador nos habría respondido con el mensaje de error sintáctico porque no comprendería la línea de instrucciones 10 que diría "**N**". Simplemente, usando **N** de esta manera, le hemos dicho al ordenador que **N** va a ser una variable numérica. La definición de la instrucción **FOR** en el lenguaje BASIC, exige que esa palabra clave sea seguida por una variable numérica, de forma que el ordenador siempre supone que lo que va detrás de la palabra **FOR** es simplemente eso.

También le hemos dicho al ordenador que **N = 32 to 255**. Y con eso le hemos declarado que la gama de valores de esa variable es en efecto una serie consecutiva de valores, empezando por 32 y terminando en 255.

Habiendo reseñado así esta instrucción y declarado esta variable numérica, continuamos instruyendo al ordenador con lo que queremos que haga después: y es lo que le reflejamos en la siguiente línea:

```
20 PRINT CHR$(N);
```

Eso le dice al ordenador que debe convertir el valor que en ese momento tenga la variable numérica designada con **N**, en el **carácter** que corresponda a ese valor. La función **CHR\$(N)** es la forma convenida con BASIC para que efectúe esa tarea de conversión. Y habiendo examinado en su memoria el valor que corresponde a la variable numérica designada con la letra **N**, y calculado el carácter correspondiente, simplemente lo **EXPONE (PRINT)** en pantalla, con lo que ya ha cumplimentado esa instrucción.

El punto y coma al final de la línea le indica al ordenador que no envíe hacia la pantalla el carácter de "retorno de carro" ni "avance de línea", para que así el siguiente carácter que tenga que exponer no aparezca en la columna de la izquierda y en la siguiente línea de la pantalla, tal y como ocurriría automáticamente si no pusieramos el signo de punto y coma. Es decir, si no lo pusieramos, expondría los caracteres uno debajo de otro, formando una columna en la pantalla, en lugar de uno después de otro, formando una pila en la pantalla.

La siguiente instrucción le dice al ordenador que cuando haya efectuado esta tarea con el primer número de la serie señalada en la instrucción **FOR** (el 32) debe volver a la línea donde está reseñada la instrucción **FOR**, y volver a hacer la misma tarea pero con OTRO valor (y podíamos decir el SIGUIENTE) que corresponda a la variable **N**. Este proceso repetitivo se conoce como **bucles**, o una serie de "rondas" en que se efectúa una determinada tarea en cada ronda. Y es uno de los más fundamentales y frecuentes aspectos de la programación y operación con ordenadores. Este bucle que empieza con la instrucción **FOR** y termina con la instrucción **NEXT**, es uno de los rasgos fundamentales de la informática, y aparece en todos los lenguajes de programación sea en una forma o en otra. Ahorra tecleo manual con tareas reiterativas y rápidamente llegarás a usarlo en tu propia programación.

Cuando el número de rondas efectuadas coincide con el límite de la serie declarada en la instrucción **FOR**, (el 255), la operación cesa porque el bucle se ha terminado, y el ordenador examina la línea que va detrás de aquella donde aparece la instrucción **NEXT**. En este caso, la que va detrás de la línea 30, y como no hay ninguna, simplemente termina la ejecución del programa, saca en pantalla el aviso **Ready**, para indicar que está PREPARADO para recibir nuevos comandos o instrucciones. Obviamente, puedes volver a decirle que repita la ejecución del programa. El programa ya está conservado en la memoria en forma segura, y permanecerá allí hasta que mandes al ordenador lo contrario; o hasta que quites la alimentación, en cuyo caso, todos los datos variables y constantes, y los programas se habrán perdido a no ser que los hayas GUARDADO en el cassette como medida de seguridad.

Este programa ilustra claramente un punto fundamental en informática: todo lo que el ordenador hace está relacionado con números. El ordenador ha mostrado su repertorio de caracteres, con letras, cifras, signos de puntuación y un surtido de símbolos especiales; y lo ha hecho, tomando como referencia para cada carácter mostrado un número. Cuando pulsas la tecla marcada **A**, el teclado envía hacia las entrañas del ordenador un número asociado con esa **A** y de allí sale hacia la pantalla ese número (u otro) que los circuitos de pantalla interpretan y reconocen como correspondiente a la letra **A** y por tanto, muestran la **A** en pantalla. Por consiguiente, a cada dato le corresponde un código numérico que es el que realmente se manipula dentro del ordenador, y se convierte a información adecuada a las personas cuando es necesario recibir un dato de ellas (entrada por teclado) o mostrarles algún dato (salida por pantalla).

Cada carácter tiene pues un número inherentemente asociado, y esa es la lista que aparece en el Apéndice III de este manual.

## 1.2.4

Por favor, no te preocupes si no comprendes todos los tecnicismos y jerga usada en esta página. Es importante explicar precisa y completamente cómo el ordenador cumplimenta tus instrucciones y consigue los resultados que le pides, pero también es probable que los comprendas claramente si cuidadosamente examinas las explicaciones. Sin embargo, si encuentras esta sección bastante complicada, saltatela y pásala a la sección 1.2.5 (ya podrás volver más adelante).

Por ejemplo, DECIMOS que el **código** -ese número en estrecha correspondencia con cada carácter-, de la letra **A** es 65. y que el código de la **a** es 97.

Pero a poco que pienses, llegarás a la conclusión de que sólo es una forma de hablar. En realidad, dentro de la máquina no hay ni **A** ni 97. No puede haberlo. Lo que sí hay es OCHO puntos en los circuitos electrónicos internos, que examinados con un voltímetro uno a uno, nos irían diciendo: **no** hay voltios, **sí** hay voltios, **no** hay voltios, **no** hay voltios, **no** hay voltios, **sí** hay voltios. Y comprenderás que hablar así es largo y con muchas posibilidades de confusión. Por lo tanto, abreviamos y consideramos a esos OCHO puntos como un solo sitio, (de ahí el **octeto**, como también se llama al bico), y decimos que en ese punto hay un 97 cuando se ha pulsado la tecla marcada con **A** en el teclado.

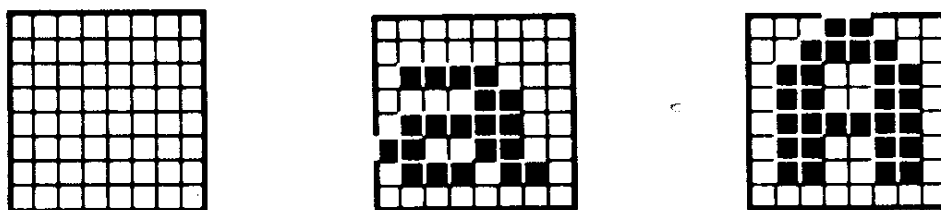
¿Y por qué 97 y no tropecientos savanta y nove? Bien, es 97 y no otro número porque todo está basado en el SISTEMA DE NUMERACION EN BASE DOS, y los sistemas de numeración ya en los tiempos de Pitágoras llevaban siglos de existencia, y no había ordenadores.

Claro que la **notación decimal** que usamos en nuestra vida diaria es tan natural que pensar en otro sistema de numeración puede parecer no sólo complicado, sino hasta aberrante. Nos es natural porque tenemos 10 dedos en las manos, pero los ordenadores no. Y en los circuitos electrónicos y en los interruptores es más fácil detectar si hay tensión o no. Por eso les es natural usar el sistema binario. Y no es complicado, como más adelante veremos. Alguien simplemente lo comparó con la destreza necesaria para cambiar sus hábitos de comida, usando el cuchillo y el tenedor en manos contrarias a las que estás acostumbrado. Y además, una vez comprendida esa notación, muchísimas de las cosas sobre ordenadores empiezan a encajar y empieza a ser patente la elegante estructura de los sistemas de numeración usados internamente.

Una vez que en las entrañas del ordenador se ha traducido la pulsación de la letra **A** detectada en el teclado, a su código correspondiente, y cuando le toca mostrarla en pantalla, consulta también sus circuitos internos para ver el **tipo** que tiene ese carácter. Y recuerda que el tipo ya desde los tiempos de Gutenberg, que inventó la imprenta, es la manera de referirse a la **forma** específica que tiene cada símbolo.

Pues bien, para definir ese tipo, en fábrica se preparan 64 puntos en los circuitos electrónicos con valores inalterables de sí tensión/no tensión que permiten al ordenador mostrarlo en pantalla. Y otra vez, no sólo para abreviar, esos 64 puntos consecutivos se dividen en OCHO grupos (octetos o bicos) con 8 puntos cada uno.

Y además, los circuitos encargados de presentar la información en la pantalla, usan esas señales eléctricas para hacer OCHO trazos, con OCHO motas cada uno, y los coloca sucesivamente uno debajo de otro, formando una cuadrícula diminuta, tal y como aparece en la figura 5:



**Figura 5:** Una cuadrícula en blanco, con la a minúscula y con la A mayúscula.

Por tanto, podemos asimilar la pantalla a un papel cuadriculado, en que cada carácter ocupa uno de los cuadritos; y además, cada uno de estos cuadritos vuelve a estar reticulado en forma de 8 filas de 8 columnas cada una, y lográndose el tipo o forma del carácter iluminando selectivamente las 64 motas que resultan. Pero mejor es mirarla como 8 trazos horizontales (que corresponden a 8 bicos) situados verticalmente uno debajo de otro.

Si no encuentras el tipo de carácter que deseas dentro del repertorio de 256 que está incrustado en la memoria del ordenador CPC464, también puedes definir el tipo que tiene y luego mostrarlo en pantalla, si usas las instrucciones disponibles en el BASIC AMSTRAD usando las palabras reservadas **SYMBOL** y **SYMBOL AFTER**, que estudiaremos en el capítulo 8.

Estos caracteres definidos por el usuario, usan una de las combinaciones posibles que pueden hacerse iluminando selectivamente o no las 64 motas mencionadas. Y si analizaras cuántas combinaciones posibles puedes hacer de esa manera te resultaría una cantidad de combinaciones que sobrepasa los 18 trillones. Sí, he dicho 18 trillones, es decir 18 seguido de 18 ceros; y además puedes agrupar varios de esos tipos para que aparezcan en pantalla ocupando cuadritos contiguos y formar figuras más complejas, por lo que la cantidad de caracteres y figuras definidas por el usuario está únicamente limitada por tu capacidad de imaginación y por tu tiempo.



### 1.2.5 Volviendo al programa

El resultado del primer programa que has tecleado no tiene muy buena apariencia. Todavía quedan restos del mensaje de salutación en la parte superior de la pantalla. Tendría mejor presentación si dejáramos la pantalla en limpio antes de comenzar a exponer otras cosas. Añadiremos una línea en el programa para que arregle esto.

Teclea lo que viene a continuación en la línea donde está situado el cursor (que ya sabes, es ese pequeño rectángulo sólido e iluminado inmediatamente debajo y a la izquierda del aviso **Ready**, y si no sabes cuál es, no sé que estás haciendo leyendo esto antes de los Fundamentos!).

```
5 CLS
RUN
```

Observa cómo la pantalla queda completamente en blanco antes de que el ordenador empiece a escribir el repertorio de caracteres a partir de la esquina superior izquierda.

Esto demuestra también otro aspecto de los lenguajes de programación, y es que no importa en qué orden teclees tus instrucciones ni que tampoco necesitas realmente listar el programa para añadirle una instrucción más. El ordenador siempre ordena (y por eso se llama ordenador) las instrucciones de acuerdo con un orden ascendente de números de línea, antes de que comience a ejecutar el programa. Compruébalo listando el programa y viendo que la última tecleada ya está en el sitio que le corresponde.

### 1.2.6 LISTANDO

Puedes comprobar fácilmente cuál es el programa que el ordenador tiene almacenado en su memoria, pidiéndole que lo liste. Teclea:

```
LIST
```

y el resultado en la pantalla será:

```
5 CLS
10 FOR N = 32 TO 255
20 PRINT CHR$(N);
30 NEXT N
Ready
```

Este programa permanecerá en la memoria del CPC464 hasta que hagas algo de lo siguiente.

- \* Apagar el ordenador
- \* Restaurar las condiciones iniciales, pulsando **CTRL**, **SHIFT** y **ESC** sucesivamente, y manteniendo pulsada cada una hasta que sueltes las tres simultáneamente.
- \* Cargues otro programa a partir del cassette, o lo cargues y ejecutes mediante **RUN**.

- \* Teclees el comando **NEW** que pone a cero todas las variables y vacía la memoria que contiene el programa, sin restaurar los valores iniciales del modo de pantalla ni de los colores.

Vamos ahora a preparar una de las teclas funcionales para que en cuanto se pulse, ella sola efectúe automáticamente la misma tarea que si hubieramos tecleado **ENTER CLS:LIST ENTER**, porque con esa facilidad aceleramos el desarrollo de los programas enormemente. Para hacerlo, teclea:

```
KEY 138, CHR$(13)+"CLS:LIST"+CHR$(13)
```

Y ahora pulsas la tecla con el punto decimal situado en el teclado numérico separado, que es el que corresponde a la **TECLA (KEY)** con código 138.

Hasta 32 teclas pueden pre-programarse de esta manera, y puedes elegir cualquiera de las del teclado para "redefinir" el valor que les corresponde, en caso de que quieras usar el teclado numérico separado para su propósito original. Consulta la descripción del comando **KEY** en el capítulo 8.

Si el tuyo es un programa largo, puedes definir esa tecla como sigue:

```
KEY 138, CHR$(13)+"CLS:LIST"
```

con lo que al pulsarla, te permitirá teclear a continuación una gama de números de línea que desees; o simplemente pulsandola dos veces sucesivas te presentará el listado completo.

Cuando estás haciendo experimentos con los colores, es posible que te veas perdido con una combinación de colores donde no es posible ver lo que aparece en pantalla porque la tinta del papel y de la pluma es la misma. Para ello puedes pre-programar otra tecla:

```
KEY 139, CHR$(13)+"mode2:ink1,0:ink0,9"+chr$(13)
```

con lo que sólo tienes que pulsar la más pequeña de las dos teclas **ENTER** (la que está en el teclado numérico separado) para que cuando te encuentres en la situación mencionada, vuelvas a la situación base con una combinación de colores visibles en pantalla (no perderás el programa en memoria).

Las teclas definidas por el usuario son restauradas siempre que se restauran las condiciones iniciales de la máquina, dado que tienen que estar disponibles para las instrucciones de los programas; por lo tanto, una vez que hayas pre-programado tus favoritas, escríbelas en un programa y guárdalas en la cinta para poder recuperarlas fácilmente.

### 1.2.7 Lo primordial de editar

Inevitablemente cometerás equivocaciones al teclear los programas. Bienvenida a esta sección a todos los que han saltado aquí desde la sección 1.2.2.

En el CPC464 se ha intentado hacer la corrección de esas equivocaciones tan simple como es posible, y al mismo tiempo evitando los problemas de volver a escribir accidentalmente caracteres que no querías cambiar.

El grupo de teclas que controla el movimiento del cursor (ese pequeño rectángulo sólido que define dónde va a aparecer el siguiente carácter que teclees) proveen medios para dirigir la atención del ordenador a aquella parte del texto que hay en pantalla que desees alterar.

Al cometer un error en una línea numerada, v.g.:

```
10 FOR N = 332 TO 255
```

dispones de varias opciones:

1. Puedes pulsar **ENTER** y volver a teclear toda la línea. La línea incorrecta será borrada de la memoria y sustituida por la línea que teclees usando ese mismo número de línea.
2. Puedes pulsar la tecla de retroceso del cursor y desplazarlo hasta que se sitúe encima del símbolo incorrecto.

```
10 FOR N = 332 TO 255
```

Observa que el carácter situado en la misma posición que el cursor aparece en video inverso. En otras palabras, el carácter que normalmente tiene el mismo color que el cursor, adopta ahora el mismo color que el fondo (el papel) de manera que se transparente a través del cursor que está situado precisamente encima de él.

Pulsa ahora la tecla de BORRAR, marcada **CLR**, y el carácter situado en la misma posición del cursor desaparecerá, y los situados a la derecha se moverán para que no queden huecos:

```
10 FOR N = 32 TO 255
```

Pulsa otra vez **ENTER**, y esta línea ya corregida, será la que el ordenador recuerde. El cursor no tienes porqué llevarlo hasta el extremo de la línea, porque el ordenador recoge toda la línea, sin tener en cuenta la posición del cursor en ese momento.

3. También puedes hacer que el cursor se sitúe en el carácter inmediatamente a la derecha del que desees borrar:

```
10 FOR N = 332 TO 255
```

Ahora pulsa la tecla de QUITAR (marcada **DEL**) con lo que desaparecerá el carácter situado inmediatamente a la izquierda del cursor, al mismo tiempo que se desplazan tanto el cursor como el resto de caracteres a la derecha sin que se vea afectado el situado precisamente en la posición del cursor, sólo el de la izquierda.

Pulsa **ENTER** y la línea quedará almacenada en la memoria como antes.

```
10 FOR N = 32 TO 255
```

### 1.2.8 Considerandos

Los métodos anteriores son buenos cuando detectas el error antes de que llegues a terminar la línea pulsando **ENTER**. La mayoría de los errores, sin embargo, permanecen inadvertidos en ese momento y sólo salen a la luz cuando intentas ejecutar el programa, momento en que el ordenador te responde con un mensaje de error. (Apéndice VIII).

Unas cuantas clases de errores provocarán que el ordenador exponga la línea defectuosa, con el cursor de edición situado en la primera columna (extremo izquierdo del papel). Si ese es el caso, puedes utilizar directamente los procedimientos mencionados anteriormente, como si hubieras detectado el error antes de concluir esa línea del programa pulsando **ENTER**.

Si al producirse el error, el ordenador no te muestra la línea con la equivocación, sino simplemente te reseña el número de línea, tendrás que listar el programa para examinar esa línea y poder corregir el problema.

### 1.2.9 Editando con el Cursor de Copiar

Primero lista el programa. (Continuamos suponiendo que estás trabajando con el programa breve de ensayo que estamos comentando, ya que el listado cabe en un solo pantallazo).

```
5 CLS
10 FOR N = 32 TO 255
20 PRINT CHR$(N);
30 NEXT N
```

El error está en la línea 20, hay una **S** en lugar del signo **\$**, que indica una función literal (el **\$** obliga al ordenador a considerar que los caracteres deben tomarse LITERALMENTE al pie de la letra, y que no puede hacer operaciones de índole cuantitativa con ellos). Puedes volver a teclear la línea 20 desde el principio y volverla a meter, o bien puedes usar el llamado **Editor de plana** (o de pantalla, para distinguirlo del caso anterior en que decimos **Editor de línea**) en la forma siguiente:

Mantén presionada la tecla **SHIFT** (cualquiera de las dos que hay en el teclado) y pulsa la tecla que hace subir al cursor, marcada (↑).

Puedes bien ir pulsando repetidamente para que suba una posición a cada pulsación, o puedes mantenerlo pulsado el tiempo suficiente como para que intervenga la repetición automática disponible en tu ordenador. En el instante en que levantes tu dedo de la tecla, se detendrá el cursor; y con un poco de práctica, pronto te familiarizarás con estos efectos. Si te has pasado, puedes volver con la tecla que baja el cursor, marcada (↓) al mismo tiempo que como antes, mantienes presionada la tecla **SHIFT**.

De esta manera, se provoca que aparezca el "CURSOR DE COPIAR" que es distinto del cursor principal aunque tengan la misma forma, y que se vaya desplazando hasta alcanzar la línea que deseas modificar. Ahora debes situarlo en el primer carácter de la línea a corregir.

```
5 CLS
10 FOR N = 32 TO 255
20 PRINT CHR$(N);
30 NEXT N
Ready
■
```

Si intentaras llevar el cursor principal a esa línea que quieres corregir, o sea, usaras las teclas de desplazamiento del cursor, sin mantener simultáneamente pulsada la tecla **SHIFT**, el ordenador no lo aceptaría como una acción válida, dado que únicamente los caracteres tecleados directamente y detrás del cursor principal son los que pueden formar parte de instrucciones y comandos aceptables por el ordenador.

Si comenzaras así y hasta reescribieras parte de la línea, podrías fácilmente escaparte del enredo en que te has metido, pulsando la tecla **ESC ANTES** de pulsar cualquiera de las teclas **ENTER** o de usar alguna de las teclas funcionales que contenga **CHR\$(13)**. Si accidentalmente tecleas el comando **NEW** y lo concluyes pulsando **ENTER**, habrás perdido para siempre todo tu programa. Así que debes ser cuidadoso para ahorrarte problemas.

Una vez que has comenzado a teclear una línea, si intentas salirte de ella sin pulsar previamente **ENTER**, el ordenador emitirá un pitido en cuanto llegues a un límite ilegal. En cuanto te salgas del problema pulsando **ENTER**, no se habrá perdido nada del programa, a no ser que hayas tecleado un número de línea que sea válido al principio de todo, en cuyo caso la línea, si existe, cuyo número coincida con el tecleado, será "reescrita" como resultado de esa operación, y tendrás que examinarla para ver cómo tienes que corregirla.

Cuando hayas situado el cursor de copiar correctamente, sigue repicando con la tecla de COPIAR, marcada **COPY** hasta que dicho cursor llegue a la posición donde quieres hacer un cambio. (Cuando te acostumbres a la velocidad del cursor de copiar, serás capaz de mantener pulsada la tecla **COPY** el tiempo suficiente como para situarlo certeramente en una sola tacada).

```
5 CLS
10 FOR N = 32 TO 255
20 PRINT CHR$(N);
30 NEXT N
Ready
20 PRINT CHR$
```

Ahora suelta la tecla **COPY** y pulsa la marcada con el signo \$, que aparecerá debajo en la línea que está escribiéndose con el cursor **PRINCIPAL**, que por supuesto se desplazará una posición a la derecha como siempre.

```
20 PRINT CHR$
```

Sin embargo, el cursor que **COPIA** tiene que ser desplazado para que se salte esa **S** errónea, y para ello mantén pulsada la tecla **SHIFT** al mismo tiempo que pulsas una vez la tecla de avance de cursor. Con ello, el cursor de copia descansará sobre el paréntesis de apertura. Ahora sueltas la tecla **SHIFT** y mantienes pulsada la tecla **COPY** hasta que llegue al final de esa línea de instrucciones; en ese momento pulsarás **ENTER** para indicar que has concluido la operación, y la línea ya corregida tal y como aparece en la parte inferior de la pantalla, sustituye a la línea incorrecta que aparece en el listado.

Puedes combinar todos estos métodos simplemente copiando enteramente la línea del programa defectuosa, y antes de concluir la operación pulsando **ENTER**, pasar a la **EDICION DE LINEA**, usando las posibilidades asociadas al cursor principal tales como directamente las teclas de desplazamiento del cursor y las teclas de borrado, (**CLR**) y quitado (**DEL**) de caracteres. Para ello, manteniendo pulsada la tecla **CTRL**, pulsas la tecla de avance de cursor o de retroceso de cursor, con lo que llevarás de un solo golpe el cursor al extremo derecho o al extremo izquierdo de la línea que se está editando.

Practica y verás que es más fácil no sólo a la hora de manejarlo en la práctica, sino también mucho más fácil de lo que parece cuando se está explicando en un texto.

Finalmente, también puedes pasar a la edición (recuerda que es palabra aprovechada para hablar de corrección) tecleando:

```
EDIT 20
```

El ordenador responderá, mostrándote dicha línea 20 en la forma:

```
20 PRINT CHR$(N);
```

Ahora puedes usar directamente las teclas de cursor junto con las teclas **CLR** y **DEL**, tal y como te hemos comentado, y cuando estés satisfecho con el resultado simplemente pulsa **ENTER** para indicar que ya VALE. Si te metes en algún lío con la edición, pulsa **ESC** y lista la instrucción de nuevo. La línea que se estaba editando cuando has pulsado **ESC** no se ha visto alterada.

Ahora lista una vez más, y verás aparecer la versión corregida del programa, y si no lo está, pues vuelve a ensayarlo otra vez.

Hasta ahora, solamente hemos comenzado a explorar el funcionamiento normal del CPC464. Vamos a echar una mirada a los otros dos modos. Teclea:

```
MODE 0
RUN
```

Observa que primero se limpia la pantalla, y luego el programa muestra el repertorio de caracteres, pero con sólo 20 en cada renglón de pantalla:

```
!"#$%&'(<>?*+-./01234
56789:;=<=>?@ABCDEFGHI
HIJKLMNOPQRSTUVWXYZ[\]^
_`ab cde fgh i j k l m n o
p q r s t u v w x y z { | }
~ ¡ ¢ £ ¤ ¥ ¦ § ¨ © ª « ¬ ® ¯
° ± ² ³ ´ µ ¶ · ¸ ¹ º » ¼ ½ ¾
¿ ¹ º » ¼ ½ ¾ ¿ ¹ º » ¼ ½ ¾
Ready
```

Para volver al modo original y normal, teclea:

```
MODE 1
RUN
```

Y ya hemos regresado al principio. Para observar la imagen en el modo de 80 columnas, teclea:

```
MODE 2
RUN
```

```
!"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMN
OPQRSTUVWXYZ[\]^_`ab cde fgh i j k l m n o
p q r s t u v w x y z { | } ~ ¡ ¢ £ ¤ ¥ ¦ § ¨ © ª « ¬ ® ¯
° ± ² ³ ´ µ ¶ · ¸ ¹ º » ¼ ½ ¾ ¿ ¹ º » ¼ ½ ¾
Ready
```

Ya hemos comenzado a trabajar, y ya debieras haber satisfecho parte de tu curiosidad inicial. Los usuarios avanzados ya estarán -me lo imagino- preparándose para convertir sus programas favoritos, en versiones que puedan ejecutar con este potente dialecto de BASIC. Los usuarios menos familiarizados con el BASIC deben proseguir a través de las secciones "primordiales" para conseguir una panorámica sobre las facetas del BASIC que son específicas en esta máquina.



## 2. Cassette (Datacorder)

Manipulando el sistema de almacenamiento en cinta.

Temas tratados en este capítulo:

- \* Similitudes y diferencias entre los cassettes de voz y de datos.
- \* Cargando programas grabados en cinta cassette  
- la cinta de Bienvenida
- \* Las opciones de velocidad de transferencia de datos
- \* Guardando programas en cintas cassettes
- \* Errores

La memoria del CPC464 sólo es capaz de conservar la información mientras se suministre energía eléctrica al ordenador. En la terminología informática, se dice que la memoria es un medio de almacenamiento "volátil" a causa de su poca retentividad. Si quieres conservar los programas y los datos cuando se interrumpe el suministro de energía (y alguna vez tendrás que apagar el ordenador) deben ser grabados en la cinta magnética contenida en la cassette, o en otro medio de almacenamiento no volátil como los diskettes.

### 2.1 Controles del Cassette

En la parte derecha del teclado del ordenador, encontrarás un equipo cassette, identificado comercialmente como DATACORDER, y que te presentamos en la figura 2.1. La mecánica de esta DUCTORA de cinta cassette es esencialmente la misma que la de los equipos de audio para la grabación y reproducción de sonidos; excepto en que la electrónica que controla los mecanismos de arrastre, grabación, etc. está específicamente optimizada para usar el cassette como medio de almacenamiento de información "digital".

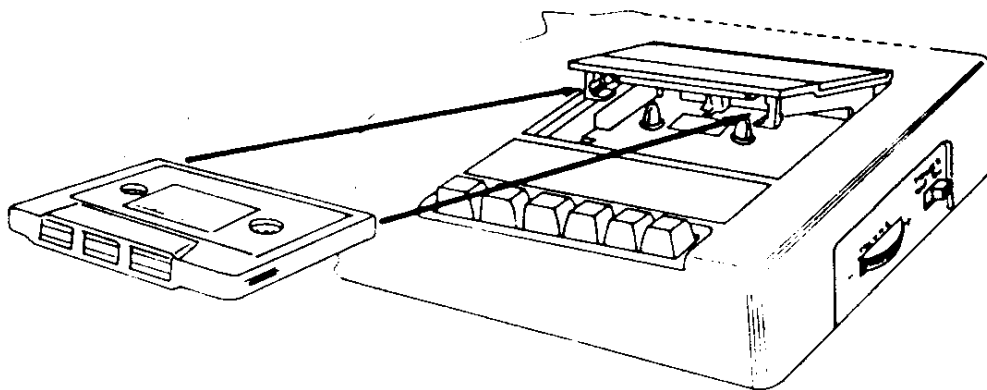
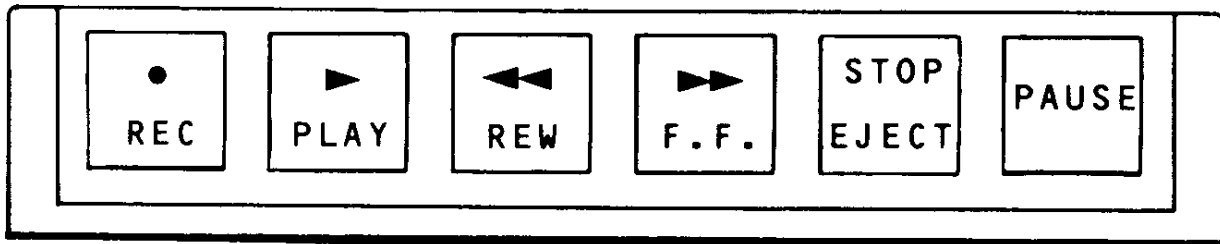


Figura 2.1: La forma correcta de insertar el cassette.

Igualmente, la operación de la fila de teclas situada en la parte superior delantera del cassette es igual que para la mayoría de los equipos de audio-cassette:



**Figura 2.2:** Los controles del cassette Datacorder CPC464.

Observa que todas estas teclas de control requieren ser presionadas (no podemos decir pulsadas) de forma más firme que las teclas empleadas para imponer información en el ordenador.

**REC** = Opera simultáneamente con **PLAY** para grabar datos cuando exigimos al ordenador que así lo haga. No puede actuar a no ser que se haya metido en el cassette una cinta cuya lengüeta de protección ante grabación (figura 2.2) y se haya bajado la tapa del cassette.

Para que efectúe su labor, debes oprimir la tecla **REC** y mientras la mantienes presionada, oprimir la tecla **PLAY**. El ordenador comenzará a transferir datos de la memoria al cassette cuando cumplimente la instrucción pertinente, o cuando ejecute el comando **SAVE** impuesto directamente por teclado.

**PLAY** = **LECTA** información de la cinta (y "lectar" es un aumentativo de leer) para trasvasarla a la memoria del ordenador. Puedes usar el comando **LOAD** simplemente para cargar en la memoria del ordenador la información extraída de la cinta, o el comando **RUN** para que además de cargarlo lo ejecute inmediatamente. Ambos puedes darlos como instrucciones en un programa, o como comandos directos por teclado.

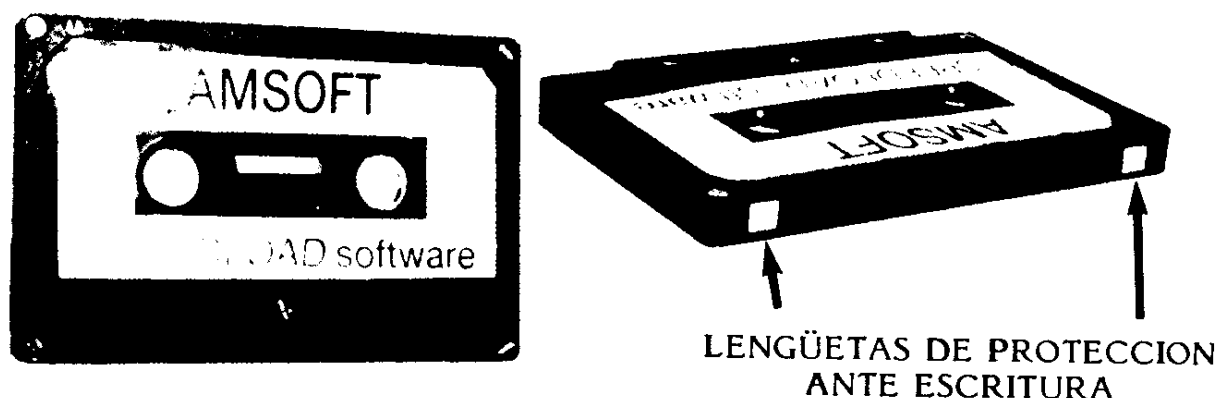
Ambas teclas **REC** y **PLAY** se reponen automáticamente cuando el mecanismo de arrastre de la cinta detecta el final del carrete.

**REW** = Rebobina la cinta, pasándola del carrete izquierdo al carrete derecho. No hay conclusión mecánica de esta función al llegar al extremo de la cinta, por lo que no debes dejar el cassette **REBOBINANDO (REWIND)** si no quieres que el motor de arrastre se sobrecaliente.

**[STOP/EJECT]** = **PARA** cualquier operación que se esté efectuando en el cassette, reponiendo las teclas a su posición normal. Si se pulsa por segunda vez esta tecla, **ALZA** la tapa del cassette para que puedas sacar la cinta o meter otra. La cinta no puede sacarse hasta que haya cesado el movimiento de arrastre.

**PAUSE** = Apretando esta tecla consigues una PAUSA en la operación de la ductora de cinta, sin alterar el modo de funcionamiento en que esté. No debes usarla cuando el ordenador está pasando información desde o hasta el cassette, ya que provocarías un error. Todas las pausas necesarias durante la grabación o la LECTACION, son manejadas por los programas propios del CPC464, por lo que estas pausas conseguidas mecánicamente no son empleadas frecuentemente.

### 2.2 Protección ante escritura (figura 2.3)



Con el fin de impedir que se borre accidentalmente la información que guardas en una cinta cassette, se dispone de unas lengüetas de protección. Estas lengüetas se pueden quitar fácilmente con unos pequeños alicates, para que no se pueda apretar la tecla **REC**, con lo que esa cinta está protegida ante escritura.

Esta faceta de protección se aplica separadamente a CADA CARA de la cinta, de manera que para proteger ambas caras de la cinta, debes quitar las dos lengüetas. Si a continuación inconscientemente, deseas hacer que vuelva a ser posible grabar en esa cinta que está protegida frente a escritura, simplemente te basta cubrir el hueco resultante al quitar la lengüeta con un trozo de cinta adhesiva.

### 2.3 Cargando las cintas

La figura 2.1 muestra la manera correcta de meter una cinta cassette en la ductora de cinta, usando la que se suministra con el CPC464.

La cinta debe estar completamente rebobinada (pasándola del carrete izquierdo totalmente hasta el carrete derecho, y si no lo está, presiona la tecla de rebobinado hasta que el motor se detenga en el extremo delantero de la cinta). Si accidentalmente se ha salido la cinta por la abertura frontal de su cajita, debes tensarla previamente y dejarla bien bobinada antes de insertarla en el equipo cassette, o la cinta y la información que contiene puede estropearse.

Observa por favor, que mientras que el mal uso de las cintas cassettes en los sistemas de audio simplemente lleva a que se escuchen mal por los daños sufridos en la superficie magnética de la cinta, ese mismo tratamiento aplicado a unas cintas informáticas, no es posible si quieres que funcionen con la suficiente fiabilidad.

Por ejemplo, cuando dañas la cinta porque atrapas un trozo suelto con la tapa del cassette, etc., y aunque compruebes que todavía puedes cargar y guardar programas en ella, lo que debes hacer inmediatamente es volver a guardar esa información en una cinta intacta y sin daños, en cuanto logres tener en la memoria del CPC464 un programa, y luego desechar esa cinta dañada antes de que te entren tentaciones de volver a usarla y en el momento más inoportuno te dé problemas.

## 2.4 La cinta de Bienvenida

Con el CPC464 se suministra una cinta cassette con unos cuantos programas de demostración sobre las posibilidades sonoras y gráficas del ordenador y de los programas RESIDENTES en el mismo: el intérprete del BASIC AMSTRAD y el llamado Sistema Operativo de la Máquina (MOS).

Parte de la función del conjunto de programas que es incorporado en fábrica dentro del ordenador y ya reside en su memoria permanentemente, denominado sistema operativo, es la de gestionar precisamente todas las operaciones con el cassette, incluyendo una serie de comandos que permiten leer y escribir datos y programas en la cinta. Los comandos más frecuentes son el de CARGA en memoria (**LOAD**), cargar y EJECUTAR el programa (**RUN**), y GUARDAR un programa en cinta (**SAVE**).

Para hacer la operativa del ordenador lo más simple posible, el CPC464 incluye ciertas funciones especiales que simplifican la manipulación de las teclas hasta que adquiere la suficiente práctica. Si ya has puesto en marcha el ordenador, y estás viendo el mensaje de "salutación" con el aviso:

Ready

que te indica que está **PREPARADO** para que le des alguna orden por teclado, puedes meter la cinta de Bienvenida tal y como te mostramos en la figura 2.1, "arredrar" el contador del cassette pulsando el pequeño botón situado a la derecha de la ventanita donde se muestra el número de vueltas, y luego pulsar la tecla de control, marcada **CTRL**, y mientras la mantienes apretada, pulsar la pequeña tecla **ENTER** en la esquina inferior derecha del teclado exclusivamente numérico. Inmediatamente el ordenador mostrará en pantalla el mensaje:

RUN"

Press PLAY then any key:

Que obviamente te indica que en cuanto aprietes **PLAY** y luego cualquier tecla, empezará a ejecutar el programa grabado en la cinta.

Para tu información, te puedo hacer notar que éste es otro de los ejemplos en que con un "toque" simultáneo de dos teclas, conseguimos enviar hacia el interior del ordenador un comando completo; como ya comentamos brevemente al hablar del comando **TECLA** en el capítulo 1. Observa que al tocar una tecla conjuntamente con otra, como **SHIFT** o **CTRL**, cambiamos el significado de esa tecla. Es una manera de escribir **TAQUIGráficamente**, con lo que una vez acostumbrados, se consigue acelerar enormemente la operación en el ordenador y el desarrollo de los programas. Seguiremos comentándolo a lo largo del manual, pero el efecto así conseguido es el mismo que si hubieras tecleado el comando **RUN** y luego pulsado **ENTER**, y lo logras más fácilmente con el toque simultáneo de dos teclas.

En la mayoría de las ocasiones, las dos teclas **ENTER** llevan a cabo la misma función, pero hay circunstancias en que la función desempeñada por la más pequeña de esas dos teclas, puede ser alterada (**redefinida**) para conseguir efectos distintos.

La tecla de la ductora de cinta marcada **PLAY** es la que te pide el ordenador que pulses, y debes hacerlo hasta que quede trabada en su posición más baja. La parte del mensaje en que te pide pulsar cualquier tecla, es una frase comúnmente usada por los programadores y que pueden confundir al principiante. Se usa para simplificar las operaciones de manera que el ordenador pueda proseguir con el programa en cuanto el operador haya efectuado la operación que se le pide.

Pero debe interpretarse siempre que se puede pulsar cualquier tecla normal, es decir, distinta de las marcadas **SHIFT**, [**CAPS LOCK**], **CTRL** y **ESC**, (y por supuesto, cualquiera de las teclas del panel de la ductora de cinta) pero inevitablemente todos los fabricantes de ordenadores tienen que hacer ciertas hipótesis con el fin de disminuir el espacio ocupado en memoria y hacer más simple la operación. Por lo tanto, siempre que aparezca la frase "**Press any key**" en esta Guía de Usuario, o en los programas suministrados por **AMSOFT** y otros vendedores, debe dar por supuesto que las excepciones mencionadas continúan aplicandose.

Observarás además que cualquiera que sea la tecla que pulses, no aparece en pantalla el símbolo correspondiente, y que lo que se logra con la pulsación es simplemente que el motor de la ductora de cinta arranque y que el mecanismo "lectador" de información comience a actuar. Si la cinta no comienza a pasar, pulsa otra tecla (la costumbre es pulsar la tecla **ENTER**, que para eso la hacemos más grande) y comprueba que no tienes presionada la tecla **PAUSE** de tu ductora de cinta. También si pulsas más de una tecla, el ordenador no lo tendrá en cuenta y comenzará igualmente a trasvasar la información desde la cinta hasta su memoria.

Observa también que en el comando no has reseñado ningún nombre de programa en particular. Y si no mencionas un nombre de programa en el comando, justamente después de las comillas, como en este caso:

run"

el ordenador sólo escrutará la cinta hasta que encuentre el primer programa grabado en ella, que es el que comenzará a cargar e inmediatamente a ejecutar, tal y como le has mandado. Cuando está CARGANDO el programa, te lo indica en pantalla mediante el mensaje:

Loading WELCOME 1 block 1

ya que así se llama -WELCOME 1- el primer programa de una serie de bloques que el fabricante de esa cinta ha preparado. En el CPC464, cada programa se GUARDA en cinta en la forma de bloques consecutivos de información (con un máximo de 2Kbs, (que ya veremos qué es) que son lectados y traspasados a la memoria del ordenador. Cada bloque de información está identificado en la cinta de manera unívoca, y el mensaje de la pantalla te indica el bloque corriente en cada momento. Después de trasvasar un bloque de información, la ductora detiene la operación momentáneamente, y luego vuelve a comenzar y casi al mismo tiempo se actualiza el mensaje que aparece en pantalla para indicarte el nuevo número del bloque corriente.

Si en cualquier momento el ordenador detecta información errónea, te mostrará un mensaje de error (Apéndice VIII) indicándote las características del error. Generalmente no es posible hacer ninguna otra cosa que no sea rebobinar la cinta e intentar volver a cargar el programa.

Supongamos que el ordenador ha cargado la cinta en memoria y comienza a ejecutar el programa. Vete leyendo las instrucciones que aparecen en pantalla y el programa de Bienvenida que gobierna el ordenador hará el resto.

## 2.5 Más-seguro o más-rápido

El CPC464 dispone de dos CADENCIAS para trasvasar datos desde y hasta la ductora de cinta:

- una cadencia supersegura de 1000 baudios
- una cadencia superrápida de 2000 baudios
- y recordando que los **baudios** (es una medida de los tiempos de la telegrafía para indicar el número de señales transmitidas por segundo, corresponderían en este caso por tanto, a 1000 bitios por segundo y 2000 bitios por segundo respectivamente).

Por lo tanto, a la cadencia de superrapidez se trasvasan datos desde y hasta la cinta dos veces más rápidamente que con la cadencia supersegura, sacrificando los márgenes para errores que algunas veces se precisan para tener en cuenta la falta de homogeneidad en las cintas de bajo coste y los errores suscitados por los diferentes alineamientos de los cabezales de lectura y escritura en las diferentes ductoras de cinta.

Para programas que se almacenen y recuperen usando la misma máquina, la cadencia de superrapidez ha demostrado que es suficientemente fiable si empleas cintas de calidad razonable. Con esa cadencia también podrás cargar la mayoría de las cintas comerciales con programas directamente y sin errores -aunque AMSTRAD siempre aconseja que todos esos programas sean grabados doblemente, con una y otra cadencia.

El ordenador automáticamente se prepara para leer a la cadencia con que está grabada una cinta. Pero cuando GUARDAS un programa, necesitas decirle al ordenador si deseas usar la cadencia superrápida, o por omisión lo hará en la cadencia supersegura que es la prescrita como normal.

Si eliges la cadencia de 2000 baudios para grabar tus programas y datos, comprueba que estás en modo directo (con el aviso **Ready** en pantalla) y teclea:

```
SPEED WRITE 1
```

con lo que se preparará para trasvasar datos a esa velocidad; y para regresar a la cadencia prescrita como normal, puedes simplemente restaurar el ordenador (en cuyo caso perderás todo lo que tengas en memoria) o simplemente teclear cuando esté PREPARADO:

```
SPEED WRITE 0
```

## 2.6 Grabando programas y datos en el cassette

BASIC tiene varios comandos en relación con la manera en que se graba información en el cassette. Lo resumimos brevemente aquí y presentamos algunos ejemplos.

### 2.6.1 SAVE "<nombre-fichero>"

El método más directo de depositar información en el cassette es usar el comando **SAVE** cuando el CPC464 muestra la palabra **Ready** después de ejecutar o listar un programa. Con el programa breve que presentaba el repertorio de caracteres visibles comentado en el capítulo 1 como objeto de nuestro comando de GUARDAR programas en cassette, trabajaremos en los ejemplos.

El <nombre-fichero> puede ser cualquier combinación de 16 caracteres (incluyendo los espacios en blanco). Si intentas emplear un NOMBRE DE FICHERO con más caracteres, no se tendrán en cuenta el carácter 17º y los siguientes. Cuando tengas el programa en la memoria del ordenador, teclea por ejemplo:

SAVE "CARACTERES"

a lo que el ordenador responderá con el mensaje:

Press REC and PLAY then any key:

Recuerda lo que hemos comentado sobre el término "cualquier tecla", y observa que la ductora de cinta se pone en marcha y el ordenador GUARDARA el programa identificándolo con el nombre fichero reseñado en el comando: CARACTERES.

### IMPORTANTE

Observa que el ordenador no puede detectar si has apretado o no las teclas del cassette correctas, así que si sólo aprietas **PLAY**, la ductora se pondrá en marcha y el programa parecerá que queda guardado pero no es así.

**PRECAUCION:** Si accidentalmente aprietas **REC** y **PLAY**, cuando lo que quieres lectar un programa de la cinta y cargar en memoria, lo que consigues es borrar el programa de la cinta, ya que estás grabando "nada". Si no lo interrumpes pulsando la tecla **ESC**, la cinta continuará marchando hasta el final y quedará completamente borrada ya que el ordenador no habrá encontrado ningún programa haciendo operaciones de grabación!. Si estás nervioso y quieres evitar la pérdida de datos por esta causa, hábituete a quitar la lengüeta de protección ante escritura de tus cassettes antes de que empieces a tener motivos para estar nervioso realmente.

Hay cuatro maneras en que puedes GUARDAR ficheros en el CPC464. Has visto el método más general, pero existen otros tres métodos alternativos para propósitos más especializados.

#### 2.6.2 SAVE "<nombrefichero>" ,A

El procedimiento es igual que antes, excepto que se interpreta el sufijo ,A por el ordenador como que requieres guardar el programa o los datos en la forma de un FICHERO TEXTUAL ASCII (literalmente con el código ASCII para cada símbolo que aparezca en el fichero), en lugar de utilizar la notación "taquizada" que el ordenador usaría si no le dices lo contrario.

Este método de conservar información se aplica primordialmente a los ficheros creados por los ELABORADORES DE TEXTOS y otros programas de aplicación que preparan ficheros para ser posteriormente tratados con otros programas.



**2.6.3 SAVE "<nombre fichero>" ,P**

La cláusula ,P en este comando le indica al ordenador que proteja la información que deposita en la cinta (y lo hace "cifrando" la información de manera que ni el servicio secreto podrá descifrarlo) y que sólo puede ser trasvasada a la memoria del ordenador usando los comandos RUN o CHAIN. Se hace para evitar que nadie pueda cargar lo del cassette en la memoria y luego detener la ejecución mediante la tecla ESC y simplemente listar el programa. Si tú mismo crees que posteriormente vas a editar o alterar el programa guardado de esa manera, mejor mantén una copia en lugar seguro y grabada en cinta sin este tipo de protección para que puedas hacerlo.

**2.6.4 SAVE "<nombre fichero>",B,<dirección inicial>, <longitud> [,<dirección entrada>]**

Esta opción te permite depositar en cinta un bloque de memoria con sus contenidos y en forma binaria, como una copia exacta de los valores alojados en cada una de las celdillas de memoria. Es necesario comunicar al ordenador la dirección inicial de la sección de memoria que quieres guardar, la longitud de la sección en bicos, y la dirección de carga en que comenzará la ejecución del programa si esa sección que guardas va a ser inmediatamente ejecutada después de ser recuperada de la cinta.

Esta posibilidad de grabar copias exactas de secciones de memoria permite que guardes directamente en el cassette la información que aparece en la pantalla, que es lo que solemos llamar VOLCADO o VERTIDO de la memoria en la cinta. Una de las formas de aprovechar esta facultad es para construir los "titulares" para programas que debido a su longitud tardan bastante en ser cargados en memoria.

**2.6.5 Ficheros sin nombre y el comando CATálogo**

Si guardas un fichero en cinta sin darle un nombre que lo identifique, usando:

SAVE ""

el ordenador lo guardará en la cinta como un "fichero sin nombre". En la cassette puedes guardar tantos ficheros como quepan en la cinta, usando los mismo nombres o distintos, o no dándoles nombres; quedarán secuencialmente uno detrás de otro, que en eso se diferencian de los discos en que es imprescindible asignar un nombre "unívoco" a cada uno de los ficheros que guardas.

Rápidamente le perderás la pista a tus programas si no les das nombres adecuados y que te recuerden de qué se trata. Y además te aconsejamos que siempre añadas algún tipo de referencia a la fecha dentro del nombre para que puedas saber cuál es la versión más reciente de tus programas y ficheros de datos.

Puedes sacar el CATALOGO con el contenido de una cinta si le tecleas el comando **CAT**, y luego llevas a cabo la operación que el ordenador te pide con el mensaje:

**Press PLAY then any key:**

El sistema irá enunciando los ficheros contenidos en la cinta, mostrando todos los <nombrs de fichero> MAYUSCULAS, seguido del número del bloque corriente, y luego mediante un solo carácter, la clase de fichero de que se trata:

\$ es un programa BASIC standard  
 % es un programa BASIC protegido  
 \* es un fichero textual ASCII  
 & es un fichero en notación binaria

Y al final de la línea, el aviso **OK** te indica que el fichero es perfectamente "lectable" y que hubiera sido cargado en memoria si le hubieses pedido que lo hiciera. Al efectuar la función **CAT** no se altera en absoluto el programa existente en la memoria del ordenador.

## 2.7 Read errors

Si en algún momento aparece en pantalla el mensaje **Read error**, es señal inequívoca de que se ha producido un **ERROR DE LECTADO** o **LECTURA**, durante la operación de carga que el CPC464 estaba realizando; la ductora de cinta continuará en marcha y el ordenador seguirá lectando los bloques que encuentre después del error, pero no intentará cargarlos a no ser que correspondan al primer bloque del programa que está intentando infructuosamente cargar.

Eso significa que después de este error, puedes detener la cinta apretando la tecla del cassette [**STOP/EJECT**], rebobinar hasta el principio de la cinta y apretar de nuevo la tecla **PLAY**. El ordenador hará otro intento de cargar el programa cuya carga fue fallida por el error, y con suerte esta vez la carga será lograda.

Los errores en las cintas son provocados por diversas causas, siendo la más común un daño accidental en la superficie de grabación y lectura de la cinta magnética por arañazos, dobleces, y otros defectos. También puede surgir de la práctica aparentemente inocente de apagar el ordenador mientras todavía está apretada la tecla **PLAY** o las teclas **REC** y **PLAY**.

Y es debido a que con una tecla pulsada, la cinta está frotando contra el correspondiente cabezal de lectado o grabación, y se puede generar un impulso con la extracorrente de ruptura al descargarse el alimentador y pasar a través de esos cabezales y dañar la cinta. Incluso, aunque la cinta por sí misma está estacionaria, este impulso de apagado (y también el de encendido) puede indudablemente, deteriorar la información en esa pequeña parte de la cinta y hacerla ilegible.

Además, la cinta estacionaria está sujeta a fricciones entre el pequeño cabestrante y el rodillo de arrastre, lo que al prolongarse excesivamente puede conducir a la creación de arrugas en la superficie de la cinta.

También se pueden producir errores en la operación de lectura si la cinta fue previamente grabada en otro CPC464 cuyo cabezal de grabación estuviera incorrectamente alineado.

También se producen errores ocasionalmente por razones puramente de azar. La cinta en cassette no fue diseñada originalmente como medio de almacenamiento de información "discreta" y como tal tiene varias desventajas que solamente se evitan en los equipos de almacenamiento en cinta profesionales mucho más complejos y considerablemente más caros.

A pesar de todo lo que pueda parecer, las cassettes efectúan una labor excelente al proveer un "medio" standard para el almacenamiento y recuperación de la información en los ordenadores, y lo hacen con un bajo coste. Las limitaciones debido a los tamaños de las partículas magnéticas de la superficie de la cinta, conjuntamente con la velocidad con que la cinta pasa por los cabezales de lectura y grabación, determinan un límite concreto sobre la cadencia con que se pueden transferir datos entre la ductora de cinta y el ordenador. Intentar sobrepasar esa cadencia más allá de la cota superior de tolerancia del sistema, provocará un funcionamiento poco fiable, particularmente con los métodos de duplicación masiva de cintas que se usa para comercializar los programas en soportes de bajo coste.

**OBSERVA:** Que las cassettes que contienen programas para otras clases de ordenador, no pueden ser cargados en el CPC464. Pueden parecer iguales, e incluso puede que hagan sonidos similares cuando se reproducen mediante un sistema de audio, pero no podrán ser cargados ni desde luego ejecutados. Si encuentras alguno que aparentemente carga y ejecuta, te rogamos encarecidamente que nos lo comuniques dándonos detalles de la clase de ordenador y del programa en cuestión. Ya tomaremos medidas.

### 2.8 Consideraciones sobre el Datacorder

Aunque el Datacorder -nombre comercial de esa ductora de cinta- acepta gustosamente todos los cassettes de cualquier duración hasta C90, debieras usar únicamente el C12 (6 minutos por cara) y como mucho el C30. Los programas grabados al final de las cintas largas son una pesadez a no ser que estés dispuesto a esperar un rato hasta que se encuentran (y que recuerdes el nombre de fichero que le diste), o que seas lo suficientemente meticuloso como para emplear el contador de cinta en un sistema INDEXADO, incluyéndolo por ejemplo dentro del nombre del fichero.

Si quieres escribir encima de un programa grabado en una cinta de larga duración, debes ser muy cuidadoso en localizar el punto de comienzo y procurar no sobre-escribir cualquier otro programa que desees conservar.

De forma general, diríamos que uses cassettes separadas con tan pocos programas como te sea posible. Las cassettes C12 son relativamente baratas y si dañaras la cinta por cualquier percance, es mucho menos atractivo sacar copias de un cassette de larga duración que sólo tiene algunas partes buenas.

Finalmente, recuerda por favor que los programas comerciales casi siempre se suministran bajo condiciones estrictas frente a las copias, y con los derechos reservados. No debes intentar copiar ni en ninguna manera duplicar programas grabados en cassette que no estén conformes con las condiciones de ventas de esos programas (algunos programas realmente te animan a que por lo menos saques una copia de reserva, y eso es lógico), y sobre todo, evitar eso de que es "sólo para un amigo". Las leyes sobre los derechos reservados se están revisando para contemplar todas las formas de duplicación de programas sin autorización, y aunque ha habido poca vigilancia hasta la fecha, esta situación cambiará sustancialmente en los próximos años, y puede que sea con efectos retroactivos. Y desde luego, hay unos ciertos valores éticos y morales que están siendo vulnerados al duplicar los programas. Y eso también importa.

### 3. Lo primordial de BASIC

Una breve introducción a la programación usando el lenguaje BASIC AMSTRAD.

Temas tratados en este capítulo

- \* Las reglas sintácticas y la descripción de la sintáxis
- \* Los comandos de exposición, los cauces de exposición y la forma de exposición.
- \* Zonas en pantalla

#### 3.1 Lo básico del BASIC

La relación fundamental del BASIC incorporado en el CPC464 con la operación interna de los ordenadores, se presenta en el Apéndice II. Si anteriormente no has programado un ordenador, intentaremos ayudarte en tu proceso de aprendizaje, pero puede que sea necesario hacer algunas hipótesis que pudieran causar confusión al principiante. Si así es, te sugerimos que ojees algunos de los muchos libros y cursos disponibles, que están planteados para presentar los aspectos fundamentales de la programación al recién llegado.

Serás capaz de sacar provecho de este capítulo si sigues los simples ejercicios presentados y que no exigen una comprensión completa de lo que pasa, aunque cuantas más reglas aprendas, más fácil te será todo lo que viene.

BASIC es el lenguaje CONGENITO en tu ordenador CPC464. En fábrica lo hemos incorporado y ya reside permanentemente en la memoria de tu ordenador. En cuanto lo enciendes, sentirás su presencia porque te saluda con la palabra:

Ready

que indica que está "dispuesto" y "preparado" para obedecer tus órdenes.

El lenguaje BASIC es el lenguaje más simple de aprender. Está organizado en torno a palabras claramente definidas y a una gramática simple, y opera de una manera completamente racional y lógica, en cuanto comprendas las reglas.

El BASIC AMSTRAD es capaz de obedecer los comandos que reseñamos en el capítulo 8. Cada comando se caracteriza y comienza por una o más **palabras clave**, que indican la operación que el ordenador hará al obedecer ese comando.

Los comandos incluyen una serie de **parámetros** -que permiten "matizar" la acción especificada por el comando y algunos de los cuales son opcionales. En general, cada parámetro puede venir designado por una expresión en la que intervienen constantes, variables y funciones. Las combinaciones de símbolos tomadas del repertorio admitido, sean letras, cifras, signos, etc., cuando están "ensartados" y deben ser tratados por el ordenador LITERALmente (es decir, letra a letra) lo denominamos datos **litéricos** o **literales**. Cuando solamente intervienen "dígitos", y lo que es más importante, ha de ser tratado como una cantidad, se denomina dato **numérico** o **numeral**, y hay diversas clases dependiendo de la precisión en enteros, reales, etc. y dependiendo del sistema de numeración empleado: decimal (base 10), hexadecimal (base 16) y binario (base 2). La información en las cintas de cassette se guarda en colecciones de datos relacionadas entre sí denominadas **ficheros**. Y se depositan y recuperan de la cinta SECUENCIALMENTE (lo que quiere decir, uno inmediatamente detrás de otro), en oposición a los ficheros que emplean como soporte los diskettes en que los ficheros se guardan con discernimiento, porque se puede DISCERNIR el sitio donde están guardados, y los cabezales de lectura y grabación pueden ser situados directamente en ese sitio, para depositar o recuperar la información.

### 3.2 La estructura de un programa en BASIC

Los programas en BASIC son series de **líneas de instrucciones**. Una línea puede incluir varias instrucciones, separadas por dos puntos, con la única limitación que el número máximo de caracteres que puede haber en la línea es de 255. Una línea puede ocupar varios renglones, pero es el número que precede a todas las líneas del programa el que delimita una línea de la siguiente. En el **modo directo**, damos comandos utilizando el teclado, que son en su mayoría iguales a las instrucciones pero que no comienzan con un número de línea. En el **modo programa**, el ordenador examina las líneas de programa existentes en su memoria y las va cumplimentando sucesivamente siguiendo un orden ascendente de números de línea a no ser que una de ellas le indique otra cosa.

El BASIC AMSTRAD permite que el usuario añada y quite líneas del programa mientras el ordenador está en el modo directo, preparado para recibir comandos y que corrija las líneas existentes. Antes de ejecutar o de LISTAR un programa, el BASIC reorganizará internamente el orden de las líneas para conservar una secuencia ascendente de números de línea, sin tener en cuenta el orden en que fueron tecleadas.

### 3.3 Imposición por teclado de una línea de programa

BASIC acepta líneas de programa siempre que tengan menos de 255 caracteres y estén terminadas pulsando **ENTER**.

Durante la imposición de una línea, es posible EDITAR la corriente línea, y usar las facilidades del cursor de COPIAR para insertar caracteres en la línea tomados de cualquier otro sitio del texto que aparece en pantalla. (Consulta capítulo 1, sección 1.2.7).

Todas las palabras clave deben estar precedidas y seguidas de un signo SEPARADOR, que puede ser bien un espacio en blanco, un signo o signo de puntuación (, : etc.) es debido a que está admitido usar variables cuyo nombre incluye una palabra clave (o palabra reservada), y obviamente el ordenador no puede distinguir el comienzo y terminación de la palabra clave cuando está "cercada" con otros caracteres que no sean separadores.

Puedes teclear las palabras clave usando letras minúsculas (sin pulsar simultáneamente **SHIFT**) o letras mayúsculas (pulsando simultáneamente **SHIFT**).

El comando **PRINT** puede abreviarse usando simplemente el signo de interrogación (?) y cuando se usa de esta manera, no se necesita el carácter delimitador. Los signos de operaciones matemáticas (+ - \* / MOD) también efectúan al mismo tiempo la función de delimitar las palabras clave, de manera que lo siguiente es válido, (aunque no te animamos a que lo emplees, dado que adquirirías malos hábitos tecleando líneas de programas sin espacio, cuando son siempre convenientes y algunas veces imprescindibles).

```
for n= 1 to 50: ?n: next
```

De igual manera, el apóstrofe (') conseguido pulsando simultáneamente **SHIFT** y la tecla marcada con 7, puede usarse en lugar de la palabra reservada **REM** cuando das instrucciones de comentario o MEMORandum.

Los espacios en blanco extras serán despreciados, y puedes usarlos para CONFORMAR el listado del programa indicando bucles, subrutinas, etc.

### 3.4 Terminología

Para describir los comandos y palabras clave del BASIC, debe usarse una terminología formal aunque simple. Cada comando se describe en la forma en que puede ser impuesto por el teclado, con cualquiera de las variables o parámetros opcionales señalados por "marcadores de posición" (varias clases de paréntesis y corchetes) que hacen referencia a elementos detallados en la descripción de la instrucción.

Estos elementos o partidas están representados por diversos nombres y encerrados entre corchetes angulados (< >). Por ejemplo, en los diversos sitios en que se exige un NUMERAL, es decir, directamente un número o bien una variable numérica, o bien una expresión que una vez evaluada dé como resultado un número, lo representaremos simplemente mediante: <numeral>.

Mientras que cuando sólo se acepte en la instrucción una constante numérica o número, escribiremos:

**<número>**

Todo lo que no está entre corchetes angulados, deberá ser tecleado tal y como se reseña. Por ejemplo, el comando **STOP**, adopta la forma:

**STOP**

Cuando en una definición de instrucción interviene una parte opcional, esa parte opcional se encierra entre corchetes cuadrados, por ejemplo si el numeral mencionado anteriormente fuera opcional, entonces aparecería:

**[<numeral>]**

Si esa parte opcional puede repetirse varias veces en la instrucción, (incluyendo la repetición 0 en que no aparece en absoluto), incluiremos un asterisco después del corchete cuadrado de cierre. Por ejemplo, una constante numérica, que requiera como mínimo la presencia de un dígito, aparecería

**<dígito>[<dígito>]\***

Y por tanto, a la hora de escribir la instrucción completa, podríamos sustituirla por:

3,  
ó 34  
ó 344  
ó 745978, etc.

En muchas descripciones de instrucción se usa una lista cuyos elementos están separados por comas. Se usa una forma abreviada, que se explica mejor mediante un ejemplo. Así:

**<lista de: <expresión> significa: <expresión>[,<expresión>] \***  
**<lista de: [#] <número> significa: [#] <número> [, [#] <número>] \***

Y a la hora de concretar la instrucción, debe ser sustituido por algo como:

# 3,4  
ó 3,4,4  
ó # 7,4,5,9,7,8, etc.

Estas LISTAS pueden ser de un solo elemento. Si la lista contiene más de un elemento, cada uno de los elementos sucesivos debe estar precedido por el signo coma, que marca el límite o delimita los elementos que el ordenador debe tratar por separado.

Los números pueden expresarse usando diferentes NOTACIONES:

**a. <números sin escala>**

... son números en los que no aparece la parte del exponente que se usa en la notación científica.



- b. <números con escala> que es la notación científica habitual en que el número está "elevado a la potencia" o según ESCALA, usando la forma:
- 2E4 (2 x 10 elevado a la potencia de 4, ó  $2 \cdot 10^4$ ), y la escala habitual es 10.
- ...y obviamente la parte del exponente puede ser positiva o negativa.
- c. <números con base>
- ...son números que expresan cantidades usando otro SISTEMA DE NUMERACION, bien sea el binario o el hexadecimal (Apéndice II):
- |  |           |
|--|-----------|
| Base 10 (la condición prescrita para omisiones)... | 100       |
| Base 16 (hexadecimal).....                         | &B40R&H64 |
| Base 2 (binario).....                              | &X1100100 |
- Y observa que mientras la H es opcional, la X es obligatoria.

### 3.5 La práctica perfecciona: Presentando PRINT

Para demostrar la manera en que funciona esta terminología, aquí hay algunos ejemplos prácticos.

Una de las instrucciones del BASIC que acepta la mayoría de las facetas de la terminología, es la correspondiente a la palabra-reservada **PRINT**. Un comando es una frase en BASIC que comienza directamente por una palabra clave y que hará que el ordenador la ejecute inmediata y directamente. Mientras que llamamos instrucción cuando esa frase está precedida por un número de línea de programa, y el ordenador cuando se le impone por teclado, simplemente la guarda en su memoria. Una FUNCION que entrega un resultado función del argumento, exige la presencia de un comando no solamente para "invocar la función", sino para poder EXPONER el resultado. Por ejemplo:

```
PRINT SQR(64)
```

Con esto, le estás pidiendo al CPC464 que exponga el resultado de aplicar la función raíz cuadrada (**SQR**) al número 64 (el argumento).

Fácil, pero está implícito que existe algún convenio para entendernos. O si no: ¿POR DONDE expone el resultado conseguido?

Efectivamente, el ordenador dispone de varios CAUCES de comunicación con el mundo exterior. Es decir, un cauce lleva los resultados a la pantalla y ya veremos que puede ser subdividida en varios "lienros". Otro cauce de comunicación es con la impresora, otro cauce de comunicación es con la ductora de cinta, etc.

La palabra reservada **PRINT** se utiliza para decirle al ordenador que exponga o "saque" algo (el resultado en este caso de aplicar la función raíz cuadrada) hasta un determinado "cauce" de salida". Y cada cauce queda identificado por un número del 0 al 9 y en las descripciones de las instrucciones aparece como <numeral de cauce>, que será la constante numérica, variable numérica o expresión cuyo valor es numérico, que determinará el cauce concreto que usará al cumplimentar esa instrucción.

Los cauces con números del 0 al 7, corresponden a "ventanas" o "lienzos" de pantalla que debemos haber estipulado previamente mediante el comando **WINDOW**.

El cauce 8, corresponde al **PORTAL** por donde sacamos en paralelo los datos hacia la impresora, y que solamente usaremos si tenemos conectada correctamente una impresora (compatible Centronics).

El cauce 9 es el que saca información hacia la ductora de cassette, donde también previamente habremos definido un **FICHERO** y lo habremos abierto adecuadamente. Por tanto, la descripción del comando **PRINT**, (usando sólo lo visto hasta ahora, que vendrá más) sería:

**PRINT**     [#<numeral del cauce>][<lista a exponer>]

Los corchetes cuadrados indican que no tiene que reseñar en el comando el <numeral del cauce> siempre y cuando quieras que saque la información hacia el órgano periférico prescrito para omisión de ese parámetro, y tenemos realmente que decir la lista de datos que queremos que exponga (y si no ponemos nada, el ordenador sacaría una línea en blanco hacia el cauce de salida que corresponda). El cauce prescrito, cuando no le indicas lo contrario, es el cauce principal #0, que corresponde a la pantalla y en la posición señalada por el cursor en ese momento. Ensaya con el comando ya clásico:

**PRINT "HOLA"**

y recuerda que debes concluirlo pulsando **ENTER** (y hasta que lo hagas, el ordenador sólo "escucha" lo que le dices y "repica" los caracteres tecleados en la pantalla, pero como el que oye llover). Sólo cuando pulsas **RETURN** ve que has concluido y pasa a examinar lo que le has tecleado y realizar lo que le pides si no detecta un error sintáctico. Solemos decir que simplemente lo que tecleas lo está metiendo en el "buzón de entrada".

El ordenador cumplimenta lo mandado y responde con:

HOLA`

situado donde corresponde según la posición corriente del cursor. Y observa además que las comillas no han sido enviadas hasta el cauce de salida. Las comillas son simplemente para señalar al **BASIC** de una manera precisa y exacta la **SARTA** de caracteres que tiene que exponer.

Teclea ahora:

```
PRINT #0, "HOLA"
```

...y el resultado es el mismo. Pero si tecleas:

```
PRINT #4, "HOLA"
```

observarás que **HOLA** aparece en la esquina superior izquierda de la pantalla, y es debido a que el primer envío al cauce número 4 -que por ahora no has estipulado usando el comando **WINDOW**, y que por tanto cubre todo el área de textos en la pantalla-, por lo que como todos los textos enviados por primera vez hacia un cauce de exposición por monitor o pantalla, la posición de comienzo es la esquina superior izquierda. El mensaje de salutación al poner en marcha el ordenador, usa el cauce #0 que es el prescrito como normal, así que en el caso anterior, ya lo habías usado y por tanto el dato a exponer se mostraba a continuación de los datos ya expuestos.

Esta facultad del BASIC AMSTRAD confiere al ordenador CPC464 un instrumento extremadamente potente, dado que te permite preparar en pantalla diversas "viñetas" usando comandos y funciones muy simples, con lo que contribuye a producir fácilmente imágenes del tipo mosaico, con una excelente presentación.

Observa por segunda vez que la sarta de caracteres encerrada entre las comillas, es aceptada **literalmente** por el ordenador, sin preocuparse en absoluto por lo que hay dentro, por eso le llamamos **lítero** o decimos que es un dato de índole **literal**. En esos líteros también puedes incluir palabras reservadas:

```
PRINT "4*4"
```

y el ordenador te responderá con:

```
4*4
```

Para indicar a BASIC que debe efectuar el producto **4\*4**, los números y el SIGNO OPERADOR (el símbolo de multiplicación **\***) debe ser accesible al BASIC, y dirigir el resultado hacia un cauce de salida. Por ejemplo:

```
PRINT 4*4
```

...y BASIC nos presentará la respuesta:

```
16
```

Observa que el número está desplazado una posición en relación con el margen izquierdo, dado que BASIC reserva este espacio para colocar cuando lo precisa el **signo monádico menos (-)** que identificaría un número negativo.

El comando **PRINT** tiene muchas otras variantes, y puede usar la posibilidad de **CONFORMAR** el resultado a exponer mediante la utilización de una serie standard de "PLANTILLAS" u "HORMA".

La <lista a exponer> en un comando **PRINT** hace referencia a la serie de datos que han de sacarse hacia el cauce especificado. Los elementos de la lista pueden ser numerales o LITERALES, incluyendo las variables litericas y las expresiones cuya evaluación da como resultado un lítero, que es tanto el contenido de una variable literica como lo que está encerrado entre comillas.

Si en el comando de exposición, se usa la cláusula **USING**, podemos exponer los números de acuerdo con la HORMA utilizada en dicha cláusula, de manera que pueden mostrarse alineados por la derecha, y con la parte fraccionaria limitada a un cierto número de dígitos.

### 3.6 PRINT USING y las ZONAS de pantalla

Al ponerse en marcha, el BASIC considera la pantalla subdividida en ZONAS que ocupan 13 posiciones a lo ancho. Cuando la instrucción **PRINT** incluye en la lista de datos a exponer una coma (,), el siguiente dato es TABULADO hasta situarse al principio de la siguiente zona. Si se dispone de menos columnas al exponer en la última de las zonas de las especificadas para esa zona, el BASIC comenzará a exponer el siguiente dato una línea más abajo. No corta el dato y pone guión para que continúe en la línea siguiente.

Si no se especifica la cláusula **USING**, el BASIC expone los números positivos precedidos por un espacio, y los negativos precedidos por el signo menos. Todos los números son separados del dato siguiente por un espacio en blanco. El punto decimal (que se usa para indicar la parte fraccionaria de un número) no se muestra si el número no posee parte fraccionaria.

El BASIC AMSTRAD no admite el uso de la tacla **TAB** como tabulador de columnas, dado que hay una considerable falta de uniformidad en los criterios sobre el significado de esta función, entre los diversos dialectos de BASIC. Al pulsar la tecla **TAB** muestra la flecha de avance a la derecha (+), que es lo mismo que si se pulsa **CTRL** y la letra **I**, pero que no tiene ningún otro propósito en la BASIC AMSTRAD.

### 3.7 PRINT TAB (<numeral entero>) (<lista a exponer>)

El efecto de esta instrucción se comprueba mejor mediante un ejemplo. Teclea este programa y observa el resultado:

```
5 MODE 2: INK 1,0: INK 0,9
10 FOR N=1 TO 5
20 ZONE 40
30 PRINT TAB(N*4) "HI", N
40 NEXT
```

Este programa muestra tanto la separación en ZONAS de exposición mediante la coma, y la función **TAB (numeral entero)**. Vuelve a ejecutarlo alterando antes la línea 10 para que sea:

```
10 FOR N=-5 to 5
```

La cláusula **TAB** hace que la primera posición del primer dato a exponer mediante la instrucción **PRINT**, avance el número de espacios en blanco especificados por el numeral entero. La zona puede estipularse dentro de la gama 1...255, como veremos en el capítulo 8.

La instrucción **PRINT USING** se emplea para conformar los resultados de los cálculos cuando es un **número real**, y poder conseguir una tabulación ordenada de las partes fraccionarias de los números. Es un concepto complicado que puede apreciarse mejor con los ejemplos prácticos, dado que la expresión general es:

```
PRINT [ # <numeral del cauce > , ] [ < lista a exponer > ] [ < cláusula USANDO > ]  
[ < separador > ]
```

que no puede calificarse como "amistosa", especialmente dado que la <cláusula USANDO> se divide a su vez en:

```
USING < literal de horma > ; [ < lista a usar > ]
```

en donde a su vez la <lista a usar> se vuelve a subdividir en:

```
< expresión > [ < separador > < expresión > ] *
```

Mejor ensaya con lo siguiente:

```
PRINT 123.456, USING "###.##";4567.896
```

apareciendo en pantalla como resultado, el siguiente:

```
123.456           %4567.90
```

que nos ilustra varios puntos. Primeramente, el primer elemento a exponer reseñado antes de la cláusula **USING** no se ve afectado. En segundo lugar, con la horma o plantilla que figura en la cláusula **USING** se reserva el número de posiciones a ocupar en pantalla por el dato numeral que viene detrás, y que por supuesto puede ser una variable en lugar de un número como en este caso. Y si este dato tiene más dígitos a la izquierda del punto decimal de los que has reservado mediante los signos "almohadilla" (#), sigue apareciendo el resultado tal y como se muestra en el ejemplo, pero precedido del símbolo de porcentaje para indicar que se ha rebasado el espacio reservado. Luego, observa que la coma reseñada después de **123.456** ha provocado que el siguiente dato se exponga al comienzo de la siguiente ZONA de exposición. Si hubiera sido un punto y coma en lugar de la coma, el segundo número se expondría separado del primero simplemente por un espacio en blanco. Los números se separan siempre por un espacio en blanco cuando aparecen en la misma línea de pantalla (por razones obvias!).

Observa además, que el segundo dato es REDONDEADO al aparecer en pantalla, y no se han TRUNCADO los dígitos que sobran por la derecha de acuerdo con la "plantilla" de exposición usada.

Ensayá ahora con esto:

```
PRINT 123.456, USING "#####.##+";4567.899
```

y verás que el signo más (+) aparecerá a continuación de mostrar el dato que está usando la plantilla. Un signo menos (-) precederá a los números negativos si se omite cualquier otra especificación.

Esta variante **PRINT USING** es una posibilidad muy aprovechable para producir tablas de resultados. Siempre te avisará si la **PLANTILLA** especificada para exponer un dato es demasiado restrictiva (usando el signo de porcentaje %).

## 4. Variables, operadores y datos

**Manipulando la información en un programa BASIC.**

**Temas tratados en este capítulo:**

- \* Conociéndose y familiarizándose
- \* Clases de variables: numéricas y literales
- \* Operadores, expresiones lógicas
- \* Tablas
- \* LISTAS

### 4.1 Detectando la palabra reservada

Observa (si ya no lo has hecho antes) que los comandos y funciones que son palabras reservadas en el BASIC AMSTRAD están delimitados por espacios en blanco, signos de puntuación, signos de operaciones numéricas, etc. Los programas son más fáciles de leer y de DEPURAR, porque aunque puedes teclear las palabras reservadas tanto en mayúsculas como en minúsculas, cuando se lista el programa, todas ellas se convierten siempre a MAYUSCULAS. Si no lo están, es una señal de que hay un error en la manera en que han sido tecleadas, y el programa no se ejecutará.

El BASIC AMSTRAD te permite que incrustes palabras clave dentro de los nombres de las variables. Una variable en BASIC es el nombre de un sitio de la memoria en el que alojas un dato específico. El nombre puede ser tan simple como una sola letra (y los nombres deben siempre empezar con una letra, no pueden empezar con un número), aunque es más fácil de leer y comprender un programa largo si empleas nombres para las variables que reflejen a qué se refieren.

**RESPUESTA=4\*4: print RESPUESTA**

Los nombres de las variables en el BASIC AMSTRAD pueden constar de 40 caracteres como máximo (y el primero debe ser una letra), y los 40 son significativos. Los nombres no pueden contener espacios en blanco, o el BASIC sólo leería los caracteres anteriores al primer espacio en blanco y luego señalaría el hecho con un error:

**Syntax error**

que nos indica que se ha impuesto una serie ilegal de caracteres como nombre de una variable (Apéndice VIII). Si quieres incluir separación entre las palabras de un nombre, usa un punto (.) en el sitio en que correspondería el espacio en blanco. Son posibles todas las formas habituales de variables múltiples, teniendo presente la necesidad de declararlas previamente mediante la instrucción **DIMensión**.

## 4.2 Formas de abreviar

Es tedioso teclear **PRINT** cada vez, así que puedes usar en su lugar el signo de interrogación, y el BASIC comprenderá que quieres decir **PRINT** (en tanto y en cuanto no esté colocado dentro de ninguna frase encerrada entre comillas). Observa que la opción **?** no exige que esté delimitado usando un espacio en blanco como para las palabras reservadas. Si la usas en una instrucción y no como un comando directo:

```
10?4*4
```

```
run
```

La respuesta también es la misma, pero ahora al listar este programa de una línea es como si hubiera habido magia...

```
List
```

```
10 PRINT 4*4
```

Y aquí también ha insertado un espacio en blanco antes de "expandir" el signo de interrogación. En instrucciones **PRINT** que usaran constantes literales encerradas entre comillas, sería...

```
10?"HOLA"
```

También puedes ser perezoso y prescindir de las comillas de cierre, como ya habrás observado al usar **CTRL** y **ENTER** para cargar el cassette de Bienvenida, ya que en pantalla aparecía el comando **RUN**". Eso mismo funciona en las líneas de programa, pero no es un buen hábito, dado que si posteriormente vuelves a editar esa línea y le añades algo, probablemente te olvides de cerrar las comillas, que al añadir algo, ya sí que son necesarias.

## 4.3 Líneas multi-instrucciones y cálculos combinados

Puedes efectuar una serie de operaciones con una sola línea de programa en BASIC, de hecho tantas como te permita la limitación de 255 caracteres por línea como máximo. Como siempre, las instrucciones deberán separarse mediante dos puntos (:). También es posible efectuar series de operaciones matemáticas:

```
?2*8/5+5-4*777E9/3
```

cuyo resultado será:

```
-1.036E+12
```

Sin embargo, es esencial comprender el orden en que BASIC reconoce los diversos **OPERADORES** matemáticos (+, -, \*, **MOD**, etc.) cometerás equivocaciones. Las prioridades son:

- ↑ Potenciación: elevar un número a una potencia dada.
- Menos monádico o unario (el signo menos para declarar un número como negativo.
- \* Multiplicación



- / División
- \ División entera: el resultado se corta dando solamente la parte entera del número y descartando la parte decimal.
- + Suma
- Resta

Todo lo contenido dentro de los paréntesis ( ) se calcula antes de cualquier otra cosa, y si el contenido de los paréntesis es precisamente un cálculo combinado, se maneja en el mismo orden de prioridades esbozado anteriormente; incluyendo cualquier paréntesis extra que hubiera dentro de los anteriores. Como es obvio, debes siempre tener tantos paréntesis de cierre o a derecha, como paréntesis de apertura o a izquierda tengas; y el no cumplirlo, suele ser la forma más frecuente de error sintáctico.

### 4.4 Progresando

Ya hemos hecho algún camino desde que presentamos la instrucción **PRINT** en la sección 3.5. Ya debes haber percibido lo suficiente de las reglas fundamentales del BASIC AMSTRAD para permitirte adentrarte en el reino propio del BASIC, genuinamente informático, en lugar de las facetas de pseudocalculadora. Las palabras clave en BASIC te las presentaremos a medida que las necesites, pero puedes consultar el capítulo 8, donde están ordenadas alfabéticamente aunque las descripciones de uso no son obvias porque falta el contexto en que aparecen.

Muchas palabras clave del BASIC dicen lo que quieren decir; el comando de **VAYA a 50**, sólo necesita para ser comprendido saber que **VAYA** en inglés es **GOTO**, y con ello se deduce que le estamos mandando ir a la línea numerada con 50 y ejecutar lo que allí encuentre. **END** es una palabra internacional, y por tanto le dice a BASIC que **TERMINE** el programa que está ejecutando y vuelva a presentarnos el aviso de **DISPUESTO** (que en inglés es **Ready**) para que podamos mandarle nuevas cosas.

El modo directo (o inmediato) te permite teclear una serie de comandos y no una sola, si separas los comandos con el signo dos puntos (:), sin embargo, una vez que concluyas esa serie de comandos (pulsando la tecla **ENTER**) serán cumplimentadas y la línea no la puedes volver a usar porque no queda como instrucciones en memoria. Aunque en el CPC464, no necesitas teclearla de nuevo porque puedes usar la posibilidad de **COPIAR** lo que hay en pantalla, mediante el cursor de copiar (y suponiendo que todavía sigue ahí, y no lo has mandado precisamente que limpie pantalla).

#### 4.5 Instrucciones condicionales y comparativas

BASIC utiliza ampliamente la capacidad propia del ordenador para hacer tareas sencillas repetitivas, a gran velocidad y sin sentirse aburrido. Unos cuantos comandos de programación están disponibles para ayudarte a preparar estos procesos repetitivos (BUCLES, los llamamos), en que hay comandos que inician, continúan y detectan cuándo debe terminarse de hacer las "rondas", porque se ha cumplido un conjunto predeterminado de condiciones.

Normalmente, el último de estas instrucciones que controlan los procesos repetitivos concierne a una EXPRESION DE RELACION. En otras palabras, qué clase de relación existe entre dos datos, y la clase viene determinada por el **operador de relación** usado en la expresión. Puedes comparar constantes con constantes, o variables con variables, o variables con constantes. Los operadores de relación son:

- < menor que
- <= menor que o igual
- = igual
- > mayor que
- >= mayor que o igual
- <> distinto (no igual)

Ahora te presentamos un breve programa que demuestra el uso de estos operadores, y está basado en un tema muy querido para nosotros. Si no estás observando en este momento nuestro saludo:

```
Amstrad 64K Microcomputer (v1)
c 1984 Amstrad Consumer Electronics plc
and Locomotive Software Ltd
```

BASIC 1.0

Ready

...pues pulsa las teclas CTRL, SHIFT y ESC, manteniendo pulsadas las tres simultáneamente, con lo que el ordenador restaurará las condiciones iniciales y presentará nuestro saludo. Ahora prosigue y teclea lo siguiente (las instrucciones para editar el programa se dan en la sección 1.2.7):

```
10 INPUT "CUAL ES TU SALARIO";SALARIO
20 IF SALARIO < 10000 THEN GOTO 30 ELSE 40
30 PRINT "EXIGE UN AUMENTO":END
40 PRINT "EXIGE UN COCHE GRANDE"
```

Ejecuta este pequeño programa, tecleando simplemente:

RUN

y el obligatorio ENTER, para que el ordenador se entere que has concluido. Aparecerá:

**CUAL ES TU SALARIO?**

y observa cómo el ordenador ha añadido automáticamente el signo de interrogación al final, cuando espera que respondas para IMPONER el dato que teclees. Contesta usando sólo números, no letras, signos de puntuación o monetarios, etc., y concluye el dato usando la tecla ENTER.

Añade ahora la línea 5 simplemente tecleándola, cuando el aviso de PREPARADO vuelva a aparecer:

**5 CLS**

Al ejecutarlo de nuevo, verás que se limpia la pantalla. Si tu primera respuesta fue menos de 10000, contesta ahora más de 10000 para que aprecies la diferencia. Ahora amplía el programa original añadiéndole lo siguiente como línea 50:

```
50 IF SALARIO > 30000 THEN PRINT  
"sobornate al contable"  
run
```

El comando END de la línea 30 termina el programa y hace que el ordenador vuelva a presentar su divisa Ready. Dado que no hay comando END en la línea 50, el programa prosigue comparando el valor de la variable SALARIO con 30000, y SI SÍ expone el mensaje correspondiente.

Sigue trabajando y añade otra línea de programa con número 60:

```
60 IF SALARIO > 25000 THEN PRINT  
".... y prestame uno de los grandes"  
run
```

Observa que si contestas por ejemplo 26000 cuando ejecutes este programa ampliado, verás que el BASIC pasa a través de la línea 50 como si no existiera (ya que la PREMISA de que el salario sea mayor de 30000 no se cumple), y pasa a efectuar la comparación y la operación reseñadas en la línea 60.

En este momento, es hora de que construyas por tí mismo un programa usando las instrucciones comentadas hasta ahora. Observa la manera en que este programa ha ido "creciendo"; y la mayoría de los programas evolucionan de esta manera, lo que nos da pie para presentarte lo que quizás sea el concepto más importante en programación y que utiliza la habilidad ideal del BASIC para organizar.

#### 4.6 Evolución: el origen de los programas

La manera en que el BASIC te permite construir programas a medida que vas avanzando y comprobando, es su FACULTAD más conveniente. Los puristas y teóricos arguyen que esta comodidad conduce a técnicas de programación "sin estructurar" y poco sistematizadas. Abordando los programas en cuanto se te ocurren las ideas; mientras que los realistas consideran que es la mejor manera de retener el interés de los estudiantes, que así puede ir comprobando el progreso que alcanza en cada etapa.

Tomemos otra vez nuestro ejemplo anterior y añadamos ahora la línea 70 en que mandamos al ordenador que haga una pausa haciendo 900 veces nada (con lo que tarda lo suficiente como para que puedas leer el texto en imagen), y que después de un "salto" VAYA a la línea 5.

```
5 CLS
10 INPUT "CUAL ES TU SALARIO"
20 IF SALARIO < 10000 THEN GOTO 30 ELSE 40
30 PRINT "EXIGE UN AUMENTO":END
40 PRINT "EXIGE UN COCHE GRANDE"
50 IF SALARIO >30000 THEN PRINT
   "sobornate al contable"
60 IF SALARIO >25000 THEN PRINT
   "... y prestame uno de los grandes"
70 for n=1 to 900: next n:goto 5
run
```

Observa que esta nueva línea, y las otras líneas de programa que añadimos, las tecleamos en minúscula para recordarte que el BASIC AMSTRAD las acepta de las dos maneras y al mismo tiempo reconoce la diferencia entre el nombre de una variable, una constante literal, y una palabra clave. Como cuando ha concluido una ronda, le mandamos que vaya a la línea 5, no es posible que este programa termine nunca; así que pulsa ESC por dos veces para escapar de él, y luego lista el programa para comprobar que el ordenador ha convertido todas las palabras clave a MAYUSCULAS, pero deja los nombres de las variables como n en minúsculas.

La línea 70 presenta lo que llamamos un BUCLE DE DEMORA, ya que el ordenador se limita a hacer variar n de 1 a 900 antes de ejecutar la tercera instrucción de esa línea: GOTO 5. Así el programa "recicla" sin acabar nunca. Somos reiterativos, como el programa. Pero ahora te recordamos que pulsando ESC una sola vez, detienes la ejecución del programa, pero pulsándola de nuevo es cuando se INTERRUMPE definitivamente y vuelves al modo de comandos, sin perder en absoluto el programa que tienes en memoria.

Lo más probable, a no ser que ESCAPES precisamente mientras el ordenador está en el bucle de demora de la línea 70, el programa se interrumpirá inmediatamente, dado que lo habrás pillado ejecutando la instrucción 10 en que está esperando que le teclees un dato. En todo caso, siempre te mostrará el número de línea vigente en el momento en que se vio interrumpido, y en este caso, lo probable es que el mensaje sea

### **Break in 10**

Pero si lo pillaras haciendo el bucle de demora, el mensaje después de la segunda pulsación de la tecla ESC, es:

### **Break in 70**

Si escapas durante ese bucle de demora de la línea 70, habrás SUSPENDIDO la operación que puede ser REANUDADA pulsando simplemente "cualquier tecla". Pero si has llegado a la interrupción del programa, también puedes hacer que SIGA a partir del punto en que estaba, si tecleas:

### **CONT**

Y el programa continuará a partir de la línea en que se le interrumpió.

Hagas lo que hagas con la tecla ESC no perderás el programa almacenado en memoria, a no ser que específicamente mandes la ordenador que "desnude" su memoria usando el comando NEW, (vas a empezar uno NUEVO) o le des un comando de restauración a las condiciones iniciales manteniendo pulsadas simultáneamente las teclas CTRL, SHIFT y ESC.

Con el CPC464 no hay necesidad de poner una "red de seguridad" para aquéllos que inadvertidamente RESTAURAN la máquina. La acción de quitar el programa existente en memoria debe hacerse deliberadamente. GUARDA todo en el cassette en cuanto tengas duda de si lo vas a querer usar posteriormente o no.

## **4.7 Más sobre variables numéricas y literales**

La esencia de la programación es el concepto de variable. Si el ordenador tratara exclusivamente con informaciones fijas, sería simplemente una ayuda electrónica al tecleado. Recuerda que si cualquier parte de una expresión matemática puede ser una variable, el resultado es ineludiblemente un dato asignado a una variable.

Las variables tienen tres atributos (cosas que se le pueden atribuir). Un nombre, una clase y una "organización". Los nombres ya los hemos comentado antes (4.1), las clases de variables son opcionales, así que podemos definir una variable de acuerdo con las reglas de la sección 3.4 en la forma siguiente:

**<nombre>[<marca de clase>]**

Las marcas de clase son:

- % para variables numéricas enteras, en que todo lo que pueda aparecer a la derecha del punto decimal (de la coma que se usa en muchos países) se desprecia. Las variables enteras ocupan menos espacio en memoria, por lo que en aquellos programas que no exijan manejar números con parte fraccionaria se ejecutarán más rápidamente si las variables se DEFinen como enteros. El comando **DEFINT** estipula que determinados nombres de variables corresponden a variables enteras. La gama de valores que puede tener una variable entera es de -32768 a +32767.
- ! para números reales, con su parte entera y con su parte fraccionaria separadas por el punto decimal. Las variables numéricas están prescritas para ser reales desde el momento en que pones en marcha el ordenador, así que si omites especificar lo contrario, se consideran reales lleven detrás o no el signo de admiración. La gama de valores que puede tener una variable numérica real es de 2.9E-39 a 1.7E+38.
- \$ indica una variable literal, cuyo valor es por tanto un lítero, o constante literal. Y un lítero es simplemente una "sarta" de caracteres cualesquiera. Lo importante es que el ordenador los trata LITERALMENTE (tal y como los escribes, al pie de la letra, etc.) y los reconoce porque están encerrados entre comillas. Por ejemplo:

**NOMBRE\$="ALVARO GARCIA"**

Usando esto en nuestro ejemplo evolutivo, añade la línea 6 y corrige la línea 60:

```
6 INPUT "Como te llamas";NOMBRE$
60 IF SALARIO >25000 THEN PRINT
    ".... y prestame uno de los grandes";NOMBRE$
run
```

Observa que hemos incluido un espacio en blanco al final del mensaje LITERAL a exponer en la línea 60 cuando se cumple la condición previa; y lo hacemos para evitar que se apelocone con el valor de la variable nombre\$, y si no ensaya no poniéndolo y así nos creerás. El signo ; al final de una instrucción **PRINT** o de una **INPUT**, suprime el envío automático por parte del ordenador del carácter de retorno de carro al final de cada una de esas instrucciones. Con el punto y coma se queda el cursor donde está, y no pasa a la línea siguiente.

También podemos practicar con variables enteras si añadimos la línea 61. Simplemente tecleala, y luego comprueba que el ordenador sigue teniendo organizado su programa. Listalo y verás:

```

5 CLS
6 INPUT "Còmo te llamas";NOMBRE$
10 INPUT "CUAL ES TU SALARIO";SALARIO
20 IF SALARIO < 10000 THEN GOTO 30 ELSE 40
30 PRINT "EXIGE UN AUMENTO":END
40 PRINT "EXIGE UN COCHE GRANDE"
50 IF SALARIO >30000 THEN PRINT
   "sobornate al contable"
60 IF SALARIO >25000 THEN PRINT
   ".... y prestame uno de los grandes";NOMBRE$
61 TARIFA.DIARIA=SALARIO/365
   PRINT "dispones de pts";TARIFA.DIARIA;" cada día"
70 FOR n=1 TO 5000: NEXT n:GOTO 5
run

```

Observa que hemos aprovechado para incrementar el número de rondas en el bucle de demora a 5000, ya que ahora exponemos más texto en pantalla. El resultado del salario por día, no aparece muy presentable y podríamos mejor REDONDEARLO para que fuera un número entero. Añade la línea 62...

```

62 REDONDEO%=TARIFA.DIARIA: PRINT
   "o bien pts";REDONDEO%;"si no re importan
   los centimos"
run

```

Observa que siempre debes reseñar la marca de clase %, para referirte a esa variable entera, ya que puedes tener una variable real con el mismo nombre y sin que se confundan por ser de clase diferente. Observa también cómo el ordenador "recorta" las líneas que son demasiado largas como para caber en un solo renglón y continúa en el renglón siguiente (y lo hace en los tres modos de imagen). Usa el modo 2 (80 columnas) para escribir programas largos o con líneas largas, dado que es mucho más fácil leer programas sin tener que volver atrás cuando se llega al final de una línea.

Para pasar al modo 2, ya sabes que simplemente tecleas:

```
MODE 2
```

Y para producir texto en negro sobre fondo blanco, lo que es bastante mas legible en el monitor CTM640, teclea directamente los siguientes tres comandos:

```

INK 1,0
INK 0,13
BORDER 13

```

Y ahora al listar verás el efecto.

#### 4.8 Conformando la imagen

Parte de la evolución en tu programa es el proceso de "pasarlo a limpio" de vez en cuando. La primera cosa que podemos hacer para que quede "bien arregladito" es RENUMERAR todas las líneas con múltiplos de 10, usando el comando RENUM. Basta que cuando veas el aviso de PREPARADO (Ready), teclees:

RENUM

Y al listarlo de nuevo verás:

```
10 CLS
20 INPUT "Cómo te llamas";NOMBRE$
30 INPUT "CUAL ES TU SALARIO";SALARIO
40 IF SALARIO < 10000 THEN GOTO 50 ELSE 60
50 PRINT "EXIGE UN AUMENTO":END
60 PRINT "EXIGE UN COCHE GRANDE"
70 IF SALARIO >30000 THEN PRINT
   "sobornate al contable"
80 IF SALARIO >25000 THEN PRINT
   ".... y prestame uno de los grandes"
90 TARIFA.DIARIA=SALARIO/365: PRINT
   "dispones de pts";TARIFA.DIARIA;" cada día"
100 REDONDEO%=TARIFA.DIARIA: PRINT
   "o bien pts";REDONDEO%;" si no te importan
   los centimos"
110 FOR n=1 to 5000: NEXT n:GOTO 10
```

Y observa que no sólo han quedado REDONDOS los números de línea, sino que también se han cambiado adecuadamente los números de línea que se citan dentro de las instrucciones (v.g. GOTO 50). Claro que no tendría mucha utilidad si el BASIC no mantuviera registrados todos los números de línea mencionados en un programa y los actualizara todos simultáneamente (cosa que en otros ordenadores...)

Vamos ahora a "maquillar" la imagen que el programa presenta en pantalla, y antes de ello vamos primeramente a hacer PASIVO el bucle de demora de la línea 110; para lo cual basta que convirtamos ese comando ACTIVO en un mero comentario o MEMOrandum, usando precisamente al principio el comando REM:

```
110 REM:FOR n=1 to 5000: NEXT n:GOTO 10
```

Insertando el comando REM: al principio de esa línea, hacemos que el resto sea meramente algo que el ordenador nos "remorará" cada vez que listemos, porque continua guardando esa línea de programa en memoria, pero no interviene en absoluto en la ejecución. En este caso, conseguimos así que no haya ninguna demora ni se vaya de un salto a la línea 10, sino que termine la ejecución del programa y quede en pantalla lo que el programa haya expuesto. Ahora teclea la línea:

```
15 mode 1
```



Con ello, fijamos el modo de imagen sin tener en cuenta el modo en que está cuando le pedimos que ejecute el programa: la línea 15 fijará el modo 1 en cuanto empiece a ejecutar. Como el comando **MODE** automáticamente al fijar el MODO, también LIMPIA la pantalla, resulta que la línea 10 que teníamos es ahora redundante, pero de todas formas la podemos dejar.

Ahora, EJECUTA el programa y vete respondiendo con los datos necesarios:

```
Como te llamas? Alvaro
CUAL ES TU SALARIO? 4000
EXIGE UN COCHE GRANDE
y sobornate al contable
dispones de pts 109.589041 cada dia
o bien pts 110
si no te importan los centimos
Ready
```

No es ni presentable ni elegante, especialmente al resultar cortadas algunas frases. Bueno, pues añade...

```
25 PRINT: PRINT
85 PRINT
```

y prepárate a corregir la línea 10, pasando al modo de edición mediante **EDIT 100**, y haz que sea:

```
100 REDONDEO%=TARIFA.DIARIA:PRINT
  "o bien pts";REDONDEO%;" si no te":PRINT
  "importan los centimos"
```

Ejecútalo de nuevo y comprobarás que todo está más "adecentado". Añade también la línea 120 para que el mensaje de PREPARADO (**Ready**) aparezca bastante más abajo:

```
120 ?::?:?:?:?:?
```

Y luego ejecuta otra vez el programa para comprobarlo. O mejor, suprime ese mensaje al mismo tiempo que aprovechas para hacer que comience a dar rondas y rondas y rondas:

```
120 GOTO 120
```

Una vez que hayas ejecutado esta versión, ya sabes que la única manera es ESCAPAR, usando **ESC**. Recuerda que el signo ? es un medio rápido de dar el comando **PRINT**. Ahora lista para comprobar el resultado que tenemos:

```

10 CLS
15 MODE 1
20 INPUT "Como te llamas";NOMBRE$
25 PRINT:PRINT
30 INPUT "CUAL ES TU SALARIO";SALARIO
40 IF SALARIO <10000 THEN GOTO 50 ELSE 60
50 PRINT "EXIGE UN AUMENTO": END
60 PRINT "EXIGE UN COCHE GRANDE"
70 IF SALARIO >30000 THEN PRINT
   "sobornate al contable"
80 IF SALARIO >25000 THEN PRINT
   ".... y prestame uno de los grandes";NOMBRE$
85 PRINT
90 TARIFA.DIARIA=SALARIO/365: PRINT
   "dispones de pts";TARIFA.DIARIA;" cada dia"
100 REDONDEO%=TARIFA.DIARIA: PRINT
   "o bien pts";REDONDEO%;" si no te": PRINT
   "importan los centimos"
110 REM:FOR n=1 to 5000: NEXT n:GOTO 10
120 GOTO 120

```

#### 4.9 Colocando el cursor

Hasta ahora, la mayoría de los comandos BASIC los hemos escrito usando la sintáxis del BASIC primitivo que puede ser comprendida por la mayoría de las máquinas que poseen algun clase de "interpretador" de BASIC. El comando **LOCATE** es una de las potentes facultades del BASIC AMSTRAD (y ésta también la tienen algunos otros) que te permite **UBICAR** el cursor de texto en cualquier posición de la pantalla:

```
LOCATE 10,4
```

coloca el cursor de textos en la décima columna de la cuarta línea, contando desde la parte superior de la pantalla. Si intentas darlo como un comando directo, el cursor sí se ubicará en las coordenadas mencionadas, pero el aviso **Ready** hará que comience una nueva línea, por lo que al fin y al cabo, el cursor terminará como siempre en la primera columna de la línea siguiente. Así que mejor añádelo como línea del programa 16:

```
16 LOCATE 10,4
run
```

y verás cómo se baja la primera solicitud de datos y cómo se "sangra" o "indenta" (se mete para adentro) de acuerdo con lo que le has mandado. La siguiente línea continúa como antes (alineada a la izquierda, es como llamamos) dado que la línea 25 lo que hace es meter un par de renglones en blanco. Pulsa **ESC** por dos veces para terminar y luego añade la línea 26:

```
26 LOCATE 10,4
```

Al ejecutar el programa, observarás que la segunda solicitud de dato el mensaje se escribe encima ("re-escribe") del primero. Puedes ahora continuar colocando el texto exactamente donde quieres que aparezca en la pantalla, y para ello te será de mucha utilidad usar las RETICULAS DE PANTALLA que con coordenadas horizontales y verticales te presentamos en el Apéndice VI.

Si lo que quieres es que todos los mensajes y preguntas aparezcan en la misma línea, una de las cosas que puedes hacer es preceder cada instrucción de ubicación del cursor con una instrucción de LIMPIE PANTALLA, para evitar confusión con los restos que pueda haber de la última exposición.

Observa que las coordenadas empleadas en el comando LOCATE pueden venir dadas como valores de variables numéricas enteras, y generar esos valores dentro del programa. Puedes practicar por tu cuenta, pero volveremos sobre el tema.

Ya hemos gastado lo suficiente por ahora el programa de salario, así que el siguiente ejemplo se basará en un tema diferente. Si quieres conservar lo que has construido hasta ahora, haz que el ordenador GUARDE el programa en cassette con los comandos que ya conoces del capítulo 2. Puede incluso que te haya empezado a gustar el programa a estas alturas, y que desees de vez en cuando consultarlo o ampliarlo con los comandos que te iremos presentando.

### 4.10 SI...PUES

Su uso es bastante inmediato y patente. **IF** <expresión lógica> **THEN** <serie de comandos>. Y con "expresión lógica" hacemos referencia a una condición o PREMISA cuyo resultado es de la clase lógica: cierto o falso.

Con la instrucción **IF** indicamos al ordenador que evalúe una expresión lógica para comprobar si es cierta, y si lo **THEN** ejecute la serie de comandos. Esta operación puede incorporarse en un programa para hacer bucles que pueden ser simplemente REPETITIVOS o RECURSIVOS (que ya comentaremos). Restaura el ordenador pulsando CTRL, SHIFT y ESC y teclea:

```
1 MODE 1
10 AMSTRAD=0
20 PRINT "AMSTRAD CPC464 colour personal
  computer"
30 AMSTRAD = AMSTRAD +1
40 IF AMSTRAD <10 GOTO 20
```

Ejecútalo y verás cómo se repite la instrucción de exposición en pantalla de la línea 20 hasta que se "cumple" la condición o premisa de la línea 40. Por lo tanto con la línea 40 le obligamos a dar un "salto hacia atrás" a la línea 20. Observa también el significado de la variable llamada **AMSTRAD** que va cambiando de valor cada vez que el ordenador efectúa una ronda de ese bucle, por lo que la podemos asimilar a un "contador de vueltas".

Si quieres ver más exactamente lo que está sucediendo con el valor de **AMSTRAD** durante este programa, puedes añadir la siguiente línea de programa:

```
35 LOCATE 1,20: PRINT AMSTRAD:LOCATE 1,AMSTRAD
run
```

Y si ha ido demasiado deprisa, lo puedes hacer más lento con un bucle de demora:

```
36 for n=1 to 500:next
```

Y ahora añadir un toque de color, (si tienes la opción CTM640) incluyendo:

```
34 BORDER AMSTRAD
```

Esta línea va cambiando el color del BORDE en justa correspondencia con los cambios de valor de la variable **AMSTRAD**. Lista otra vez el programa:

```
1 MODE 1
10 AMSTRAD=0
20 PRINT "AMSTRAD CPC464 colour personal
  computer"
30 AMSTRAD = AMSTRAD +1
34 BORDER AMSTRAD
35 LOCATE 1,20:PRINT AMSTRAD:LOCATE 1,AMSTRAD
36 FOR n=1 TO 500:NEXT
40 IF AMSTRAD <10 GOTO 20
run
```

Y como no tenemos duda de que todos queréis ver todos los colores disponibles en el CPC464, cambia la línea 40, para que sea:

```
40 IF AMSTRAD <26 GOTO 20
```

Ejecuta el programa y verás todos los colores disponibles, comenzando con el más oscuro y terminando con el blanco brillante. Puedes hacer más aclaratorio el mensaje que expones en la línea 35, si la editas (corriges) para que sea:

```
35 LOCATE 1,20:PRINT "Borde";AMSTRAD:
LOCATE 1,AMSTRAD
run
```

Y ya que estamos tratando de los diferentes BORDES (y puede que te guste más la palabra MARCO), cuando el programa haya concluido y el ordenador esté PREPARADO para aceptar tus comandos, teclea:

**BORDER 14,6**

Y verás cómo el borde o marco "parpadea" entre los colores 14 y 6. Tendrás que esperar hasta el siguiente capítulo para que te demos más explicaciones sobre la manera en que funcionan los gráficos y los colores. Pero para redondear esta sección, teclea el comando:

**ink 1,18,16**

y para finalizar, el comando:

**speed ink 1,5**

Verás que la CADENCIA (velocidad) con que el CPC464 cambia la TINTA que usa, es como para ir a buscarse una aspirina y restaurar las condiciones iniciales del ordenador. Recuerda hacer que GUARDE el programa en la cinta, antes de que se RESTAURE a sí mismo, si quieres impresionar a tus amigos.

#### 4.11 Tablas

Algunos programas requieren un montón de variables para poder "alojar" los datos que el programa procesa. Y se puede hacer, pero a menudo es difícil seguirles la pista a todas esas variables que están usando para almacenar información. Afortunadamente, BASIC ha previsto esta situación y la ha resuelto usando lo que denominamos TABLAS. También lo solemos llamar ringlas, porque es una palabra más cercana a la palabra inglesa ARRAYS, que responde al concepto de un conjunto de cosas dispuestas según una REGLA determinada.

¿Pero qué es una tabla? Te lo puedes preguntar y llegar a la conclusión de que en esencia, es un grupo de variables homogéneas con un determinado arreglo entre ellas, que nos permite seleccionar cada una de ellas mediante un nombre común y uno o varios números ordinales específicos.

La mejor manera de terminar de comprenderlo es tomar un ejemplo. Considera un programa que simula un juego de cartas, y que por tanto tenemos que tratar con cartas seleccionadas al azar.

Obviamente no podemos sacar la misma carta dos veces, y por lo tanto debemos mantener un REGISTRO de lo que ha pasado con cada carta individual. Una manera simple de hacerlo sería asignar a cada carta una variable con nombre distinto, y luego dar como valor de la variable uno entre dos valores; por ejemplo, un 1 para indicar que esa carta ya ha salido, y un 0 para indicar que todavía está en el montón.

Evidentemente, eso nos llevaría a usar 52 variables aisladas y de nombre diferente (a no ser que estés jugando con cartas trucadas) y además, tendrías que recordar cuál es el nombre de la variable que has asociado a cada carta. Es un caso ideal para ver cómo entran en acción las TABLAS o ringlas (lo que prefieras decir).

Primeramente, necesitamos dar a este conjunto de variables un nombre común, y vamos a darle por ejemplo el de CARTA. Ahora, para poder seleccionar una carta específica -un elemento de esa tabla- podemos simplemente darle un número. Así, podríamos usar los primeros 13 números para definir el palo de OROS, los siguientes 13 para el de COPAS, y así sucesivamente. Con ello nos quedaría que CARTA(6) sería el 6 de oros, mientras que CARTA(11) sería la sota y CARTA(13) sería el rey, y CARTA(14) el as de copas, y así sucesivamente. Bastante sencillo, parece.

Desafortunadamente, no podemos empezar planteándonoslo así y llevar a cabo el programa directamente, sino que tenemos que ser justos con el ordenador y avisarle para que OCUPE un espacio en memoria reservado para este grupo de variables. El comando DIM se usa para que sepa la dimensión del espacio que queremos que ocupe. Es decir, este comando DIM fija las dimensiones de la tabla y en nuestro ejemplo, con la baraja que hemos dicho, necesitamos ocupar espacio para alojar las 52 variables que forman parte de la tabla. Por tanto, tendríamos que teclear el comando:

**DIM CARTA (52)**

Y a pesar que le decimos al ordenador que "ocupe" espacio en memoria para 52 variables, es una de las situaciones en que no nos hace exactamente lo que le hemos dicho; ya que realmente reserva espacio para 53 elementos. Pero ya verás que lo hace por nuestra conveniencia. (El espacio de más que ocupa es para la CARTA que corresponde al número 0).

Ahora ya podemos empezar a escribir la estructura de un programa, o mejor un PARRAFO del programa que sea el que extrae una carta del montón:

```
10 DIM CARTA(52)
20 FOR X = 1 TO 52
30 LET CARTA(X)=0
40 NEXT X
```

```
1000 AZAR = RND(52)+1
1010 IF CARTA(AZAR) = 1 THEN GOTO 1000
1020 CARTA(AZAR) = 1
1030 RETURN
```

Observa que en el párrafo inicial, en la primera línea, ponemos la instrucción **DIM**. Y es debido a que siempre necesitamos comunicar al ordenador las dimensiones de la tabla que vamos a usar; y además, hay que tener cuidado si posteriormente queremos cambiarle las dimensiones, es preciso que antes le mandemos que **LIBRE** el espacio ocupado, mediante el comando **ERASE**, como ya veremos más adelante.

En las líneas 20 a 40 hacemos que todos los valores de la tabla sean 0. Luego en el **PARRAFO** que comienza en la línea 1000, hacemos que elija una carta al **AZAR** (que eso es lo que hace la función **RND(52)**), y que después de elegirla, compruebe que no había sido sacada anteriormente. Y si lo ha sido, el valor ya sería 1, por lo que le mandamos que **VAYA** a sacar otracarta al azar, y así continuará hasta que encuentre una que todavía permanezca en el montón. Cuando ya la encuentra, hacemos que cambie el valor correspondiente del elemento de la tabla para que nos indique que ahora ya ha sido sacada, y luego le mandamos que **VUELVA**, y eso lo hace el comando **RETURN**, al punto donde le corresponda.

Es bastante natural que el manejo de tablas no esté restringido a aquéllas que sólo tienen una dimensión, (en cuyo caso también se les suele llamar **RISTRAS**), sino que pueda ampliarse al número de dimensiones que deseemos. Y se logra simplemente añadiendo más números de "selección" al dar el nombre de la variable. Por ejemplo, una tabla de 1000 elementos, arreglados en 10 grupos de 100, y en que cada subgrupo de 100 está igualmente "arreglado" en 10 grupos de 10, se le comunicaría al ordenador mediante el comando:

```
DIM RINGLA(10, 10, 10)
```

Esta técnica es muy útil para dividir grupos enormes en subconjuntos o subgrupos más pequeños y por tanto, más manejables. En el ejemplo de la baraja, nos sería todavía más fácil tener en cuenta los cuatro palos, con 13 cartas cada uno, que componen la baraja y haber definido la **TABLA** en la forma:

```
DIM CARTA (4, 13)
```

Y claro, ahora si como primer **SUBINDICE** (que así llamamos a esos números) tiene el valor 1, serían oros, si tiene el valor 2 serían copas, si tiene el valor 3 serían espadas, y si tiene el valor 4 serían bastos. Mientras que el segundo de los subíndices nos diría exactamente de qué carta se trata, si es un 1 sería un As, y si es un trece sería un rey. Y también es lógico que podamos usar estas **TABLAS** no sólo para variables numéricas enteras o reales, sino también para variables literales. Y supongo que ya habrás adivinado que el nombre de la tabla deberá en ese caso, terminar con el signo \$.

Y supongo también que ya estás pensando en aplicaciones de esto para llevar las "fichas" de tus compañeros de estudios, la gente que ha reservado asiento en el cine, o la colección de discos que posees. Y muchas más.

#### 4.12 Series de datos

El comando **DATA** puede establecerse para reseñar en un programa una "serie" de **DATOS**, (y por tanto, son **CONSTANTES**, y no variables); y luego hacer que el ordenador **APUNTE**, mediante el comando **READ**, cada una de esas **CONSTANTES** de la serie como valor de las **VARIABLES** que deseemos. Los datos de la serie deben aparecer en una línea de programa que comience con la palabra reservada **DATA**, y separando las constantes que intervienen en la serie mediante el signo **,**. El ordenador irá apuntando esos datos cada vez que ejecute un comando **READ**, en forma **SECUENCIAL**; es decir, prepara un "puntero" que le indique en cada momento cuál es la última constante de la serie que le hemos mandado que apunte mediante el comando **READ**, y cada vez que apunte una constante de la serie como valor de la variable mencionada en el comando, avanzará el puntero una posición.

Veamos un ejemplo, teclea:

```
10 READ X,Y,Z
20 PRINT X;"+";Y;"+";Z;" = ";X+Y"Z
30 DATA 1,3,5
```

Las constantes de la serie pueden ser numéricas o literales, o de ambas clases. No te preocupes cuando no te quepan en una sola línea, simplemente comienza otra línea con la palabra reservada **DATA** al principio. Cuando el ordenador tropieza en el programa con una instrucción **READ**, interpreta que queremos que apunte la constante que le toca como valor de la variable mencionada en la instrucción; así que no le importa ni dónde le hemos reflejado ni cuántas instrucciones **DATA** le hemos dado. El sigue cambiando **SECUENCIALMENTE** su "puntero" interno. Lo único que tienes que cerciorarte es que concuerdan perfectamente en número y en clase, las constantes que pones en las instrucciones **DATA**, con las variables que mencionas en las instrucciones **READ**.

La única manera de interrumpir esta asociación **SECUENCIAL** de constantes a variables, es hacer que el ordenador **REPUNTE** al sitio que le menciones en lugar de al siguiente que le toque. Usas el comando **RESTORE**, y le indicas el sitio a donde quieres que **REPUNTE**, dándole a continuación un número de línea de programa, que obviamente debe corresponder al de una serie **DATA**. Y también, si no mencionas ningún número de línea a la que quieras que repunte, lo hará a la primera que haya en el programa. El ejemplo siguiente ilustra el uso de estos comandos:



```

10 FOR C =1 TO X
20 READ X$
30 PRINT X$;" ";
40 NEXT C
50 RESTORE
60 GOTO 10
70 DATA HOLA, COMO, TE, ENCUENTRAS, HOY

```

Observa que aunque la línea con la serie de constantes está colocada en el ejemplo al final del programa, la puedes numerar y por tanto, colocar donde te convenga.

En el ejemplo, lo único que hacemos con las constantes es apuntarlas sucesivamente como valores de la variable X\$ que luego exponemos en pantalla. Pero también podemos hacer que el ordenador las apunte en una variable, y luego hacer con la variable lo que deseemos. Por ejemplo, usarla en el comando que hace que el ordenador SUENE, comando SOUND. Teclea:

```

10 FOR n=1 TO 30
20 READ s
30 SOUND 1, s, 40, 5
40 NEXT n
50 DATA 100, 90, 100, 110, 120, 110, 100, 0
60 DATA 130, 120, 110, 0, 120, 110, 100, 0
70 DATA 100, 90, 100, 110, 120, 110, 100, 0
80 DATA 130, 0, 100, 0, 120, 150

```

Y si no oyes nada, y el programa lo has tecleado bien, ajusta el control de volumen en el costado derecho del ordenador.

Para concluir esta breve introducción al BASIC, aquí hay un programa que te permite jugar a una variante de las "21" con el CPC464. Demuestra cómo se usan muchas de las posibilidades del BASIC, y debieras comprenderlo fácilmente merced al uso de nombres significativos para las variables. Puedes embellecerlo con gráficos, y añadir ambiente con sonido; y en general, desarrollar el tema de la evolución de los mejores programas BASIC a partir de un núcleo "humilde".

El objeto del juego es conseguir acercarte al total de 21, añadiendo el valor facial de las cartas en tu mano, y luego la casa intenta igualarte o acercarse todavía más al total 21, sin sobrepasarlo para no verse "quebrado". Después de teclear la línea 1, usa el comando AUTO para que genere automáticamente los números de línea.

```
1 REM A LAS VEINTIUNA
10 REM INICIO
20 YC=2:CC=2
30 ASES=0
40 CACES=0
50 S=0
60 T=0
70 DIM MANOS$(4)
80 MANOS$(1)="TREBOLES"
90 MANOS$(2)="CORAZONES"
100 MANOS$(3)="PICAS"
110 MANOS$(4)="DIAMANTES"
120 CLS
130 DIM CARTA (52)
140 FOR X=L TO 52
150 CARTA (X)=0
160 NEXT X
170 REM REPARTE DOS NAIPES PARA CADA JUGADOR
180 LOCATE 10,3)
190 PRINT "TU";SPC(15)"LA BANCA"
200 LOCATE 3,5
210 GOSUB 740
220 S=S+F
230 IF F=11 THEN ASES=ASES+1
240 LOCATE 3,6
250 GOSUB 740
260 S=S+F
270 IF F=11 THEN ASES=ASES+1
280 LOCATE 24,5
290 GOSUB 740
300 T=T+F
310 IF F=11 THEN CACES=CACES+1
320 LOCATE 24,6
330 GOSUB 740
340 T=T+F
350 IF F=11 THEN CACES=CACES+1
360 REM INPUT SIGUES (P) O QUEDAS (Q)
370 X$=INKEY$:IF X$(">"Q" AND X$(">"P" THEN 370
380 IF X$="Q" THEN 560
390 LOCATE 3,YC+5
400 YC=YC+1
410 GOSUB 740
420 S=S+F
430 IF F=11 THEN ASES=ASES+1
```

```

440 REM PUNTUACION Y ASES
450 IF S<22 THEN 370
460 IF ASES = 0 THEN 500
470 ASES = ASES-1
480 S=S-10
490 GOTO 450
500 LOCATE 12,19
510 PRINT "QUIEBRAS"
520 PRINT:PRINT"OTRA PARTIDA (S/N)"
530 X$=INKEY$:IF X$<>"S" AND X$<>"N" THEN 530
540 IF X$="S" THEN RUN
550 END
560 IF T>16 THEN GOTO 700
570 CC=CC+1
580 LOCATE 24,CC+4
590 GOSUB 740
600 T=T+F
610 IF F=11 THEN CACES=CACES-1
620 IF T<21 THEN 560
630 IF CACES = 0 THEN 670
640 CACES = CACES-1
650 T=T-10
660 GOTO 620
670 LOCATE 12,19
680 PRINT "TU GANAS"
690 GOTO 520
700 LOCATE 12,19
710 IF T<S THEN 680
720 PRINT"GANA LA BANCA"
730 GOTO 520
740 REM REPARTE NAIPE
750 LET AZAR=INT( RND(1)*52+1)
760 IF CARTA(AZAR)=1 THEN GOTO 750
770 CARTA(AZAR)=1
780 F=AZAR-13*INT(AZAR/13)
790 IF F=0 THEN F=13
800 IF F=1 OR F>10 THEN GOTO 850
810 PRINT F;" DE ";
820 IF F>10 THEN F=10
830 PRINT MANOS$(INT((AZAR-1)/13)+1)
840 RETURN
850 IF F=11 THEN PRINT "JOTA DE ";
860 IF F=12 THEN PRINT "DAMA DE ";
870 IF F=13 THEN PRINT "REY DE ";
880 IF F<>1 THEN GOTO 820
890 F=11
900 PRINT "AS DE"
910 GOTO 830

```

Pulsa **S** si "sigues" (y tu siguiente carta se sumará a las que ya tengas), o **Q** si "quedas" y juega la casa. No pretendemos que sea la última palabra en juegos de cartas con el CPC464, pero te proporciona el hueso sobre el que puedes añadir carne en forma de gráficos y sonido.

#### 4.13 Expresiones Lógicas

Una de las diferencias mayores entre una calculadora y una computadora, es la habilidad del ordenador para manejar operaciones **lógicas** en instrucciones condicionales, tal como **SI...PUES**. Para hacer eso, los operadores lógicos tratan los valores sobre los que operan de acuerdo a su **equivalente** en el sistema de numeración **binario**; y además, lo hacen tomando individualmente los **bits** HOMOLOGOS de cada operando y lo tratan por separado. Aunque la descripción y utilización es completamente **LOGICA** (pero de lógica matemática), es bastante difícil describirla en términos simples sin recurrir a las definiciones precisas. Pero vamos a intentarlo. (y el Apéndice III ayudará).

Las dos partes de una expresión lógica, se conocen como **operandos** y el signo de operación es normalmente una breve palabra que llamamos **operador lógico**. Por lo tanto, la sintaxis de una expresión lógica es

**<operando>[<operador lógico><operando>]**

y el operando puede ser a su vez cualquier **<numeral entero>**, cualquier **<expresión relacional>**, o cualquier **<expresión logical>**.

Y te resaltamos que el numeral entero debe estar dentro de la gama permitida de valores, y en caso contrario se produce un **Error 6**.

Esto que llamamos operaciones **logicales**, está como veremos, estrechamente vinculado a esas palabras que en la vida real nos sirven para conectar frases condicionales, como por ejemplo en ésta:

Si hace sol y **si** nieva, o **si** llueve, iré al museo.

Pero es algo más complicado porque el ordenador sólo nos interpretará correctamente **si no** hay ambigüedades.

Los operadores lógicos según el orden de **PRIORIDAD** en la ejecución, son simbolizados en BASIC con las palabras clave **AND**, **OR** y **XOR**. Y los resultados de una operación con dos operandos de **un solo** bit, es como sigue:

<b>AND</b>	Da como resultado 0 a no ser que ambos operandos sean 1.
<b>OR</b>	Da como resultado 1 a no ser que ambos operandos sean 0.
<b>XOR</b>	Da como resultado 1 a no ser que ambos operandos sean iguales.

Representando con 1 y 0 los dos símbolos de un sistema de numeración binario.

```

440 REM PUNTUACION Y ASES
450 IF S<22 THEN 370
460 IF ASES = 0 THEN 500
470 ASES = ASES-1
480 S=S-10
490 GOTO 450
500 LOCATE 12,19
510 PRINT "QUIEBRAS"
520 PRINT:PRINT"OTRA PARTIDA (S/N)"
530 X$=INKEY$:IF X$(">"S" AND X$(">"N" THEN 530
540 IF X$="S" THEN RUN
550 END
560 IF T>16 THEN GOTO 700
570 CC=CC+1
580 LOCATE 24,CC+4
590 GOSUB 740
600 T=T+F
610 IF F=11 THEN CACES=CACES-1
620 IF T<21 THEN 560
630 IF CACES = 0 THEN 670
640 CACES = CACES-1
650 T=T-10
660 GOTO 620
670 LOCATE 12,19
680 PRINT "TU GANAS"
690 GOTO 520
700 LOCATE 12,19
710 IF T<S THEN 680
720 PRINT"GANA LA BANCA"
730 GOTO 520
740 REM REPARTE NAIPE
750 LET AZAR=INT( RND(1)*52+1)
760 IF CARTA(AZAR)=1 THEN GOTO 750
770 CARTA(AZAR)=1
780 F=AZAR-13*INT(AZAR/13)
790 IF F=0 THEN F=13
800 IF F=1 OR F>10 THEN GOTO 850
810 PRINT F;" DE ";
820 IF F>10 THEN F=10
830 PRINT MANOS$(INT((AZAR-1)/13)+1)
840 RETURN
850 IF F=11 THEN PRINT "JOTA DE ";
860 IF F=12 THEN PRINT "DAMA DE ";
870 IF F=13 THEN PRINT "REY DE ";
880 IF F(">"1 THEN GOTO 820
890 F=11
900 PRINT "AS DE"
910 GOTO 830

```

Pulsa **S** si "sigues" (y tu siguiente carta se sumará a las que ya tengas), o **Q** si "quedas" y juega la casa. No pretendemos que sea la ultima palabra en juegos de cartas con el CPC464, pero te proporciona el hueso sobre el que puedes añadir carne en forma de gráficos y sonido.

#### 4.13 Expresiones Lógicas

Una de las diferencias mayores entre una calculadora y una computadora, es la habilidad del ordenador para manejar operaciones **lógicas** en instrucciones condicionales, tal como **SI...PUES**. Para hacer eso, los operadores lógicos tratan los valores sobre los que operan de acuerdo a su **equivalente** en el sistema de numeración **binario**; y además, lo hacen tomando individualmente los **bits** HOMOLOGOS de cada operando y lo tratan por separado. Aunque la descripción y utilización es completamente LOGICA (pero de lógica matemática), es bastante difícil describirla en términos simples sin recurrir a las definiciones precisas. Pero vamos a intentarlo. (y el Apéndice III ayudará).

Las dos partes de una expresión lógica, se conocen como **operandos** y el signo de operación es normalmente una breve palabra que llamamos **operador lógico**. Por lo tanto, la sintáxis de una expresión lógica es

**<operando>[<operador lógico><operando>]**

y el operando puede ser a su vez cualquier **<numeral entero>**, cualquier **<expresión relacional>**, o cualquier **<expresión logical>**.

Y te resaltamos que el numeral entero debe estar dentro de la gama permitida de valores, y en caso contrario se produce un **Error 6**.

Esto que llamamos operaciones **logicales**, está como veremos, estrechamente vinculado a esas palabras que en la vida real nos sirven para conectar frases condicionales, como por ejemplo en ésta:

Si hace sol y **si** nieva, o **si** llueve, iré al museo.

Pero es algo más complicado porque el ordenador sólo nos interpretará correctamente **si no** hay ambigüedades.

Los operadores lógicos según el orden de PRIORIDAD en la ejecución, son simbolizados en BASIC con las palabras clave **AND**, **OR** y **XOR**. Y los resultados de una operación con dos operandos de **un solo** bit, es como sigue:

<b>AND</b>	Da como resultado 0 a no ser que ambos operandos sean 1.
<b>OR</b>	Da como resultado 1 a no ser que ambos operandos sean 0.
<b>XOR</b>	Da como resultado 1 a no ser que ambos operandos sean iguales.

Representando con 1 y 0 los dos símbolos de un sistema de numeración binario.

## Más BASIC

El operador **AND** es el más comúnmente empleado.

```
PRINT 10 AND 10
```

Y el ordenador te responderá 10.

```
PRINT 10 AND 12
```

Y el ordenador te responderá 8.

```
PRINT 10 AND 1000
```

Y te responderá de nuevo 8.

Es debido a que los números 10 y 1000 tienen como equivalente en el sistema de numeración binaria (Apéndice II), respectivamente los siguientes:

```
      1010
1111101000
```

Y la operación **AND** (también llamada **YLIACION**) va examinando el par de bits homólogos cada vez y sólo coloca en la posición homóloga del resultado un 1 cuando ambos operandos son 1. Por lo tanto, el resultado es:

```
0000001000
```

...cuyo equivalente en el sistema decimal es precisamente 8, como nos decía el ordenador. Pero en BASIC, es más normal usar ese operador **AND** para detectar si se cumplen simultáneamente dos condiciones (...Y SI...). Aquí hay un programa autoexplicativo:

```
10 CLS:INPUT "Numero del dia";dia
20 INPUT "Numero del mes";mes
30 IF dia=25 AND mes=12 GOTO 50
40 GOTO 10
50 PRINT "¡Feliz Navidad!"
```

La operación **OR** (también llamada **OLIACION**) opera bit a bit de la misma manera, pero ahora el resultado es 1, salvo que ambos operandos sean 0. Usando la misma pareja de números que antes, tendríamos:

```
PRINT 1000 OR 10
      1002
```

Y haciendo la operación hallando el equivalente binario, tendríamos:

```
      1010
1111101000
```

Que nos da como respuesta:

```
1111100010
```

que corresponde en el sistema decimal a nuestro resultado de 1002.

Todo eso significa que podemos usar el operador **OR** para detectar si se cumple una u otra de dos condiciones, incluyendo el caso de que se cumplan las dos al mismo tiempo. Y aquí hay un programa ejemplo:

```
10 CLS:
20 INPUT "Numero del mes";mes
30 IF mes=12 OR mes=1 OR MES=2 GOTO 50
40 CLS:GOTO 10
50 PRINT "Debemos estar en invierno"
```

Y también existe el operador lógico **NOT** (también llamado **NEGACION**) que opera únicamente con un operando (operadores "monádicos") y que en el caso de que el operando tenga un solo bit, podemos decir que el resultado es 0 cuando el operando es 1 y viceversa. Se emplea obviamente para comprobar que **NO** se cumple una determinada condición. Como en el programa:

```
10 CLS
20 INPUT "Numero del mes";mes
30 IF NOT(mes=6 OR mes=7 OR mes=8) goto 50
40 CLS:GOTO 10
50 PRINT "¡No puede ser verano!"
```

La consideración final más importante es que podemos combinar un cierto número de condiciones lógicas y con varios operadores lógicos, como te mostramos en este ejemplo:

```
10 CLS:INPUT "Numero del dia";dia
20 INPUT "Numero del mes";mes
30 IF NOT(mes=12 OR mes=1) AND dia=29 GOTO 50
40 CLS:GOTO 10
50 PRINT "Ni es Diciembre, ni es Enero, y puede que sea
un año bisiesto"
```

Ahora bien, ten en cuenta que el resultado de una expresión de relación es también un número que corresponde a 0 o a -1. Y además, la representación en el sistema binario del número decimal 0, es que todos los bits sean 0, mientras que la del número -1 es que todos los bit sean 1. El resultado de una operación lógica con dos operandos que sean expresiones de relación, nos producirá como resultado el número decimal -1 (cierto) o el número decimal 0 (falso).

Comprueba esto, añadiendo la línea 60 y 70 al programa anterior:

```
60 PRINT NOT(mes=12 OR mes=1)
70 PRINT (mes=12 OR mes=1)
```



Y cuando ejecutes el programa, teclea 29 como el día, y digamos, 5 como el mes, con lo que se producirá la respuesta según la línea 50, y los valores reales que son resultado de las expresiones lógicas de las líneas 60 y 70, aparecerán en las líneas más abajo.

Finalmente, el operador **XOR** (igual que el operador **OR** pero **eX**cluyendo el caso en que se cumplen las dos condiciones) produce el resultado **CIERTO** siempre que ambos argumentos sean distintos. La tabla resumen de todas estas combinaciones, es lo que se llama una **TABLA DE LA VERDAD**; que es una forma conveniente de ilustrar lo que sucede en la operación lógica cuando los operandos sólo tienen un bit.

Operando A	1	0	1	0
Operando B	0	1	1	0
Resultado AND	0	0	1	0
Resultado OR	1	1	1	0
Resultado XOR	1	1	0	0

## 5. Lo primordial de Gráficos

Siguiendo tu camino con los colores y gráficos del CPC464.

Temas comentados en este capítulo:

- \* Modos de imagen y motas
- \* Los colores
- \* Tinta, Papel y Pluma
- \* Dibujando líneas
- \* Ventanas o viñetas

### 5.1 Peculiaridades de la máquina

La descripción y aplicación del BASIC AMSTRAD hasta ahora han estado basadas en especificaciones standard en la industria. La mayoría de las operaciones puramente aritméticas sólo necesitan ligeros ajustes al pasar de un **dialecto** de BASIC a otro; sin embargo, los comandos BASIC que controlan los gráficos (y, la ubicación del cursor de textos) son más específicos de cada máquina, y deben ser comprendidos cuidadosamente para sacar el mayor provecho de tu microordenador.

Como siempre, las palabras clave del BASIC las presentaremos en **negrita** para destacarlas, y también están relacionadas alfabéticamente con una breve descripción en el capítulo 8.

#### 5.1.1 Selección de colores

El NEGRO (la ausencia de color o iluminación en la pantalla) se considera como un color para el propósito de describir las posibilidades de colores del CPC464 y los diversos comandos asociados a esas posibilidades.

El BORDE puede fijarse a "cualquier par de colores" sin tener en cuenta el modo de imagen, y no se restaura a las condiciones iniciales cuando se dicta al ordenador el comando **MODE**. Puede hacerse que parpadee o palpite entre dos colores, o que muestre un color de manera permanente.

El número de TINTAS disponibles, sólo puede aprovecharse simultáneamente dependiendo del MODO DE IMAGEN elegido. También cada TINTA puede fijarse para que palpite con una pareja de colores; o que corresponda a un solo color. El PAPEL y la PLUMA usada en textos y la PLUMA usada en gráficos, puede estipularse para que corresponda a una de las tintas disponibles.

### 5.1.2 Las relaciones entre PLUMA, PAPEL y TINTA

Excepto cuando se especifica el parpadeo entre dos colores, hay dos TINTAS que afectan al texto mostrado en la pantalla: una de esas TINTAS determina el color con que escribe la PLUMA; mientras que la otra determina el color con que se da el "tinte" al PAPEL.

**NB:** El número asociado con el comando **PAPER** es la **INK** declarada para ese número, y **NO** el número de color enunciado en el Apéndice VI. Igualmente, el número asociado con el comando **PEN** es la **INK** declarada para ese número, y **NO** el número de color del Apéndice VI.

El número en el comando **PAPER** está prescrito que sea 0 cuando se omite otra mención explícita de número. Mientras que el número del comando **PEN** está prescrito a 1 para las omisiones. Para hacer que la TINTA del PAPEL número 0 sea "verde", cuyo número de color es el 9, tendremos por tanto que teclear:

**INK 0,9**

Igualmente, para estipular que la TINTA de la PLUMA número 1 sea "negro", cuyo número de color es 0, teclearemos:

**INK 1,0**

Por supuesto que poniendo el PAPEL a la misma TINTA que usa la PLUMA no podrás observar el texto que haya en pantalla.

La escritura de texto puede hacerse que sea "transparente" (y el cursor cuadrado que indica textos, siempre está puesto para que sea transparente) u "opaca", usando una serie de caracteres de control que permiten ampliar provechosamente los comandos para gráficos del BASIC. Con la opción de transparencia, puedes no tener en cuenta el color del papel y sobrescribir los gráficos, o puedes sobrescribir completamente el fondo. Este breve programa ilustra los efectos disponibles, pero restaura antes el ordenador pulsando **CTRL, SHIFT y ESC**, como ya estás acostumbrado.

```
10 MODE 1
20 INK 2,19
30 DRAW 200,200,2
40 LOCATE 1,21
50 PRINT "1 NORMAL"
60 PRINT CHR$(22)+CHR$(1)
70 ORIGIN 0,0
80 DRAW 500,200,2
90 LOCATE 12,18
100 PRINT"2 TRANSPARENTE"
110 PRINT CHR$(22)+CHR$(0)
120 LOCATE 22,15
130 PRINT"3 NORMAL DE NUEVO"
```

El primer comando **DRAW** hace que el ordenador DIBUJE un segmento, operación que hace antes de que "pongamos" la opción de transparencia, lo que hacemos en la línea 60 mediante el envío a pantalla de **CHR\$(22) + CHR\$(1)**, mientras que el segundo segmento que dibujamos en la línea 80 lo hace ya con la opción transparencia puesta. Observa cómo los puntos en que se solapan han cambiado de color, y cómo (con la opción transparencia), la tinta que rellena el carácter, ha sido completamente sobreescrita.

Canjea la posición de las instrucciones que hacen que el ordenador "ponga" la opción transparencia (línea 60) y que "quite" la opción transparencia (línea 110), y observa cómo esto afecta a la imagen en pantalla. Una lista completa de todos estos comandos adicionales la damos en el Apéndice VI.

### 5.2 Modos de imagen en pantalla

Hay tres MODOS en que la imagen en pantalla (con textos y gráficos) aparece:

#### a) Normal

Modo 1: 40 columnas x 25 líneas, 4 TINTAS para textos  
320x200 motas, accesibles individualmente con 4 colores

#### b) Modo multicolores:

20 columnas x 25 líneas, 17 TINTAS para textos  
160x200 motas accesibles individualmente con 16 colores.

#### c) Modo alta resolución:

80 columnas x 25 líneas, 2 TINTAS para textos  
640x200 motas, accesibles individualmente con 2 colores.

Como puedes ver, la diferencia estriba en la cantidad de motas (elementos individuales pictóricos (pixels)) que caben horizontalmente en cada línea. No te confundas con las pequeñas barras en la parte delantera del tubo de un televisor, que es una peculiaridad del equipo monitor de televisión que no guarda ninguna relación.

Cada uno de los tres diferentes MODOS en que usamos la pantalla, es lo que llamamos modo de imagen o de pantalla, y solamente puede aparecer uno en cada momento usando el lenguaje BASIC. Al cambiar de MODO hace que la pantalla se LIMPIE completamente, incluyendo todas las VENTANAS de texto o gráficos que hayamos estipulado (el mismo efecto que con el comando **CLS** y **CLG**) pero no afecta en absoluto al programa contenido en la memoria.

Los cambios de modo pueden hacerse con instrucciones BASIC, o pueden darse por teclado directamente con el comando correspondiente.

### 5.2.1 MODE 0

Corresponde a imágenes multicolores

Pueden aparecer simultáneamente en pantalla 16 de los 27 colores disponibles, y cada elemento individual de la imagen puede ser programado en cuanto al color. La imagen está compuesta de 160 motas en cada línea horizontal, y 200 en cada columna vertical. Una "retícula" para preparar figuras en este modo, aparece en el Apéndice VI.

En el MODO 0, hay por lo tanto posibilidad de mostrar 20 TIPOS (caracteres) en cada una de las 25 líneas.

### 5.2.2 MODE 1

Es el modo prescrito como standard

El MODO 1 está prescrito en el CPC464 y es el adoptado al ponerlo en marcha. En este modo se pueden mostrar simultáneamente 4 de los 27 colores disponibles, aunque puedes conmutar rápidamente a través de todos los 27, si lo deseas. La imagen contiene 320 motas a lo ancho por 200 motas a lo alto. También en el Apéndice VI aparece una retícula adecuada para este modo.

En el MODO 1, hay por tanto 40 TIPOS (caracteres) en cada una de sus 25 líneas.

### 5.2.3 MODE 2

Es el modo de alta resolución

El MODO 2 permite que se usen simultáneamente 2 colores de tinta, y primordialmente se elige por su capacidad para producir 80 caracteres de texto en cada una de sus 25 líneas, lo que hace que un programa sea mucho más fácil de escribir y de examinar al poderse ver mucho más de cada programa con una sola pasada.

El MODO 2 permite 640 motas en el sentido horizontal, juntamente con 200 motas en cada columna vertical.

### 5.2.4 Ensayando

Con el CPC464 completamente restaurado, teclea el programa:

```
5 REM DEMOSTRACION EJEMPLOS DE GRAFICOS
10 MODE 1
15 INK 2,0
16 INK 3,6: REM
```

```

17 BORDER 1: REM AZUL OSCURO
20 CLG: REM CLAREADO DE PANTALLA
30 b%=RND*5+1 :REM PONIENDO VARIABLES ENTERAS
   PSEUDO ALEATORIAS
40 c%=RND*5+1
50 ORIGIN 320,200 :REM FIJA ORIGEN GRAFICO
60 FOR a = 0 TO 1000 STEP PI/30
70 x%=100*COS(a)
80 MOVÉ x%,x% :REM MUEVE EL CURSOR GRAFICO
90 DRAW 200*COS(a/b%),200*SIN(a/c%),3
   :REM DIBUJA LA LINEA
91 IF INKEY$()="" THEN 20
100 NEXT :REM RETROCEDE A 60 SINO SE INTERRUMPE EN 91
110 GOTO 20

```

Ahora EJECUTA el programa. pulsa "cualquier tecla" para conseguir otra imagen diferente. Así se demuestran varias de las facultades importantes de los **circuitos** y programas del CPC464. Con ellos el CPC464 "escribe" en la pantalla suavemente sin tirones ni temblores, y mediante comandos que consiguen efectos muy complicados con un mínimo de esfuerzo. Las instrucciones **REM**, simplemente sirven para reMEMOrar, pero no necesitas incluirlas para que el programa funcione; te ayudan a tí y particularmente a la gente que no escribió el programa pero lo está estudiando para comprender mejor el trabajo que hace.

Observando los números de línea, puedes detectar también algunas que son consecuencia de reflexiones y recensiones, y aunque podríamos presentar los números en intervalos constantes, RENUMerando, te ayuda a ver la manera en que hemos desarrollado los programas a partir de estructuras simples iniciales.

Guarda este programa en cassette, mediante, por ejemplo, el comando:

**SAVE "GRAFICO 5.5.84"**

Este otro programa de demostración, PINTA una imagen coloreada con distinto "cariz":

```

new
10 a$=INKEY$: REM PULSA UNA TECLA PARA
   INICIAR UNA NUEVA SECUENCIA
20 IF a$="" THEN 10
30 CLS
40 m=INT(RND*3):REM SELECCIONA UN NUMERO
   ALEATORIAMENTE ENTRE 0 Y 3
50 IF m>2 THEN 40:REM PRUEBA DE NUEVO
60 MODE m
70 i1=RND*26:REM TOMA VALORES PARA INK ALEATORIAMENTE
80 i2=RND*26
90 IF ABS(i1-i2)<5 THEN 70
100 INK 0,i1:INK 1,i2

```

```

110 s=RND*5+3
120 ORIGIN 320,-100
130 FOR x= -1000 TO 0 STEP s
140 MOVE 0,0
150 DRAW x,300: DRAW 0,600
160 MOVE 0,0
170 DRAW -x,300: DRAW 0,600
180 a$=INKEY$
190 IF a$(">") THEN 30: REM INTERRUMPE EL BUCLE
    PRESIONANDO UNA TECLA
200 NEXT X
210 GOTO 10

```

Tanto este como el programa anterior, ilustran simples conceptos matemáticos de una forma colorística y televisiva. Ambos básicamente hacen sumas de números generados al AZAR a partir de un determinado "gérmen" para garantizar que cada una de las imágenes tienen diferente cariz, y muestran los resultados como segmentos aleatoriamente situados.

Tu CPC464 es también un excelente "impreso" electrónico para dibujar funciones y curvas geométricas, y una de las que tienen un cariz más conocido es la que corresponde a una "onda sinusoidal".

```

10 REM DIBUJA SINUSOIDE
20 MODE 2
30 INK 1,21
40 INK 0,0
50 CLS
60 DEG
70 ORIGIN 0,200
80 FOR n=0 TO 720
90 y=SIN(n)
100 PLOT n*640/720,198*y,1
110 NEXT

```

La instrucción **PLOT**, hace que la máquina **PINTE** una sola **mota**, en la parte de la pantalla determinada por las coordenadas mencionadas en la instrucción. En este ejemplo, y mediante el bucle que empieza en la línea 80 y culmina en la 110, aparece la curva del seno, pintada mota a mota.

El CPC464 dispone de muchos comandos, simples y potentes, que añaden efectos especiales al programa anterior. Por ejemplo, con:

```
15 BORDER 6,9
```

Hacemos que el BORDE presente sucesivamente los colores correspondientes a los números 6 y 9. El ritmo o cadencia de palpitación, está prescrito a su valor habitual. Para hacer que el programa deje de dar "rondas" sin cesar, ESCapa (y por dos veces para interrumpirlo, porque con una sola, simplemente suspendes la operación), luego añade la instrucción:

```
120 GOTO 50
```

Observa que el borde no cesa de palpitir aunque el programa ya ha concluido, y es debido a que el borde que enmarca la imagen se controla independientemente del resto del programa. Para detener el parpadeo del borde y hacer que sea permanentemente azul brillante, cambia la línea 15 para que sea:

```
15 BORDER 2
```

Y observarás que al concluir la EJECUCION del programa, cesa la palpitación del borde. Para cambiar el color con que la PLUMA PINTA la línea y para cambiar la tinta del PAPEL que usa, cambia las instrucciones INK respectivas, en las líneas 30 y 40. Ahora, cuando le hagas que LISTE el programa, aparecerá:

```
10 REM DIBUJA SINUSOIDE
15 BORDER 2
20 MODE 2
30 INK 1,2
40 INK 0,20
50 CLS
60 DEG
70 ORIGIN 0,200
80 FOR n=0 TO 720
90 y=SIN(n)
100 PLOT n*640.720,198*y,1
110 NEXT
120 GOTO 50
```

El número 1 al final de la instrucción **PLOT** de la línea 100, le dice al ordenador que PINTE la curva con el color correspondiente a la TINTA 1 (que es la de la PLUMA), elegida en la instrucción número 30. Comprueba la definición del comando **PLOT** en la lista de palabras clave del capítulo 8, y verás exactamente cómo afectan al comando los diversos PARAMETROS que puedes reseñar.

Si miras concienzudamente la curva que se está pintando en la pantalla, verás que no es una línea perfectamente continua, sino que esta formada por segmentos diminutos. El más pequeño de esos segmentos, es lo que hemos llamado MOTA anteriormente (y que como es un elemento "pictórico", también se le llama "pixel").



### 5.2.5 El invisible cursor para gráficos

Ya has ensayado algunas de las maneras en que puedes usar en los programas la capacidad gráfica del CPC464, y algunos de los conceptos y comandos han tenido la oportunidad de mostrar su actuación. Cuando quieres sacar rectas y curvas en la pantalla, hay que vigilar algunas consideraciones importantes para evitar confusiones.

El primer punto a vigilar es el estado presente de la memoria, en el recinto reservado al programa. El ordenador recuerda cómo tienes en cada momento estipulados los colores, incluso aunque le hayas dado el comando **NEW** para DESNUDAR la memoria de programa. Para restaurar todas las condiciones iniciales, prescritas en fábrica para que actúen al poner en marcha el ordenador, debes pulsar sucesiva y simultáneamente **CTRL**, **SHIFT** y **ESC** para volver a esas condiciones, sin necesidad de apagar y encender de nuevo, pero por supuesto, debes hacer que **GUARDE** todo en el cassette, antes de restaurar.

Vamos a comprobarlo, tecleando simplemente:

**NEW:CLS**

...después de hacer que el ordenador interrumpa la ejecución del programa. Teclea ahora:

**DRAW 100,100**

El comando **DRAW** pide al ordenador que **DIBUJE** una línea recta a partir del punto en que se encuentre en ese momento el cursor de gráficos (invisible) hasta el punto cuyas coordenadas x e y, son las mencionadas en el comando (100,100). Aunque es invisible, el cursor de gráficos es un "ente" que la máquina maneja internamente para saber las coordenadas del punto recién aceptado por el último comando que hayas hecho en relación con gráficos.

Pero sí puedes saber las coordenadas donde está **UBICADO** en cada momento el cursor de gráficos, mediante las funciones **XPOS** e **YPOS**. Después del comando **DRAW** anteriormente dado al ordenador, es lógico que si tecleas:

**PRINT XPOS**

Te dé como respuesta/

100

(y lo mismo ocurriría con **YPOS** en este momento).

Observa que aunque debido a la longitud de texto, llegues hasta la parte inferior de la pantalla, con lo que provocas que se DESPLIEGUE la imagen subiendo una línea, y que a pesar de que también subirá la imagen gráfica que puedas tener, las coordenadas correspondientes a la ubicación del cursor de gráficos, continúan con los valores que tenían. Puedes fácilmente ensayar con esto, manteniendo la tecla de BAJADA DE CURSOR (↓) hasta que la pantalla quede limpia, y preguntar en esa situación la UBICACION del cursor de gráficos. Continuará con la misma coordenada x, y con la misma coordenada y.

Para especificar el color con que queremos que DIBUJE una línea, añade un **parámetro** más al final del comando **DRAW**. (Puedes ver la descripción que hemos hecho del comando **PLOT** en el programa anterior, porque funciona de la misma manera).

Debes primeramente, especificar la TINTA, y recuerda que sólo puedes usar los números de colores admitidos en el modo de pantalla en que estés. Para ensayar con esto, teclea:

```
10 MODE 1
20 INK 0,10
30 ORIGIN 0,0
40 INK 1,26
50 INK 2,0
60 DRAW 320,400,1
70 DRAW 640,0,2
```

Aquí hay un ejemplo que usa todos los conceptos y comandos mencionados hasta ahora, al mismo tiempo que presenta una pareja nueva de conceptos fructíferos. Observa cómo en la primera instrucción estipulamos las condiciones de color de la TINTA usada con la PLUMA y con el PAPEL, para asegurar que se anula el efecto de las condiciones que hubiera en la memoria del CPC464, y producir los resultados que queremos:

```
10 INK 0,0:INK 1,26:INK 2,6:INK 3,18: BORDER 0
20 REM dibuja figuras regulares
30 mode 1:DEG
40 PRINT "3,4 o 6 lados ? ";
50 LINE INPUT p$
60 IF p$="3" THEN sa=120:GOTO 100
70 IF p$="4" THEN sa=135:GOTO 100
80 IF p$="6" THEN sa=150:GOTO 100
90 GOTO 50
100 PRINT:PRINT "Calculando";
105 IF p$="3" THEN ORIGIN 0,-50,0,640,0,400
    ELSE ORIGIN 0,0,0,640,0,400
110 DIM cx(5),cy(5),r(5),lc(5)
120 DIM np(5)
130 DIM px(5,81),py(5,81)
140 st=1
```

```

150 cx(1)=320:cy(1)=200:r(1)=80
160 FOR st=1 TO 4
170 r(st+1)=r(st)/2
180 NEXT st
190 FOR st=1 TO 5
200 lc(st)=0:np(st)=0
210 np(st)=np(st)+1
220 px%(st,np(st))=r(st)*SIN(lc(st))
230 py%(st,np(st))=r(st)*COS(lc(st))
240 lc(st)=lc(st)+360/r(st)
245 IF (lc(st) MOD 60)=0 THEN PRINT ". ";
250 IF lc(st) < 360 THEN 210
252 px%(st,np(st)+1)=px%(st,1)
254 py%(st,np(st)+1)=py%(st,1)
260 NEXT st
265 CLS:ink 1,2
270 st=1
280 GOSUB 340
290 LOCATE 1,1
300 EVERY 25,1 GOSUB 510
310 EVERY 15,2 GOSUB 550
320 EVERY 5,3 GOSUB 590
330 GOTO 330
340 REM dibuja un circulo con mas de 3,4 o
    6 lados
350 cx%=cx(st):cy%=cy(st):lc(st)=0
360 FOR x%=1 TO np(st)
370 MOVE cx%,cy%
380 DRAW cx%+px%(st,x%),cy%+py%(st,x%),
    1+(st MOD 3)
390 DRAW cx%+px%(st,x%+1),cy%+py%(st,
    x%+1),1+(st MOD 3)
400 NEXT x%
410 IF st=5 THEN RETURN
420 lc(st)=0
430 cx(st+1)=cx(st)+1.5*r(st)*SIN(sa+lc(st))
440 cy(st+1)=cy(st)+1.5*r(st)*COS(sa+lc(st))
450 st=st+1
460 GOSUB 340
470 st=st-1
480 lc(st)=lc(st)+2*sa
490 IF (lc(st) MOD 360) <> 0 THEN 430
500 RETURN

```

```

510 ik(1)=1+RND*25
520 IF ik(1)=ik(2) OR ik(1)=ik(3) THEN 510
530 INK 1,ik(1)
540 RETURN
550 ik(2)=1+RND*25
560 IF ik(2)=ik(1) OR ik(2)=ik(3) THEN 550
570 INK 2,ik(2)
580 RETURN
590 ik(3)=1+RND*25
600 IF ik(3)=ik(1) OR ik(3)=ik(2) THEN 590
610 INK 3,ik(3)
620 RETURN

```

Cuando ejecutes este programa, te hará una pregunta (línea 40), y contesta tres en esta ocasión para conseguir más rápidamente resultados. El programa te anunciará después: **Calculando**, y te irá exponiendo un punto cada pocos segundos (línea 245) para indicar que todavía "se lo está pensando" y confirmar que el programa sigue ejecutándose.

Las SUBROUTINAS (párrafos del programa que empiezan en una determinada línea y terminan con el comando de VUELVA (**RETURN**)), que son CITADAS en las líneas 300, 310 y 320, obligan a que la imagen parpadee con diferentes tintas, y a las "cadencias" determinadas por el comando **EVERY**, que fija CADA cuánto tiempo debe hacerlo. Por lo tanto, si quieres ralentizar el parpadeo, cambia las línea 300 a 320 para que digan:

```

300 EVERY 250,1 GOSUB 510
310 EVERY 150,2 GOSUB 550
320 EVERY 50,3 GOSUB 590

```

Para ver lo que has conseguido, estudia el comando **EVERY** en el capítulo 8, ya que es una de las facultades más aprovechables del BASIC AMSTRAD. Uno de los efectos interesantes de este comando **EVERY**, n segundos según su TEMPORIZADOR m, vaya a la subrutina que comienza en la línea mencionada, es la manera en que va apilando estas recuestas si se interrumpe el programa pulsando la tecla **ESC**, sólo "una vez".

Al hacer esto, suspende la operación del programa durante unos pocos segundos y luego reanúdala pulsando "cualquier tecla". La imagen palpitara frenéticamente, ya que las temporizaciones solicitadas han "sido puestas en la cola" y se apresuran a tomar la vez. En cada una de las "colas" que el sistema mantiene, sólo hay un espacio limitado, así que después de un rato, un nuevo comando **EVERY** se ve desechado hasta que haya espacio disponible porque ya se ha vaciado parte o enteramente alguna de las colas.

### 5.3 Ventanas o viñetas

El usuario puede seleccionar hasta 8 ventanas o viñetas de texto en los que escribir caracteres, y también una ventana o viñeta de gráficos en que pueda pintar. Las ventanas estipuladas se anulan cuando se cambia el modo de imagen. Estudia las descripciones correspondientes del capítulo 8.

**NB:** Si la ventana de texto es equivalente a toda la pantalla (el valor prescrito para omisiones), el DESPLIEGUE rápido de las imágenes se consigue por los circuitos electrónicos. Si la ventana de textos es menor que el área disponible en pantalla, el CPC464 consigue el "despliegue" de las imágenes por programas internos, que por lo tanto es mucho más lento.

El comando **WINDOW**, especifica los márgenes izquierdo/derecho/superior/inferior que corresponden al CAUCE de comunicación con esa ventana, y por tanto a la posición que ocupa la ventana dentro de la pantalla. Esas ventanas o viñetas pueden solaparse unas con otras, permitiendo así un método rápido para rellenar las viñetas. Antes de comenzar a explorar esta nueva facultad, teclea:

```
KEY 139, "mode 2:paper 0:ink 1,0:ink 0,9:
      list"+chr$(13)
```

Con esto haces que al pulsar la pequeña de las teclas **ENTER**, se limpie y restaure el sistema y se estipulen colores visibles de manera que no te pierdas en una combinación con la misma **TINTA** para el **PAPEL** y la **PLUMA**. El ejemplo fabrica una serie de ventanas a lo ancho de la pantalla, e ilustra dos puntos importantes:

```
5 MODE 0
10 FOR n=0 TO 7
20 WINDOW # n,n+1,n+6,n+1,n+6
30 PAPER # n,n+4
40 CLS # n
50 FOR c=1 TO 200:NEXT
60 NEXT
```

El primero de esos puntos es que cada nueva imagen sobre-escribe la anterior, y el segundo que queremos destacar es que los mensajes siguen enviándose por el CAUCE número 0 en todo momento (a no ser que explícitamente digas otra cosa). Antes de hacer cualquier otra cosa, LISTA el programa, y verás cómo sigue siendo enviado a través del CAUCE 0.

Ensayá ahora:

```
LIST # 5
```

Y luego:

**CLS # 6**

para comprender lo que hemos dicho, a saber que el último CAUCE hacia la pantalla que se emplee sobre-escribirá todo lo demás, mientras que el mensaje típico del sistema **Ready** continúa siendo enviado a través del CAUCE 0; incluso aunque hayas hecho que el listado se envíe a través del CAUCE número 5.

Además es posible usar un comando para obligar al ordenador a que CANJEE ventanas, mediante el comando **SWAP**. Añade, por ejemplo la línea 55:

**55 IF n=3 THEN WINDOW SWAP 7,0**

Y puede que supongas que enviará el mensaje **Ready** por el cauce 7. Pero ejecútalo y comprueba. Al desarrollar este programa tan ¡simple!, apreciarás la manera en que el CPC464 utiliza sus potentes facultades en cuestión de manipular viñetas dentro de la pantalla.

## 6. Lo primordial de Sonidos

Los efectos sonoros son reproducidos por un altavoz incorporado en el propio ordenador. Si estás usando el modulador/alimentador MP1 y un televisor, gira por tanto el control de volumen de tu TV a un mínimo.

El **VOLUMEN** del sonido puede ajustarse usando el botón **VOLUME** en el costado derecho del ordenador. El sonido puede también ser enviado al enchufe de entrada auxiliar de tu cadena estéreo, a través del enchufe de entrada/salida (I/O) en la parte izquierda del panel trasero del ordenador. Eso te permitirá escuchar el sonido generado por tu ordenador en estéreo, a través de altavoces o auriculares de alta fidelidad.

El **CPC464** **SUENA** tan bien como aparenta. Para conseguir aprovechar las facultades sonoras en forma inimaginable, necesitas comprender previamente la filosofía que gobierna las estructuras de tiempos y tonos.



Temas tratados en este capítulo:

- \* Períodos (duración) del tono
- \* Haciendo que SUENE
- \* Envolventes
- \* "CHORROS" de notas y sincronización

Si todo lo que quieres hacer es simplemente que el ordenador emita un pitido, teclea:

```
PRINT CHR$(7)
```

Y no vayas más allá, con lo que ya habrás conseguido todo lo que pueden dar otros ordenadores personales. Pero si lo haces, te perderás algunas de las facultades más interesantes y motivadoras del **CPC464**, así que esta sección es un fundamento que te dará una visión general del tema, seguida de un análisis pormenorizado de las palabras reservadas y de cómo dar los comandos. Proporcionará al programador los fundamentos para saber cómo construir un "chorro" de notas, inventar diversas clases e instrumentos musicales, y estructurar sus "tonadas" usándolos.

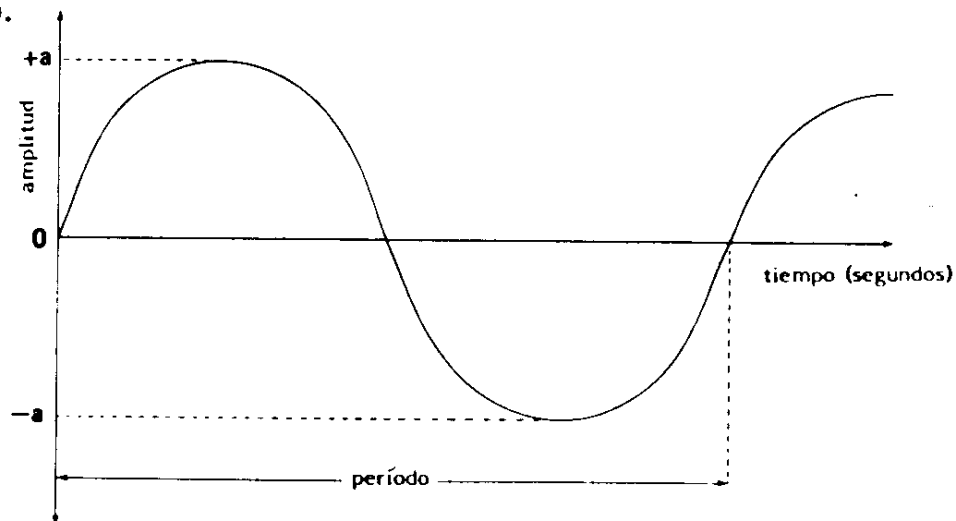
## 6.1 Fundamentos del comando SOUND

Cuando escuchas el sonido de una NOTA MUSICAL standard, generada por un diapasón, hay varios atributos que debes considerar:

- 1) El tono y las variaciones del tono en el tiempo que dura la nota.
- 2) Volumen y variaciones en el volumen en el intervalo de tiempo que dura la nota.
- 3) Duración de la nota.

## 6.2 EL TONO

En una nota musical pura, el tono es con mucho, el atributo más importante. Una nota musical pura puede describirse como una oscilación regular, y como todas las oscilaciones, tiene FRECUENCIA, PERIODO y AMPLITUD.



**Figura 1:** Las características de las notas musicales puras.

La frecuencia es el número de oscilaciones que da en cada segundo, mientras que el período es la duración de cada una de esas oscilaciones. La amplitud es el nivel máximo alcanzado y está relacionado con el volumen, y no concierne en absoluto con la definición del TONO de una nota.

Fácilmente verás que frecuencia y período estás relacionadas unas a otras simplemente por:

$$\text{Frecuencia (expresada en ciclos por segundo)} = 1/\text{Período (expresado en segundos)}$$

La relación entre la frecuencia y el <período del tono> emitido por el altavoz del ordenador con el comando SOUND, viene dado por:

$$\text{Frecuencia (expresada en ciclos por segundo)} = 125/\text{<período del tono>}$$



Por lo tanto, un período del tono de 1000 daría como resultado un tono con frecuencia de 125 cps, y un período del tono de 125 daría como resultado un tono (bastante común en telefonía) de 1000 cps.

Cualquiera de estas magnitudes pudiera usarse para estipular el TONO, pero en el BASIC AMSTRAD hemos elegido la que hemos llamado <período> del tono. No te confunda el hecho que a medida que crece de valor el período, el tono se hace más grave. La gama de períodos disponible es amplia (0 a 4095) y debe designarse por un numeral entero. Y a pesar de ello, puede tener errores de redondeo en el tono emitido, pero nadie lo notará excepto quizás, un músico o un crítico musical. Consulta también el Apéndice VII.

Cuando en un instrumento musical tradicional se toca una determinada nota, el TONO, puede variar, y algunas veces deliberadamente por "vibrato". Usando el comando que permite definir la ENVOLVENTE DE TONO, abreviado ENT, podemos estipular la forma específica en que debe variar en PASOS discretos el tono, a lo largo de toda la duración de la nota; y esa envolvente de tono puede ser mencionada en el comando SOUND. Igualmente sucede con la ENVOLVENTE DE VOLUMEN, abreviado ENV.

### 6.3 Volumen de las notas

El volumen es simplemente la medida de lo fuerte que suena. Para estipular el nivel inicial de volumen por cada nota generada en el CPC464, con el comando SOUND, disponemos de una gama de valores de 0 a 15.

Este valor para el volumen de la nota emitida, también puede mencionarse en el comando SOUND, y haremos referencia a él, diciendo simplemente <volumen>, y recuerda que puede ser una variable numeral entera.

Si ya has ensayado con algunas notas simples, habrás notado la poca gracia que tienen las notas puras, ya que el ordenador todavía no ha "elaborado" los otros atributos de la nota. La diferencia con los instrumentos musicales condicionales, radica que en ellos cuando se toca una nota el volumen va comenzando a crecer durante la primera parte de la nota -denominado el "ataque" se mantiene más o menos constante durante un rato, y luego vuelve a decaer cuando está terminando la nota emitida -denominada la "caída",

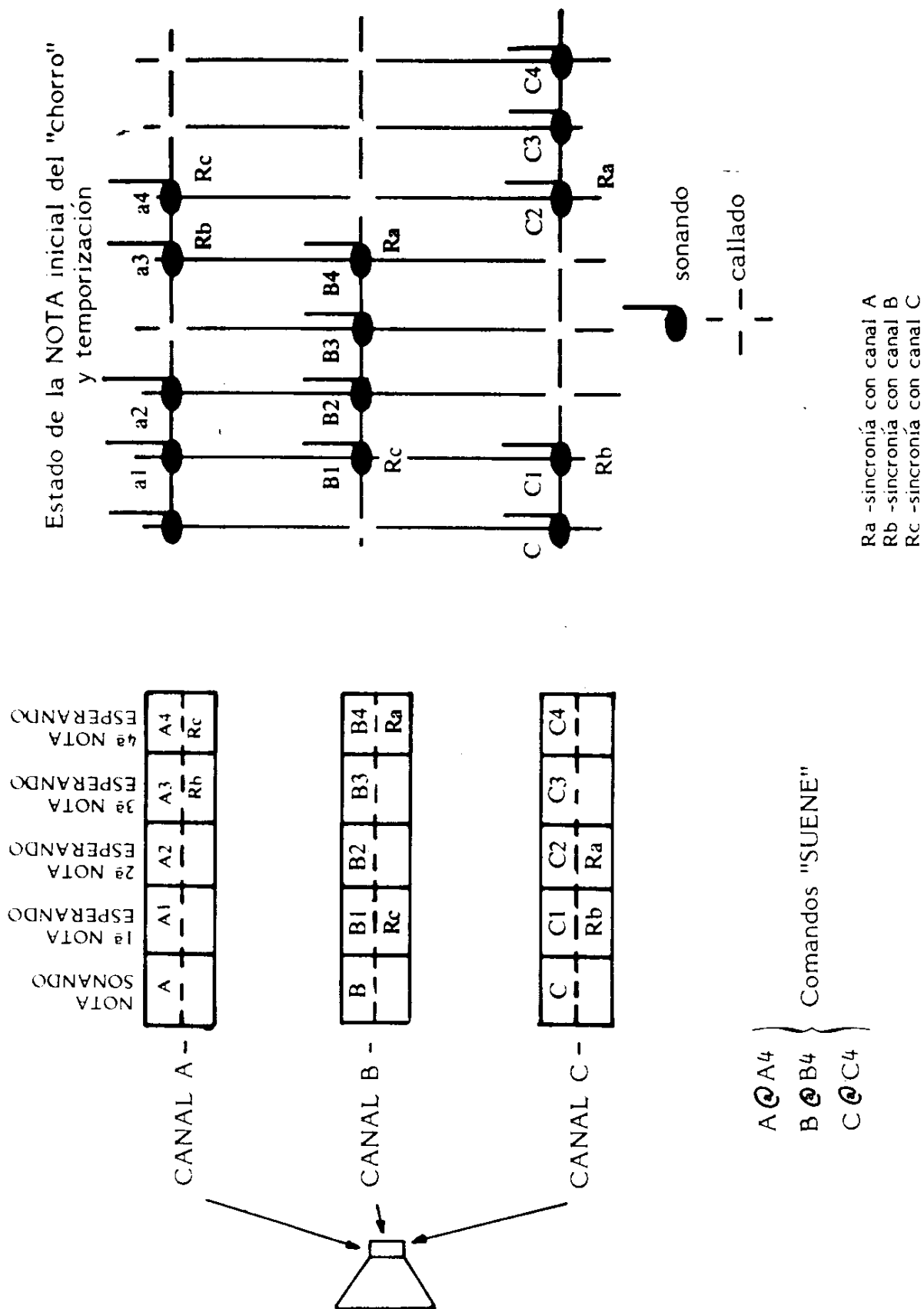
Además, cada instrumento musical tiene una forma diferente de "atacar" las notas, mantenerlas, y de "caída" que le es específico, y que en el ordenador CPC464 podemos simular si ajustamos el volumen para que varíe en pasos o secciones a lo largo de la duración de la nota. Y es lo que denominamos <envolvente de volumen>, y que podemos alterarla con el comando que usa la palabra clave ENV.

#### 6.4 Duración de las notas

La longitud en tiempo de las notas es una de las facetas básicas de cualquier composición musical; la unidad musical para la duración o longitud en tiempo de las notas es la CORCHEA, y hay mínimos, semicorcheas, etc. que simplemente son múltiplos o divisores de esta unidad básica. Pero como además las TONADAS pueden tocarse con diferente ritmo o velocidad, es necesario que esta unidad básica de duración o período de referencia para la corchea "standard", tenga que determinarse.

Para que no tengas que molestarte, hay un parámetro del comando **SOUND** conocido como **<duración>**, que ha de ser una variable numérica entera, y en que una unidad representa una centésima de segundo (1/100 segundo). Observa que también queda conformada por el parámetro **<envolvente de volumen>**, y de esta forma evitamos las posibles confusiones.

ACORDES en los canales sonoros



## 6.5 Otros SONIDOS

El ruido aleatorio ("ruido blanco") también puede ser generado por el ordenador mediante el comando **SOUND**, pero no es una nota musical. Puede añadirse como fondo de las melodías para crear variaciones interesantes, o simplemente usarlo por sí solo para efectos especiales. Y observa que la explosión es básicamente un ruido blanco con un parámetro de <envolvente de volumen> específico. Este ruido consta de tonos cuya frecuencia varía aleatoriamente y que duran lo estipulado en el parámetro <período de ruido> del comando **SOUND**. Una de las importantes facetas de este ordenador es que aunque hay tres **CANALES** disponibles separadamente y con la posibilidad de estipular tres <períodos de tono>, sólo un <período de ruido> puede aparecer en la combinación de canales que se elija.

## 6.6 Sonidos múltiples y canales

La mayoría de las piezas musicales se escriben usando dos claves como mínimo: clave en FA y clave el SOL. Para simular este aspecto en el CPC464, se dispone de tres canales de sonido denominados A, B, C. En todos ellos se puede emitir sonidos independientemente, o puede hacerse que estén sincronizados. La selección del canal por donde se envía la nota, se efectúa mediante el parámetro <estado de canal> del comando **SOUND**.

## 6.7 "Chorros" de notas en los canales

Cada canal de sonido recibe las notas que debe emitir, una a una, formando un "chorro". En el ordenador se dispone de espacio para formar estos "chorros" sólo para cinco comandos **SOUND** sucesivos; y uno de ellos, el que ocupe la posición inicial, decimos que es el activo, y las otras cuatro posibles notas que están esperando o "a la cola". El sistema operativo del CPC464 puede continuar realizando otras tareas mientras se están emitiendo sucesivamente las notas que forman el chorro de cada canal, y sólo vuelve cuando necesita conseguir más notas sonoras a emitir.

## 6.8 Estado del canal

La función **SQ** aplicada a un determinado canal, nos entrega un valor correspondiente al estado de ese canal, y mediante ese valor podemos saber cómo es el chorro de notas que hay en ese canal; es decir, lugares sin notas, notas **ACORDES**, retenciones, etc. Puede usarse el comando **ON SQ GOSUB** como una interrupción al sistema para que preste atención a la sección del programa que genera los sonidos, cuando el estado del canal es uno determinado.

## 6.9 Retenciones y ACORDES

Para forzar que las notas sean emitidas al unísono, existe la posibilidad de marcarlas como ACORDES. En ese caso, la marca se coloca sobre aquellas notas de los diversos chorros que deben sonar simultáneamente, y se hace mediante el parámetro correspondiente del comando **SOUND**. Es muy útil para iniciar la temporización de los chorros de notas, cuando la melodía contiene pausas o intervalos de reposo, que no ocurren simultáneamente en los canales.

También es posible disponer la facultad de **RETENER** una determinada nota del chorro de manera que se logre la sincronización global de los canales (i.e. al arranque de una tonada), y puede ser con un cierto retardo, y para todo ello es necesario haber marcado previamente la nota de ese chorro de forma adecuada. También se dispone del comando **RELEASE** que permite se libere un chorro de notas que estuviera retenido.

## 6.10 Comandos de generación de tonos

Forma del comando: **SOUND G,H,I,J,K,L,M**

Siendo:

- G:** Estado del canal
- H:** Período del tono
- I:** Duración de la nota
- J:** Volumen de arranque
- K:** Envolvente de volumen
- L:** Envolvente de tono
- M:** Período del ruido

El comando **SOUND** lo que hace es emitir un sonido, conformado por los parámetros G a M, hacia los circuitos generadores de sonido del ordenador. Los valores de los parámetros G a M han de ser enteros y dentro de la gama permitida, y únicamente es obligatorio siempre especificar los dos primeros. Los restantes son opcionales, pero tienen valores prescritos para cuando son omitidos, que comentaremos inmediatamente.

### 6.10.1 Descripciones de los parámetros: G: Estado del canal

Gama de valores 1 a 255.

Prescrito para misiones: ninguno (obligatoriamente debe darse un valor).

Con el CPC464 es posible emitir tres sonidos diferentes al unísono. Eso lo logra porque tiene tres **CANALES** de sonido, llamados A, B, C, donde el órgano central del ordenador va construyendo un **CHORRO DE NOTAS**, que los circuitos generadores de sonido procesarán y enviarán al altavoz.

El estado del canal viene dado por un numeral entero con notación decimal, que debe ser convertido al equivalente en binario, para poder entender el significado, ya que es el estado de cada uno de los 8 bits el que especifica la acción correspondiente al sonido enviado al canal o canales pertinentes.

DECIMAL	BIT Nº	ACCION
1	0 BAN	envía la NOTA al canal A
2	1	envía la NOTA al canal B
4	2	envía la NOTA al canal C
8	3	marca ACORDE con canal A
16	4	marca ACORDE con canal B
32	5	marca ACORDE con canal C
64	6	retención de canal
128	7 BUN	oclusión de canal

(BAM corresponde al Bit Anterior del Número, que es el bit más significativo (en inglés LSB). BUN es el Bit Ultimo del Número que es el menos significativo (en inglés MSB).

Algunos ejemplos con valores en el sistema decimal son:

2 = envía el SONIDO al canal B

5 = 4+1 envía el SONIDO a los canales A y C

98 = 64+32+2 envía el SONIDO al canal B, marcándolo como ACORDE con el canal C y establece retención del canal B.

Es importante recordar que si se estipula que haya un ACORDE entre dos o más canales, es necesario marcar el parámetro de <estado del canal> en cada uno de ellos en el momento oportuno.

La RETENCION detiene la elaboración de ese sonido, y libera el chorro de notas que haya detrás del que está actuando, y no puede ser usado el canal hasta que sea LIBERADO con el comando **RELEASE**, o sea DERRIBADO dicho canal mediante un comando **SOUND** con el parámetro pertinente.

Cuando se especifica OCLUSION de un canal o canales, fijando el valor adecuado en el parámetro <estado> de un comando **SOUND**, se ejecuta inmediatamente la cancelación de todo el chorro de notas que pueda haber en espera, y cualquier nota que esté sonando en ese momento queda inmediatamente cortada. El canal queda desactivado.

#### H: Período del tono

Gama de valores: 0 a 4095.

Prescrito para omisiones: ninguno (valor obligatorio).

Estipula que la frecuencia del sonido a generar (tono de la nota), sea la que resulte de evaluar la fórmula:

$$\text{frecuencia} = 125000 / \text{período del tono}$$

Si se usa el valor 0, no se generará ningún tono, lo que puedes aprovechar cuando sólo requieras RUIDO.

### **I: Duración**

Gama de valores: -32768 a +32767

Valor prescrito para omisiones: 20

Para valores positivos, este parámetro establece la duración en centésimas de segundo. Cuando es igual a 0, la duración viene controlada por la longitud de la envolvente de volumen que se especifique.

Cuando el valor de la duración es negativo, su valor absoluto representa el número de veces que se repetirá la envolvente de volumen especificada.

### **J: Volumen de la nota**

Gama de valores: 0 a 15 (0 a 7 si no se especifica envolvente de volumen).

Valor prescrito para omisiones: 12 (4 cuando no se especifica envolvente de volumen).

Es el volumen de arranque o "ataque" de la nota, y se ve alterado cuando se especifica una envolvente de volumen. El cero corresponde a sin volumen inicial.

### **K: Envolvente de volumen**

Gama de valores: 0 a 15

Prescrito para omisiones: 0

El valor de este parámetro especifica una envolvente definida previamente mediante el comando ENV. Cuando se usa el 0 como valor de este parámetro, el volumen no puede cambiarse con ningún comando ENV, y queda constante durante 2 segundos y al nivel de volumen que tenga el ordenador.

### **L: Envolvente de tono**

Gama de valores: 0 a 15

Prescrito para omisiones: 0

Este parámetro especifica una envolvente de tono definida previamente, con el comando ENT. Si se usa el valor 0 para el parámetro, no puede cambiarse el tono por ningún comando ENT y queda constante en todo el <período de tono>.

**M: Período de ruido**

Gama de valores: 0 a 15

Prescrito para omisiones: 0

Especifica el ruido a añadir a la nota generada con el comando. Si se usa el valor 0, no se añade ningún ruido. Recuerda que solamente puede estipularse un período de ruido cada vez. Lo que significa que todos los canales que tengan estipulado ruido, recibirán el mismo ruido.

**6.11 La envolvente de volumen usando el comando ENV**

El inicio del sonido (ataque) y la culminación (caída) se requieren para hacer que las notas tengan vida propia. El comando de **ENV**olvente de volumen permite establecer el perfil del volumen a lo largo del tiempo de duración de la nota. Es conveniente antes de describir dicho perfil al ordenador, usar un papel a escala para esbozar tus requisitos, como en el ejemplo:

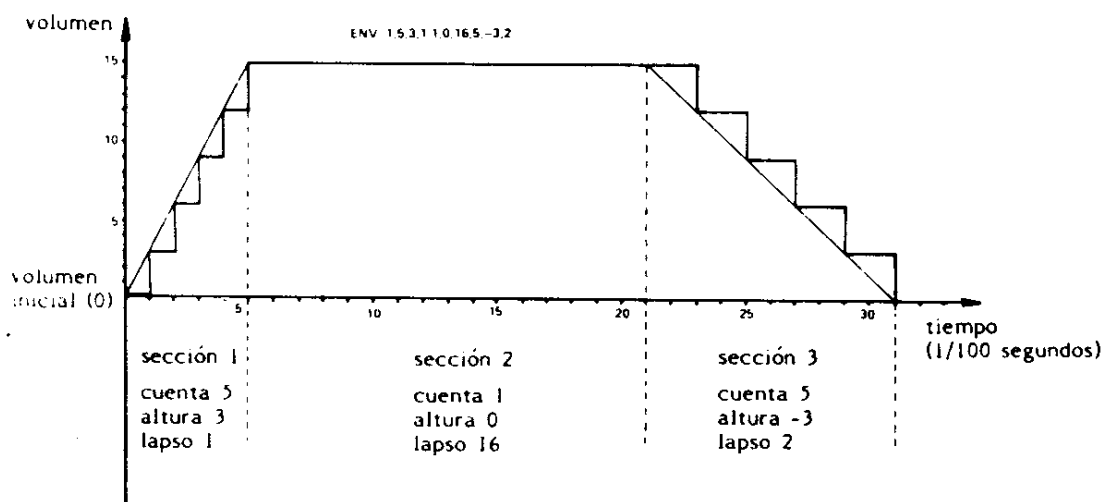


Figura 3: Ejemplo de envolvente de volumen.

La curva debe ser traducida a números de modo que puedan reseñarse en el comando. Para hacerlo, subdivídela en secciones, en cinco secciones como máximo. En cada sección debes aproximar la curva primero como una línea recta y luego como una línea quebrada o escalonada, con el número de escalones o pasos que quieras, y que llamaremos **<cuenta de pasos>**; y a estos pasos les corresponderá una cierta longitud en tiempo que llamaremos **<lapso>** y que mediremos en centésimas de segundo.

Al ir efectuando cada paso para dar un incremento o una disminución de volumen, que denominamos **<altura del paso>**. Si no se especifican pasos en una sección, corresponderá al volumen establecido en la sección inmediatamente anterior.



**NB:** Al intentar simular la envolvente específica de un instrumento musical, sucede a menudo que el volumen inicial y final de cada nota será 0, así que el volumen de arranque que estipules en el comando **SOUND** habrá de ser 0, y todo el volumen cambiado según la envolvente.

**Forma del comando ENV:**

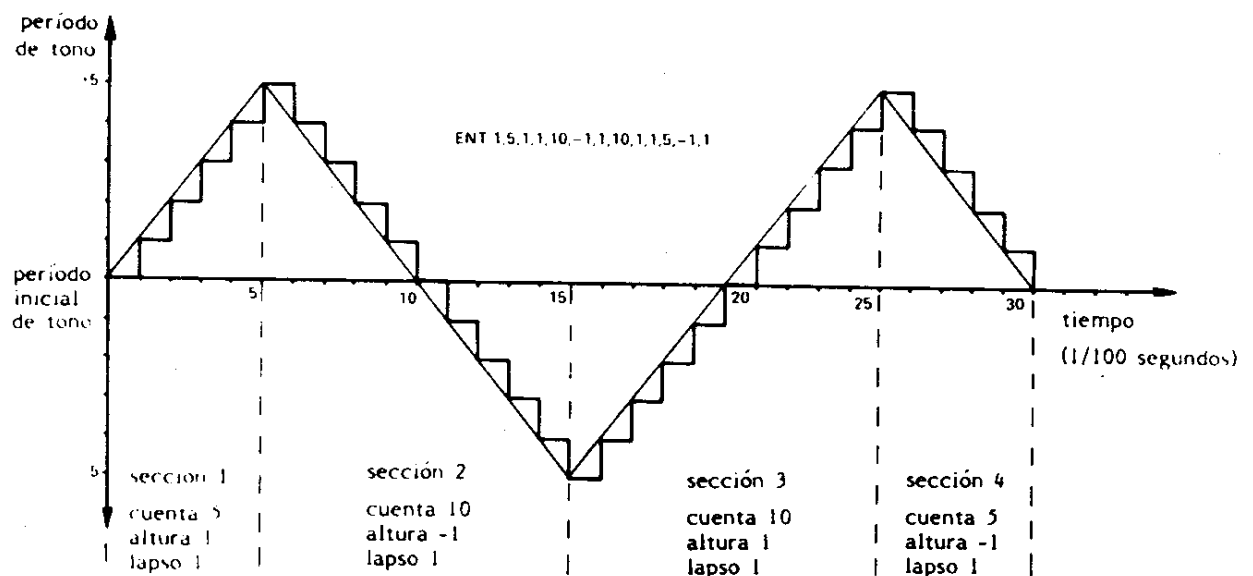
**ENV N,P1,Q1,R1,P2,Q2,R2,P3,Q3,R3,P4,Q4,R4,P5,Q5,R5**

N: Número de la envolvente	gama 1 a 5.
P1 a 5: Cuenta de pasos	gama 0 a 127
Q1 a 5: Altura del paso	gama -128 a +127
R1 a 5: Tiempo del paso	gama 0 a 255

El número de envolvente es un parámetro obligatorio. Además es obligatoria una sección completa por lo menos. Por ejemplo, puedes omitir las secciones 4 y 5 y decidir sólo las tres primeras secciones de la envolvente).

## 6.12 La envolvente de tono y el comando ENT

La envolvente de tono tiene exactamente el mismo tipo de construcción que la envolvente de volumen, sólo que aquí lo que se varía no es el volumen sino el tono de sonido a generar; en otras palabras, lo que se llama "vibrato", consistente en variar alternativamente entre dos tonos cercanos. Después de decidir la forma en cuando a los valores de tono que quieres que tenga esa nota, esbózalos como antes en papel a escala y divídelo en secciones y pasos, como en el ejemplo:



**Figura 4:** Ejemplo de envolvente de tono.

La mayor diferencia entre las envolventes de volumen y las envolventes de tono, es que éstas últimas no tienen ningún efecto sobre la duración de la nota; esta duración queda definida en el comando **SOUND** o en el comando **ENV**. Por lo tanto, si la envolvente de tono dura menos que la nota, a continuación se genera el tono constante durante el tiempo que resta de duración de la nota. Sin embargo, si la duración de la envolvente de tono, sobrepasa en tiempo a la duración de la especificada para la nota, las últimas secciones y pasos de la envolvente de tono no se generan.

Para repetir continuamente una cierta envolvente de tono durante todo el período de tiempo especificado para la nota, usa un número negativo para identificar la envolvente. (Pero sólo en el parámetro del comando **ENT**, y no en el parámetro del comando **SOUND**).

#### Forma del comando ENT:

**ENT S,T1,V1,W1,T2,V2,W2,T3,V3,W3,T4,V4,W4,T5,V5,W5**

S: Número de envolvente	gama 1 a 15 (negativo para repetir)
T1 a 5: Cuenta de pasos	gama 0 a 239
V1 a 5: Altura del paso	gama -128 a +127
W1 a 5: Tiempo del paso	gama 0 a 255 (centésimas de segundo)

El número de envolvente es obligatorio, y además completar cada una de las secciones individuales que uses.

Cuando se define un <número de envolvente>, todos los valores previamente estipulados son derogados. Si se especifica una envolvente de cualquier clase, sin ninguna sección, se tomarán como parámetros los correspondientes a la envolvente con número cero.

### 6.13 Otras funciones y comandos asociadas

#### **SQ(x)**

Siendo x el número de canal, que es 1 para el canal A, 2 para el canal B, y cuatro para el canal C. (Como corresponde a la notación interna binaria).

Como ya te mostramos en la tabla explicativa de los bits del parámetro estado de canal, sólo te diremos que esta función entrega como resultado un número entero en la gama 0 a 255, que tendrás que analizar según su equivalente binario para poderlo interpretar. Lo único que varía con respecto al parámetro del comando **SOUND** son los bits 0 a 2, y el bit 7, pero repetimos todos para mayor claridad:

Bit 0, 1, 2 número de espacios libres en el "chorro" examinado (de 0 a 4 en decimal).

Bit 3	la nota inicial del chorro está marcada con ACORDE con canal A
Bit 4	la nota inicial del chorro está marcada con ACORDE con canal B
Bit 5	la nota inicial del chorro está marcada con ACORDE con canal C
Bit 6	la nota inicial del chorro está marcado con RETENCION
Bit 7	el canal está actuando en ese momento.

Los Bits 3 a 6 (ACORDES y RETENCION) los comentamos al tratar el <estado de canal>. Además de la facilidad para examinar el estado de un canal, puede emplearse para INHIBIR la la señal de interrupción que provoca la ejecución del comando **ON SQ GOSUB**, descrito en el siguiente párrafo.

## ON SQ GOSUB

### ON SQ(y) GOSUB numero de linea

Siendo y el número de canal, que según los bits del equivalente en binario, será 1, 2 ó 4, para los canales A, B, C, respectivamente.

Esta instrucción solamente se ejecuta cuando ha habido una señal de INTERRUPCION, provocada por estar vacío el canal mencionado cuando no está INHIBIDO por algún comando **SQ** o **SOUND** anterior; y al ejecutarse obliga a un DESVIO hacia el número de línea mencionado en la instrucción. La subrutina a la que se desvía, debe terminar con la instrucción **RETURN**, para que VUELVA como hace habitualmente. Todos los canales tienen la misma prioridad en cuanto a la generación de interrupciones, y cuando hay un "vacío" en el canal mencionado en la instrucción, no sólo se provoca la ejecución de la subrutina, sino que la ha INHIBIDO para generar otras posibles interrupciones; por lo que dentro de la propia subrutina necesitarás quitar la inhibición e interrupciones, si es que las vas a necesitar de nuevo.

## RELEASE

### RELEASE z

Siendo z el número del canal que se ve LIBERADO. Observa que aquí el número de canal está en la gama de 1 a 7 para poder "liberar" varios canales con un solo comando.

Como se describió en el parámetro estado de canal del comando **SOUND**, es posible establecer en cualquier nota que se envía a un canal, una marca de RETENCION. El comando **RELEASE** simplemente quita esas marcas. El número del canal tiene su significado de acuerdo con la composición del número equivalente en binario, lo que nos permite "liberar" cualquier combinación de canales con un solo comando.

Bit 0: Canal A

Bit 1: Canal B

Bit 2: Canal C

## 7. Imprimiendo y jugando

El CPC464 no necesita ningún acoplador adicional para operar con uno o dos controladores de juego (joystick), y para usar una impresora compatible con la norma Centronics.

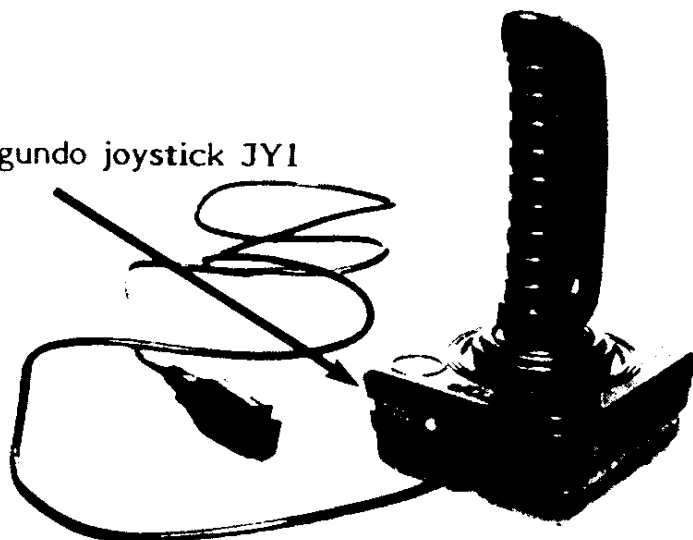
Temas tratados en este capítulo:

- \* "Joysticks"
- \* Impresoras en paralelo
- \* Acoplado

El modelo de mandos para juegos, llamados también palancas, AMSOFT JY1 es un equipo adicional que seguro desearás adquirir si usas el ordenador CPC464 con programas que incorporen esta posibilidad interesante para movimientos y disparos dentro de un juego. El JY1 puede enchufarse en la parte trasera de tu ordenador, usando el enchufe de 9 vías marcado **USER PORTS (I/O)**. El ordenador Amstrad CPC464 puede usarse con dos "joysticks" simultáneamente, enchufando el segundo de ellos en el correspondiente conector del primero.

No hay que hacer ninguna observación especial sobre cómo conectarlos al CPC464. Simplemente se inserta directamente la clavija del extremo del cable del joystick en el enchufe de 9 pines, previsto en la parte exterior del ordenador y marcado **USER PORTS (I/O)**, que es el "portal" para entradas y salidas hasta/desde el ordenador. También puede acoplarse un segundo joystick si es necesario para el juego, insertando su clavija en el enchufe previsto en la base del primer joystick JY1. Las conexiones de estas clavijas y enchufes se enumeran en el Apéndice V al final de este manual, junto con todas las otras funciones y conectores del CANALIZADOR de entrada/salida.

Conector para un segundo joystick JY1



Joystick JY1

## 7.1 Jugando con joysticks

Los programas incorporados en el ordenador CPC464 AMSTRAD, admite uno y dos joysticks, que son tratados como si fueran parte del teclado, y por tanto, puede examinarse el dato que están enviando como si fueran teclas; es decir, podemos usar el comando **INKEY** y la función **INKEY\$**, que guardan estrecha relación con las **INDAGACIONES** de valores tecleados. Nota que si existe un botón para disparar en tu joystick, es probable que sea el que dentro de la terminología AMSTRAD para el CPC464, corresponde al **GATILLO 2** (en inglés, "Fire 2", pero no hacen "fuego").

Se dispone de una función para examinar directamente el estado de cada uno de los **JOYSTICKS**, y es **JOY(0)** para el primero, y **JOY(1)** para el segundo. El valor resultante de la función tiene significación al emplear la notación binaria, con la que indica el estado exacto de sus controles en el momento de la última exploración del teclado. Como se efectúan 50 "barridos" del teclado por segundo, el valor obtenido es virtualmente el estado en ese mismo instante de la posición del joystick (que en inglés significa "bastón de jugar").

Los valores **CLAVE** correspondientes a los joysticks, se muestran en la tabla siguiente, siendo **CLAVE** el valor que hay que usar como argumento en la función **INKEY**, mientras que **ESPEJO** es la tecla equivalente del teclado.

Primer Joystick	JOY(0)	CLAVE	Segundo Joystick	JOY(1)	CLAVE	ESPEJO
Arriba	Bit 0	72	Arriba	Bit 0	48	6
Abajo	Bit 1	73	Abajo	Bit 1	49	5
Izquierda	Bit 2	74	Izquierda	Bit 2	50	R
Derecha	Bit 3	75	Derecha	Bit 3	51	T
Gatillo 2	Bit 4	76	Gatillo 2	Bit 4	52	G
Gatillo 1	Bit 5	77	Gatillo 1	Bit 5	53	F

Observa que cuando el CPC464 examina el estado del segundo joystick, no puede decir si realmente corresponde al joystick o a las teclas equivalentes que figuran en la columna **ESPEJO**. Pero en la práctica no es probable que haya conflicto de interpretación, y así se puede usar el teclado como segundo joystick (cuando todavía no lo has adquirido).

Cuando se usa el **AMSOFT JY1**, el segundo es idéntico al primero, y se conecta en el enchufe de la base del considerado primero. No se requiere cableado especial para poder usar este segundo joystick.

El conector en la parte trasera del ordenador que sirve de **PORTAL DE ENTRADA/SALIDA**, afecta joysticks standard para otros ordenadores personales, aunque no suele permitir la utilización de un segundo joystick a no ser que se emplee un adaptador especial. Sin embargo, **NO** intentes usar uno de estos como segundo joystick y enchufarlo en el conector del **AMSOFT JY1**.

Los escritores de programas pueden considerar la posibilidad de permitir al comienzo de sus programas que el usuario elija operación con los joysticks o con las teclas de cursor, pudiendo emplear en lugar de la tecla **COPY** o de cualquier otra, el gatillo de disparo.

### 7.2 Acoplando la impresora

El CPC464 AMSTRAD admite utilizar una impresora que responda a la norma de interconexión "Centronics" standard en la industria.

El cable impresora se construye simplemente por conexiones directas y una a una entre el conector marcado **PRINTER** en el ordenador y el conector de la impresora. Observa que en la tarjeta de circuito impreso del ordenador hemos dejado dos 'pistas' menos que en el conector de la impresora; eso te permite usar conectores standard para tarjetas de circuito impreso directamente.

Los detalles exactos de la interconexión se muestran en el Apéndice V.

El cable deberá construirse de modo que el pin 1 del ordenador conecte con el pin 1 de la impresora, el pin 19 del ordenador con el pin 19 de la impresora, etc. Y con los pines 18 y 36 de la impresora sin conectar a ningún pin del ordenador.

En particular, la fila inferior de pistas en el ordenador está numerada a partir de 19 (y no como pudieras esperar, a partir de 18 dado que sólo hay 17 pistas en la fila superior); con el fin de que cada cablecillo que se use se conecte exactamente en la pista con el mismo número tanto por la parte del ordenador como por la parte de la impresora.

El ordenador usa la señal **BUSY** en el pin 11 para indicar si está OCUPADO, y así poderse sincronizar con la impresora, para lo que esperará cuando la impresora no está seleccionada o está OFF LINE.

No se requiere ningún comando de preparación por parte del usuario, y la salida se dirige hacia la impresora en cuanto especifiques en cualquier comando de salida el CAUCE número 8, como por ejemplo:

**LIST #8**

que hará que el programa BASIC que haya en memoria sea listado, siempre y cuándo sea de la clase admitida para listado (no esté protegido).

Dentro de los programa, puedes usar instrucciones para que expongan datos por impresora usando el mismo número de cauce; por ejemplo:

**PRINT #8, "Este es el curso para la impresora"**

Muchas impresoras automáticamente pasan al renglón siguiente cuando el extremo de la línea a imprimir sobrepasa el margen permitido, así que comprueba el manual de tu impresora. Pero observa que el BASIC AMSTRAD puede hacer esa misma función, si le especificas la ANCHURA de la línea a imprimir mediante el comando **WIDTH**. El valor prescrito cuando omities explícitamente otro, para la anchura de las impresoras es de 132 columnas, pero puedes fijarlo al valor que precises, por ejemplo con **WIDTH 80**.

Cuando se le da como valor de anchura el de 255, el BASIC AMSTRAD no se preocupará de si la línea sobrepasa el margen derecho y delegará esta función completamente a la propia impresora. BASIC mantiene internamente un contador con la posición que en cada momento ocupa el cabezal de impresión, y puedes examinar el valor de ese contador, y por tanto de la POSICION del cabezal de la impresora mediante el comando **POS**, como por ejemplo:

```
IF POS( # 8 ) > 50 THEN GOTO 100
```

El CPC464 envía una marca de AVANCE DE LINEA, que corresponde en ASCII a **CHR\$(10)** y una marca de RETORNO DE CARRO, que corresponde a **CHR\$(13)**, al final de cada una de las líneas que quiere imprimir. La impresora habitualmente dispone de un conmutador con el que puedes elegir la entrada de datos apropiada, y será obvio inmediatamente cuáles son las exigencias prescritas como standard y para omisiones en cuanto intentes imprimir la primera vez.

### 7.3 Imprimiendo los gráficos

El manual suministrado con tu impresora, especificará los CODIGOS DE CONTROL, que generalmente pueden ser enviados por el BASIC, al darle el comando:

```
PRINT CHR$(n)
```

Algunas de las impresoras tienen caracteres gráficos similares a muchos de los que posee AMSTRAD y que están enumerados en el Apéndice III, pero es muy poco probable que todos los números asignados a los caracteres correspondan de manera exacta, de forma que tendrás que preparar tu propia tabla de conversión de acuerdo con la impresora en cuestión.

Aunque la interconexión con la impresora está planteada para impresoras matriciales de bajo coste, admite por supuesto impresoras de margarita cuando responden a la norma de acoplamiento adecuada; y también admitirá trazadoras de gráficos e impresoras por chorro de tinta con varios colores de tinta. La clave de todas estas posibilidades de compatibilidad radica en que el CPC464 utiliza la norma standard para transferencia de datos en paralelo.

## 8. BASIC AMSTRAD.

### Guía de referencia

Una lista ordenada con todas las palabras clave del BASIC AMSTRAD e ilustrada por ejemplos, presentando la forma precisa de utilización y las palabras clave asociadas.

Temas tratados en este capítulo:

- \* La notación empleada
- \* Caracteres especiales y su significado
- \* Todas las palabras clave (reservadas) en el BASIC AMSTRAD

Este capítulo es un resumen conciso de las funciones que puede llevar a cabo el interpretador BASIC incorporado en el CPC464. El surtido de posibilidades disponibles, representa una implementación standard en la industria del lenguaje BASIC, potentemente ampliada para estar acorde con los avanzados circuitos electrónicos del CPC464.

#### 8.1 Notación

##### Caracteres Especiales

& o &H	Prefijo para constantes hexadecimales
&X	Prefijo para constantes binarias
:	Separador de los comandos incluidos en una misma línea.
#	Prefijo de número de CAUCE (por donde fluye la información).

##### Literales

Las **constantes literales** pueden tener de 0 a 255 caracteres de longitud y puede intervenir en expresiones literales, uniéndose mediante el operador de CONCATENACION o EMPALME, simbolizada por el signo +, siempre que el resultado de la expresión sea otro **lítero** con menos de 255 caracteres. (Son consideradas como 'sartas' de caracteres, ocupando una posición por carácter incluyendo el espacio en blanco).

Las **constantes numéricas** pueden ser enteras o reales. Las enteras han de estar en la gama -32768 a +32767; mientras que las reales tienen poco más de 9 dígitos de precisión, y han de estar en la gama -1.7E+38 a +1.7E+38, con el valor absoluto más pequeño para representar la 0 de aproximadamente 2.9E-39.

Los nombres de las **variables** llevan como último carácter del nombre un símbolo designador de la clase a que pertenece: % numerales enteros, ! numerales reales, \$ literales.



Una expresión es cualquier combinación de constantes variables y operadores que puede ser evaluada y el resultado de la evaluación asignado a una variable de la clase pertinente. Habrá expresiones numerales y expresiones literales.

Si el resultado de la expresión no cae dentro de la gama permitida, o bien el argumento empleado en una función no corresponde con los valores aceptados, o bien el valor dado a un parámetro de un comando BASIC no es válido, será rechazado mostrando el correspondiente mensaje de error.

Los CAUCES internos para la transferencia de información entre los circuitos centrales y periféricos del ordenador, están IDENTIFICADOS mediante lo que llamamos número del cauce. Hay números de cauce prescritos para el monitor, la impresora, la ductora de cassette, y el teclado.

## PALABRAS RESERVADAS

En esta sección enumeramos en orden alfabético las palabras reservadas o **palabras clave** del lenguaje BASIC AMSTRAD, presentandolas en la siguiente forma:

### PALABRA CLAVE

Sintáxis/Función

Ejemplo

Descripción

PALABRAS CLAVE asociadas

## IMPORTANTE:

Las **palabras clave** pueden representar:

**ACCIONES:** que con ciertos parámetros han de ser llevadas a cabo por el ordenador, y que pueden darse en forma de **COMANDOS** o de **INSTRUCCIONES** (precediéndolas de un número de línea).

**FUNCIONES:** con los argumentos correspondientes y que pueden ser usadas en comando, instrucciones y expresiones.

## Paréntesis

Las parejas de ( ) se requieren como parte de las acciones o de las funciones. En las descripciones y en el texto usamos otros tipos de paréntesis, que no deben ser nunca tecleados como parte de la instrucción. Son: los corchetes cuadrados [ ] para datos o parámetros opcionales. Los corchetes angulados < > para reflejar conceptos mencionados en el texto.

### Comillas

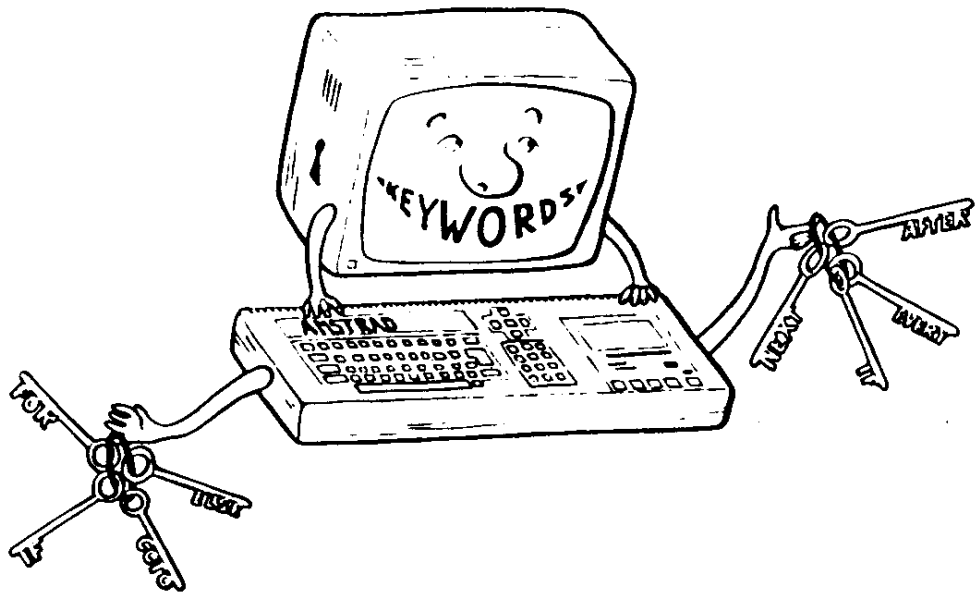
Únicamente las comillas " " forman parte intrínseca del lenguaje BASIC, y se usan para delimitar exactamente las constantes literales. Aunque no es aconsejable, en ciertas ocasiones puede prescindirse de las comillas de cierre.

Los apóstrofes ' ' se usan para resaltar o destacar ciertos aspectos en las descripciones, pero no aparecerán en ninguna parte para evitar confusiones, y sólo los usaremos incluyéndolos como cualquier otro signo dentro de una <expresión literal>.

### Tecleando

BASIC convierte todas las palabras clave impuestas en el teclado usando las 'letras minúsculas' a las correspondientes 'letras mayúsculas' cuando se lista el programa. Los ejemplos que aquí mostramos son en mayúsculas, dado que así aparecen al listar el programa; pero si los tecleas usando minúsculas, te será más fácil detectar los errores de tipografía más rápidamente, dado que la palabra mal tecleada, continuará apareciendo en minúsculas al ser listado el programa.

Las palabras CLAVE han de estar delimitadas por separadores, dado que en el BASIC AMSTRAD está permitido incrustar palabras clave en los nombres de las variables y en las constantes literales, y por tanto, LIST CODE es aceptable como nombre de una variable.



## ABS

**ABS**((expresion numeral))

```
PRINT ABS(-67.98)  
67.98
```

**FUNCION:** El resultado entregado es el valor absoluto del argumento (lo que significa que si el número es negativo lo convierte en positivo).

**CLAVES asociadas:** SGN

## AFTER

**AFTER** (expresion entera) [, (expresion entera)] **GOSUB** (numero de linea).

```
AFTER 200,2 GOSUB 320
```

**ACCION:** Se desvía a la subrutina después de que ha transcurrido un período de tiempo dado. La primera expresión entera es el parámetro que fija el período de demora en unidades de 0,02 segundos; y la segunda expresión entera -en la gama 0 a 3- es el parámetro que indica cuál de los cuatro temporizadores (cronómetros) disponibles es el que se emplea en esta acción.

**CLAVES asociadas:** EVERY, REMAIN

## ASC

**ASC** ((expresion literal))

```
PRINT ASC("X")  
88
```

**FUNCION:** Obtiene como resultado el código ASCII correspondiente al primer carácter de un lítero o expresión literal (siempre y cuando corresponda un carácter ASCII).

**CLAVE asociada:** CHR\$

## ATN

**ATN**(*<expresion numeral>*)

```
PRINT ATN(1)
0.785398163
```

**FUNCION:** Calcula el arco que tiene por tangente el valor especificado como argumento. (Convierte la expresión que figura como argumento previamente a un número real en radianes, dentro de la gama  $-\pi/2$  a  $+\pi/2$ ).

**CLAVES asociadas:** SIN, COS, TAN, DEG, RAD

## AUTO

**AUTO** [*<número de línea>*][*<salto>*]

```
AUTO 100,50
```

**ACCION:** Se usa exclusivamente como COMANDO para generar automáticamente números de línea del programa. El parámetro número de línea establece el número a partir del que debe comenzar, y el parámetro salto establece el que hay entre dos números de línea consecutivos. Los valores prescritos para omisión son 10 para el número de línea y 10 para el salto.

Cuando un número de línea del programa existente en memoria va a ser sobre-escrito como consecuencia de esta acción, el BASIC inserta un asterisco (\*) como aviso.

## BIN\$

**BIN\$** (*<expresion entera positiva>*)[*<expresion entera>*]

```
PRINT BIN$(64,8)
01000000
```

**FUNCION:** El resultado de esta función es una 'sarta' de símbolos binarios 1 y 0, representando el valor equivalente al del argumento *<expresión entera positiva>*, y formado por tantos 'bitos' como indique el segundo argumento de la función.

**CLAVES asociadas:** HEX\$, STR\$

## BORDER

**BORDER** (color)[, (color)]

**BORDER** 3, 2

**ACCION:** Cambia el color del 'borde' de la pantalla. Cuando se especifican los dos argumentos para el color, el borde alterna entre los dos colores con una 'cadencia' estipulable mediante el comando **SPEED INK**. La gama de colores para el borde es de 0 a 26.

**CLAVES** asociadas: **SPEED INK**

## CALL

**CALL** (expresion direccional)[, (lista de): (parametros)]  
**CALL** &BD19

**ACCION:** Permite que se 'cite' una SUBROUTINA desarrollada externa e independientemente de BASIC. Usala con precaución, porque no es una acción con la que el inexperto pueda experimentar. El ejemplo es relativamente inofensivo, dado que lo que hace es obligarle a esperar al siguiente 'cuadro' de la imagen en el monitor, lo que es especialmente aprovechable para sincronizar los movimientos de figuras cuando se trabaja con programas de 'animación'.

**CLAVE** asociada: **UNT**

## CAT

**CAT**

**CAT**

**ACCION:** El BASIC comienza a examinar la cinta del cassette y a mostrar los nombres de los ficheros que va encontrando. Esta acción no afecta al programa existente en memoria, y por tanto, puede usarse para verificar que un programa ha sido **GUARDADO** antes de alterar el programa. El BASIC te pide que pongas en marcha el cassette, y el mensaje con que anuncia el contenido de la cinta es:

**FILEMANE BlockNumber      Flag OK**

con el número de bloque pertinente y el 'banderín' que indica el modo en que se ha grabado el fichero. De acuerdo con:

- \$ un fichero con programa en BASIC
- % un fichero en BASIC 'protegido'
- \* un fichero con 'texto' de caracteres ASCII
- & un fichero en notación binaria

Si el fichero no fue grabado por BASIC, pueden aparecer otros caracteres.

CLAVES asociadas: **LOAD, RUN, SAVE.**

## **CHAIN**

### **CHAIN MERGE**

**CHAIN** (nombre del fichero)[, (expresión de números de línea)]

**CHAIN MERGE** (nombre del fichero)[, (expresión de números de línea)]

[;DELETE (gama de números de línea)]

**CHAIN "TEST", 350**

**ACCION:** El comando **CHAIN** permite el **ENGARCE** de programas 'cargando' en la memoria y desde la cinta el programa mencionado en el comando; sustituyendo al programa existente, pero conservando los valores de las variables que hubiera en memoria. La expresión de números de línea le indica la línea del nuevo programa a partir de la que debe empezar la subsecuente ejecución del programa.

Con la variante **CHAIN MERGE**, además de cargar el programa, lo **CONGREGA** con el programa existente en memoria y manteniendo los valores y variables que haya.

Si no se menciona ningún <nombre de fichero>, el BASIC extrae de la cinta el primer fichero válido que encuentre. Cuando el primer carácter anterior del nombre del fichero es el signo de admiración (!) el BASIC quita este signo del nombre del fichero para buscar en cinta, y suprime los mensajes que habitualmente genera durante el trasvase de cinta a memoria.

Aunque tanto **CHAIN** como **CHAIN MERGE** retienen todas las variables presentes en memoria, se descartan las funciones definidas por el usuario así como los ficheros abiertos. Las condiciones **ON ERROR** se restauran, las series de datos se **RESTORE** a la primera **DATA**, y se olvidan las posibles **DEFINT**, **DEFREAL**, **DEFSTR** que hubiera en el programa anterior, y todos los bucles **FOR** y **WHILE** y los desvíos a subrutinas pendientes se cancelan. Los ficheros protegidos no admiten la variante **MERGE**.

La opción DELETE permite suprimir antes de la carga, la gama de número de líneas especificada en el parámetro.

CLAVES asociadas: LOAD, MERGE

## CHR\$

CHR\$((expresion entera))

PRINT CHR\$(100)

d

FUNCION: Entrega como resultado el carácter equivalente al argumento numeral de la función, (usando el repertorio de caracteres AMSTRAD CPC464 reseñado en el Apéndice III).

CLAVES asociadas: ASC, LEFT\$, RIGHT\$, MID\$, STR\$

## CINT

CINT ((expresion numeral))

10 n=578.76543

20 PRINT CINT(n)

RUN

579

FUNCION: El resultado es la parte entera redondeada del argumento, que debe estar en la gama -32768 a +32767.

CLAVES asociadas: CREAL, INT, FIX, ROUND, UNT

## CLEAR

CLEAR

CLEAR

ACCION: Coloca a ceros o a blancos todas las variables numéricas y literales que haya en memoria.

## **CLG**

**CLG[(tinta invisible)]**

**CLG**

**ACCION:** Limpia la pantalla destinada a gráficos.

**CLAVES** asociadas: **CLS, ORIGIN**

## **CLOSEIN**

**CLOSEIN**

**CLOSEIN**

**ACCION:** Cierra el fichero en cassette definido como fichero de ENTRADA. Los comandos **NEW** y **CHAIN MERGE** también abandonan cualquier fichero que haya abierto.

**CLAVES** asociadas: **OPENIN, CLOSEOUT**

## **CLOSEOUT**

**CLOSEOUT**

**CLOSEOUT**

**ACCION:** Cierra el fichero en cassette definido como fichero de salida.

**CLAVES** asociadas: **OPENOUT, CLOSEIN**

## **CLS**

**CLS[(numeral del cauce)]**

**CLS**

**ACCION:** Limpia la 'viñeta' definida en la pantalla a través del cauce mencionado.



## CONT

CONT

CONT

ACCION: Hace que SIGA la ejecución del programa después de **\*Break\***, **STOP**, o **END**, siempre que el programa no se haya alterado posteriormente. Sólo se admite como COMANDO.

## COS

**COS** ((expresion numeral))

?COS(34)

-0.848570274

y

deg: ?cos(34)

0.829037573

FUNCION: Calcula el COSENO del valor dado como argumento. El argumento de la función se considera dado en 'radianes' a no ser que explícita y previamente se haya estipulado que sean 'grados', mediante el comando **DEG**.

CLAVES asociadas: **IN**, **TAN**, **ATN**, **DEG**, **RAD**

## CREAL

**CREAL**((expresion numeral))

```
5 DEFINT n
10 n=75.765
20 d=n/34.6
30 PRINT d
40 PRINT CLEAR(n)
50 PRINT n./55.4
run
2.19653179
76
1.37184116
Ready
```

**FUNCION:** Convierte el argumento en un número real (en oposición a números enteros).

**CLAVE asociada:** CINT

## DATA

**DATA** (lista de):(constante)

```
10 REM Lista de correctores
20 DIM Nombres$(10)
30 DIM Apellidos$(10)
40 FOR n=1 to (10)
50 READ Nombres$(n)
60 READ Apellidos$(n)
65 PRINT Nombres$(n); " Apellidos$(n)
70 DATA Julio, Sierra Frade, Julio, Legido Gonzalez, Leopoldo,
  Porras Larios, Enrique, Diaz Correa, Lourdes, Legido Gonzalez,
  Yolanda, Bellette Jimenez, Agustin, Rodriguez Guaza, Juan,
  Martinez del Rio, Miguel Angel, Ortega Plana, Yolanda, Cabello
  Jaime
90 NEXT
```

**ACCION:** Es una instrucción no ejecutable, que permite definir en el programa una serie de datos en la forma de constantes. Es una de las posibilidades más ampliamente usada con BASIC, ya que permite agrupar ciertos datos en series encabezadas por la palabra reservada **DATA** que son 'secuencialmente' empleados en el programa, cada vez que se 'apuntan' como valores de variables, mediante la instrucción **READ**. Las constantes de la instrucción **DATA** deben ser coherentes con las variables de la instrucción **READ** correspondiente.

**CLAVES asociadas:** READ, RESTORE

**DEF FN**

DEF FN(nombre)[((*parametros formales*))]=(*general expresion*)

```
10 DEF FNintereses(capital)=1.14*capital
20 INPUT "Cual es el capital resultante";capital
30 PRINT "El capital prestado mas los intereses
    anuales";FNintereses(capital)
```

ACCION: BASIC permite que se definan y USEN funciones definidas por el usuario con varios argumentos pero un solo resultado literal o numeral. Esta instrucción constituye la definición de la función, que luego puede ser utilizada en la misma manera que las funciones 'intrínsecas' del BASIC (tales como COS, SIN, etc.).

Para obtener el resultado de la función definida, basta nombrarla en el programa sustituyendo los argumentos 'formales' por los argumentos 'actuales' en cada momento. La función debe definirse antes de que pueda ser citada en el programa.

**DEFINT**  
**DEFSTR**  
**DEFREAL**

DEFtype (*gana(s) de letras*)

```
DEFINT I-N
DEFSTR A,W-Z
DEFREAL
```

ACCION: Permite definir la clase de variables según el primer carácter anterior de su nombre; y corresponde a variables enteras, literales y reales sucesivamente. Siempre el nombre de la variable debe comenzar con una letra, pero puede ser en minúscula o en mayúscula.

CLAVES asociadas: LOAD, RUN, CHAIN, NEW, CLEAR

**DEG**

DEG

DEG

**ACCION:** Fija para los argumentos de las funciones 'trigonométricas' la medida en grados. La condición prescrita para omisiones y a la puesta en marcha es la medida en 'radianes' para los argumentos de las funciones trigonométricas. Este comando cambia esa prescripción hasta que sea restaurada mediante los comandos **CLEAR**, **RAD** o se cargue un nuevo programa desde el cassette.

**CLAVES asociadas:** **RAD**

## **DELETE**

**DELETE** (gama de números de línea)

**DELETE 100-200**

**ACCION:** Los números de línea comprendidos en la gama mencionada se **SUPRIMEN** del programa existente en memoria. Si encuentra un error durante esta acción, se perderán las líneas ya suprimidas, úsala con cuidado y comprueba la gama de números a suprimir antes de terminar el comando.

**CLAVES asociadas:** **NEW**, **CHAIN**

## **DI**

**DI**

```
10 CLS
20 TAG
30 EVERY 10 GOSUB 100
40 X1=RND*320:X2=RND*320
50 Y=200+RND*200
60 FOR X=320-X1 TO 320+X2 STEP 2
70 DI:PLOT 320,0,1:MOVE X-2,
  Y:PRINT " ";:MOVE X,Y:PRINT "#";:EI
80 NEXT
90 GOTO 40
100 MOVE 320,0
110 DRAW X+8,Y-16,0
120 RETURN
```

**ACCION:** Permite **INHIBIR** las señales de **interrupción** generadas en los diversos circuitos de la máquina, exceptuando la interrupción de programa **\*Break\***; y hasta que vuelvan a ser explícitamente 'facultadas', mediante el comando **EI** o lo sean implícitamente al concluir la subrutina incitada por la interrupción, que debe terminar con el comando **RETURN**.

Se emplea cuando un programa desea continuar todas sus operaciones sin verse materialmente "interrumpido" por señales enviadas desde otros circuitos, como por ejemplo cuando dentro del programa hay dos instrucciones 'compitiendo' para apoderarse de una impresora, canal de sonido, etc. En el ejemplo, el programa principal y el párrafo que comienza en la instrucción 100 (que es puesto en marcha de acuerdo con la instrucción 30) están compitiendo por la pantalla de gráficos.

CLAVE asociada: EI

## DIM

**DIM (lista de):(variables subscritas)**

```
10 CLS:PRINT "Introduzca 5 nombres....":PRINT
20 DIM B$(5)
30 FOR N=1 TO 5
40 PRINT "Nombre"N"por favor";
50 INPUT B$(N)
60 NEXT
70 FOR N=1 TO 5
80 PRINT "Muy inteligente por tu parte";B$(N);
  "haber comprado un CPC464"
90 NEXT
```

**ACCION:** Hace que se OCUPE el espacio destinado a tablas, de acuerdo con el tamaño de la tabla especificada mediante los 'sufijos' que seleccionan los elementos de la tabla. BASIC exige que las tablas (ringlas) sean declaradas antes de poder usarlas en el programa, pero admite la carencia de definición previa cuando no hay más de 10 elementos en la tabla.

Para seleccionar un elemento de la tabla, ha de usarse el mismo NOMBRE que se ha dado al establecer las dimensiones, y usar el sufijo o sufijos correspondientes a ese elemento. El tamaño de las tablas sólo está limitado por la memoria disponible.

CLAVES asociadas: ERASE

## **DRAW**

**DRAW** (x coordenadas), (y coordenadas) [, (tinta usada)]

**DRAW 200, 200, 13**

ACCION: 'Dibuja' un segmento en la pantalla a partir de la posición corriente del cursor de gráficos hasta la posición señalada por los parámetros **coordenada x**, y **coordenada y**. El parámetro 'tinta' determina la usada al dibujar el segmento.

Hay muchos ejemplos en el capítulo 5.

CLAVES asociadas: **DRAW, PLOT, PLOT, MOVE, MOVER, TEST, TESTR, XPOS, YPOS, ORIGIN**

## **DRAWR**

**DRAWR** (incremento de x ), (incremento de y ) [, (tinta usada)]

**DRAWR 200, 200, 13**

ACCION: 'Dibuja' un segmento a partir de la posición corriente del cursor de gráficos y hasta la posición dada por lo parámetros **coordenada x** y **coordenada y**, en valores relativos a la posición actual del cursor.

CLAVES asociadas: **DRAW, PLOT, PLOT, MOVE, MOVER, TEST, TESTR, XPOS, YPOS, ORIGIN**

## **EDIT**

**EDIT** (numero de linea)

**EDIT 110**

ACCION: Coloca la línea de programa mencionada en el parámetro, en el modo adecuado para que pueda ser EDITADA (revisada y corregida). Véase sección 1.4.

CLAVES asociadas: **LIST**

**EI****EI****EI**

ACCION: 'Faculta' las señales de interrupción generadas por los circuitos internos y que están 'inhibidas' por un comando **DI** anterior.

CLAVES asociadas: **DI**

**END****END****END**

ACCION: 'Termina' el programa. En el BASIC AMSTRAD está explícito un comando de **FIN** cuando llega a la última línea de programa. Con el comando **END** se cierran todos los ficheros en cassette y se vuelve al modo directo de ingreso de comandos (los 'chorros de notas' pendientes de ser emitidos, lo harán hasta que queden vacíos).

CLAVES asociadas: **STOP**

**ENT**

**ENT** (numero de envoltura) [(secciones de envoltura)]

10 **ENT** 1, 100, 2, 20

20 **SOUND** 1, 100, 1000, 4, 0, 1

ACCION: Es posible definir la envolvente de tono correspondiente a una 'nota musical' generada con el comando **SOUND**. La envolvente de tono define las variaciones de frecuencia que han de tener lugar a lo largo de la duración de la nota. Véase el capítulo 6 y el Apéndice VII para una descripción completa.

El parámetro <número de envolvente> ha de ser una expresión numeral entera cuyo valor resultante debe estar en la gama 1 a 15, e identifica esa envolvente de tono. Si se usa un número negativo, se repetirá esa envolvente de tono hasta que concluya el sonido generado.

Se pueden dar hasta cinco parámetros denominados <secciones> y constituidos a su vez por otra serie de tres parámetros denominados: <cuenta de pasos>, <altura del paso>, <tiempo del paso>, o bien por series de dos parámetros, denominados: <período de tono>, <tiempo de paso>.

La primera de las variantes especifica un cambio RELATIVO en relación con el tono usado en el paso anterior. La segunda especifica un tono en valor absoluto para todo el paso.

Los parámetros indican:

**cuenta de pasos** da el número de pasos en esa sección, y debe ser un numeral entero en la gama 0 a 239.

**altura del paso** da el incremento o decremento sobre el tono del paso anterior, y es una expresión entera dentro de la gama -128 a +127.

**tiempo del paso** indica la duración del paso en centésimas de segundo, y debe ser una expresión entera en la gama 0 a 255 (considerando el 0 igual que el 256).

**período de tono** da el nuevo valor para el **período** (relacionado con la inversa de la frecuencia) del tono, y ha de estar en la gama 0 a 4095.

En el comando **SOUND** se hace que se genere una nota musical con un determinado tono inicial, y luego se pueden especificar 1 de 15 envolventes de tono para vivificar dicha nota. Si no se especifica ninguna envolvente, o el número que se menciona en el comando no corresponde a ningún identificativo de envolvente, el tono permanecerá constante a lo largo de la duración de la nota. La envolvente de tono no tiene ningún efecto sobre la duración del sonido generado. Si quedan pasos de la envolvente de tono cuando concluye el sonido, simplemente son despreciados.

Las envolventes de tono identificadas con números negativos, indica que serán repetidas incesantemente hasta que concluya el sonido pertinente.

Las parámetros de las envolventes de tono, se evalúan cuando se ejecuta el comando y se conservan los resultados para uso posterior, cuando se cita esa envolvente. Pero por definir envolventes de tono no se provoca que se re-ejecute el comando de sonido correspondiente. Cada vez que se declara una envolvente de tono, los valores concernientes a la envolvente que posee el mismo número identificativo, son abandonadas.

Cambiando una envolvente mientras un sonido generado la está usando, (o está pendiente de usarla), pero ya ha sido incluida en el 'chorro' de notas a emitir por altavoz, los efectos no están determinados (y puede que sean interesantes).

Especificando una envolvente con un número identificativo pero sin ninguna sección, cancela los valores corrientes para esa envolvente. Cualquier uso posterior de la envolvente será desechado y se utilizarán los parámetros prescritos para omisiones.



CLAVES asociadas: ENV, SOUND

## ENV

ENV (numero de envoltura)[, (secciones de envoltura)]

10 ENV 1, 100, 2, 20

20 SOUND 1, 100, 1000, 4, 1

ACCION: Cuando se genera un sonido, es posible variar su volumen a lo largo de la duracion del sonido, definiendo una envolvente de volumen. Se usa el comando ENV con una serie de tres parámetros, denominados:

<cuanta de pasos>, <altura de paso>, <tiempo del paso>

y define cómo varía el volumen en los sucesivos pasos. La **cuanta de pasos** debe ser una expresión entera de 0 a 127; la **altura del paso** una expresión entera en la gama -128 a +127, y el **tiempo del paso** se medirá en unidades de centésimas de segundo, y será una expresión entera dentro de la gama 1 a 256.

El <número de envolvente> es el parámetro 'identificador' de dicha envolvente y ha de ser un numeral entero dentro de la gama 1 a 15.

Se puede usar una serie de cinco parámetros denominados secciones de envolvente, y a su vez, cada uno de ellos puede ser una serie de tres parámetros, denominados **cuanta de pasos**, **altura del paso**, **tiempo del paso**, o bien una serie de dos parámetros denominados:

**envolvente prescrita**, **período de envolvente**.

La primera de las variantes, especifica una sección de envolvente cuyos parámetros están controlados PROGRAMALMENTE, significando lo siguiente:

**cuanta de pasos** establece el número de pasos de que consta la sección y ha de ser una expresión numeral entera en la gama 0 a 127.

**altura del paso** da el incremento o decremento de la amplitud del paso en relación con la amplitud del paso anterior, y debe ser un valor entero en la gama -128 a +127.

(Si la cuenta de pasos es cero, el valor estipulado para la amplitud corresponde con el valor con que haya terminado la sección de envolvente anterior, o con el valor de arranque reseñado en el comando SOUND).

**tiempo de paso** especifica la duración del paso en centésimas de segundo, y debe ser una expresión numeral entera dentro de la gama 0 a 255 (siendo el 0 considerado como 256).

La segunda variante especifica una sección de envolvente que va a ser ejecutada bajo control CIRCUITAL del equipo, siendo:

**envolvente prescrita** el valor que se aloja en el 'registro de formas de envolvente' (el registro 15 en notación octal).

**período de envolvente** es el valor que se aloja en el 'registro de período de envolvente' (registros 13 y 14, en notación octal).

Los valores de las envolventes prescritas, no tienen un tiempo de paso asociado, por lo que se ejecuta inmediatamente la siguiente sección de la envolvente. Es aconsejable que en la siguiente sección haya una pausa de la duración adecuada (y si no hubiera siguiente sección, se efectúa una pausa de dos segundos).

El comando **SOUND** estipula el volumen inicial de la NOTA y como parámetro se puede especificar 1 de 15 envolventes de volumen. Si no se especifica ninguna, o una que no haya sido previamente definida e identificada, el volumen permanecerá constante a lo largo de la duración del sonido.

Estipulando, en el parámetro de altura del paso un 0, con una cuenta de pasos que sea distinto de 0, se consigue que el volumen permanezca constante e igual al último valor estipulado, durante toda la duración del paso.

Los parámetros de la envolvente de volumen se evalúan cuando se ejecuta el comando, y se conservan los resultados para uso posterior, pero usar una envolvente de volumen no implica que sea re-ejecutado el comando.

Cada vez que se estipula una envolvente de volumen determinada, se pierden los valores previos. Cambiando una envolvente mientras algún sonido la está usando, o cuando hay una nota pendiente de ser emitida, producirá efectos indeterminados (pero posiblemente interesantes).

Especificando una envolvente sin secciones, cancela cualquiera de las estipulaciones previas. Cualquier uso posterior de la envolvente será ignorado, y los parámetros prescritos para omisiones, usados en su lugar.

**CLAVES asociadas: ENT, SOUND**

**EOF****EOF****PRINT EOF****-1**

**FUNCION:** Examina si el cabezal lector del cassette ha alcanzado el Final del Fichero, cuando lo ha hecho, el resultado es -1 (cierto), y en caso contrario 0 (falso).

**CLAVES asociadas:** OPENIN

**ERASE**

**ERASE** (lista de; (nombre de variable))

**ERASE A, B\$**

**ACCION:** Cuando ya no se requiere usar en el programa una TABLA (ringla), puede conseguirse espacio LIBRE en la memoria, con el comando **ERASE**, para disponer de mayor espacio en memoria.

**CLAVE asociada:** DIM

**ERR****ERL****ERR****ERL**

```

10 CLS
20 ON ERROR GOTO 1000
30 DATA LOURDES, YOLANDA, LETICIA, CONSUELO, ANTONIA
40 READ A$
50 PRINT A$
60 GOTO 40
70 REM que empieza la deteccion de errores
1000 IF ERR=4 THEN PRINT "No tengo datos que copiar"
1010 IF ERL<70 AND ERL>20 THEN PRINT
    ".....en las lineas 30-60"
1020 END

```

**FUNCIONES:** Estas funciones son auténticas variables (y no es preciso darles ningún argumento), y sus valores corresponden al 'número de error' y 'número de línea' donde se detectó el error durante la ejecución del programa. Consulta la lista de mensajes de error en el Apéndice VIII.

CLAVES asociadas: ON ERROR, ERROR

## **ERROR**

**ERROR** (expresion entera)

### **ERROR 17**

**COMANDO:** Inicia la acción de error con un número de error dado. El error puede ser uno ya usado y reconocido por BASIC (Apéndice VIII) en cuyo caso la acción adoptada es la misma que se adoptaría si tal error hubiera sido detectado por BASIC. Los números de error superiores a aquéllos reconocidos por BASIC pueden ser usados por el programa para señalar sus propios errores.

CLAVES asociadas: ON ERROR, ERR, ERL

## **EVERY**

**EVERY** (expresion entera) [, (expresion entera)] **GOSUB** (numero de linea)

**EVERY 500, 2 GOSUB 50**

**ACCION:** El CPC464 mantiene un reloj para medida del tiempo, que puede usar para preparar cuatro 'cronómetros' diferentes. El comando **EVERY** obliga a BASIC a examinarlos regularmente, y a hacer que cada cierto intervalo de tiempo se desvíe del programa para ejecutar la subrutina citada en la instrucción. El segundo parámetro especifica el 'cronómetro' a emplear, por lo que debe estar en la gama 0 a 3, y el primer parámetro cada cuanto tiempo tiene que efectuar la subrutina cuyo número de línea viene reflejado por el parámetro del comando **GOSUB** asociado.

CLAVES asociadas: AFTER, REMAIN

## EXP

EXP ((*expresion numeral*))

```
PRINT EXP(6.876)
968.743625
```

FUNCION: El resultado es el número **E** elevado a la potencia dada por el argumento. (El número **E** es la base de los logaritmos naturales y aproximadamente es 2.7182818).

CLAVE asociada: LOG

## FIX

FIX ((*expresion numeral*))

```
PRINT FIX(9.99999)
9
```

FUNCION: A diferencia de la función **CINT**, la función **FIX** da como resultado exactamente la parte entera de su argumento. Es decir, los dígitos que aparecen antes del punto decimal, (y por tanto, sin ningún redondeo, como en la función **CINT**).

CLAVES asociadas: CINT, INT, ROUND

## FOR

FOR (*variable simple*) = (*inicio*) TO (*fin*) [STEP (*salto*)]

```
FOR DAY=1 to 5 STEP 2
```

ACCION: Marca el comienzo de un párrafo del programa que debe ser REPETIDO el número de veces determinado por la variable simple que controla el 'bucle', mediante los valores inicial y final que ha de tener, y del 'salto' que debe efectuar a cada ronda. Si no se especifica el parámetro STEP se toma como 1.

CLAVES asociadas: NEXT

## **FRE**

**FRE** ((expresion numeral))

**FRE** ((expresion literal))

**PRINT FRE** (0)

**PRINT FRE** (" ")

**FUNCION:** El resultado refleja la cantidad de memoria que no ha sido usada por BASIC, cuando el argumento de la función es un numeral. Cuando el argumento es literal, v.g. **FRE(" ")** se provoca una "recogida de basuras" antes de calcular el espacio disponible en memoria.

## **GOSUB**

**GOSUB** (numero de linea)

**GOSUB** 210

**ACCION:** Hace que BASIC se 'desvíe' (VÁya Y venGA) a la subrutina que comienza en esa línea especificada.

**CLAVE asociada:** RETURN

## **GOTO**

**GOTO** (numero de linea)

**GOTO** 90

**ACCION:** Hace que BASIC 'brinque' al número de línea especificado. (VAYA, pero no vuelva aquí).

## HEX\$

HEX\$ ((expresion entera positiva))[, (expresion entera)]

```
PRINT HEX$(65534)
FFFE
```

FUNCION: El mismo argumento expresado en notación HEXADECIMAL. (Apéndice II).

CLAVES asociadas: BIN\$, STR\$

## HIMEM

HIMEN

```
?HIMEN
43903
```

FUNCION: Corresponde a una de las VARIABLES del SISTEMA, que determina la dirección máxima usada por BASIC, y puede también emplearse en expresiones numerales en la forma normal.

Antes de alterar el valor de la máxima dirección de memoria, usando el comando MEMORY, es aconsejable examinar el valor de HIMEM, simplemente con mm=HIMEM. Posteriormente puedes estipular que el sistema vuelva a emplear esa dirección, mediante el comando MEMORY mm.

CLAVES asociadas: FRE, MEMORY

## IF

IF

IF (expresion logica) THEN (opcion) [ELSE (opcion)]

IF (expresion logica) GOTO (numero de linea) [ELSE (opcion)]

```
IF A>B THEN A=C ELSE A=D
```

```
IF A>B GOTO 1000 ELSE 300
```

**ACCION:** Hace que se examine SI el resultado de la expresión lógica es 'cierto' o 'falso'; y que se ejecuten las acciones reflejadas después de la palabra **THEN**, cuando el resultado es cierto, o que salte a la parte **ELSE**, o simplemente a la siguiente línea de programa.

Estas instrucciones condicionales pueden 'anidarse', sin más limitación que la longitud de línea, sin más que incluir una instrucción **IF** dentro de la serie de comandos que concierne a la parte **THEN** o a la parte **ELSE** de otra instrucción **IF** anterior.

**CLAVES** asociadas: **THEN, ELSE, GOTO, OR, AND, WHILE**

## INK

**INK** (ink), (color) [, (color)]

**INK** 0, 1

**ACCION:** Depende del modo corriente de pantalla (Capítulo 5), la gama de valores que pueden tener los parámetros; pero cambia el número de color, usado para la **TINTa** de la 'pluma' usada o del **TINte** del 'papel' empleado, de acuerdo con la tabla de números de colores del Apéndice VI.

**CLAVES** asociadas: **PEN, PAPER**

## INKEY

**INKEY** ((expresion entera))

```
10 CLS:IF INKEY(55)=32 THEN 30 ELSE 20
20 CLS:GOTO 10
30 PRINT "Estas presionando" [SHIFT] y V"
40 FOR t=1 TO 1000:NEXT:GOTO 10
```

**FUNCION:** El resultado de la función depende del estado en que se encuentre la tecla cuyo <código de entrada> corresponde al argumento de la función. El teclado se escruta completamente en dos centésimas de segundo, y con esta función se puede **INDAGAR** el estado de una tecla determinada. Es particularmente útil para detectar respuestas S/N, dado que la pulsación de la tecla **SHIFT** puede no afectar al resultado, si se usa una de las opciones de interpretación. En el ejemplo anterior se detecta si están en ese momento pulsadas las dos teclas **SHIFT** y **V**.



Los resultados de la función corresponden con:

Valor entregado	SHIFT	CTRL	KEY
-1	cualquiera	cualquiera	NO PULSADA
0	NO PULSADA	NO PULSADA	PULSADA
32	PULSADA	NO PULSADA	PULSADA
128	NO PULSADA	PULSADA	PULSADA
160	PULSADA	PULSADA	PULSADA

CLAVES asociadas: INPUT, INKEY\$

## INKEY\$

```

10 CLS
20 PRINT "Eres un tipo inteligente (S o N) ?"
30 a$=INKEY$: IF a$="" GOTO 30
40 IF a$="N" OR a$="n" THEN PRINT
   "entonces debes comprarme!":END
50 IF a$="S" OR a$="s" THEN
   PRINT "Eres excesivamente modesto!!!":END
60 GOTO 20

```

**FUNCION:** Suministra como resultado el carácter ASCII correspondiente a la última tecla que se haya pulsado (o la que esté pulsada en el momento en que se ejecuta el comando. Si no hay ninguna tecla pulsándose, ni ninguna pulsación anterior que esté pendiente de tratar, el resultado es el 'lítero vacío' (""). Es útil para no tener que esperar a que el usuario pulse la tecla ENTER, y poder proseguir el programa inmediatamente. En el ejemplo se examina el resultado constantemente en la línea 30.

CLAVES asociadas: INPUT, INKEY

## INP

INP ((numero del portal))

```
PRINT INP(&FF77)
```

**FUNCION:** El resultado es el valor depositado en el 'portal de entrada' cuya dirección viene dada por el argumento de la función.

CLAVES asociadas: OUT, WAIT

## INPUT

**INPUT** [(#(numeral del cauce),)[;][(constante literal entre comilladas);](lista de):(variable)) or **INPUT** [(#(numeral del cauce),)[;][(constante literal entre comilladas);](lista de):(variable))

```
10 CLS
```

```
20 INPUT "Dame dos numeros, separados por una  
coma";A,B
```

```
30 PRINT A=B THEN PRINT "Los dos numeros son  
iguales"
```

```
40 IF A>B THEN PRINT A "es mayor que" B
```

```
50 IF A<B THEN PRINT A "es menor que" B
```

```
60 CLEAR:GOTO 20
```

**ACCION:** INGRESA datos procedentes de los circuitos de entrada del ordenador, a través del CAUCE cuyo número se menciona en la instrucción, y los IMPONE como valores de las variables mencionadas en la instrucción. La mayor parte de las veces se usa para ingresar datos procedentes del teclado, y ofrece la posibilidad de incluir un mensaje para que sea 'expuesto' usando el mismo CAUCE de transferencia. Si se incluye un punto y coma después del mensaje a exponer, se obliga a que lo termine con un signo de interrogación, mientras que la coma suprime dicho signo. Similarmente, un punto y coma antes de dicho mensaje, impide que se genere un 'retorno de carro' al final de los datos que se están imponiendo por teclado. Los datos ingresados deben ser coherentes con la variable sobre los que se imponen; en caso contrario (y el más frecuente es teclear O en lugar de 0) y se saca el aviso:

**?Redo from start**

repitiendo además el mensaje que hayas especificado en la instrucción, para que puedas 'volver a teclear desde el principio'.

Todas las respuestas deben concluir pulsando la tecla **ENTER**.

Cuando se indica un parámetro de CAUCE correspondiente a la ductora de cassette, no tiene objeto incluir el mensaje de ayuda, pero si se especifica, será desechado por los programas internos de gestión del cassette; por lo que el mismo programa puede ingresar datos a través de distintos cauces, sin más que cambiar el numeral identificativo de cauce.

Cada dato recuperado de un fichero será impuesto como valor de cada variable de la lista reseñada en la instrucción, secuencialmente; y deben ser ambas de la misma clase. Para una variable numérica, el dato debe estar terminado por una coma, un retorno de carro, un espacio en blanco, o la marca de final de fichero. Para una variable literal, el dato será tomado con su auténtica índole de **lítero**, cuando esté encerrado entre comillas. (Y queremos recalcar que **lítero** implica 'al pie de la letra', sin tocar, alterar o hacer nada con él).

Cuando no están entrecomillados, los datos literales se consideran terminados en las mismas situaciones mencionadas para los datos numéricos.

CLAVES asociadas: LINE INPUT, READ, INKEY\$

## INSTR

INSTR([(expresion entera),](expresion literal), (expresion literal))

PRINT INSTR(2, "BANANA", "AN")

FUNCION: Sirve para examinar el literal dado por el primer argumento, y determinar dónde empieza a aparecer el literal dado por el segundo argumento, el valor resultante es la posición correspondiente al segundo argumento. El parámetro numérico opcional indica la posición a partir de la que debe comenzar la búsqueda; y si se omite, la búsqueda comienza a partir del primer carácter del argumento donde busca.

CLAVES asociadas: MID\$, LEFT\$, RIGHT\$

## INT

INT ((expresion numeral))

PRINT INT(-1.995)  
-2

FUNCION: El resultado es el número entero inmediatamente inferior al valor del argumento. Por lo tanto, es igual que la función **FIX** si el argumento es positivo, pero con argumentos negativos da como resultado un número entero negativo cuyo valor absoluto es la parte entera redondeada del argumento.

CLAVES asociadas: CINT, FIX, ROUND

## JOY

JOY ((expresion entera))

10 IF JOY(0)=8 THEN GOTO 100

## BASIC: Palabras Clave

**FUNCION:** Proporciona como resultado un número decimal entero que 'calibra en binario' el estado del control de juegos correspondiente al argumento. La 'calibración binaria' responde a la siguiente tabla:

Bit	Decimal
0: Arriba	1
1: Abajo	2
2: Izquierda	4
3: Derecha	8
4: Gatillo 2	16
5: Gatillo 1	32

CLAVES asociadas: INKEY

## KEY

**KEY** (expresion entera), [CHR\$(n)+](expresion literal) [CHR\$(n)]

**KEY** 140, "RUN"+CHR\$(13)

**COMANDO:** Asocia a la TECLA identificada por el primer parámetro -cuyo valor corresponde al **código de entrada**-, la expresión literal reseñada como segundo parámetro; de manera que cada vez que se pulse esa tecla, se consigue el mismo efecto que si se hubiera tecleado directamente toda la expresión literal asociada.

Se suele decir que la tecla es 'programable', o que es una tecla 'funcional', o que su código es 'expandible'.

En el CPC464 hay 32 posibles teclas, con códigos de entrada en la gama 128 a 159 (Apéndice III), y se le pueden asociar expresiones literales con 32 caracteres de longitud como máximo; y siempre que el número total de caracteres empleados en todas las asociaciones no sobrepase de 120.

CLAVES asociadas: KEY DEF

## KEY DEF

**KEY DEF**(numero de tecla), (repite) [, (normal) [, (turno) [, (control) ]]]

**KEY DEF** 46, 1, 63

**ACCION:** Este comando REDEFINE el código de entrada asignado a la tecla dada por el primer parámetro, uno o varios códigos de entrada de acuerdo con los parámetros opcionales que se especifiquen en este comando.

El ejemplo anterior convierte la tecla marcada con N para que al pulsarla se interprete lo mismo que si se pulsa la tecla marcada ?. (El código de entrada (Apéndice III) de la tecla n usada en minúsculas, mientras que el código ASCII decimal del carácter ? es 63). Para dejar esa tecla en sus condiciones normales:

```
KEY DEF 46, 1, 110
```

ya que el código ASCII decimal del carácter n es 110.

CLAVES asociadas: KEY

## LEFT\$

**LEFT\$**((expresion literal), (expresion entera))

```
10 CLS
20 A$ = "AMSTRAD"
30 B$ = LEFT$(A$, 3)
40 PRINT B$
RUN
```

(LIMPIAR PANTALLA)

AMS  
Ready

**FUNCION:** El resultado es una constante literal, obtenida segregando tantos caracteres ANTERIORES del argumento literal como indique el argumento numérico. Si la longitud del argumento literal es insuficiente, el resultado es todo el argumento.

CLAVES asociadas: MID\$, RIGHT\$

## LEN

**LEN**((expresion literal))

```
A$="AMSTRAD":PRINT LEN(A$)
```

**FUNCION:** El resultado es el número de caracteres que forman el argumento de la función. Están incluidos todos los caracteres, incluyendo los espacios en blanco (ya que el argumento es una constante **literal**).

## LET

[LET](variable) = (expresion)

LET X=100

ACCION: Asigna a la variable mencionada la constante que resulte de evaluar la expresión colocada a la derecha del signo igual. La palabra reservada LET es un "recuerdo" del BASIC original, pero en el BASIC AMSTRAD es opcional y el ejemplo anterior puede simplemente escribirse como: X=100

## LINE INPUT

LINE INPUT [(#<numeral del cauce>),][;][constante literal entre comilladas;](variable literal)  
 LINE INPUT [(#<numeral del cauce>),][;][constante literal entre comilladas;](variable literal)

LINE INPUT A\$

LINE INPUT "NOMBRE";N\$

ACCION: Ingresa toda una LINEA desde el CAUCE indicado, y la IMPONE como valor de la variable literal mencionada en la instrucción. El cauce prescrito para omisiones es, como siempre, igual en todo que el comando INPUT, excepto en que todo lo que se IMPONGA es dado como valor de la variable, incluyendo los delimitadores como comas y espacios en blanco, antes del **retorno de carro**.

CLAVES asociadas: READ, INPUT, INKEY\$, INPUT\$

## LIST

LIST [(<gama de numeros de linea>)][(, #numeral del cauce)]

LIST 100-1000, #1

**ACCION:** El comando **LIST** hace que se **LISTE** a través del CAUCE especificado, las líneas de programa incluidas en la gama mencionada en el comando. El cauce #0 es el prescrito y corresponde a la pantalla del monitor; el cauce #8 corresponde a la impresora.

Puede suspenderse temporalmente el listado, pulsando una vez la tecla **ESC** y reanudarse pulsando la barra espaciadora. Una doble pulsación **ESC** permitirá **ESCAPARSE** definitivamente.

Además puede dirigirse el listado a través de los cauces previamente definidos en pantalla para que aparezcan en ventanas separadas y así servir de ayuda en la **depuración** de programas sin tener que sobre-escribir todo el área de pantalla.

Puede suprimirse la COTA superior de la gama para obtener el listado desde un número de línea hasta el final; o suprimirse la COTA inferior de la gama para obtener el listado desde el principio hasta un número de línea; vg.

**LIST-200 or LIST 30-**

## LOAD

**LOAD** (nombre de fichero)[, (expresion direccional)]

**LOAD "INVENT"**

**ACCION:** **RECUPERAR** un programa previamente **DEPOSITADO** en la cinta cassette, se usa este comando **LOAD** para hacer que se **CARGUE** en la memoria del ordenador, reemplazando cualquier programa que hubiera en ella, o si se usa el parámetro opcional con una **dirección determinada**, cargar un fichero **binario** en la memoria a partir de la dirección mencionada. Véase el capítulo 2.

## LOCATE

**LOCATE** [(#(numeral de cauce),](x coordenada), (y coordenada)

```
10 MODE 1
20 LOCATE 20,12
30 PRINT CHR$(249)
```

**ACCION:** El comando **LOCATE** hace que el ordenador **UBIQUE** el cursor de texto en la posición correspondiente a las coordenadas **absolutas** mencionadas en el comando. Si se especifica el numeral de cauce, el origen de coordenadas será el correspondiente a la esquina superior izquierda de la ventana asociada a dicho cauce. El cauce prescrito para omisiones es toda la pantalla del monitor.

**CLAVES asociadas:** **WINDOW**

## **LOG**

**LOG**((expresion numeral))

?**LOG**(9999)  
9.21024037

**FUNCION:** Calcula el logaritmo natural del argumento; y el resultado es una constante numérica real aunque el argumento sea entero.

**CLAVES asociadas:** **EXP, LOG10**

## **LOG10**

**LOG 10**((expresion numeral))

?**LOG10**(9999)  
3.99995657

**FUNCION:** Calcula el logaritmo en base 10 del argumento; y el resultado es una constante numérica real aunque el argumento sea entero.

**CLAVES asociadas:** **EXP, LOG**



## LOWER\$

**LOWER\$**((expresion literal))

```
A$="AMSTRAD":PRINT LOWER$(A$)
amstrad
```

**FUNCION** Convierte a letras minúsculas todas las letras mayúsculas que aparezcan en el argumento.

Se utiliza para uniformizar las posibles respuestas ingresadas por teclado.

CLAVE asociada: **UPPER\$**

## MAX

**MAX**((lista de:(expresion numeral))

```
10 n=66
20 PRINT MAX(1,n,3,6,4,3)
```

**FUNCION:** El resultado es el número con valor absoluto MAXIMO que aparezca en la lista argumento.

CLAVE asociada: **MIN**

## MEMORY

**MEMORY** (expresion direccional)

```
MEMORY &20AA
```

**ACCION:** Cambia la cantidad de memoria disponible para el programa BASIC, estableciendo como 'cima' de direcciones la que se menciona en el comando.

CLAVES asociadas: **HIMEM, FRE**

## MERGE

**MERGE** [(nombre de fichero)]

**MERGE** "PLAN"

**ACCION:** El comando **MERGE** hace que el ordenador CONGREGUE en memoria dos programas: el ya existente en memoria y el procedente del cassette que habrá de tener el nombre mencionado en el comando. Si no se menciona ninguno, el BASIC intentará CONGREGAR el primer fichero de la **clase** adecuada que encuentre en la cinta.

Si el primer carácter del nombre de fichero es el signo de admiración, prescindirá de él al buscar el fichero en la cinta y de los habituales mensajes que genera para facilitar el manejo del cassette.

Cuando desees añadir un programa en cinta al que ya tienes en memoria, sin que éste se vea sobre-escrito por el que añades, deberás **RENUMERAR** adecuadamente el programa que tienes en memoria para que sus números de línea no coincidan con el programa que añades.

Se cancelan todas las variables, funciones de usuario, y ficheros. Se cancelan también todas las definiciones de clases de variables, y se **arredran** todos los punteros internos. Además se dejan sin efecto todas las posibles pautas para casos de error.

Observa que no podrás CONGREGAR ficheros, si el procedente de la cinta no fue previamente GUARDADO utilizando la clase A (ficheros ASCII).

CLAVES asociadas: **LOAD**, **CHAIN** **MERGE**, **SAVE**

## MID\$

**MID\$**[(literal, (expresion entera) [(expresion entera)])]

```
A$="AMSTRAD":PRINT MID$(A$,2,4)
MSTR
A$="AMSTRAD":b$=MID$(A$,2,4):PRINTb$
MSTR
```

**FUNCION:** Como función, se usa **MID\$** para segregar los caracteres MEDIANEROS del argumento y formar por tanto, un sub-lítero de él. El primer parámetro entero que especifica la posición donde comienza la segregación; y el segundo parámetro entero la longitud del sub-lítero resultado de la función. (Si se omite, se toma hasta el final del argumento).

**ACCION:** Como comando e **instrucción**, permite reemplazar en el valor de una variable literal los caracteres MEDIANEROS. El primer parámetro entero especifica la posición del primer carácter a reemplazar, y el segundo parámetro entero el número de caracteres que se reemplazará.

CLAVES asociadas: (como función) **LEFT\$, RIGHT\$**

## MIN

**MIN**((lista de:(expresion numeral))

```
PRINT MIN(3, 6, 2.999, 8, 9)
      2.999
```

**FUNCION:** El resultado es el número con valor absoluto MINIMO que aparezca en la lista argumento.

CLAVE asociada: **MAX**

## MODE

**MODE** (expresion entera)

```
MODE 1
```

**ACCION:** Estipula como nuevo modo de pantalla el mencionado en el parámetro (0, 1 ó 2). Y deja la pantalla en **INK 0**, que puede no corresponder al tinte fijado para el papel. Todas las ventanas de texto y gráficos posibles, quedan restauradas, y los cursores de texto y gráfico se ubican en su posición base.

CLAVES asociadas: **WINDOW, ORIGIN**

## MOVE

**MOVE** (x coordenadas), (y coordenadas)

```
MOVE 34, 34
```

**ACCION:** MUEVE el cursor de gráficos a la posición cuyas coordenadas relativas se mencionan en el comando. YPOS y XPOS son las funciones correspondientes para conocer la posición corriente del cursor de gráficos.

**CLAVES asociadas:** MOVER, PLOT, PLOTR, DRAW, DRAWR, ORIGIN, TEST, TESTR, XPOS, YPOS

## **MOVER**

**MOVER** (incremento de x), (incremento de y)

**MOVER** 34, 34

**ACCION:** MUEVE el cursor de gráficos usando como coordenadas RELATIVAS a la posición anterior del cursor, los parámetros horizontal y vertical especificados en el comando.

**CLAVES asociadas:** MOVE, PLOT, PLOTR, DRAW, DRAWR, TEST, TESTR, XPOS, YPOS

## **NEW**

**NEW**

**NEW**

**ACCION:** Este comando VACIA la memoria dejándola preparada para un NUEVO programa. Las definiciones de teclas no se pierden, ni se limpia la pantalla. Es un comando del que no puedes recobrar ni el programa ni las variables existentes en memoria, por lo que sólo es un poco menos peligroso que el restaurado total del ordenador o el apagado. Usalo con cuidado.

## NEXT

**NEXT** [(lista de:(variable))]

**FOR** n=1 **TO** 1000:**NEXT**

**ACCION:** Delimita el final de un bucle reiterativo. Es decir, el comando **NEXT** obliga a que se efectúe OTRA ronda si las condiciones del bucle lo exigen, o que se pase a otra instrucción. Es más rápido cuando se omite mencionar el 'contador de rondas' del bucle. En el ejemplo anterior sería más lento poner

**NEXT** n

**CLAVES** asociadas: **FOR**

## ON GOSUB ON GOTO

**ON** (expresion entera) **GOSUB** (lista de:(numero de linea))

**ON** (expresion entera) **GOTO** (lista de:(numero de linea))

10 **ON** DAY **GOSUB** 100,200,300,400,500

10 **ON** RATE **GOTO** 1000,2000,3000,4000

**ACCION:** SEGUN el valor del parámetro entero mencionado después de **ON**, hará que el ordenador VAYA a ejecutar una de las instrucciones mencionadas después de **GOTO**. Si el resultado es 1, hará que en la lista aparezca en primer lugar, si es 2 la que aparezca en segundo lugar. En el ejemplo, SEGUN el valor de la variable DIA, así procederá: cuando DIA=1 irá a la 1000, cuando DIA=2 irá a la 2000.

La variante con **GOSUB** simplemente hace que VAYA Y venGA al párrafo que comienza en ese número de línea.

**CLAVES** asociadas: **GOTO**, **GOSUB**

## ON BREAK GOSUB

**ON BREAK GOSUB** (numero de linea)

```
10 ON BREAK GOSUB 40
20 PRINT "ejecutando el programa"
30 GOTO 20
40 CLS
50 PRINT "pulsando [ESC] dos veces llama
  la rutina GOSUB"
60 FOR t=1 TO 2000:NEXT
70 RETURN
```

ACCION: Cuando se produzca una **interrupción** en la ejecución del programa (vg. pulsando dos veces la tecla de **ESC**) **proseguirá** la ejecución del programa en la subrutina mencionada.

CLAVES asociadas: **ON BREAK STOP**, **RETURN**

## ON BREAK STOP

**ON BREAK STOP**

```
10 ON BREAK GOSUB 40
20 PRINT "ejecutando el programa"
30 GOTO 20
40 CLS
50 PRINT "pulsando [ESC] dos veces llama a
  la rutina GOSUB"
60 FOR t=1 TO 2000:NEXT
65 ON BREAK STOP
70 RETURN
```

ACCION: CUANDO se ve 'atrapado' en una **interrupción** y está por tanto ejecutando la subrutina pertinente, la instrucción **ON BREAK STOP**, desactiva el 'cepo', sin ningún otro efecto inmediato. En el programa anterior, que incluye **ON BREAK STOP**, el 'cepo' **ON BREAK GOSUB** operará solamente una vez.

CLAVES asociadas: **ON BREAK GOSUB**

## ON ERROR GOTO

**ON ERROR GOTO** (numero de línea)

```
10 ON ERROR GOTO 80
20 CLS
30 PRINT"si hay un error, prefiero"
40 PRINT"listar el programa"
50 PRINT"a si puedo ver donde me equivoco"
60 FOR t=1 TO 4000:NEXT
70 GOTO 200
80 CLS:PRINT"hay un error en la línea";ERL:PRINT
90 LIST
```

**ACCION:** CUANDO haya un error al estar ejecutándose un programa, hará que el ordenador VAYA al número de línea mencionado. En este ejemplo, en la línea 70 se producirá el error.

**CLAVES** asociadas: ERR, ERL, RESUME

## ON SQ GOSUB

**ON SQ ((canal)) GOSUB** (numero de línea)

```
ON SQ (2) GOSUB 2000
```

**ACCION:** CUANDO el sistema detecta que existe un hueco vacío en el 'chorro de notas' del canal de sonido mencionado en la instrucción, generará una señal de interrupción, que obligará al ordenador a desviarse a la subrutina mencionada en la instrucción. Los números de **canal** corresponden a:

- 1 para el canal A
- 2 para el canal B
- 4 para el canal C

**CLAVES** asociadas: SOUND, SQ

## OPENIN

**OPENIN** (nombre de fichero)

**100 OPENIN "¡INFORMACION"**

**ACCION:** ABRE como fichero de ENTRADA el mencionado después de **OPENIN**; y prepara en memoria el 'buzón de entrada' de la información procedente del cassette.

Si se menciona como carácter anterior del nombre de fichero el ?, se suprimirán los mensajes de ayuda al manejo del cassette y el programa extraerá del cassette el primer bloque de datos y lo pondrá a disposición del programa.

**CLAVES asociadas:** COSEIN, OPENOUT

## OPENOUT

**OPENOUT** (nombre de fichero)

**OPENOUT "¡DATOS"**

**ACCION:** ABRE como fichero de SALIDA el mencionado después de **OPENOUT**; y prepara en memoria el 'buzón de salida' de la información procedente del cassette.

Si se menciona como carácter anterior del nombre de fichero el ?, se suprimirán los mensajes de ayuda al manejo del cassette y el programa extraerá del cassette el primer bloque de datos y lo pondrá a disposición del programa.

Cada bloque de datos consta de 2Kbs, y no se transferirá a la cinta hasta que el buzón de salida esté lleno, o se **CIERRE** el fichero de SALIDA, usando la instrucción **CLOSEOUT**.

**NB:** El comando **NEW** también **VACIA** el buzón de salida, con la subsecuente pérdida de datos.

**CLAVES asociadas:** CLOSEOUT, OPENIN



## ORIGIN

**ORIGIN** (x), (y) [, (izquierda), (derecha), (arriba), (abajo)]

```
10 CLS:BORDER 13
20 ORIGIN 0, 0, 50, 590, 350, 50
30 DRAW 540, 350
40 GOTO 20
```

**ACCION:** Estipula el punto **ORIGIN** para el cursor de gráficos. Los parámetros opcionales de este comando fijan los límites de una nueva ventana de gráficos, que será operativa en todos los modos admitidos de pantalla, debido a la técnica de coordenadas empleadas.

El **ORIGIN** es habitualmente el punto de coordenadas 0,0 situado en la esquina inferior izquierda de la ventana, y las coordenadas aumentan verticalmente y hacia la derecha. En el Apéndice VI, presentamos retículas para poder planificar tus gráficos correctamente.

Si cualquiera de los corners de la ventana corresponde a una posición que se sale de la pantalla, se tomará automáticamente la posición "visible" más alejada en esa dirección.

**CLAVES** asociadas: **WINDOW**

## OUT

**OUT** (numero de portal), (expresion entera)

```
OUT &F8F4, 10
```

**ACCION:** Envía el valor del parámetro entero (que debe estar en la gama 0 a 255) al portal de **SALIDA** correspondiente a la dirección mencionada.

**CLAVES** asociadas: **INP, WAIT**

## PAPER

**PAPER** [(#(numeral de cauce),](tinta usada)

```
10 MODE 0
20 FOR p=0 TO 15
30 PAPER p:CLS
40 PEN 15-p
50 LOCATE 6,12:PRINT "PAPER"p
60 FOR t=1 TO 500: NEXT t
70 NEXT p
```

**ACCION:** Estipula el **TIN**Te correspondiente al **PAPEL** (fondo) de la pantalla. Cuando se exponen caracteres en la pantalla de textos, la posición del carácter se rellena con la **TINTA** correspondiente a **PAPER** antes de sobre-escribir el carácter con la tinta correspondiente a **PEN**, a no ser que se haya elegido el modo transparente.

El número de colores que puede aparecen en pantalla depende del modo elegido. Si el parámetro de tinta mencionado en el comando, no está disponible, se tomará el valor prescrito para omisiones.

**CLAVES** asociadas: **INK**, **WINDOW**, **PEN**

## PEEK

**PEEK**((expresion direccional))

```
10 MODE 2
20 INK 1,0: INK 0,12 : BORDER 12
30 INPUT "Direccion de inicio para su examen";inicio
40 INPUT "Direccion del final para su examen";final
50 FOR n= inicio TO final
60 VALUE$=HEX$(PEEK(n),2)
70 PRINT VALUE$;
80 PRINT" en ";HEX$(n),4),
90 NEXT
```

**FUNCION:** El resultado es en notación decimal, el contenido de la **celdilla** de memoria cuya dirección viene expresada por el argumento de la función. El ejemplo muestra un programa 'utensilio' que te permite **PINZAR** y observar en pantalla el contenido de la memoria de lectura y escritura del CPC464, junto con la correspondiente **dirección**.

**CLAVES** asociadas: **POKE**

## PEN

PEN [#(numeral del cauce),](tinta usada)

PEN 1,2

ACCION: Estipula la tinta a usar con la PLUMA cuando se usa una determinada ventana (correspondiente al cauce mencionado). Si no se menciona explícitamente un cauce, el comando afecta al cauce número 0, que como siempre, corresponde a toda la pantalla.

CLAVES asociadas: INK, PAPER

## PI

PI

PRINT PI

3.14159265

```

10 REM Dibujo en perspectiva
20 MODE 2
30 RAD
40 INK 1,0
50 INK 0,12
60 BORDER 9
70 FOR N= 1 TO 200
80 ORIGIN 420,0
90 DRAW 0,200
100 REM dibuja angulos en puntos de fuga
110 DRAW 30*N*SIN(N*PI/4), (SIN(PI/2))*N*SIN(N)
120 NEXT
130 MOVE 0,200
140 DRAWR 0,50
150 DRAWR 40,0
160 WINDOW 1,40,1,10
170 PRINT"Ahora puede terminar el programa del
    Ahorcado!"

```

FUNCION: Realmente es una **variable** numérica real intrínseca del ordenador. Representa la relación entre la circunferencia y el diámetro de un círculo, y el valor con el que se opera es de 3.141592653468251. Se usa ampliamente en las rutinas de gráficos tal como la presentada en el ejemplo.

CLAVES asociadas: DEG, RAD

## PLOT

**PLOT** (x coordenada), (y coordenada) [, (tinta usada)]

```
10 MODE 2:PRINT "Pulse 4 numeros, separados
  por comas":PRINT
20 PRINT "Pulse X origen (0-639),
  Y origen (0-399), radio y angulo
  del salto ":INPUT x,y,r,s
30 ORIGIN x,y
40 FOR angulo = 1 to 360 STEP s
50 XPOINT = r*cos(angulo)
60 YPOINT = r*cos(angulo)
70 PLOT XPOINT,YPOINT
74 REM MOVE 0,0
75 REM DRAW XPOINT,YPOINT
80 NEXT
```

**ACCION:** Es más rápido que ensayos con los parámetros 320, 200, 20, 1 como primera respuesta al ejemplo. Luego activa la línea 75 (quita REM) y desactiva la línea 70 (precedela de REM) para ver la diferencia.

Observa que en el primer caso se PINTA el contorno del círculo, y en el segundo se dibuja un círculo sólido. Recuerda también que en este programa se usan RADIANTES para las magnitudes angulares, (que es el modo prescrito como normal), por lo que el ángulo en cada paso es mucho mayor de 1 grado (aproximadamente 57 grados). Puedes teclear el comando 25 DEG para pasar a grados y ver el efecto.

**CLAVES asociadas:** DRAW, DRAWR, PLOT, PLOTR, MOVE, MOVER, ORIGIN, TEST, TESTR, XPOS, YPOS

## PLOTR

**PLOTR** , (x coordenada), (y coordenada) [, (tinta usada)]

```
10 MODE 2:PRINT "Pulse 4 numeros,
  separados por comas":PRINT
20 PRINT "Pulse X origen (0,369),
  Y origen (0-399), radio y angulo
  del salto":INPUT x,y,r,s
30 ORIGIN x,y
40 FOR angulo = 1 to 360 STEP s
50 XPOINT = r*COS(angulo)
60 YPOINT = r*COS(angulo)
70 PLOTR XPOINT, YPOINT
80 NEXT:GOTO 40
```

**ACCION:** Ensayá 320, 0, 20, 1 como respuestas. PLOTR es similar a DRAWR, pero solamente PINTA la mota cuyas coordenadas RELATIVAS se mencionan en el comando.

**CLAVES asociadas:** DRAW, DRAWR, PLOT, PLOTR, MOVE, MOVER, ORIGIN, TEST, TESTR, XPOS, YPOS

## POKE

**POKE** (expresion direccional), (expresion entera)

**POKE** &00FF, 10

**ACCION:** Permite acceder directamente la celdilla de memoria cuya dirección se menciona, y hacer que se PUNZE en ella el equivalente binario del parámetro entero que se menciona, que por tanto debe estar en la gama 0 a 255.

Usala con cuidado porque puedes afectar el comportamiento del programa de manera impredecible.

**CLAVE asociada:** PEEK

## POS

**POS**(#(numeral de cauce)

**PRINT POS**(#0)

1

**FUNCION:** El resultado es la posición corriente del cursor en el cauce mencionado como argumento. En este caso, no hay valor prescrito para omisiones del numeral de cauce, y además, si lo omites se provocará un mensaje de error sintáctico.

Cuando el cauce corresponde a la pantalla, el resultado refleja la coordenada X del cursor de texto, en relación con el costado izquierdo de la ventana correspondiente.

Cuando el cauce corresponde a impresora, el resultado refleja la posición del cabezal de impresión siendo el valor 1 el correspondiente al margen izquierdo. En la cuenta, sólo se incluyen los caracteres ASCII con códigos mayores de 32 (&1F).

Cuando el cauce corresponde al cassette, es lo mismo que para la impresora.

**CLAVES asociadas:** VPOS

## PRINT

**PRINT** [(#(numeral de cauce),)[(print lista)][(clausula USING)][(separador)]

**PRINT** #0, "abc"

ACCION: La explicación completa aparece en las páginas finales de este capítulo.

CLAVES asociadas: USING, TAB, SPC

## RAD

**RAD**

**RAD**

ACCION: Estipula que las magnitudes angulares se tomen en RADIANES.

CLAVES asociadas: DEG, SIN, COS, TAN, ATN

## RANDOMIZE

**RANDOMIZE** [(expresion numeral)]

10 RANDOMIZE 23  
20 PRINT RND (6)

ACCION: El BASIC tiene la habilidad de generar una secuencia pseudo-aleatoria (cada número depende del que le precede) utilizando como **gérmen** el parámetro mencionado en el comando ALEATORICE. Si se omite el parámetro, se requerirá al operador que imponga un valor por teclado, que ha de estar en la gama -32768 a 32768.

Si no se utiliza el comando **RANDOMIZE** en el programa, la secuencia generada es siempre la misma cada vez que se ejecute el programa. Si se usa la variante **RANDOMIZE TIME** el **gérmen** de la secuencia depende del TIEMPO transcurrido desde un determinado momento, con lo que resulta una aleatorización prácticamente completa.

CLAVES asociadas: RND

## READ

**READ** lista de:(variable)

```
10 FOR X=1 TO 4
20 READ N$
30 PRINT N$
40 DATA JOSE, LUIS, JAVIER, PEDRO
50 NEXT
```

**ACCION:** Con la instrucción **READ** se logra que el sistema **APUNTE** correlativamente como valor de las variables las constantes reseñadas en las instrucciones **DATA**. El sistema mantiene un 'puntero' interno para conocer las constantes que ya lleva apuntadas a las variables. Con la instrucción **RESTORE** se le obliga a que **ARREDRE** dicho puntero.

**CLAVES** asociadas: **DATA**, **RESTORE**

## RELEASE

**RELEASE** (canales de sonido)

**RELEASE** 4

**ACCION:** Cuando se envía a un canal de sonido una **NOTA**, se la puede marcar como **RETENIDA**. Con el comando **RELEASE**, se le queda **LIBERADA** la del canal o canales mencionados por el parámetro de dicho comando. El valor del parámetro está **calibrado** en el sistema binario, de la siguiente forma: A=bit 0, B=bit 1, C=bit 2. Por lo tanto, un valor de 4 en el parámetro (en binario 0100) afecta sólo al canal de sonido C.

**CLAVE** asociada: **SOUND**

## REM

**REM** (el resto de la línea)

```
10 REM Monstruo Hiperespacial Intergalactico
    a la caza de los Invasores por AMSOFT
20 REM Copyright AMSOFT 1984
```

**ACCION:** Colocando en una línea de programa la palabra reservada **REM** se le hace que reMEMOre todo lo que se escriba después y hasta el 'retorno de carro'; pero que no tenga ningún efecto sobre el programa. Puede sustituirse la palabra clave **REM** por el apóstrofe ' . Por supuesto, el apóstrofe no puede formar parte de ninguna constante literal.

## **REMAIN**

**REMAIN** ((expresion entera))

```
REMAIN (3)  
PRINT #6, REMAIN(0);
```

**FUNCION:** Desactiva el temporizador de demora especificado (en la gama 0 a 3).

Lee la cuenta que **QUEDA** en el temporizador de demora. Se devuelve 0 si el temporizador de demora no estaba activado.

**CLAVES** asociadas: **AFTER**, **EVERY**

## **RENUM**

**RENUM** [(nuevo numero de linea)][, [(anterior numero de linea)][, (incremento)]]

```
RENUM  
RENUM 100,, 100
```

**ACCION:** Renumera las líneas del programa a partir del número de línea antiguo, al que hace corresponder el número de línea nuevo, y en saltos dados por el tercero de los parámetros. Si se omite el parámetro de salto, renumera de 10 en 10. Si se omite el número de línea viejo, renumera a partir del primer número de línea del programa. Si se omite el número de línea nuevo, renumera empezando con el 10.

Durante la renumeración, el sistema se preocupa de cambiar los números **destinos** de saltos y desvíos. Los números de línea válidos para un programa deben estar en la gama 1 a 65535.

En el primero de los ejemplos en que se han omitido los tres parámetros, el resultado de la renumeración será un programa comenzando con el número de línea 10 e incrementando de 10 en 10 los números de línea subsiguientes.



## RESTORE

RESTORE [(*numero de línea*)]

RESTORE 300

```
10 FOR N=1 TO 6
20 READ A$
30 PRINT A$ " ";
40 DATA restaurada, puede, ser, releida
50 NEXT
60 PRINT
70 RESTORE
80 GOTO 10
```

ACCION: El sistema mantiene un 'puntero' interno señalando -en las series de constantes utilizadas en el programa- la última que ha apuntado como valor de una variable. Con el comando **RESTORE** se le obliga a que REPUNTE dicho puntero hacia la serie de constantes que corresponde al número de línea mencionado como parámetro. Si se omite ese número de línea se conseguirá que repunte a la primera constante de la primera serie de constantes que haya en el programa.

CLAVES asociadas: **READ, DATA**

## RESUME

RESUME[(*numero de línea*)]  
o RESUME NEXT

RESUME 300

ACCION: Cuando se ha colocado en el programa un 'cepo', mediante la instrucción **ON ERROR GOTO nnn**, y se ha 'atrapado' un error durante la ejecución; se puede incluir en el tratamiento subsecuente del error, el comando **RESUME** para hacer que se REANUDE la ejecución del programa, a partir del número de línea mencionado en la instrucción. En lugar del número de línea, también se puede reflejar la palabra clave **NEXT**, con lo que la reanudación se hará a partir de la instrucción SIGUIENTE a la que se produjo el error.

CLAVES asociadas: **ON ERROR GOTO**

## RETURN

**RETURN**

**RETURN**

**ACCION:** Indica la culminación de una subrutina, y hace que el sistema VUELVA a la instrucción inmediatamente siguiente a la que produjo el último desvío.

**CLAVÉS asociadas:** GOSUB, ON n GOSUB, ON SQ GOSUB, AFTER n GOSUB, EVERY n GOSUB, ON BREAK GOSUB

## RIGHT\$

**RIGHT\$ ((expresion literal), (expresion entera))**

```
10 CLS
20 A$ = "AMSTRAD"
30 B$ = RIGHT$(A$, 3)
40 PRINT B$
RUN
[ LIMPIA PANTALLA ]
RAD
Ready
```

**FUNCION:** Segrega del parámetro literal de la función tantos caracteres ULTERIORES como indique el parámetro entero de la función. El resultado es pues, el sublítero colocado en la parte derecha del argumento, y si dicho argumento tiene menos longitud que la exigida por el segundo argumento numérico, el resultado es exactamente igual al argumento.

**CLAVES asociadas:** MID\$, LEFT\$

## RND

**RND(((expresion numeral)))**

```
10 RANDOMIZE 23
20 PRINT RND(6)
```

**FUNCION:** Recaba un número aleatorio de la secuencia generada internamente, en el intervalo 0 a 1 y es el siguiente en secuencia, cuando el argumento se omite o es positivo; la repetición del último recabado cuando es 0; o siempre el primero cuando el argumento es negativo.

el comando **RANDOMIZE** del programa anterior, asegura que el resultado de **RND(6)** es siempre el mismo número cada vez que se ejecuta el programa (0.146940658), con lo que se destruirá el AZAR entre jugadas.

CLAVES asociadas: **RANDOMIZE**

## ROUND

**ROUND** ((expresion numeral)[, (expresion entera)])

```
10 x=0.123456789
20 FOR r=9 TO 0 STEP -1:PRINT r, ROUND(x, r):NEXT
25 x=123456789
30 FOR r=0 TO -9 STEP -1
40 PRINT r, ROUND (x, r)
50 NEXT
```

**FUNCION:** Entrega el valor del primer argumento numeral REDONDEADO a la potencia de 10 señalada por el segundo argumento de la función. Si el segundo argumento es negativo, el valor obtenido es un entero seguido por tantos ceros después del punto decimal como indique el segundo argumento.

CLAVES asociadas: **INT, FIX, CINT, ABS**

## RUN

**RUN** (expresion literal)

**RUN "BIENVENIDO"**

**ACCION:** Carga un programa desde el cassette en la memoria del ordenador, y comienza a EJECUTARLO inmediatamente. Si no se menciona nombre del programa (ie. se teclea **RUN "**) el sistema cargará y ejecutará el primer fichero que encuentre en la cinta. Si el carácter anterior de la expresión literal es ! se suprimirán los mensajes de ayuda para manipulación del cassette.

**NB:** Este comando implica que BASIC realice previamente un comando **NEW**, inmediatamente antes de comenzar a leer en la cinta.

**CLAVES asociadas: LOAD**

## **RUN**

**RUN[(numero de línea)]**

**RUN 100**

**ACCION:** Comienza a EJECUTAR el programa existente en memoria a partir de la línea mencionada; o a partir de la primera línea si no se menciona ninguna.

Con este comando, se cancelan todas las funciones de usuario estipuladas y se anulan todos los valores de las variables. Asimismo, se anulan todos los posibles convenios sobre nombres de variables; y se vacían todos los buzones de salida de los ficheros en cinta, con la consiguiente pérdida de información. Usa el comando **GOTO nn**, si no quieres perder esa información.

**CLAVES asociadas: LOAD, GOTO**

## **SAVE**

**SAVE (nombre de fichero)[, (tipo de fichero)][, (parametros binarios)]**

**SAVE "PROG" ,P**

**ACCION:** El comando **SAVE** hace que se **GUARDE** en la cinta el programa existente en memoria, con el nombre de fichero mencionado. Dependiendo del valor del último parámetro del comando, el programa guardado será un fichero de la **clase**:

ASCII	cuando el parámetro sea ,A
Protegido	cuando el parámetro sea ,P
Binario	cuando el parámetro sea ,B (usado para que se <b>GUARDE</b> la información contenida en ciertas partes de la memoria, v.g. la referente a pantalla).

**CLAVES asociadas: LOAD, RUN <nombre fichero>, MERGE, CHAIN, CHAIN MERGE**

## SGN

**SGN**((expresion numeral))

```
10 INPUT "Cual es tu Balance Bancario actual";CAJA
20 IF SGN(CAJA) <1 GOT 30 ELSE 40
30 PRINT "Horror, que mal":NED"
40 PRINT"Andas mejor que yo, eh!"
```

**FUNCION:** El valor entregado depende del **SIGNO** del argumento de la función; siendo -1 si el argumento es menor que 0 (negativo), siendo 0 si el argumento es igual a 0, y siendo 1 cuando el argumento es mayor que 0 (positivo).

**CLAVES asociadas:** ABS

## SIN

**SIN**((expresion numeral))

```
PRINT SIN(PI/2)
1
```

**FUNCION:** El resultado es siempre una constante numérica **real**, igual al **SENO** del argumento; que será considerado en **radianes**, a no ser que previa y explícitamente se haya declarado que son **GRADOS**.

**CLAVES asociadas:** COS, TAN, ATN, DEG, RAD

## SOUND

**SOUND**(estado de canal), (periodo de tono), (duracion) [, (volumen) [, (envolvente de volumen) [, (envolvente de tono) [, (periodo de nudo)]]]]]

```
SOUND 1, 200, 1000, 7, 0, 0, 1
```

ACCION: Las facetas del SONIDO del CPC464 es una de las características más potentes y complejas del CPC464, y se explica detalladamente en el capítulo 6.

CLAVES asociadas: ENV, ENT

## SPACE\$

SPACE\$(expresion entera)

SPACE\$(190)

FUNCION: El resultado es una constante literal formada exclusivamente por el número de ESPACIOS en blanco expresado por el valor del argumento.

CLAVES asociadas: PRINT, SPC, TAB

## SPEED INK

SPEED INK (expresion entera), (expresion entera)

```
5 INK 0,9,12:INK 1,0,26
10 BORDER 12,9
20 SPEED INK 50,20
```

ACCION: Se puede hacer que la tinta del papel o de la pluma, y el borde de la pantalla parpadee entre dos posibles colores asociados al número de tinta: el comando **SPEED INK** permite variar la CADENCIA de parpadeo entre esos dos colores. El primer parámetro entero especifica el tiempo correspondiente al primer valor de tinta, y el segundo parámetro el correspondiente al segundo valor de tinta. Los tiempos se miden en unidades de 2 centésimas de segundo (con suministro a 50Hz).

Debes ejercer cuidadosamente tus apreciaciones, para evitar efectos de 'perceptibilidad mesmérica' al elegir los colores y cadencias de repetición.

CLAVES asociadas: INK, BORDER

## SPEED KEY

**SPEED KEY** (demora de comienzo), (periodo de repeticion)

**SPEED KEY 20,3**

**ACCION:** Si se mantienen pulsadas un determinado tiempo, las teclas del CPC464 repetirán automáticamente el envío de las señales pertinentes con el período designado por el primer parámetro y con la demora en el arranque designada en el segundo parámetro. El ajuste se hace en incrementos de 0,02 segundos y ambos parámetros deben estar en la gama 1 a 255. Los valores prescritos para omisiones son 10 y 10 unidades de tiempo respectivamente.

Las demoras en el arranque que tengan valores muy pequeños pueden afectar a las rutinas 'quita rebotes' de las teclas. La velocidad de exploración del teclado que internamente se efectúa no se ve afectada por este comando. No todas las teclas son **tipomáticas**, y cuando el usuario redefine el significado de una tecla, puede establecer este atributo de tipomatismo para esa tecla.

**CLAVES** asociadas: **KEY DEF**

## SPEED WRITE

**SPEED WRITE** (expresion entera)

**SPEED WRITE 1**

**ACCION:** En el sistema se pueden efectuar las transferencias de información entre el ordenador y el cassette eligiendo entre dos **velocidades**, y con el comando **SPEED WRITE**, se fija la **VELOCIDAD** de **ESCRITURA** en el cassette: si el valor del parámetro es 1 la velocidad elegida es 2000 baudios, y si es 0 la de 1000 baudios.

Cuando lee información de la cinta, el CPC464 establece automáticamente la velocidad de lectura al valor correspondiente a la que se usó al escribirla; por lo que no es elegible por el usuario la velocidad de 'lectado'.

Cuando se usan cintas magnéticas de dudosa calidad de grabación y lectura, se recomienda usar la velocidad de 1000 baudios para una fiabilidad máxima (véanse las notas del capítulo 2 para más información).

CLAVES asociadas: SAVE

## SQ

**SQ((canal))**

```
10 MODE 1
20 FOR n=20 TO 0 STEP-1
30 PRINT n;
40 SOUND 1,10+n,100,7
50 WHILE SQ(1)>127:WEND
60 NEXT
```

**FUNCION:** La función **SQ** se utiliza para examinar el ESTADO DEL CANAL de sonido correspondiente al argumento; y el valor resultante está calibrado en binario de la siguiente manera. Los bits 0, 1, 2 indican el número de 'huecos' existentes en el CHORRO DE NOTAS pendientes de ser emitidas por el altavoz.

Los bits 3, 4 y 5 indican si la primera de las notas del chorro (si hay realmente chorro) está marcada como ACORDE con algún otro canal de sonido.

El bit 6 indica si la citada nota de cabecera está RETENIDA.

El bit 7 si el canal correspondiente está corrientemente ACTIVADO.

El argumento de la función también está calibrado en binario, con el bit 1 para el canal A, el bit 2 para el canal B y el bit 3 para el canal C.

Por tanto, un valor decimal de 5 hace que el comando afecte a los canales A y C.

CLAVES asociadas: ON SQ GOSUB

## SQR

**SQR((expresion numeral))**

```
PRINT SQR(9)
3
```

**FUNCION:** Entrega como resultado la raíz cuadrada del argumento (que ha de tener un valor positivo).

CLAVES asociadas: PRINT



## STOP

STOP

```
300 IF n<0 THEN STOP
```

ACCION: El comando **STOP** hace que se pare la ejecución del programa, pero que permanezca en un estado tal que pueda hacerse que **SIGA** usando el comando **CONT**. Suele usarse como instrucción dentro del programa en determinados puntos durante la **depuración** del programa y luego se suprime.

CLAVES asociadas: **CONT**, **END**

## STR\$

STR\$(*expresion numeral*)

```
PRINT STR$(&766)
1984
```

```
PRINT STR$(&1010100)
84
```

FUNCION: Convierte el argumento **numeral** a la constante **literal** equivalente. Como en los ejemplos, puede usarse para hallar los equivalentes en el sistema decimal de números en notación binaria o hexadecimal; pero ten siempre presente que el resultado es de INDOLE LITERICA.

CLAVES asociadas: **VAL**, **PRINT**, **HEX\$**, **BIN\$**

## STRING\$

STRING\$(*expresion entera*), (*especificador del caracter*)

```
PRINT STRING$(&16, "*")
*****
```

FUNCION: Entrega como resultado una constante **LITERAL** formada por tantos caracteres como especifique el primer argumento entero, y del carácter asignado en el segundo argumento.

CLAVES asociadas: SPACE\$

## SYMBOL

SYMBOL (numero de caracter), (lista de:(filas)

```
5 MODE 2
10 SYMBOL AFTER 90
20 SYMBOL 93, &80, &40, &10, &8, &4, &2, &1
30 FOR n=1 TO 2000
40 PRINT CHR$(93);
50 NEXT
60 GOTO 60
```

ACCION: Permite redefinir el **tipo**, los ocho **trazos** con que se expone en pantalla. Del SIMBOLO cuyo código CPC464 es el designado por el primer parámetro del comando; sustituyéndolo por el resultante de utilizar los 8 parámetros de la lista subsiguiente.

Para preparar la lista es conveniente usar una cuadrícula de 8 x 8, y SI rellenar o NO rellenar cada uno de los 64 cuadritos de acuerdo con el tipo o figura que desees; y considerar cada una de las 8 filas como un número según el sistema **binario** de numeración, y expresarlo en el comando por el equivalente en el sistema que desees. En pantalla, el sistema mostrará una mota con el tinte fijado para el papel cuando haya un 1 (SI), y una mota con la tinta fijada para la pluma cuando haya un 0 (NO).

En el ejemplo, el tipo definido es una pequeña raya diagonal, y será accesible por teclado usando la tecla marcada ] cuyo código CPC464 y ASCII es 93.

CLAVES asociadas: SYMBOL AFTER

## SYMBOL AFTER

SYMBOL AFTER (expresion entera)

```
SYMBOL AFTER 90
```

**ACCION:** DESPUES de este comando queda fijado el número de caracteres definibles por el usuario a un valor dependiente del parámetro del comando. Si se omite, se considera que el parámetro es 240, dejando por tanto 16 caracteres como definibles por el usuario. Si el valor de parámetro es 32 todos los caracteres con códigos CPC464 incluidos en la gama 32 a 255, se convierten en redefinibles por el usuario. Siempre que el sistema ejecuta este comando, todos los caracteres que tenga previamente definidos el usuario, se restauran al tipo prescrito en el sistema.

CLAVES asociadas: SYMBOL

## TAG

TAG[#(numeral de cauce)]

```
10 MODE 2
11 BORDER 9
14 INK 0,12
15 INK 1,0
20 FOR n=1 TO 100
30 MOVE 200+n,320+n
40 TAG
50 IF n<70 GOTO 60 ELSE 70
60 PRINT"Hola";:GOTO 80
70 PRINT" Adios";
80 NEXT
90 GOTO 20
```

**ACCION:** El texto enviado hacia un cauce dado, puede ser redireccionado para que se escriba en la posición de cursor de gráficos. Esto permite que se mezclen textos y símbolos con gráficos. El <numeral del cauce> está prescrito a #0 si se omite el parámetro. El trazo superior de la celdilla del carácter es 'pegado como una etiqueta' al cursor de gráficos, y aparecerán caracteres de control no visivos (eg. teclear nueva línea), si la instrucción PRINT no va seguida de un punto y coma ; .

CLAVES asociadas: TAGOFF

## TAGOFF

TAGOFF[#(numeral de cauce)]

TAGOFF #0

## BASIC: Palabras clave

ACCION: Cancela el TAG para un cauce dado, y envía el texto a la posición previa del cursor de textos en el punto en que se invocó TAG.

CLAVES asociadas: TAG

## TAN

TAN(*expresion numeral*)

PRINT TAN(45)

FUNCION: Calcula la tangente del ángulo dado por el argumento de la función, que debe estar en la gama -200.000 a +200.000, con la medida en radianes prescrita para omisiones.

CLAVES asociadas: COS, SIN, ATN, DEG, RAD

## TEST

TEST(*x coordenada*, *y coordenada*)

PRINT TEST(300, 300)

FUNCION: Examina y da como resultado el valor de la tinta especificada en ese momento para esa posición de la pantalla de gráficos dada por las coordenadas ABSOLUTAS.

CLAVES asociadas: TESTR, MOVE, MOVR, PLOT, PLOTR, DRAW, DRAWR

## TESTR

TESTR(*incremento de x*, *incremento de y*)

TESTR(5, 5)

FUNCION: Examina y da como resultado el valor de la tinta en la posición de la pantalla de textos, cuyas coordenadas RELATIVAS a la posición corriente del cursor de gráficos corresponde con los argumentos de la función.

CLAVES asociadas: TEST, MOVE, MOVER, PLOT, PLOT, DRAW, DRAW

## TIME

TIME

```
10 DATUM = INT(TIME/300)
20 TICKER=((TIME/300)-DATUM)
30 PRINT TICKER;
40 GOTO 20
```

FUNCION: Entrega como resultado el TIEMPO transcurrido desde la puesta en marcha del ordenador, descubriendo aquellos períodos de tiempo en que se efectuaran transferencias de datos con la cinta cassette. La unidad de medida del tiempo es de 1/300 de segundo.

## TRON TROFF

TRON

TROFF

TRON

ACCION: BASIC incluye esta facilidad para RASTREAR la ejecución de un programa; y hace que se PONGA en RASTREO con el comando TRON y que se QUITA el RASTREO con el comando TROFF. Con esta operación de rastreo los números de línea del programa que está ejecutándose, aparecen en pantalla entre corchetes cuadrados [ ], justamente antes de ser obedecidas las instrucciones de esa línea.

CLAVES asociadas: RUN

## UNT

UNT((expresion direccional))

PRINT UNT(&FF66)

**FUNCION:** El argumento ha de corresponder a una de las posibles **direcciones** de las 'celdillas de memoria'; ie. un numeral en la gama 0 a 65535 cuyo equivalente en el sistema BINARIO NATURAL (**sin signo**) tenga una representación de 16 **bits**. El resultado de la función es el número entero equivalente en la gama -32768 a +32767.

CLAVES asociadas: INT, FIX, CINT, ROUND

## UPPER\$

UPPER\$(*expresion literal*)

```
PRINT UPPER$("amstrad")
AMSTRAD
```

**FUNCION:** El resultado es igual al argumento literal sustituyendo todas las letras minúsculas que aparecen en el argumento por letras mayúsculas.

CLAVES asociadas: LOWER\$

## VAL

VAL(*expresion literal*)

```
10 A$="7 es mi numero de la suerte"
20 PRINT VAL(A$)
```

**FUNCION:** Extrae el posible dato numérico que haya en el argumento literal; y que para ser efectivo debe estar colocado al principio del dato literal. Se usa en la mayoría de los casos para convertir un dato **literal** en un dato de índole **numeral**.

CLAVES asociadas: STR\$

## VPOS

VPOS(**#**(numeral de cauce))

PRINT VPOS(**#**0)

FUNCION: El resultado refleja la posición VERTICAL del cursor de textos correspondiente al numeral de cauce argumento de la función; que es IMPRESCINDIBLE.

CLAVES asociadas: POS

## WAIT

WAIT (numero de portal), (mascara)[, (inversion)]

WAIT &FF34, 20, 25

ACCION: Suspende cualquier otra operación haciendo que el sistema ESPERE hasta que el **portal** mencionado en el comando, haya un valor particular dentro de la gama 0 a 255: en realidad, este comando obliga al BASIC a que entre en un bucle -leerá el valor presente en el portal mencionado, lo OLEará (operación XOR) por el parámetro **inversión**, y el resultado intermedio que obtenga lo YLlará (operación AND) por el parámetro **máscara**; y continuará repitiendo las operaciones del bucle HASTA que el resultado final sea distinto de cero.

Si tecleas el ejemplo anterior, no tendrás más remedio que restaurar el ordenador para poder escapar del bucle.

CLAVES asociadas: INP, OUT

## WEND

### WEND

```

10 MODE 1:REM BASIC CLOCK TIME ROUTINE
20 INPUT "Pulse la hora actual, minutos, y
    segundos (h,m,s)";horas,minutos,segundos
30 CLS:datum = INT(TIME/300)
40 WHILE horas<13
50 WHILE minutos<60
60 WHILE tick<60
70 tick=(INT(TIME/300)-datum)+segundos
80 LOCATE 70,4
90 PRINT #0,USING "## ";horas,minutos,tick
100 WEND
110 tick=0
115 segundos=0
120 minutos=minutos+1
130 GOTO 30
140 WEND
150 minutos=0
160 horas=horas+1
170 WEND
180 horas=1
190 GOTO 40

```

ACCION: El bucle **WHILE/WEND** ejecuta reiteradamente una sección de programa MIENTRAS que una determinada condición es cierta. Este ejemplo usa un bucle **WHILE/WEND** para mostrar la elegancia de los programas construidos según este enfoque. Ahora puedes añadir todo el conjunto de facetas de un surtido de relojes diferentes. El comando **WEND** termina el bucle **WHILE**.

CLAVES asociadas: **WHILE**

## WHILE

**WHILE**(expresion logica)

**WHILE** DAY <0

ACCION: Un comando **WHILE** hace que se ejecute repetidamente una sección de programa MIENTRAS que sea cierta la condición mencionada en el comando. El comando **WHILE** señala el inicio del bucle, y establece la condición. El comando **WEND** culmina el bucle **WHILE**.

CLAVES asociadas: **WEND**



## WIDTH

**WIDTH** (expresion entera)

**WIDTH 86**

**ACCION:** Le dice al BASIC la ANCHURA de la impresora medida en posiciones de impresión; con lo que el BASIC insertará oportunamente los retornos de carro necesarios.

**CLAVES asociadas:** PRINT, POS

## WINDOW

**WINDOW** [(#(numeral de cauce),](izquierda),(derecha),(arriba),(abajo)

```
10 MODE 1
20 BORDER 6
30 WINDOW 10,30,7,18
40 PAPER 2:PEN 3
50 CLS
60 PRINT CHR$(143);CHR$(242);"ESTA ES LA LOCALIZACION"
70 PRINT "1,1 EN VENTANA DE TEXTO"
80 GOTO 80
```

**ACCION:** Asocia una VENTANA DE TEXTO al cauce mencionado, y marca los límites de la ventana.

**CLAVES asociadas:** ORIGIN

## WINDOW SWAP

**WINDOW SWAP** (numeral de cauce),(numeral de cauce)

**WINDOW SWAP 0,2**

**ACCION:** Produce un CANJE de ventanas. Por ejemplo, los mensajes de BASIC enviados por el cauce #0 puede intercambiarse con otra de las ventanas definidas para resaltar ciertos aspectos durante el desarrollo de los programas y la operación.

**CLAVES asociadas:** WINDOW, PEN, PAPER, TAG

## **WRITE**

**WRITE** [(#(numeral de cauce),)][(lista escrita)]

**WRITE** #2, "HOLA", 4, 5  
"HOLA", 4, 5

**ACCION:** A través del cauce mencionado, **ESCRIBE** la lista de datos mencionada en la instrucción, escribiendo exactamente los separadores y comillas que aparecen en la lista.

**CLAVES asociadas:** PRINT

## **XPOS**

**XPOS**

**PRINT XPOS**

**FUNCION:** Refleja la posición **HORIZONTAL** del cursor de gráficos.

**CLAVES asociadas:** YPOS, MOVE, MOVER, ORIGIN

## **YPOS**

**YPOS**

**PRINT YPOS**

**FUNCION:** Refleja la posición **VERTICAL** del cursor de gráficos.

**CLAVES asociadas:** XPOS, MOVE, MOVER, ORIGIN

## ZONE

**ZONE** (expresion entera)

```
10 PRINT 1,2,3  
20 ZONE 19  
30 PRINT 4,5,6
```

**ACCION:** Cambia la anchura de las ZONAS prescritas en pantalla y aprovechadas mediante el signo , al exponer datos en pantalla; cambiando el valor prescrito al arranque de 13 posiciones de anchura de zona, al valor mencionado en el comando, dentro de la gama 1 a 255.

Se restauran las condiciones iniciales con los comandos **NEW**, **LOAD**, **CHAIN** y **RUN** "nombre de fichero".

**CLAVES** asociadas: **PRINT**, **WIDTH**

## PRINT

**PRINT**[(#(numeral de cauce),)[(lista de datos)][(cláusula mascara)][(separador)]

(lista de datos) es (expresion)[(separador) (dato a exponer)]\*

(dato a exponer) es (expresion)

o bien

**SPC**((expresion entera))

**TAB**((expresion entera))

Para sacar información por el cauce de grabación en cassette.

```
10 OPENOUT "DATA"
20 PRINT #9, "Hola"
30 CLOSEOUT
```

Para sacar información por el cauce de impresión

```
10 BOBSSALARY =23*PI
20 PRINT #8, USING "####.##";BOBSSALARY
30 PRINT #0, BOBSSALARY
RUN
72256.6311
Ready
```

[Mientras, en la impresora.....]

```
72256.63
```

ACCION: El comando **PRINT** hace que el sistema EXPONGA los datos a través del CAUCE mencionado.

Cuando se usan **separadores** el BASIC los expone en formato 'libre', donde una coma a continuación del dato a exponer hará que se coloque al comienzo de la siguiente ZONA; mientras que un punto y coma los pondrá a continuación del dato inmediatamente anterior. (Si el dato es numérico sacará un blanco).

La cláusula **USING** permite especificar una 'horma' o 'máscara' que **conforme** la apariencia de los datos expuestos; y ha de ser un literal, variable o constante, en el que sólo aparezcan los símbolos especificados en la tabla anexa.

La cláusula **SPC (n)** hace que se inserten n ESPACIOS en blanco antes de exponer el siguiente dato, y siendo n un numeral entero dentro de la gama 0 a 255. (Si es negativo se toma el valor 0), y si es más ancho que la anchura del periférico correspondiente (pantalla, impresora, etc.), se reduce al correspondiente valor congruente con dicha anchura. Observa que no tiene que delimitarse esta cláusula ni con coma ni con punto y coma; se supone en todo momento que hay un punto y coma.

La cláusula TAB (nn) desplaza el cursor hasta la posición horizontal indicada por nn, que ha de estar en la gama 0 a 255. (Si es negativo se supone la posición 1, y si sobrepasa la anchura del periférico pertinente se reduce al número congruente con dicha anchura.

Si la posición nn exigida es igual o sobrepasa la posición corriente del mecanismo expositor (cursor, cabezal, etc.), se insertan los espacios convenientes para situarlo en la posición exigida. Pero si es menor que la posición presente, se enviará un 'retorno de carro' seguido por los espacios convenientes para situarlo en la posición exigida, con lo que aparecerá en la siguiente línea. Tampoco es necesario terminar esta cláusula con un separador, ya que se supone en todo momento que hay un punto y coma.

## USING "&lt;Literal conformador&gt;"

## Símbolos para datos numéricos

Especificador	Dígitos Posibles	Caracteres en campos	Definición	Ejemplo
#	1	1	Campo numérico	####
.	0	1	Punto decimal	##.
+	0	1	Expone signo delantero o trasero.	###
			Un número positivo tendrá +	###+
			<b>Los números negativos no pueden tener - delante</b>	
-	0	1	Signo trasero. Expone - si negativo, si no blanco	##.##-
**	2	2	Asteriscos delante	**#####
\$	1	2	Signo dólar flotante colocado antes del dígito delantero	\$#####
**\$	2	3	Relleno de asteriscos y dólar flotante	**\$###
,	1	1	Usa coma cada tres dígitos (sólo a la izquierda del punto decimal)	##,###,##
↑↑↑↑	0	4	Formato exponencial. Se ajusta el número de modo que el dígito delantero sea no-cero.	##,##↑↑↑↑

## Para datos literales

!			Sólo el primer carácter.	!
/ /			Campo de longitud constante e igual a nn+2	
&			Campo de longitud variable	&

## 9. Información de programación avanzada

### Temas tratados en este capítulo

- \* Posiciones 'legales' de texto
- \* Caracteres de control
- \* El sistema operativo
- \* Estructura de interrupciones

### 9.1 Ubicación del cursor y códigos de control

En diversidad de programas de aplicación, el cursor de textos puede situarse fuera de la ventana vigente en pantalla. Diferentes operaciones fuerzan a que el cursor se sitúe en una posición 'legal' antes de que puedan ser llevadas a cabo; y son:

- la exposición de un carácter
- mostrando el cursor de texto (el 'rectangulito')
- obedecer los códigos de control señalados con un asterísco en la lista presentada en las páginas siguientes.

El método para forzar al cursor a situarse en una posición legal, es el siguiente:

- a. Si el cursor está a la derecha del margen derecho, se desplaza a la columna extremo-izquierda de la línea inmediatamente inferior.
- b. Si el cursor está a la izquierda del margen izquierdo, se desplaza a la columna extremo-derecha de la línea inmediatamente superior.
- c. Si el cursor está por encima del margen horizontal superior, la ventana se desplaza (se despliega) una línea hacia abajo, y el cursor se sitúa en la primera línea superior de la ventana.
- d. Si el cursor está debajo del margen inferior, la ventana se desplaza (se despliega) una línea hacia arriba, y el cursor se sitúa en la línea más inferior de la ventana.

Las comprobaciones y operaciones se efectúan además en el orden en que la hemos mencionado. Las posiciones ilegales del cursor puede ser 0 o negativas cuando está situado fuera de la ventana por la izquierda o por arriba.

Los códigos ASCII decimales de los caracteres (Apéndice III) que caen dentro de la gama 0 a 31, no muestran ningún símbolo al ser enviados a la pantalla de textos (y pueden hacer que el sistema operativo se quede 'colgado' si no se usan juiciosamente) sino que son interpretados como CODIGOS DE CONTROL (y llamados también 'caracteres motivos' para distinguirlos de los 'visivos').

Algunos de ellos, y especialmente el de código decimal 27, hace que los caracteres enviados a continuación ESCAPEN del convenio habitual de significados. Y pueden usarse como **parámetros** de la acción ejercida por ese 'código de control'.

Cuando se envían a la pantalla de gráficos estos caracteres de control, mostrarán un símbolo relacionado con su función cuando son impuestos por medio del teclado (e.g. CTRL y G), pero solamente ejercerán el control que tienen asociado, si son enviados usando directamente el comando correspondiente (v.g. **PRINT CHR\$(&07)**). (Véase la descripción de la palabra reservada **TAG** en el capítulo 8).

Los códigos señalados con un asterísco en la lista siguiente, fuerzan al cursor a situarse en una posición legal en la corriente ventana, antes de que sean obedecidos; pero pueden dejar el cursor en una posición ilegal. Los códigos y su significado se describen primero por su valor en notación hexadecimal (&XX) y luego por el equivalente decimal.

**Comandos adicionales mediante códigos de control: no accesibles generalmente vía el teclado, usando la tecla CTRL.**

Código	Nombre	Parámetro	Significado
&00 0	NUL		Ningún efecto. Se ignora.
&01 1	SOH	0..255	Expone el símbolo correspondiente al valor del parámetro. Permite mostrar los símbolos dentro de la gama 0 a 31.
&02 2	STX		Vuelve invisible el cursor de texto.
&03 3	ETX		Vuelve visible el cursor de texto. Retiene internamente la exposición del cursor, y sólo lo libera cuando está esperando un ingreso por teclado.
&04 4	EOT	0..2	Fija modo de pantalla, congruente con el parámetro MOD 4. Equivale al comando <b>MODE</b> .
&05 5	ENQ	0..255	Envía el carácter correspondiente al parámetro a la posición del cursor de gráficos.
&06 6	ACK		Activa una pantalla de texto (véase &15, NAK).
&07 7	BEL		Produce pitido. Observa que con esto se cierran y vacían los canales de sonido.



Código	Nombre	Parámetro	Significado
&08 8 *	BS		Retrocede el cursor una posición.
&09 9 *	TAB		Avanza el cursor una posición.
&0A 10 *	LF		Desplaza el cursor avanzando una línea.
&0B 11 *	VT		Desplaza el cursor una línea hacia arriba.
&0C 12 *	FF		Limpia la ventana de texto y coloca el cursor en la esquina superior izquierda. Equivalente al comando CLS.
&0D 13 *	CR		Retorno de carro. Desplaza el cursor al borde izquierdo de la ventana dentro de la línea vigente.
&0E 14 *	SO	0..15	Fija el tinte del papel, al valor del parámetro MOD 16. Equivalente al comando PAPER.
&0F 15	SI	0..15	Fija la tinta de la pluma, al valor del parámetro MOD 16. Equivalente al comando PEN.
&10 16 *	DLE		Suprime el carácter corriente, y rellena esa posición con el valor corriente del tinte del papel.
&11 17 *	DC1		Limpia desde el borde izquierdo de la ventana hasta -e incluyendo- la posición corriente del cursor. Las posiciones afectadas se rellenan con el valor corriente del tinte del papel.
&12 18 *	DC2		Limpia desde -e incluyendo- la posición corriente del cursor hasta el borde derecho de la ventana. Las posiciones afectadas se rellenan con el valor corriente para el tinte del papel.
&13 19 *	DC3		Limpia desde el comienzo de la pantalla hasta -e incluyendo- la posición corriente del cursor. Las posiciones afectadas se rellenan con el valor corriente para el tinte del papel.

Código	Nombre	Parámetro	Significado
&14 20 *	DC4		Limpia desde -e incluyendo- la posición corriente del cursor hasta el fin de la ventana. Las posiciones afectadas se rellenan con el valor corriente para el tinte del papel.
&15 21	NAK		Desactiva la pantalla de texto. Por tanto, no reaccionará a nada de lo que se le envía hasta después de recibir el código ACK (&06).
&16 22	SYN	0..1	Parámetro MOD 2 activa(1)/desactiva(0) la opción de transparencia.
&17 23	ETB	0..3	Parámetro MOD 4 fija la modalidad de gráficos: 1 ' XOR''' 2 ' AND''' 3 ' OR'''
&18 24	CAN		Intercambia la tinta del papel y pluma.
&19 25	EM	0..255 0..255 0..255 0..255 0..255 0..255 0..255 0..255 0..255	Véase la forma de definir TIPOS de caracteres por el usuario. Equivale al comando <b>SYMBOL</b> . Exige 9 parámetros, el primer parámetro identifica el número del carácter que se redefine. Los siguientes 8 parámetros especifican cada uno de los 8 TRAZOS del carácter al exponerlo en pantalla. Y se comienza a partir de la esquina superior izquierda, desplazándose hacia la derecha hasta completar un trazo, y luego continuando con los trazos inferiores, hasta terminar el último trazo en la esquina inferior derecha del carácter.
&1A 26	SUB	1..80 1..80 1..25 1..25	Define y sitúa una ventana en la pantalla. Equivale al comando <b>WINDOW</b> . Los dos primeros parámetros especifican el costado izquierdo y derecho de la ventana, correspondiendo el valor más pequeño al costado izquierdo, y el mayor al derecho.

Código	Nombre	Parámetro	Significado
			Los segundos dos parámetros especifican el largero superior e inferior de la ventana -el valor más pequeño se toma como largero superior, el mayor como largero inferior.
&1B 27	ESC		Ningún efecto. Se ignora.
&1C 28	FS	0..15 0..31 0..31	Elige como tinta una pareja de colores. Equivalente al comando <b>INK</b> . El primer parámetro (MOD 16) especifica el número de tinta. Los siguientes dos (MOD 32) los colores elegidos.
&1D 29	GS	0..31 0..31	Elige para el borde un par de colores. Equivalente al comando <b>BORDER</b> . Los dos parámetros (MOD 32) especifican los dos colores.
&1E 30	RS		Desplaza el cursor a la esquina superior izquierda de la ventana.
&1F 31	US	1..80 1..25	Ubica el cursor en una determinada posición de la ventana corriente. Equivalente al comando <b>LOCATE</b> . El primer parámetro señala la columna; el segundo la línea.

## 9.2 El sistema operativo

Las 'labores domésticas' del CPC464 son llevadas a cabo por un conjunto de programas organizados y grabados permanentemente en la memoria, que se conocen con el nombre de **sistema operativo** del ordenador. El sistema operativo es como un 'director de orquesta', o más bien debe verse como un 'guardia de tráfico' entre los diversos circuitos del ordenador, incluyendo las entradas y salidas.

Primordialmente, es el intermediario entre los circuitos electrónicos y el interpretador BASIC. Así por ejemplo, cuando en tu programa pides que se ponga en marcha el parpadeo de texto, el BASIC interpreta tus instrucciones y pasa los parámetros correspondientes al sistema operativo que se encarga realmente de la tarea. Por lo tanto, el primero -el BASIC- determina lo que se requiere, y el segundo -el sistema operativo- lleva a cabo la realización de las acciones subsecuentes.

El sistema operativo de las máquinas modernas se conoce también, generalmente, con el término inglés de '**firmware**', para indicar que está a medio camino (firme) entre el **hardware** (duro) y el **software** (blando). Y está constituido por las rutinas que escritas en código máquina y grabadas de manera permanente e indeleble en la memoria, son reclamadas por el interpretador BASIC para que lleven a cabo las acciones conducentes a la cumplimentación de las instrucciones y comandos que le has mandado.

Aunque es prácticamente (casi) imposible que se quede 'colgado' el ordenador CPC464 cuando estás ejecutando un programa escrito en BASIC, a no ser que utilices mal el comando **ON BREAK GOSUB**, sí es relativamente fácil que causes un 'gripado de meninges' que sólo pueda resolverse mediante una restauración completa de las condiciones iniciales, con la subsecuente pérdida de todo el programa, datos, etc., si te embarcas en la excitante aventura del código máquina. (Si no quieres emplear esos términos tan rebuscados, dí simplemente que son **CIRCUITOS** electrónicos donde radica toda la "mollera congénita" de la máquina).

Si estás tentado de ir 'hincando' valores por toda la memoria, mediante el comando **POKE**, y 'cediendo' el control de la máquina, usando el comando **CALL**, por donde se te ocurra, guarda primero tu programa en le cassette, y habiendo sacado un listado, antes de hacerlo; o puede que tengas que arrepentirte después. El amplio y potente sistema operativo del CPC464 se describe en la "**Guía para el Usuario Avanzado**", y se sale del ámbito de esta guía para el usuario principiante.

### 9.3 Interrupciones

El CPC464 aprovecha ampliamente la estructura de **interrupciones** que posee el microprocesador Z80, para disponer de un sistema operativo que incluye varias facultades de **multitarea**, de las que has podido apreciar los comandos **AFTER** y **EVERY**, descritos en el capítulo 8. La prioridad de los 'cronometradores' que generan regularmente señales de interrupción, es:

- Ruptura **ESC** **ESC**
- Temporizador 3
- Temporizador 2 (y los tres canales de sonido)
- Temporizador 1
- Temporizador 0

Deberás incluir el tratamiento por interrupciones, después de considerar las consecuencias sobre los posibles estados intermedios de las variables en el momento de la interrupción. La propia subrutina que atienda las interrupciones, debe evitar interacciones no deseables con el estado de las variables en el programa principal.

Los canales de sonido disponen de circuitos para interrupciones, independientes y de igual prioridad. Una vez que cualquier canal ha suscitado una interrupción, no se ve interrumpido por ningún otro; permitiendo así que los comandos de sonido compartan las variables con inmunidad ante los efectos mencionados más arriba por la estructura de interrupciones de los temporizadores.

Cuando en el programa se suscita una interrupción para un canal de sonido, éste se tramitará inmediatamente si en el canal el 'chorro de notas' pendientes no está completo; y en caso contrario, se tramitará dicha interrupción cuando arranque la siguiente nota y haya, por tanto, sitio para poner otra al final del 'chorro'. Como al tramitarse la interrupción se inhibe el funcionamiento de los circuitos generadores de interrupciones, la subrutina debe, antes de terminar, quitar esa inhibición si se desea seguir recibiendo las posibles señales de interrupción posteriores.

El intento de enviar un sonido a un canal, o la comprobación del estado de un canal, también inhiben el funcionamiento de los generadores de señales de interrupción.

La prioridad de la secuencia **ESC ESC**, sobre todas las otras posibles señales de interrupción, asegura que la operación del programa **BASIC** puede ser suspendida temporalmente sin pérdida del programa, siempre y cuando no se hayan efectuado acciones auxiliares para cerciorarse de la integridad del programa mediante una de las diversas técnicas de protección.

### 9.3 Ensamblador AMSOFT

Con el fin de programar frecuentemente en código máquina, será necesario usar el **ensamblador**. El ensamblador **AMSOFT** consta de un ensamblador reubicable del procesador **Z80**, junto con editor, desensamblador y monitor.

## 10. Posibilidades de interrupción

Temas comentados en este capítulo:

- |                                 |        |
|---------------------------------|--------|
| * DESPUES DE equis tiempo...    | AFTER  |
| * CADA equis tiempo...          | EVERY  |
| * ¿Cuánto tiempo QUEDA para...? | REMAIN |
| * Los cronometradores           |        |

Ya habrás observado una de las mayores innovaciones del CPC464. Es su habilidad -única en el mercado- para tratar **interrupciones** dentro de los programas BASIC; lo que significa que BASIC AMSTRAD es capaz de efectuar varias tareas simultánea y separadamente dentro de un mismo programa. Esta facultad de los ordenadores se denomina algunas veces como 'multitarea', y se consigue mediante los comandos **AFTER** y **EVERY**.

Esa misma facultad también queda demostrada claramente en la manera en que se maneja la generación y emisión de sonidos por los tres canales dispnibles, permitiendo construir **CHORROS DE NOTAS EN ESPERA**, y **ACORDES ENTRE CANALES**.

Todo lo referente a los intervalos de tiempo en el CPC464 está gobernado por un 'reloj patrón' consistente en un circuito electrónico de precisión que incorpora un cristal de cuarzo, y de diversos circuitos electrónicos que dentro del ordenador se preocupa de temporizar y sincronizar las labores y tareas que realiza el ordenador, tales como la generación de señales de barrido para el monitor, y el envío constante de impulsos hacia el microprocesador, etc. Siempre que hay un circuito que desempeña una función relacionada con el tiempo, está en estrecha correlación con las señales procedentes del reloj maestro.

El avance en programación reside en los comandos **AFTER** y **EVERY**, que para mantener el enfoque comprensible del BASIC AMSTRAD con respecto a sus usuarios, hacen precisamente lo que dicen (y los que no sabéis inglés, volver a mirar el epígrafe de este capítulo para poneros en igualdad de condiciones inmediatamente).

### 10.1 AFTER

### DESPUES DE

El comando **AFTER** se incluye en un programa BASIC para conseguir que se ejecuten determinadas tareas **DESPUES DE** transcurrido un determinado tiempo desde la última vez que se ejecutaron. En el CPC464 se dispone de cuatro 'cronometradores', cada uno de los cuales puede tener asignado que transcurrido un determinado tiempo, obligue al programa a hacer un alto en sus operaciones y pasar a ejecutar una tarea predeterminada. La sintáxis para esta instrucción es:

**AFTER** (expresion entera) [, (expresion entera)] **GOSUB** (numero de linea)

El primero de los parámetros enteros, especifica CADA cuánto tiempo tiene que efectuarse la tarea expresada por un 'párrafo' del programa que comienza en el número de línea mencionado en la instrucción. Este intervalo de tiempo se mide en 1/50 de segundo.

El segundo parámetro entero especifica cuál de los cuatro posibles cronometradores es el encargado de llevar la cuenta del tiempo transcurrido. Por lo tanto, este parámetro debe estar en la gama 0 a 3; y si se omite, se presupone el cronometrador 0.

Cuando ha transcurrido el intervalo especificado, se CEDE el control de la máquina a la subrutina, de la misma manera que cuando se ejecuta un comando **GOSUB** en el programa. Como la subrutina terminará con un comando **RETURN**, hará que el control VUELVA al programa principal que continuará ejecutándose justo en el punto donde se vio interrumpido.

Los cronometradores (temporizadores) tienen diferentes prioridades para sus señales de interrupción. El temporizador 3 tiene la máxima perioridad y el temporizador 0 la mínima.

En cualquier punto del programa se puede incluir el comando **AFTER**, con lo que se 'arredran' (ponen a cero) los cronómetros mencionados en la instrucción, y comienzan su labor a partir de ese mismo instante.

Como los temporizadores son los mismos que los usados en el comando **EVERY**, un comando **AFTER** anula cualquier comando **EVERY** anterior que use el mismo número de temporizador; y viceversa.

Veamos un ejemplo:

```
10 MODE 1:X=0
20 AFTER 45 GOSUB 100
30 AFTER 100,1 GOSUB 200
40 PRINT "AMSOFT"
50 WHILE X<100
60 LOCATE #1,30,1:PRINT #1,X:X=X+1
70 WEND
80 END
100 PRINT "perifericos"
110 RETURN
200 PRINT "y software"
210 RETURN
```

Observa el uso de dos CAUCES de comunicación con la pantalla, lo que implica dos ventanas; lo que permite a las subrutinas que tratan las interrupciones exponer mensajes independientemente de los que exponen el programa 'principal' (líneas 50 a 80).

DESPUES de haber transcurrido el tiempo preestablecido en una u otra línea del programa, éste se DESVIA hacia la subrutina mencionada en la instrucción y efectúa la tarea descrita en ese 'párrafo'.

## 10.2 EVERY

## CADA

El comando **EVERY** hace que CADA cierto tiempo, el programa suspenda la operación que esté ejecutando y se DESVIE para efectuar una determinada tarea, y que una vez terminada VENGA a continuar la que estaba haciendo. Se dispone además de cuatro cronómetros (temporizadores), a cada uno de los cuales se le puede encargar de interrumpir la ejecución del programa principal. La sintáxis de este comando es:

**EVERY** (expresion entera) [, (expresion entera)] **GOSUB** (numero de linea)

El primer parámetro entero, especifica el intervalo de tiempo entre dos 'desvíos' sucesivos al párrafo del programa que comienza en el número de línea citado en la instrucción. El tiempo se mide en 1/50 de segundo.

El segundo parámetro entero, especifica cuál de los cuatro posibles temporizadores es el encargado de generar las señales de interrupción. El valor de este parámetro debe estar en la gama 0 a 3; y si se omite, se adopta el temporizador 0.

Cuando ha transcurrido el intervalo de tiempo especificado, automáticamente se produce el desvío a la subrutina correspondiente, igual que ocurre cuando se incluye un comando **GOSUB** en un punto del programa. Los temporizadores tienen prioridades diferentes en cuanto a sus señales de interrupción. El temporizador 3 tiene la máxima prioridad, y el temporizador 0 la mínima.

```
10 MODE 1:X=0
20 P200=0:EVERY 10 GOSUB 100
30 P200=0:EVERY 12,1 GOSUB 200
40 PRINT "AMSOFT"
50 WHILE X<200
60 LOCATE #1,30,1:PRINT #1,X:X=X+1
70 WEND
80 LOCATE 1,20:END
100 DI:PEN P100:LOCATE 1,2:PRINT "perifericos":EI
105 IF P100=0 THEN P100=1 ELSE P100=0
```



```
110 RETURN
200 PEN P200:LOCATE 1,3:PRINT "y software"
205 IF P200=2 THEN P200=3 ELSE P200=2
210 RETURN
```

OBSERVA el uso de los comando **DI** y **EI**, que 'inhiben' la tramitación de otras posibles interrupciones antes de empezar a ejecutar las tareas propias, y quitan esa inhibición una vez que han terminado, respectivamente. Eso tiene como efecto demorar la tramitación de las interrupciones suscitadas por el temporizador 1 (que tienen mayor prioridad), durante el tiempo en que se está tramitando la interrupción suscitada por el temporizador 0. Por lo tanto, impidiendo que los parámetros de ubicación del cursor o de color de pluma usados en la rutina 100 a 110, se vean alterados por el programa principal.

### 10.3 REMAIN

### QUEDA

Esta función **REMAIN** entrega como resultado el tiempo que **QUEDA** por transcurrir en el temporizador dado por el argumento de la función. Al mismo tiempo inhibe la generación de señales de interrupción por parte de ese temporizador, y entrega el valor 0 si el temporizador está previamente inhibido. Su sintáxis es:

**REMAIN((expresion entera))**

## Apéndice I:

### El Arte de lo posible

El recién llegado a la informática, comienza por establecer algunos puntos de referencia que le ayuden a conseguir una panorámica sobre las capacidades y limitaciones del ordenador. Esta sección está pensada para darle una guía muy general sobre lo que sucede y su porqué.

### Fuera el hechizo!

Y aunque la única razón para comprar tu CPC464 haya sido el aprovechar los imaginativos juegos disponibles para funcionar en el 'cacharro', es probable que todavía te estés preguntando sobre diversos aspectos de los ordenadores que vienen bajo el término genérico de **CIRCUITOS** (que como son "duros" de alterar, los ingleses lo llaman 'hardware').

Esos aparatos que recoges y te llevas a casa: el ordenador con su teclado, el monitor, los cables de conexión, etc. son precisamente la 'circuitalia'. De hecho, es todo lo que no es específicamente la 'programalia', que como es 'blanco', y por tanto fácilmente alterable, los ingleses lo llaman 'software'.

Independientemente de los nombres usados, la única innovación que presentan los ordenadores es: ¡¡los PROGRAMAS están imbuidos en los CIRCUITOS!!

Ciertas facetas en la manera que funciona un ordenador, están producidas merced a los circuitos: cosas como la imagen polícroma de un aparato de TV o de un monitor; y es misión de los programas hacer uso de esta capacidad de los circuitos para producir caracteres y figuras en la pantalla específicamente diseñados. Los circuitos controlan realmente el haz de electrones proyectado sobre la superficie electroluminiscente interna de la pantalla de un tubo de televisión y hacerla que se ilumine. Los programas añaden el orden y la inteligencia para comunicar a los circuitos cuándo y dónde tienen que actuar. Añaden temporización, control y procesado consecutivo para conseguir el efecto del despegue de un avión, o de algo más vulgar como una carta apareciendo en pantalla cuando tipografías en el teclado.

### ¿Qué hace que un ordenador sea mejor que otro?

Los circuitos sin programas no valen nada. Los programas sin los circuitos ni siquiera tienen dónde residir; el valor del ordenador comienza cuando se reúnen ambos para efectuar diversas tareas. Hay algunas consideraciones fundamentales que pueden emplearse para graduar el rendimiento de ambos programas y circuitos integrados en un **sistema**.

Los puntos de referencia generalmente aceptados para los ordenadores personales, son en estos momentos:

1. La **resolución** en pantalla: la mota más pequeña que todavía es discernible.

Es una combinación de factores, incluyendo el número de colores diferentes manejables por el operador, el número de áreas distintas que pueden diferenciarse en la pantalla (las 'motas'). El número de caracteres tipográficos que pueden mostrarse en una sola área de pantalla.

Hallarás que tu CPC464 compara muy favorablemente en todos estos aspectos con cualquier máquina de precio similar.

2. El interpretador BASIC.

Virtualmente todo ordenador casero incorpora un interpretador BASIC que permite al usuario comenzar a crear programas y a usar las facultades de los circuitos. El lenguaje de programación incorporado (BASIC) que se suministra con tu máquina, es por sí mismo un programa -un inmensamente complicado e intrincado programa que ha evolucionado durante un millón de hombres/años de experiencia desde que se 'inventó' en los Estados Unidos.

Este 'código de instrucciones simbólicas, de propósito general, para principiantes' es con mucho, el lenguaje informático más ampliamente usado en el mundo, y como cualquier otro lenguaje se presenta con una variedad de 'dialectos' locales. La versión en el CPC464 es uno de los dialectos más ampliamente compatibles, y podrá ejecutar muchos de los programas BASIC comunes y escritos para operar con el sistema operativo de disco CP/M. Es una implementación de BASIC muy rápida -lo que en otras palabras significa que efectúa sus cálculos rápidamente- y mientras puede que no te concierna demasiado que un ordenador puede tardar 0,05 segundos en multiplicar 3 por 5, y mostrar el resultado; otro ordenador puede tardar 0,075 segundos en hacer lo mismo; y cuando un programa que dibuja modelos gráficos en la pantalla, puede exigir muchos millares de simples cálculos relativos, la diferencia entre 0,05 segundos y 0,075 segundos en una operación llega a acumularse a una diferencia en tiempo total muy considerable.

Frecuentemente, oirás que se usa el término 'código máquina'. Código máquina es una forma basta de código de instrucciones que pueden pasarse al microprocesador. El tarda menos tiempo en interpretar lo que se le ha pedido hacer, y se las arregla para producir el resultado de 5 a 15 veces más rápidamente que una operación equivalente que tuviera que pasar a través del interpretador BASIC. Pero se tarda de 5 a 50 veces más escribir un programa equivalente en código máquina comparado con llevar a cabo la misma tarea global usando BASIC.

El BASIC en tu ordenador AMSTRAD, está entre los más rápidos, más potentes y completos que puedas encontrar en cualquier ordenador personal, e incorpora muchas muchas facilidades para que el programador BASIC experto venza parte de la inherente dificultad de un intérprete de 'lenguaje de alto nivel' para efectuar sorprendentes efectos dinámicos, visuales y musicales.



### 3. Ampliando y completando

La mayoría de los ordenadores prestan atención a la necesidad de añadir equipos y artefactos adicionales: impresoras, 'joysticks' para juegos, ductoras de disco. Paradójicamente, algunos de los ordenadores personales de más éxito exigen la incorporación de unidades adicionales conocidas como 'acopladores e interfaces' antes incluso de que pueda conectarse una simple impresora o un controlador de juegos.

El comprador no siempre considera anticipadamente sus necesidades en el futuro, porque de lo contrario, una máquina que incorporara adecuadamente el acoplador necesario para una impresora compatible con una norma standard como Centronics, y además lo necesario para el control de juegos, puede que realmente sea más barato a la larga.

El ordenador CPC464 tiene incorporado un 'portal de salida' para impresora Centronics; facilidades para hasta dos joysticks de juegos; una salida de sonido estéreo; y un completo bus de interconexión que puede usarse para equipos de almacenamiento en disco, memorias preprogramadas adicionales, impresoras según la norma RS232, etc.

Una memoria preprogramada que sólo permite operaciones de lectura. **Memoria de Sólo Lectura** (Read Only Memory) es un circuito integrado en el que está grabada información concerniente a un programa. El BASIC que se suministra con tu ordenador, está grabado en uno de esos circuitos, y es posible que otros programas sean suministrados así, ya sea para suplementar el programa incorporado, o sustituir completamente sus funciones por otras.

Las consolas de juegos en TV, usan programas grabados en 'cartuchos'. Y esos cartuchos son en esencia, también circuitos de memorias preprogramadas, alojadas en una carcasa plástica y con una forma u otra de ser enchufado rápidamente para permitir meterlo y sacarlo muy fácilmente. Por lo tanto, esta clase de circuitos memorizadores proveen la misma función que las cassettes en las que depositas tus programas. Sin embargo, tienen la ventaja de que cargan la información en la memoria del ordenador de una forma virtualmente instantánea en comparación con los varios minutos que tarda un programa largo en ser transferido de la cassette al ordenador; y por tanto, su mayor ventaja es una de absoluta conveniencia, que hay que examinar contra su total imposibilidad de ser usadas para almacenar información que saques de tu ordenador y que puedas llevar a otro ordenador de la misma manera que haces con el cassette Datacorder de tu CPC464.

Por tanto, con el bus de interconexión, tienes garantizado que tu ordenador podrá ir incorporando la mayoría de los desarrollos que en programación y electrónica se hagan, y el sistema CPC464 tiene las posibilidades de ampliación muy completas y extremadamente documentadas.

#### 4. Sonidos

Las posibilidades sonoras de un ordenador determinan si suena o no como moscardones dentro de un bote de hojalata, o puede producir una reproducción aceptable de un instrumento musical electrónico.

El ordenador CPC464 usa un generador de sonidos con 3 canales y 8 octavas, que puede producir una calidad musical muy aceptable, con control completo sobre las envolventes de volumen y de tono. Además, el sonido está dividido según una configuración estéreo, donde uno de los canales corresponde a la salida izquierda y el otro canal a la derecha, con el tercero situado en el medio.

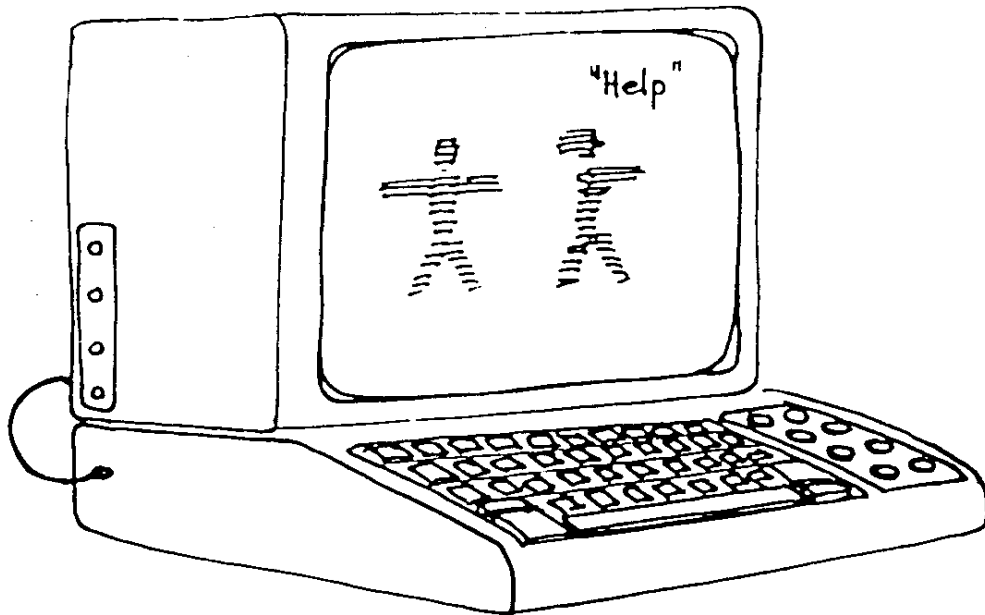
Eso ofrece un panorama considerable para escribir programas que repiquen gráficamente en la pantalla los efectos sonoros para acompañar el movimiento en los juegos típicos modernos.

Al final, tú tomarás tu decisión sobre cuál de estas facetas es más importante en tu caso. Esperemos que ensayes todas ellas para sacar el máximo provecho de tu ordenador.

#### ¿Por qué no puede?

Con toda la potencia de la moderna tecnología, los usuarios se preguntan a menudo por qué incluso una máquina tan avanzada como el CPC464, es aparentemente incapaz de mostrar la clase de imágenes vistas en cualquier aparato de televisión.

¿Por qué, por ejemplo, no puede un ordenador animar una figura de una persona paseando a través de la pantalla en una forma natural? ¿Por qué todos los ordenadores representan el movimiento con figuras que parecen hechas con cerillas?



La respuesta aunque compleja, es simple. No debes ser persuadido a creer que la imagen producida de tu ordenador tiene algo de la argucia empleada en las imágenes de televisión. La imagen de televisión usa información y electrónica de tipo continuo, con la que se puede describir una gama virtualmente infinita de grados entre los extremos de luz y oscuridad pasando por todos los colores del espectro. Este proceso significa que en términos de medida de la información (y la unidad de medida es el **bitio**) la imagen de un televisor requeriría del orden de 20 veces más memoria que las imágenes que la requerida para la imagen producida por un ordenador personal. Y eso implica pasar de memorias en **Kilobitios** a **Megabitios**. Y eso es sólo parte del problema, dado que para dar movimiento a esa imagen requiere que esa enorme cantidad de memoria sea procesada a elevada velocidad (alrededor de 50 veces por segundo toda). Puede hacerse, pero sólo con máquinas que cuestan alrededor de mil veces más que un ordenador personal como menos y en este momento.

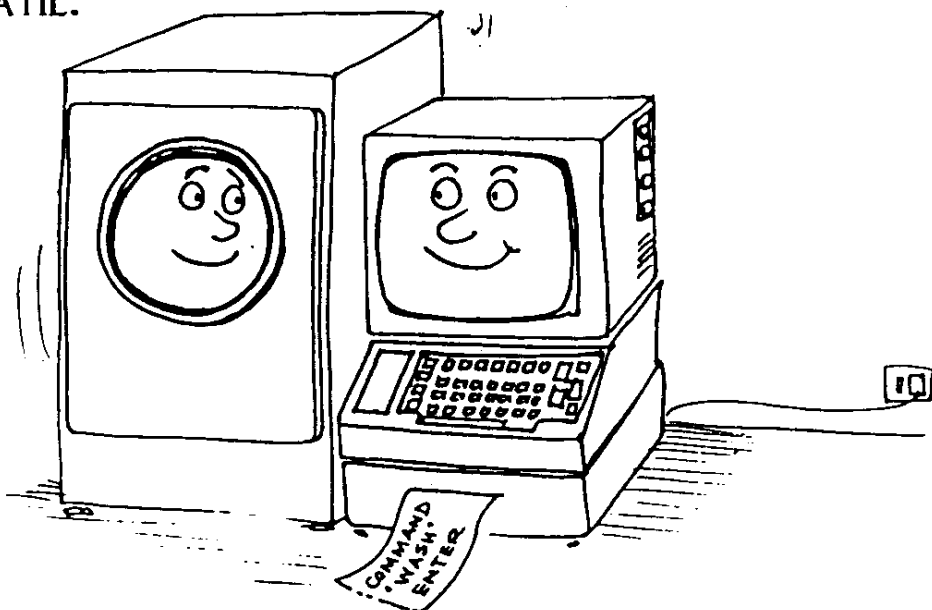
Hasta que el coste de las memorias de alta velocidad disminuya impresionantemente (lo que sucederá sin ninguna duda), los ordenadores pequeños tienen que apañárselas con una relativamente pequeña cantidad de memoria para controlar la imagen en pantalla, lo que resulta en menor resolución y movimientos espasmódicos, un diseño cuidadoso de los circuitos y una buena programación tienen un largo camino para aprovechar lo mejor en esta situación, pero todavía están lejos los ordenadores baratos que puedan reproducir movimientos armoniosos e imágenes naturales de la misma manera comparable incluso a una película de dibujos animados.

¿Por qué no puedes simplemente sentarte al ordenador y teclear una página de simple texto en la máquina?

Tampoco te confunda el hecho que el ordenador parece una máquina de escribir con una pantalla en lugar de papel. La pantalla no es un simple papel 'electrónico', es en realidad un monitor que con el teclado forma la **consola**, que es el medio de comunicarte con los interpretadores de programas o con los propios programas que ocupan la memoria de la máquina. Hasta que le digas lo contrario, el ordenador examinará e interpretará todos los caracteres procedentes del teclado como comandos o instrucciones. Y en la mayoría de los ordenadores, simplemente los 'escuchará' y repicará en la pantalla, hasta que pulses la tecla que le indica VALE, momento en que examinará lo que le has tecleado, y si no tiene sentido para el programa que gobierna su operación, rechazará lo tecleado comentando:

Sin embargo, puede simplemente suceder que el programa que reside en la memoria de tu ordenador en ese momento, está pensado para **elaboración de textos**, en cuyo caso podrás ser capaz de teclear palabras según tu propia discreción, y meterlas hacia la memoria del ordenador, y hacer que se impriman por impresora, logrando casi que fuera una auténtica y aventajada máquina de escribir. Pero para hacer todo ello, primero tendrías que **instruir** a la máquina para que supiera hacerlo, para lo que trasvasarías del cassette a la memoria de la máquina el programa pertinente.

Un ordenador parece que combina diversos equipos que se han hecho familiares en el hogar y en la oficina: el monitor similar al televisor, el teclado a una maquina de escribir, el cassette, etc.; pero debes recordar que las similitudes son generalmente sólo superficiales, y que el ordenador es una combinación de 'circuitalia familiar' a la que la 'programalia' le confiere una personalidad completamente diferente y extraordinariamente **VERSATIL**.



## Apéndice II:

### Una introducción a las intimidades de los ordenadores

#### ¿Quién teme a la jerga?

Como con todas las industrias especializadas, la informática ha desarrollado su propio 'idiolecto' como una forma breve de comunicar conceptos complicados que exigen muchas palabras y largas frases para ser explicados con el lenguaje del vulgo. No es simplemente que la industria de alta tecnología sea culpable de esconderse detrás de una aparente pantalla de humo formada por 'palabras secretas', jerga y terminología; la mayoría de todos nosotros estamos contra esas barreras puestas a la comprensión, pero sin embargo, erigidas en todas las profesiones y oficios principales.

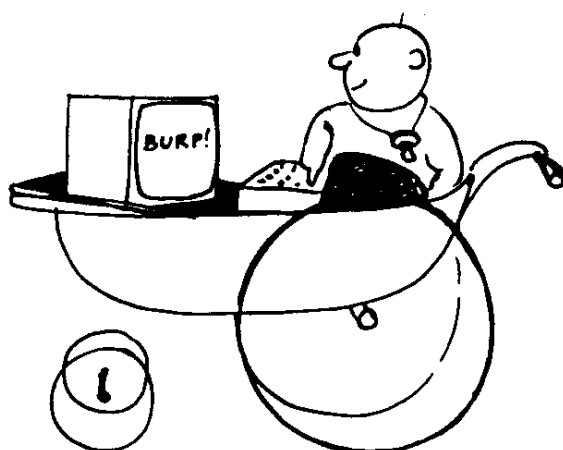
Una diferencia importante es que la confusión, por ejemplo, en la 'jerga legal' viene de la manera en que se usan las palabras, y no por las palabras en sí mismas, como es el caso del ordenador. La mayoría de la gente que aumenta su familiarización con la terminología informática, se saldrá del camino para usar las palabras en la forma más directa posible, para minimizar la complejidad de la comunicación (y obviamente para teclear menos).

No te confundas tampoco por el aparente 'simple lenguaje' usado en informática, ya que no es un asunto literario, sino una ciencia precisa, y aparte de la sintaxis de las frases comunicadas al ordenador, la estructura de comunicación es muy directa, y no puede en absoluto ser confusa o ambigua. Los maestros de informática y cómputos, somos todavía meros aprendices en el arte de transmitir conocimientos y el significado exacto que guía a un programador en la construcción de su programa.

Habiendo dicho esto, aunque sea ahora obvio o no lo sea el significado de un programa y un ordenador, hay todavía muchos aspectos que pueden ser analizados y clasificados como elegantes o desordenados, y se está poniendo un mayor énfasis en abordar formalmente la construcción de programas ahora que la inicial marejada levantada por la revolución microinformática se está asentando.

La informática es rápidamente comprendida por muchos jóvenes que aprecian la precisión y simplicidad de las ideas, y la manera en que pueden comunicarse. Seguro que no encontrarás muchos 'abogados con 10 años de edad, y sin embargo encuentras multitud de niños de 10 años que ya son programadores.





## Las bases del BASIC

Virtualmente, cualquier ordenador casero está dotado del lenguaje llamado BASIC, que permite escribir los programas en la jerga más cercana al lenguaje vulgar disponible hoy.

El significado de donde BASIC derivó su nombre (en lo referente a 'principiantes=beginners') ya no tiene ningún significado particular dado el grado de complejidad de los lenguajes, y que hay programas extremadamente complejos y potentes escritos usando el lenguaje BASIC. Sin embargo, no hay duda que el acierto en la elección del nombre, ha traído a muchos principiantes por su promesa de suministrarles la BASE para adentrarse en la maraña de lenguajes informáticos, y eso sí ha contribuido de forma significativa a su universalidad. A partir de aquí, las palabras comúnmente usadas que forman el glosario de términos informáticos serán presentadas imprimiéndolas primeramente en **negrita**. No intentes aprenderlas sólo con las siguientes secciones, también hay un índice y está el resto del manual.

## La Base

BASIC es un lenguaje informático que interpreta un cierto surtido de frases imperativas, y que efectúa operaciones sobre los datos, cuando en la máquina hay un programa **ejecutándose**. A diferencia del vocabulario medio humano de 5 a 8.000 palabras (más todos los diferentes modos verbales que pueden usarse), el lenguaje BASIC tiene suficiente con cerca de 200 palabras **clave**. Los programas para ordenadores que se escriben usando BASIC, tienen que seguir reglas rígidas en cuanto al uso de esas palabras. La **sintaxis** es precisa, y cualquier intento de comunicar con el ordenador usando expresiones coloquiales y vulgares (lenguaje) dará como resultado el frío y mecánico mensaje:

**Syntax Error**

y no es tan restrictivo como a primera vista aparece, dado que el lenguaje BASIC está diseñado primordialmente para manipular números: los **datos numéricos**. Las palabras son en esencia, una ampliación de los operadores matemáticos ya familiares (+ / - etc.) y el concepto más importante para los principiantes es 'aprehender' el hecho que el ordenador sólo puede trabajar con datos e información que utiliza un convenio universal basado en el sistema de numeración DOS para dar nombres a las señales eléctricas que se transfieren entre el **microprocesador** (que es la unidad central en los grandes ordenadores), los circuitos memorizadores y canalizadores, y las diversos circuitos de los equipos **periféricos**.

### NUMEROS POR FAVOR

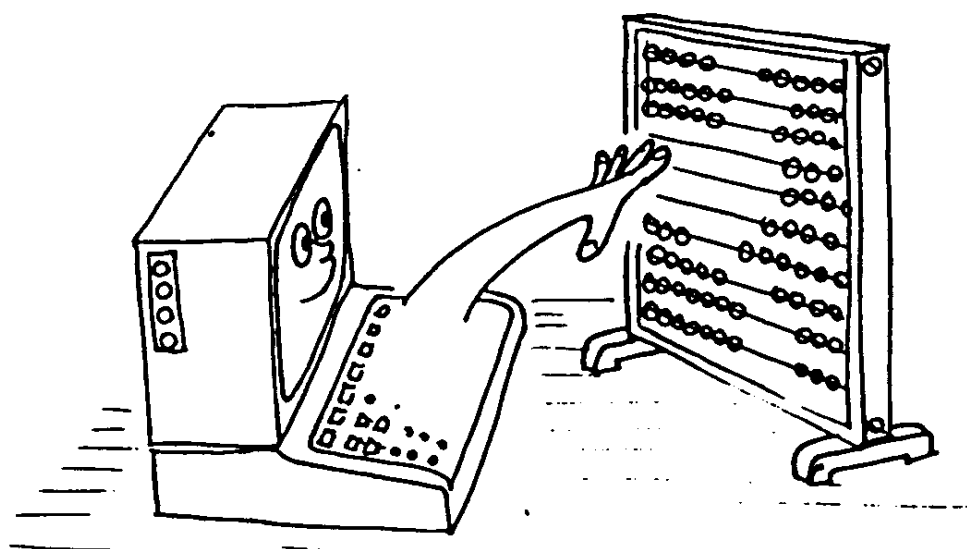
Si un ordenador se usara para conservar las obras completas de Cervantes, y nos armáramos con el instrumental adecuado, es bastante obvio que no podríamos encontrar ninguna letra o palabra en ninguna parte del sistema. Pero sí podemos encontrar sitios en el sistema donde la presencia o ausencia de señal eléctrica, adecuadamente interpretada, guarda una estrecha relación con las inmortales palabras de Cervantes.

Es mucho más simple entonces hablar de que BASIC interpreta las letras y palabras como **números** únicamente, que el ordenador puede manipular usando las operaciones matemáticas habituales y las comparaciones estudiadas como **lógica de Boole**, que le permiten determinar si un número es mayor que otro, y si una letra va antes que otra de acuerdo con el **alfabeto** que maneje; y 'elegir' la tarea a realizar dependiendo de si se cumplen o no ciertas condiciones.

Y además, para que pueda realizar una tarea, se le tiene que desglosar pormenorizada y exactamente en una serie de esas operaciones elementales, cada una con su **frase** convenida con él, y esa serie de frases es lo que denominamos PROGRAMA.

Y además, depende del lenguaje utilizado ese detalle puede ser mayor o menor. Y si por ejemplo, operas en código máquina es normal que el proceso de multiplicación se lo tengas que detallar como repetición de sumas del multiplicando, tantas veces como indica el multiplicador, porque el microprocesador utilizado no sabe más que sumar. En BASIC alguien preparó ese detalle, lo introdujo en la máquina y decimos que la máquina ya sabe multiplicar. Y por ello, ya puedes decir en BASIC una instrucción que obligue al ordenador a multiplicar, por ejemplo, 35 por 10 y te dé el resultado en lo que has llamado **variable A**. Y se lo tendrías que decir de acuerdo con la sintaxis, en la forma: **LET A=35\*10**.

Para llevar a cabo lo que has requerido, en algún sitio de la unidad central de proceso (con las siglas inglesas CPU), se **graba** el operando 10, y en otra parte el otro operando 35. Luego, la maquina bajo el control del interpretador BASIC, lo que hace en realidad es considerar el sitio donde tiene alojado en 10 como un **contador** del que va restando 1, al mismo tiempo que va **acumulando** la suma de  $35+35+35+\dots$  en otro sitio del circuito, y detiene el proceso cuando ese contador ha llegado a 0, momento en que el resultado de 350 es enviado (y no olvides que lo que viajan son electrones, y esto es sólo es una cierta forma de hablar) a otro sitio para que sea **sacado** y **mostrado** como respuesta.



Si todo este proceso te suena demasiado aburrido, tendrás razón, y habrás descubierto la primera y más importante verdad sobre los ordenadores. Un ordenador es simplemente una máquina 'extremadamente tonta y extremadamente rápida', por lo que sólo puede hacer tareas simples y repetitivas, pero las hace a toda velocidad y sin cansarse.

En un ordenador sólo **DOS** estados son reconocidos en las señales eléctricas de sus circuitos; y esos estados sólo necesitan por tanto **dos** símbolos para ser representados, de ahí que se llame **sistema binario**, y a esos dos símbolos se denominen **BIToS**; y los habituales son 'sí' y 'no', 'A' y 'B'; y los que tú quieras. Todo esto conecta además con la mencionada lógica de **BOOLE** basada en la 'certeza' y 'falsedad' de la vida normal. Y a partir de ahí se construye todo lo demás.

El proceso de considerar únicamente dos estados distinguibles en una determinada situación, es la esencia del término binario y de los que usando combinaciones de éste, llamamos **digital** y **discreto** (los electrónicos también hablan de circuitos **basculadores** o **biestables**, porque solamente poseen dos estados estables y entre los que 'basculan'). Descomponiendo cualquier dato en combinaciones de 'ringlas de biestables' es posible de una manera ordenada y sistemática representar cualquier información, manipular posteriormente dicha información, y volver a convertirla para presentarla con sentido humano. Todo eso aunque hecho a velocidades increíbles de millones de cambios de estado en cada segundo, acaba llevando su tiempo y es el empleado por el ordenador en hacer una operación.

Además, el **microprocesador** tiene un tiempo finito para desarrollar una tarea que ha de cumplir antes de que pueda comenzar a trabajar en la siguiente tarea. Estos tratamientos binarios, son los que permiten pasar del 'blanco' al 'negro' y por combinaciones inteligentes de blanco y negro obtener toda la escala de grises, por ejemplo.

Si la respuesta última analizando **un solo punto** sólo puede ser 0 ó 1, no hay posibilidad de ser 'casi' correcto, a no ser que se examinen varios puntos al mismo tiempo. El hecho de que los ordenadores pueden algunas veces cometer errores cuando manejan datos numéricos, es debido a la limitación de puntos binarios que puede analizar considerándolos como un grupo; y por tanto, con la precisión que pueden conseguir sobre números decimales sobre los que operan. Y de ahí los errores que cometen en redondeos, v.g. decir 0,999999999 cuando deben simplemente un 1.

En una palabra, si sólo se mira un cable dentro del ordenador, sólo hay dos estados posibles, y lo normal es decir: si hay tensión hay un 1, y si no hay tensión hay un 0; y para pasar de ahí, no queda más remedio que utilizar cientos de miles de cables.

### Bits y 'Bicos'

Sucede que estamos acostumbrados a comprender las cantidades y reflejarlas en un número basado en el sistema **decimal** que tiene como base el número 10; y que por tanto, dispone de 10 símbolos diferentes de los que partir (y esos **diez** símbolos se llaman obviamente **dígitos** porque dígito viene de dedo). Y te recuerdo que son:

0 1 2 3 4 5 6 7 8 9

En el sistema binario sólo se dispone de dos símbolos y esos dos símbolos obviamente se llaman bits y los habitualmente empleados son: 0 y 1. Pero naturalmente, podríamos haber usado cualesquiera otros dos símbolos, y quizá evitaríamos la cantidad de confusiones provocadas por el hecho de que también se emplean como dígitos.

Pero igual que en el sistema binario formamos "sartas de dígitos" para designar cantidades superiores a 9, de igual manera formamos sartas de bits, para representar situaciones y cantidades y letras y cifras y todo lo que queramos. Y usando esos símbolos en mi ordenador, cuando escribo mi nombre queda grabado como:

0100101001110101011011000100100101101111

que por supuesto, no hay quien sea capaz de manejar (salvo el ordenador).

Por lo tanto, y como sucede en otras cosas, hay que dividir para vencer, y empieza a tener más sentido si te la presento separada en la forma siguiente:

0100 1010 0111 0101 0110 1100 0100 1001 0110 1111

Esta separación y agrupación de bits, equivale realmente a examinar simultánea y sucesivamente diversos puntos internos dentro del ordenador. Y puedes ver que hay grupos de cuatro, que llamamos cuartetos, y grupos de ocho, que llamamos octetos, por no decir dos cuartetos, u ocho bits.

Es un convenio internacional la manera en que se forman esos cortes y grupos para que todos entendamos su significado. Y el grupo más importante es precisamente el número de puntos que el ordenador internamente mira al mismo tiempo. Y aunque ese grupo depende del ordenador en cuestión, sí se le ha dado un nombre adecuado: se le llama 'bocado', porque es lo que se traga en una sola tacada. Por supuesto que lo han hecho con la palabra inglesa correspondiente 'byte' (que también es mordisco) y los hispano-parlantes usan varias con preponderancia de 'bico' (que más que un mordisco es un beso).

Además ya va siendo generalmente aceptado, hacerlo corresponder al octeto, debido precisamente a que la inmensa mayoría de los ordenadores personales actuales es la cantidad de bits que tratan de una sola vez, traen de la memoria o meten en la memoria, envían hacia los circuitos de pantalla, traen del teclado, sacan hacia la impresora, etc.

Y además, también existen convenios internacionales para ver la equivalencia de esta NOTACION, con las otras notaciones empleadas para datos numéricos (el sistema de numeración en base 2) y datos **literales** (el código ASCII).

Veamos primero el sistema de numeración en base 2. Si recuerdas las matemáticas estudiadas, verás que una determinada cantidad puedes decir que es 7 si empleas la notación decimal, y puedes decir que es 00000111 si empleas notación binaria. Y para ello basta considerar que los bits de ese bico son meramente indicativos de si has de usar o no la correspondiente **potencia** del número 2. Es decir:

## EN LAS ENTRAÑAS DE LA MAQUINA

$2^7$   $2^6$   $2^5$   $2^4$   $2^3$   $2^2$   $2^1$   $2^0$

0 0 0 1 0 1 1 1

NO NO NO SI NO SI SI SI



$2^4 + 2^2 + 2^1 + 2^0 =$

$= 16 + 4 + 2 + 1 = 23$

Por tanto, con un bico (de los de 8 bits) el número equivalente en el sistema decimal más alto que puedes conseguir, será:

11111111 = 255

00000000 = 0

Y en total habrá por tanto, 256 números diferentes; y ese número ya empieza a serte familiar en cuanto empiezas a trabajar con ordenadores.

Te sonarán en particular mucho más sus múltiplos, que van correspondiendo a las cantidades y números representables mediante parejas de bicos, cuartetos de bicos, y octetos de bicos. Porque si compruebas con una pareja de bicos podemos llegar a distinguir  $256 \times 256 = 65.536$  que divididos por 1.024 (que ya veremos, es la famosa **K**), nos da exactamente 64 que sin ninguna duda es el número de bicos que tiene la memoria de muchísimos ordenadores (y si no has oído hablar de 64K, mejor que no sigas leyendo).

0 1 2 3 4 5 6 7

0 \_ \_ \_ \_ \_ \_ \_

1 \_ \_ \_ \_ \_ \_ \_

2 \_ \_ \_ \_ \_ \_ \_

3 \_ \_ \_ \_ \_ \_ \_

4 \_ \_ \_ \_ \_ \_ \_

5 \_ \_ \_ \_ \_ \_ \_

6 \_ \_ \_ \_ \_ \_ \_

7 \_ \_ \_ \_ \_ \_ \_

8 \_ \_ \_ \_ \_ \_ \_

9 \_ \_ \_ \_ \_ \_ \_

En la figura, se pueden apreciar 80 bandejas dispuestas regularmente ('ringlas') que pueden servirnos para simbolizar claramente lo que veríamos si pudieramos examinar simultáneamente 80 puntos **biestables**. Pero mejor consideremos que examinamos **sucesivamente** 10 sitios y en cada sitio es el exámen **simultáneamente** de 8 puntos de los circuitos internos. Si los asignamos a celdillas capaces de contener un bico (un 1 o un 0) tendremos un trozo de memoria con 10 BICOS de 8 bits.

Pues bien, si continuaras añadiendo renglones horizontales con 8 celdillas cada una hasta que llegaras al número 1.023, tendrías exactamente la representación de 1K de memoria, ya que habría 1.024 bits.

Y si siguieras hasta llegar al número 65535, tendrías los 65.536 bicos (64 x 1.024) que serían las 64K mencionadas.

Afortunadamente, los que construyen el interpretador BASIC son los que tienen que conocer todas estas cosas para que tú no las necesites, y puedas llegar a ser un programador eficiente sin una comprensión completa de los bits, los bicos, etc. Aunque una apreciación de todo lo que significa te ayudará enormemente en cuanto comiences a dibujar gráficos y avances en la ciencia informática. Por lo tanto, merece la pena emplear algún esfuerzo en adquirir esa comprensión, y el significado de los diversos números 256, 1.024, tanto en su notación decimal como en su notación binaria, y en la que veremos en el siguiente párrafo la notación hexadecimal.

**Sin embargo...** simple y elegante como es, la notación binaria es tediosa y propensa a inexactitudes que no pueden ser detectadas con una mirada. La notación binaria tiene asociada íntimamente otros sistemas de notación que realmente son los que usan los programadores al tratar de estos temas. Se denomina normalmente hexadecimal (y viene de 16), y se abrevia normalmente 'hex'. Coincide con el sistema de numeración con base 16, y como  $16 = 2^4$  basta separar adecuadamente la notación binaria en grupos de cuatro (cuartetos) y comprobar que a todos los efectos son equivalentes. Los 16 'hexitos' (y muchos prefieren llamarlos 'dexitos' para evitar interpretar el prefijo hexa como 6, dado que ya existen los hexágonos), se simbolizan normalmente por:

0 1 2 3 4 5 6 7 8 9 A B C D E F

o más bien

0 1 2 3 4 5 6 7 8 9 A b c d E F

para aprovechar que así los puedes representar sin confusión incluso en los relojes llamados digitales.

Con esta notación, un bico de 8 bits se convierte en 2 dexitos. Y basta multiplicar el equivalente decimal del primer dextito por el número 16 y sumarle el equivalente del segundo dextito para obtener el equivalente decimal de un bico de 8 bits.

En la tabla siguiente, y para resaltarte el hecho de que el 0 y el 1 con que simbolizamos habitualmente los BITOS, puedes ser simbolizados con cualquier otra cosa, te resumimos todo en la tabla siguiente:

Decimal	Binario	CPC464	Hexadecimal
0	0	fl	0
1	1	*	1
2	10	*fl	2
3	11	**	3
4	100	*flfl	4
5	101	*fl*	5
6	110	**fl	6
7	111	***	7
8	1000	*flflfl	8
9	1001	*flfl*	9
10	1010	*fl*fl	A
11	1011	*fl**	B
12	1100	**flfl	C
13	1101	**fl*	D
14	1110	***fl	E
15	1111	****	F
16	10000	*flflflfl	10

A través de toda esta guía, designamos los números hexadecimales precediéndoles del signo **&**, v.g. **&D6**; ya que es el símbolo más comúnmente usado por los programadores cuando están empleando técnicas de lenguaje máquina. Aunque todavía prefieren los lenguajes **ensambladores** que es todavía otra forma más **nemotécnica** de representar ya no sólo datos numéricos y literales a usar en el programa, sino también las claves o códigos de las operaciones que el microprocesador tiene que efectuar.

Si lo que quieres es hallar el equivalente hexadecimal de un número decimal simplemente tienen que efectuar la división entera, tomando el número decimal como dividendo y 16 como divisor. El cociente resultante será un número menor que 16, que constituye el primer dextito del equivalente, y el resto que también será menor que 16, constituye el segundo dextito del equivalente. Prueba con el decimal 214, y verás que el cociente es 13 y el resto 6; que en notación hexadecimal, 13 es la D y 6 también se simboliza con 6. Y que por tanto, el equivalente es **&D6**. (Y no te dejes llevar de la tentación, y considerar **&D6** como 13+6 ó 136; ahora ya sabes que es  $13 \times 16 + 6 = 214$ ).

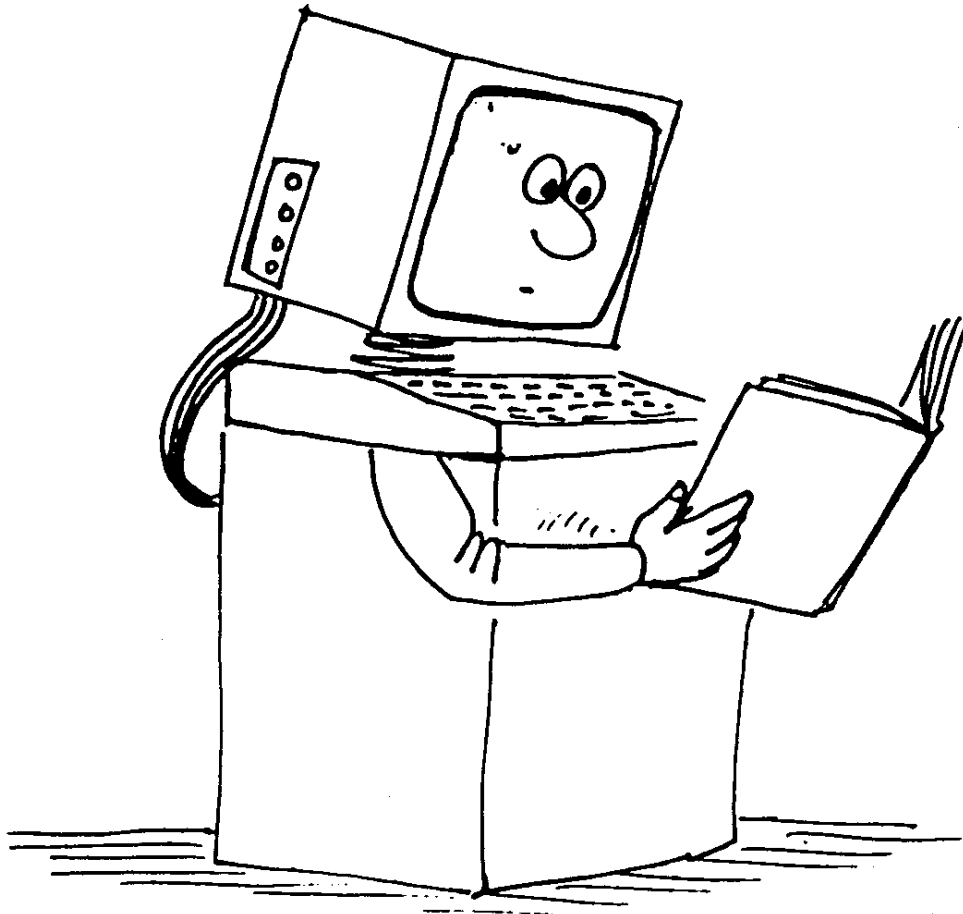
Es el mismo proceso cuando lees un número decimal (y hay mucha gente que prefiere decir un número denario, para evitarse problemas con la parte fraccionaria (decimal) de un número), tal como 89 que en realidad es  $8 \times 10 + 9$ . Simplemente sucede que multiplicar por 10 nos es mucho más simple que multiplicar por 16.

Si has llegado hasta aquí sin que se te haya armado un lío, estás bien dispuesto para comenzar el camino de la informática. Puede que incluso te preguntes de qué va todo esto, y que no es necesario. Un ordenador es un aparato que maneja conceptos e ideas muy simples: y simplemente ocurre que efectúa esas tareas a una gran velocidad (millones de veces



por segundo) y con una enorme capacidad para almacenar y recordar todos los datos que le has impuesto, y los resultados intermedios de los muchos millares de simples sumas y 'pases' de un circuito a otro para conseguir un resultado.

Si quieres proseguir estudiando, hay realmente millares de libros disponibles sobre el tema, algunos tenderán a dejarte más confuso de lo que estabas antes de comenzar a leerlos; y unos pocos, realmente te guiarán durante el camino, desvelándote la sencillez de las relaciones fundamentales que existen entre los sistemas de numeración y la manera en que el ordenador los maneja.



Y no se nos olvida que también trabaja con datos **literales**, pero sencillamente lo hace 'al pie de la letra', usando el convenio internacional ASCII. Basta que apliques lo anterior a la sarta de unos y ceros con que empezamos para que te sea fácil convertirla usando la notación hexadecimal en esta otra:

4A 75 6C 69 6F

Y luego consultar la tabla de códigos ASCII para poder descifrarla y ver que dice simplemente:

J u l i o

## Apéndice III:

### Los caracteres ASCII y los caracteres gráficos del CPC464

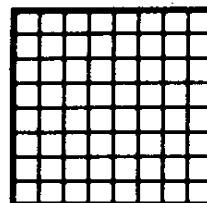
#### III. 1 ASCII

Para referencia, reproducimos aquí el repertorio de caracteres standard ASCII usando las notaciones decimal, octal y hexadecimal, juntamente con el carácter 'visivo' o de control que le corresponde. Asimismo, la composición gráfica de los caracteres usados en el CPC464 se muestran a continuación.

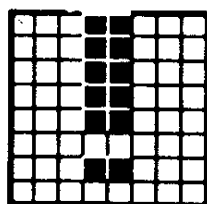
DEC	OCTAL	HEX	ASCII characters	DEC	OCTAL	HEX	ASCII	DEC	OCTAL	HEX	ASCII
0	000	00	NUL ((CTRL)A)	50	062	32	2	100	144	64	d
1	001	01	SOH ((CTRL)B)	51	063	33	3	101	145	65	e
2	002	02	STX ((CTRL)C)	52	064	34	4	102	146	66	f
3	003	03	ETX ((CTRL)C)	53	065	35	5	103	147	67	g
4	004	04	EOT ((CTRL)D)	54	066	36	6	104	150	68	h
5	005	05	ENO ((CTRL)E)	55	067	37	7	105	151	69	i
6	006	06	ACK ((CTRL)F)	56	070	38	8	106	152	6A	j
7	007	07	BEL ((CTRL)G)	57	071	39	9	107	153	6B	k
8	010	08	BS ((CTRL)H)	58	072	3A	:	108	154	6C	l
9	011	09	HT ((CTRL)I)	59	073	3B	;	109	155	6D	m
10	012	0A	LF ((CTRL)J)	60	074	3C	<	110	156	6E	n
11	013	0B	VT ((CTRL)K)	61	075	3D	=	111	157	6F	o
12	014	0C	FF ((CTRL)L)	62	076	3E	>	112	160	70	p
13	015	0D	CR ((CTRL)M)	63	077	3F	?	113	161	71	q
14	016	0E	SO ((CTRL)N)	64	100	40	@	114	162	72	r
15	017	0F	SI ((CTRL)O)	65	101	41	A	115	163	73	s
16	020	10	DLE ((CTRL)P)	66	102	42	B	116	164	74	t
17	021	11	DC1 ((CTRL)Q)	67	103	43	C	117	165	75	u
18	022	12	DC2 ((CTRL)R)	68	104	44	D	118	166	76	v
19	023	13	DC3 ((CTRL)S)	69	105	45	E	119	167	77	w
20	024	14	DC4 ((CTRL)T)	70	106	46	F	120	170	78	x
21	025	15	NAK ((CTRL)U)	71	107	47	G	121	171	79	y
22	026	16	SYN ((CTRL)V)	72	110	48	H	122	172	7A	z
23	027	17	ETB ((CTRL)W)	73	111	49	I	123	173	7B	{
24	030	18	CAN ((CTRL)X)	74	112	4A	J	124	174	7C	
25	031	19	EM ((CTRL)Y)	75	113	4B	K	125	175	7D	}
26	032	1A	SUB ((CTRL)Z)	76	114	4C	L	126	176	7E	-
27	033	1B	ESC	77	115	4D	M				
28	034	1C	FS	78	116	4E	N				
29	035	1D	GS	79	117	4F	O				
30	036	1E	RS	80	120	50	P				
31	037	1F	US	81	121	51	Q				
32	040	20	SP	82	122	52	R				
33	041	21	!	83	123	53	S				
34	042	22	"	84	124	54	T				
35	043	23	#	85	125	55	U				
36	044	24	\$	86	126	56	V				
37	045	25	%	87	127	57	W				
38	046	26	&	88	130	58	X				
39	047	27	'	89	131	59	Y				
40	050	28	(	90	132	5A	Z				
41	051	29	)	91	133	5B	[				
42	052	2A	*	92	134	5C	\				
43	053	2B	+	93	135	5D	]				
44	054	2C	,	94	136	5E	^				
45	055	2D	-	95	137	5F	_				
46	056	2E	.	96	140	60					
47	057	2F	/	97	141	61	a				
48	060	30	0	98	142	62	b				
49	061	31	1	99	143	63	c				

## III. 2 Repertorio de caracteres específico de la máquina CPC464

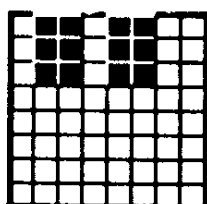
Los caracteres reproducidos aquí, están mostrados usando una **retícula** de 8x8, que es la que se emplea para exponer los caracteres en la pantalla del CPC464. Los caracteres definidos por el usuario pueden ser agrupados y colocados unos contiguos con otros para lograr figuras especiales. (Véase el comando **SYMBOL** en el capítulo 8).



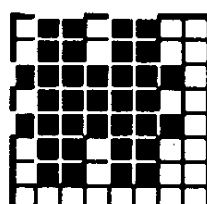
32 &H20  
&X00100000



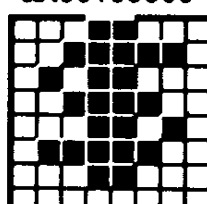
33  
&H21  
&X00100001



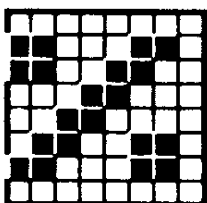
34  
&H22  
&X00100010



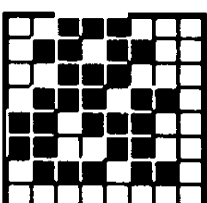
35  
&H23  
&X00100011



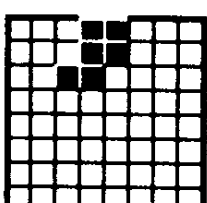
36  
&H24  
&X00100100



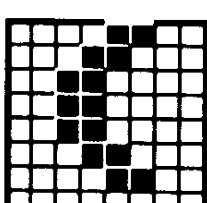
37  
&H25  
&X00100101



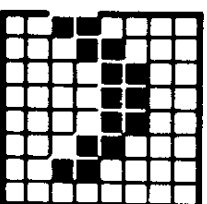
38  
&H26  
&X00100110



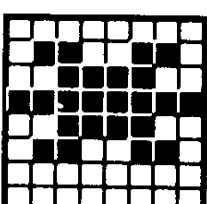
39  
&H27  
&X00100111



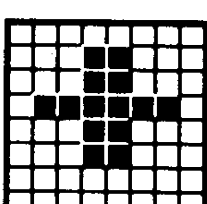
40  
&H28  
&X00101000



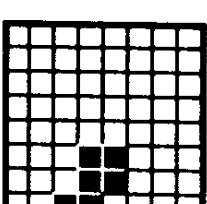
41  
&H29  
&X00101001



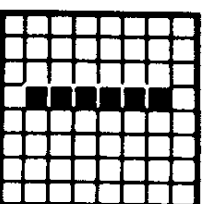
42  
&H2A  
&X00101010



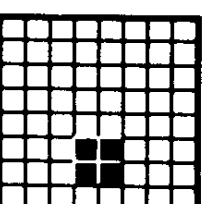
43  
&H2B  
&X00101011



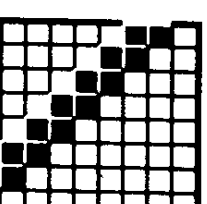
44  
&H2C  
&X00101100



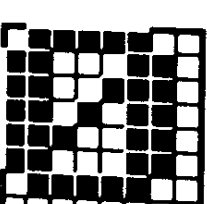
45  
&H2D  
&X00101101



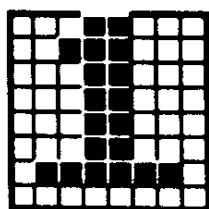
46  
&H2E  
&X00101110



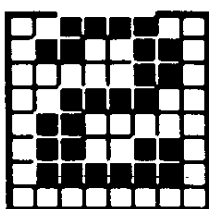
47  
&H2F  
&X00101111



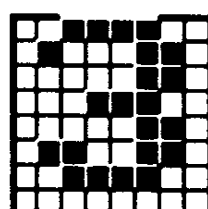
48  
&H30  
&X00110000



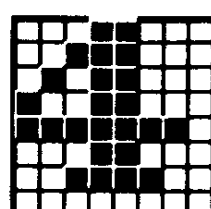
49  
&H31  
&X00110001



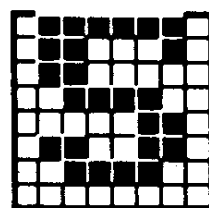
50  
&H32  
&X00110010



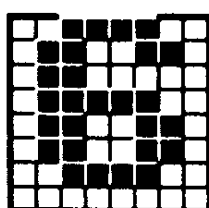
51  
&H33  
&X00110011



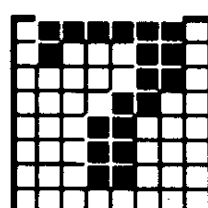
52  
&H34  
&X00110100



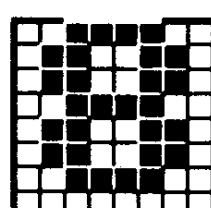
53  
&H35  
&X00110101



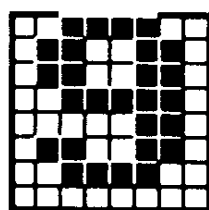
54  
&H36  
&X00110110



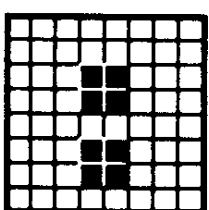
55  
&H37  
&X00110111



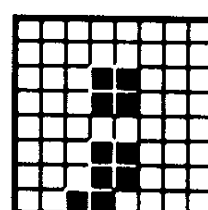
56  
&H38  
&X00111000



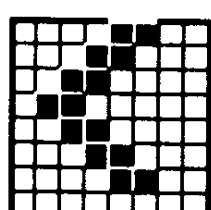
57  
&H39  
&X00111001



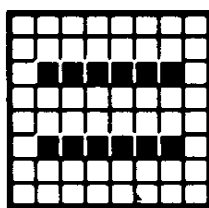
58  
&H3A  
&X00111010



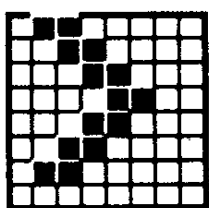
59  
&H3B  
&X00111011



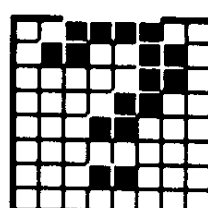
60  
&H3C  
&X00111100



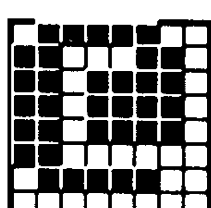
61  
&H3D  
&X00111101



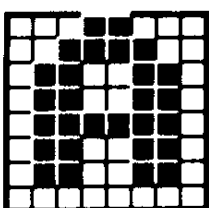
62  
&H3E  
&X00111110



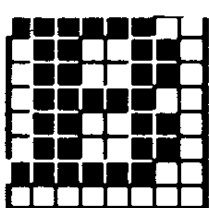
63  
&H3F  
&X00111111



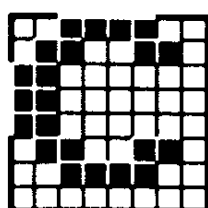
64  
&H40  
&X01000000



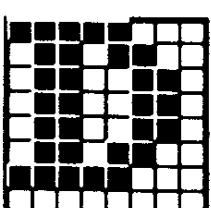
65  
&H41  
&X01000001



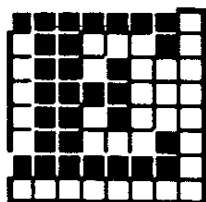
66  
&H42  
&X01000010



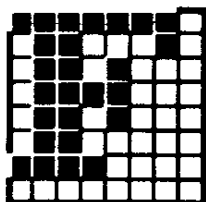
67  
&H43  
&X01000011



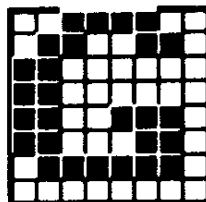
68  
&H44  
&X01000100



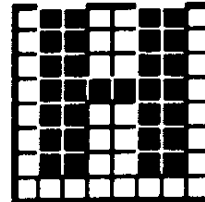
69  
&H45  
&X01000101



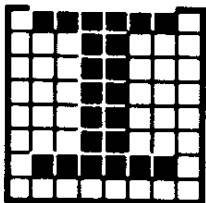
70  
&H46  
&X01000110



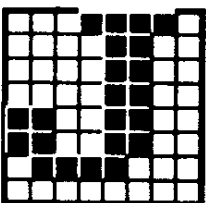
71  
&H47  
&X01000111



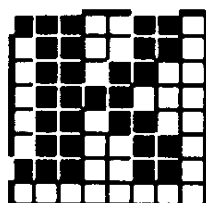
72  
&H48  
&X01001000



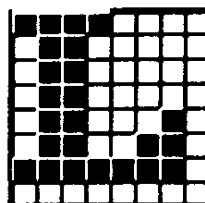
73  
&H49  
&X01001001



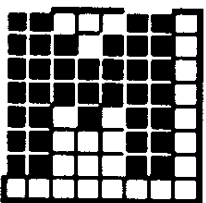
74  
&H4A  
&X01001010



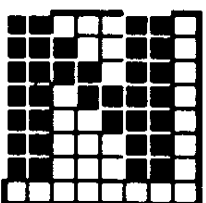
75  
&H4B  
&X01001011



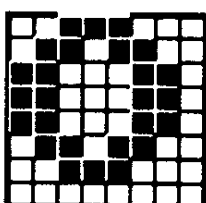
76  
&H4C  
&X01001100



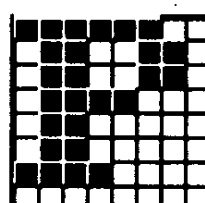
77  
&H4D  
&X01001101



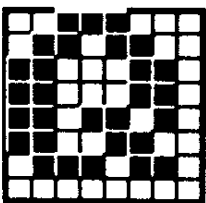
78  
&H4E  
&X01001110



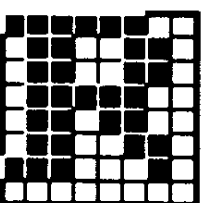
79  
&H4F  
&X01001111



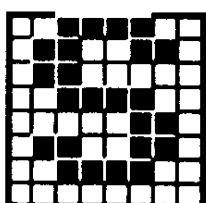
80  
&H50  
&X01010000



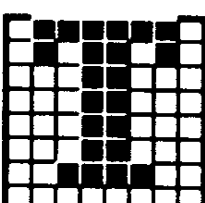
81  
&H51  
&X01010001



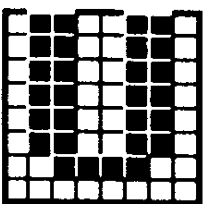
82  
&H52  
&X01010010



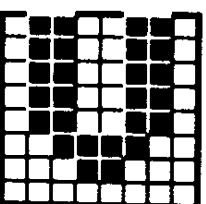
83  
&H53  
&X01010011



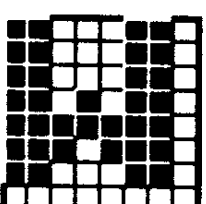
84  
&H54  
&X01010100



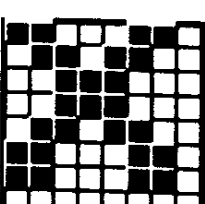
85  
&H55  
&X01010101



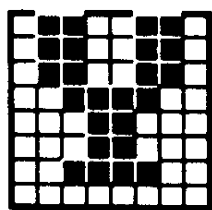
86  
&H56  
&X01010110



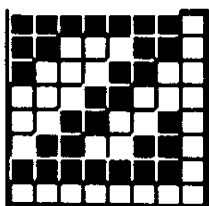
87  
&H57  
&X01010111



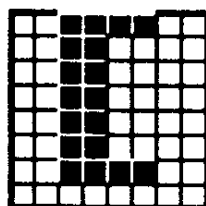
88  
&H58  
&X01011000



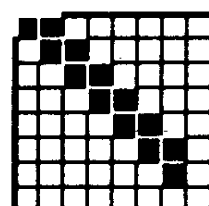
89  
&H59  
&X01011001



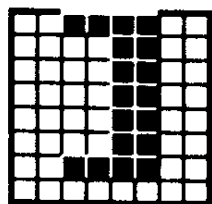
90  
&H5A  
&X01011010



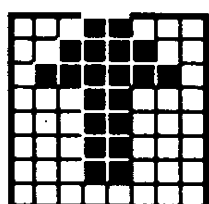
91  
&H5B  
&X01011011



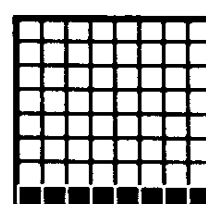
92  
&H5C  
&X01011100



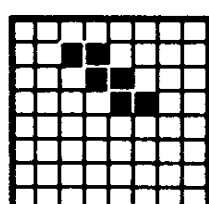
93  
&H5D  
&X01011101



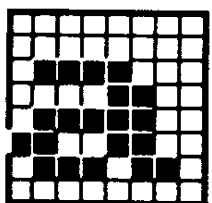
94  
&H5E  
&X01011110



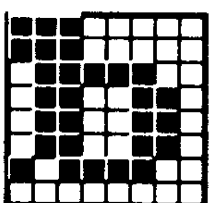
95  
&H5F  
&X01011111



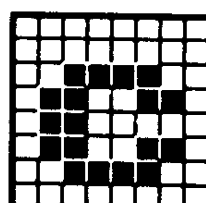
96  
&H60  
&X01100000



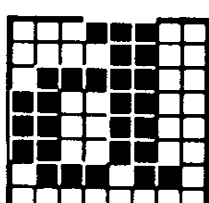
97  
&H61  
&X01100001



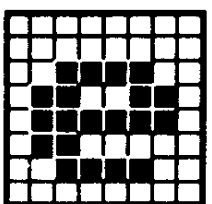
98  
&H62  
&X01100010



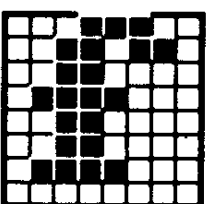
99  
&H63  
&X01100011



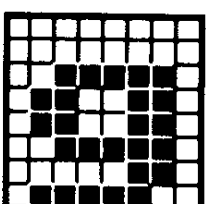
100  
&H64  
&X01100100



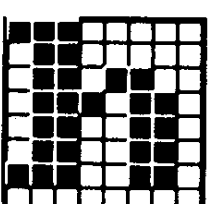
101  
&H65  
&X01100101



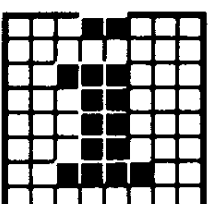
102  
&H66  
&X01100110



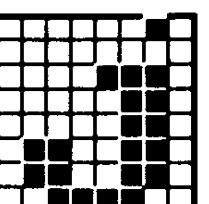
103  
&H67  
&X01100111



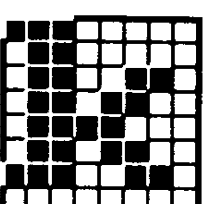
104  
&H68  
&X01101000



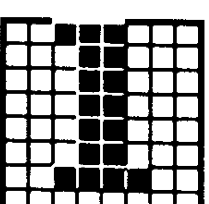
105  
&H69  
&X01101001



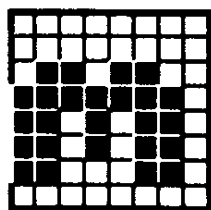
106  
&H6A  
&X01101010



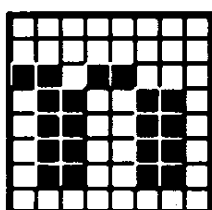
107  
&H6B  
&X01101011



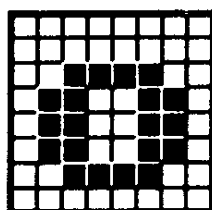
108  
&H6C  
&X01101100



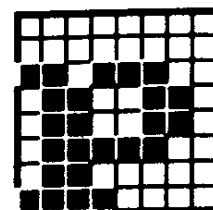
109  
&H6D  
&X01101101



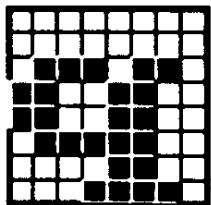
110  
&H6E  
&X01101110



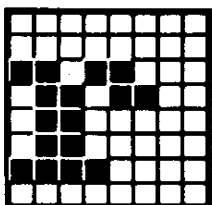
111  
&H6F  
&X01101111



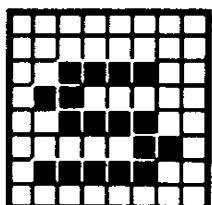
112  
&H70  
&X01110000



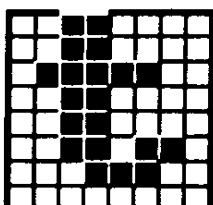
113  
&H71  
&X01110001



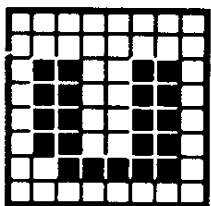
114  
&H72  
&X01110010



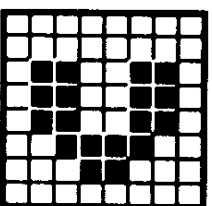
115  
&H73  
&X01110011



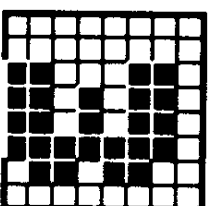
116  
&H74  
&X01110100



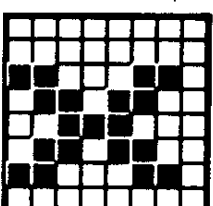
117  
&H75  
&X01110101



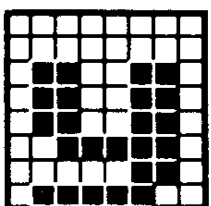
118  
&H76  
&X01110110



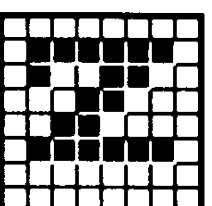
119  
&H77  
&X01110111



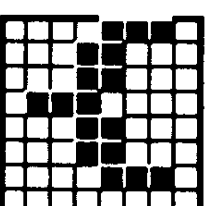
120  
&H78  
&X01111000



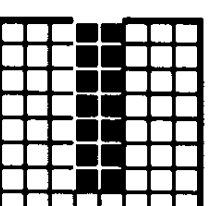
121  
&H79  
&X01111001



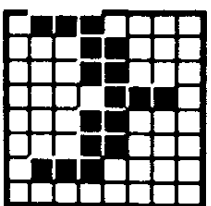
122  
&H7A  
&X01111010



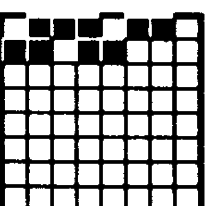
123  
&H7B  
&X01111011



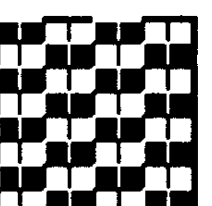
124  
&H7C  
&X01111100



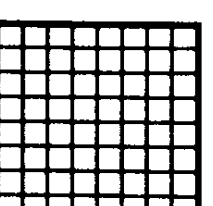
125  
&H7D  
&X01111101



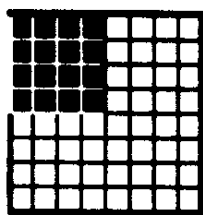
126  
&H7E  
&X01111110



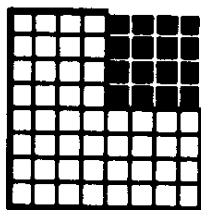
127  
&H7F  
&X01111111



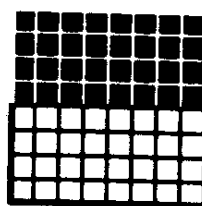
128  
&H80  
&X10000000



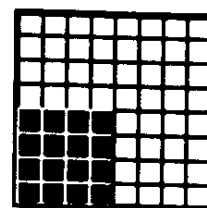
129  
&H81  
&X10000001



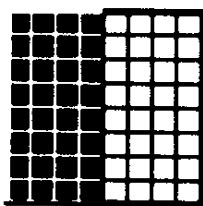
130  
&H82  
&X10000010



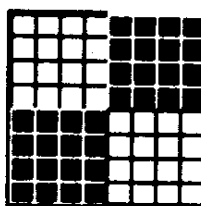
131  
&H83  
&X10000011



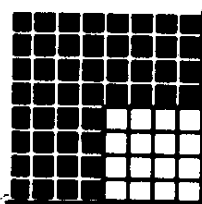
132  
&H84  
&X10000100



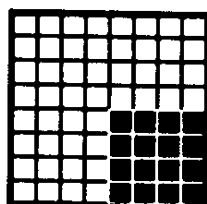
133  
&H85  
&X10000101



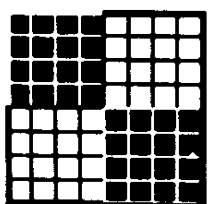
134  
&H86  
&X10000110



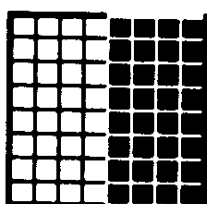
135  
&H87  
&X10000111



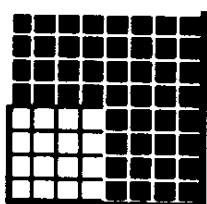
136  
&H88  
&X10001000



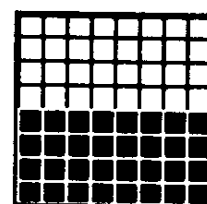
137  
&H89  
&X10001001



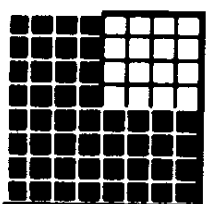
138  
&H8A  
&X10001010



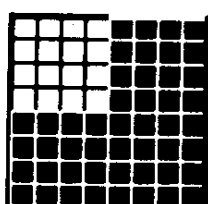
139  
&H8B  
&X10001011



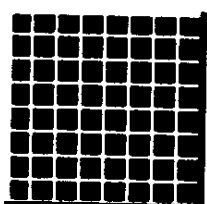
140  
&H8C  
&X10001100



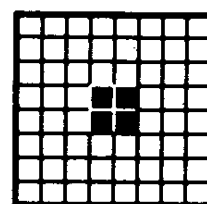
141  
&H8D  
&X10001101



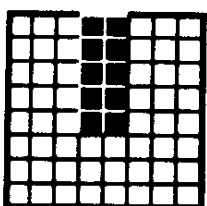
142  
&H8E  
&X10001110



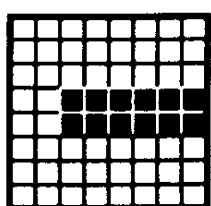
143  
&H8F  
&X10001111



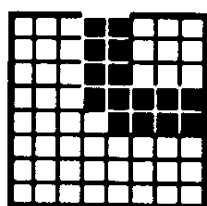
144  
&H90  
&X10010000



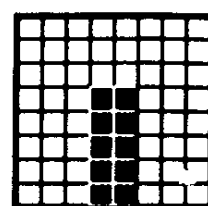
145  
&H91  
&X10010001



146  
&H92  
&X10010010

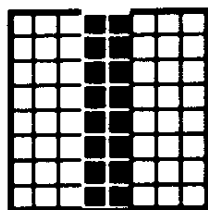


147  
&H93  
&X10010011

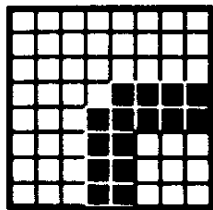


148  
&H94  
&X10010100

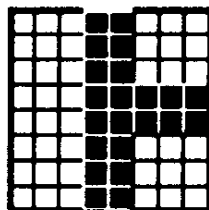




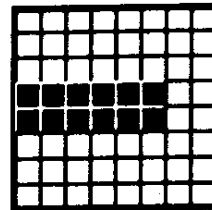
149  
&H95  
&X10010101



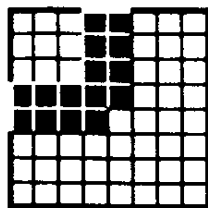
150  
&H96  
&X10010110



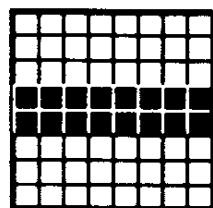
151  
&H97  
&X10010111



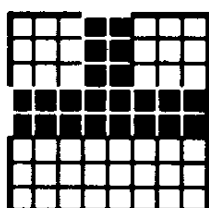
152  
&H98  
&X10011000



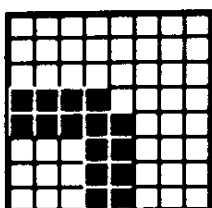
153  
&H99  
&X10011001



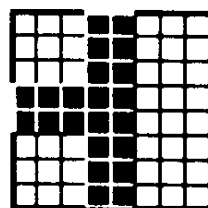
154  
&H9A  
&X10011010



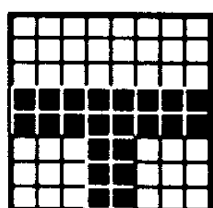
155  
&H9B  
&X10011011



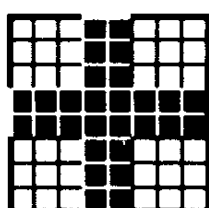
156  
&H9C  
&X10011100



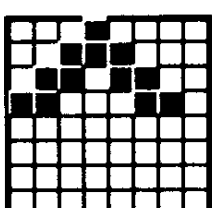
157  
&H9D  
&X10011101



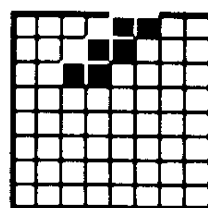
158  
&H9E  
&X10011110



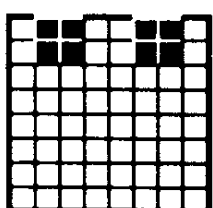
159  
&H9F  
&X10011111



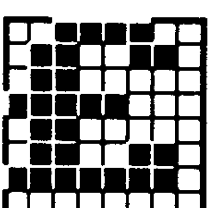
160  
&HA0  
&X10100000



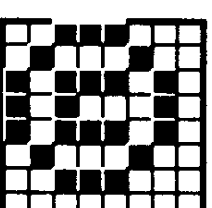
161  
&HA1  
&X10100001



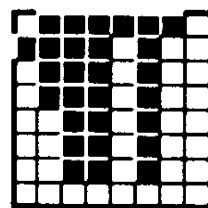
162  
&HA2  
&X10100010



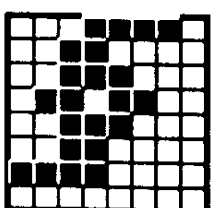
163  
&HA3  
&X10100011



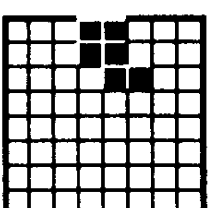
164  
&HA4  
&X10100100



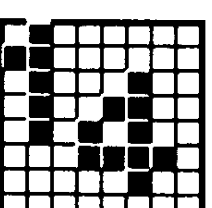
165  
&HA5  
&X10100101



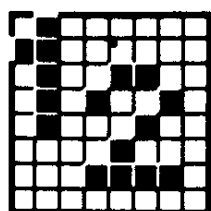
166  
&HA6  
&X10100110



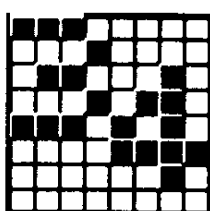
167  
&HA7  
&X10100111



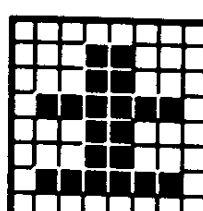
168  
&HA8  
&X10101000



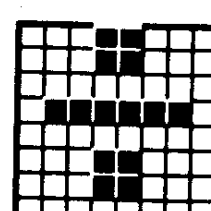
169  
&HA9  
&X10101001



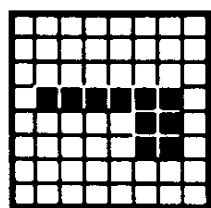
170  
&HAA  
&X10101010



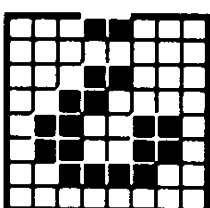
171  
&HAB  
&X10101011



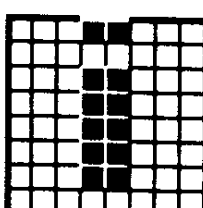
172  
&HAC  
&X10101100



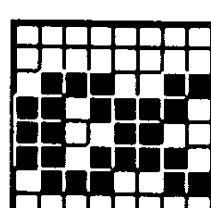
173  
&HAD  
&X10101101



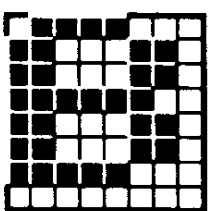
174  
&HAE  
&X10101110



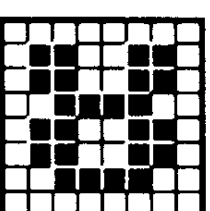
175  
&HAF  
&X10101111



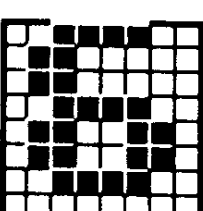
176  
&HB0  
&X10110000



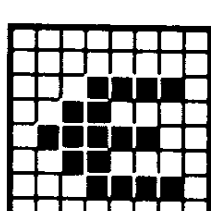
177  
&HB1  
&X10110001



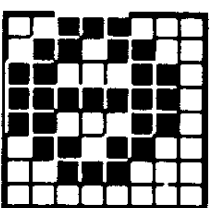
178  
&HB2  
&X10110010



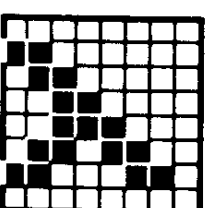
179  
&HB3  
&X10110011



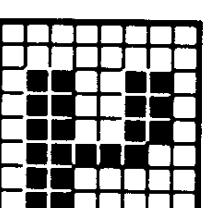
180  
&HB4  
&X10110100



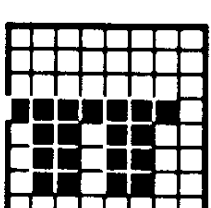
181  
&HB5  
&X10110101



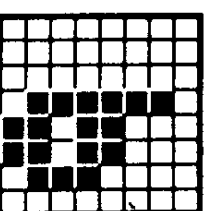
182  
&HB6  
&X10110110



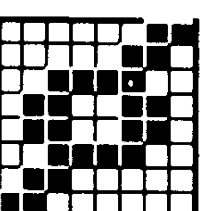
183  
&HB7  
&X10110111



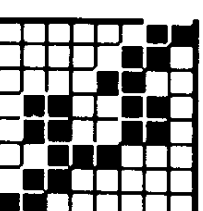
184  
&HB8  
&X10111000



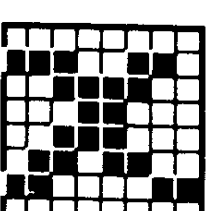
185  
&HB9  
&X10111001



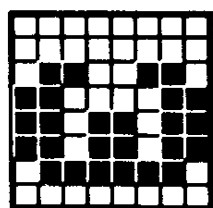
186  
&HBA  
&X10111010



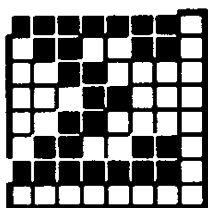
187  
&HBB  
&X10111011



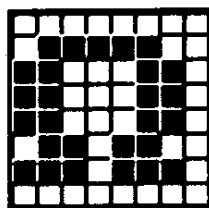
188  
&HBC  
&X10111100



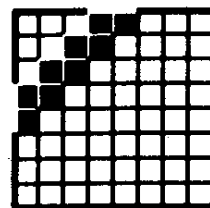
189  
&HBD  
&X10111101



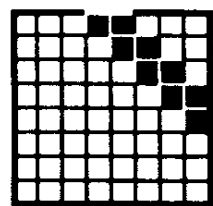
190  
&HBE  
&X10111110



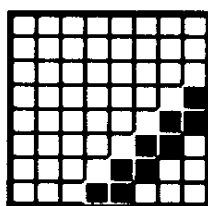
191  
&HBF  
&X10111111



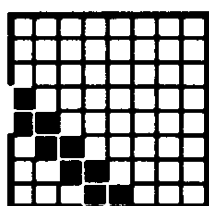
192  
&HC0  
&X11000000



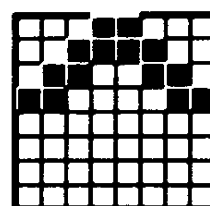
193  
&HC1  
&X11000001



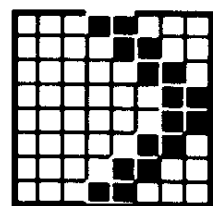
194  
&HC2  
&X11000010



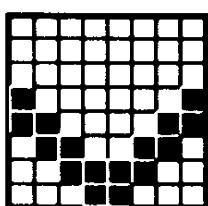
195  
&HC3  
&X11000011



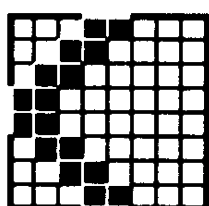
196  
&HC4  
&X11000100



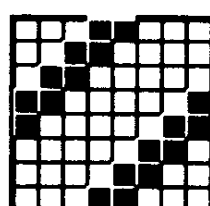
197  
&HC5  
&X11000101



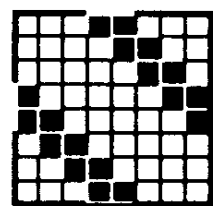
198  
&HC6  
&X11000110



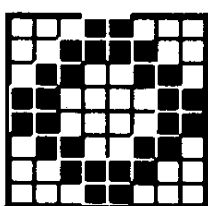
199  
&HC7  
&X11000111



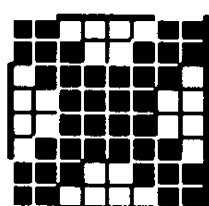
200  
&HC8  
&X11001000



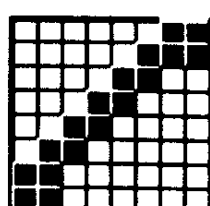
201  
&HC9  
&X11001001



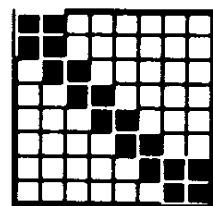
202  
&HCA  
&X11001010



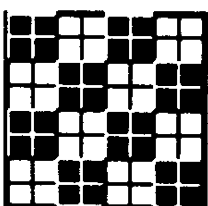
203  
&HCB  
&X11001011



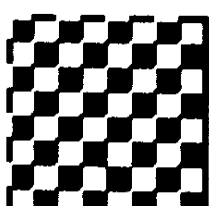
204  
&HCC  
&X11001100



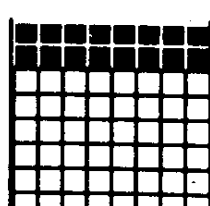
205  
&HCD  
&X11001101



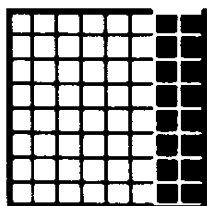
206  
&HCE  
&X11001110



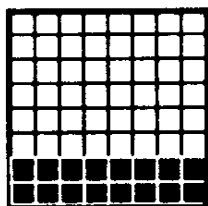
207  
&HCF  
&X11001111



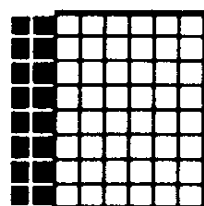
208  
&HD0  
&X11010000



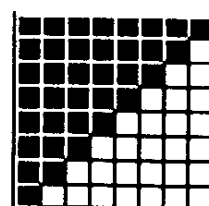
209  
&HD1  
&X11010001



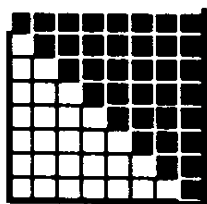
210  
&HD2  
&X11010010



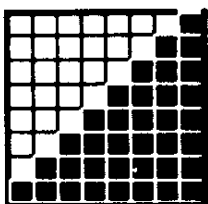
211  
&HD3  
&X11010011



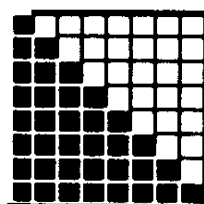
212  
&HD4  
&X11010100



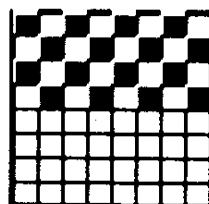
213  
&HD5  
&X11010101



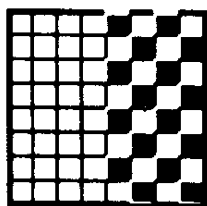
214  
&HD6  
&X11010110



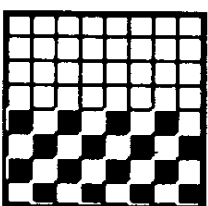
215  
&HD7  
&X11010111



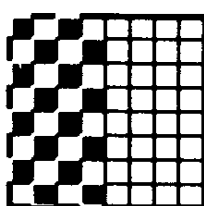
216  
&HD8  
&X11011000



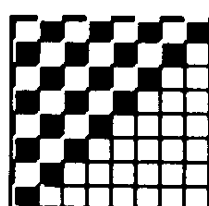
217  
&HD9  
&X11011001



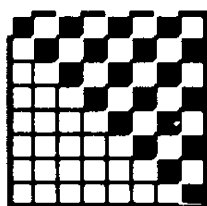
218  
&HDA  
&X11011010



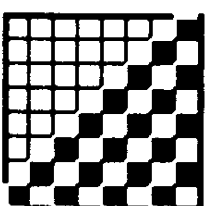
219  
&HDB  
&X11011011



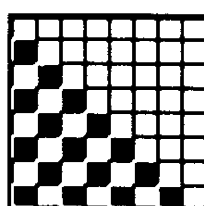
220  
&HDC  
&X11011100



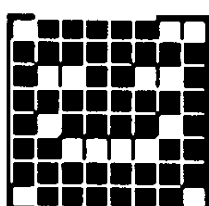
221  
&HDD  
&X11011101



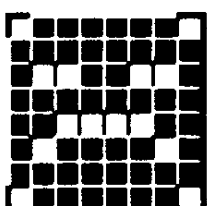
222  
&HDE  
&X11011110



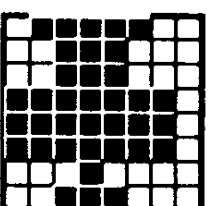
223  
&HDF  
&X11011111



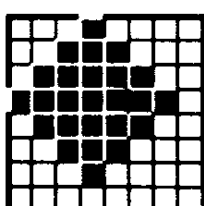
224  
&HE0  
&X11100000



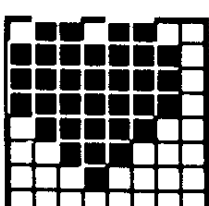
225  
&HE1  
&X11100001



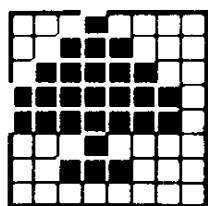
226  
&HE2  
&X11100010



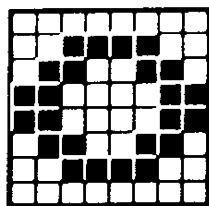
227  
&HE3  
&X11100011



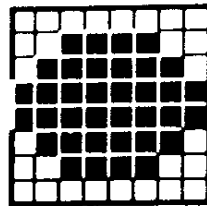
228  
&HE4  
&X11100100



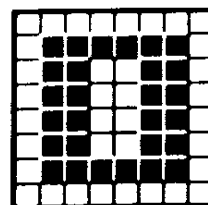
229  
&HE5  
&X11100101



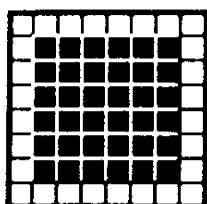
230  
&HE6  
&X11100110



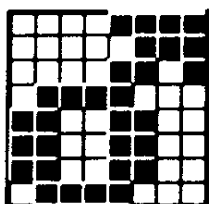
231  
&HE7  
&X11100111



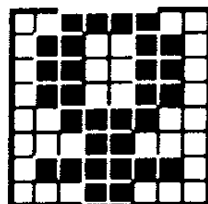
232  
&HE8  
&X11101000



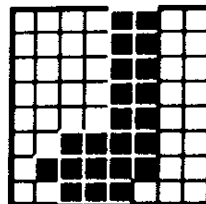
233  
&HE9  
&X11101001



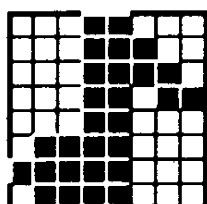
234  
&HEA  
&X11101010



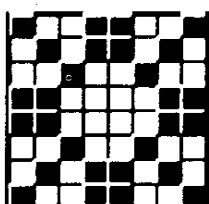
235  
&HEB  
&X11101011



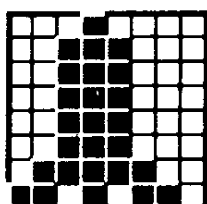
236  
&HEC  
&X11101100



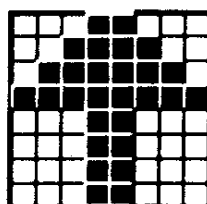
237  
&HED  
&X11101101



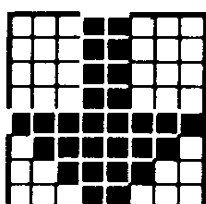
238  
&HEE  
&X11101110



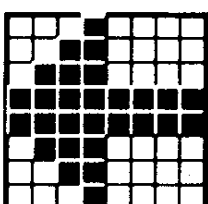
239  
&HEF  
&X11101111



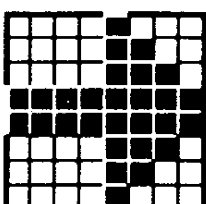
240  
&HF0  
&X11110000



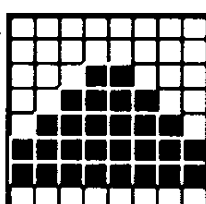
241  
&HF1  
&X11110001



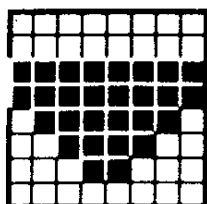
242  
&HF2  
&X11110010



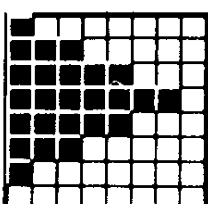
243  
&HF3  
&X11110011



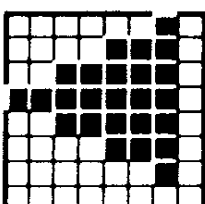
244  
&HF4  
&X11110100



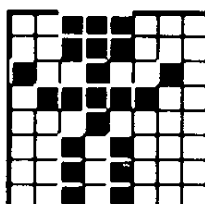
245  
&HF5  
&X11110101



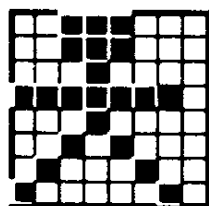
246  
&HF6  
&X11110110



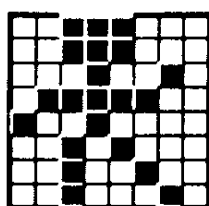
247  
&HF7  
&X11110111



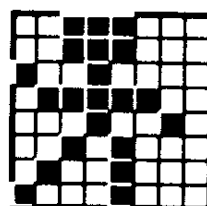
248  
&HF8  
&X11111000



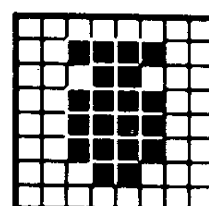
249  
&HF9  
&X11111001



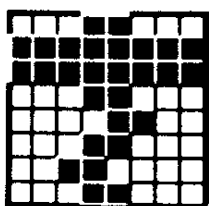
250  
&HFA  
&X11111010



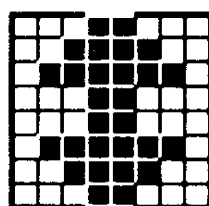
251  
&HFB  
&X11111011



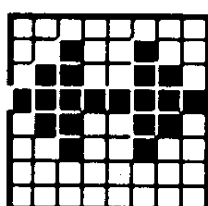
252  
&HFC  
&X11111100



253  
&HFD  
&X11111101

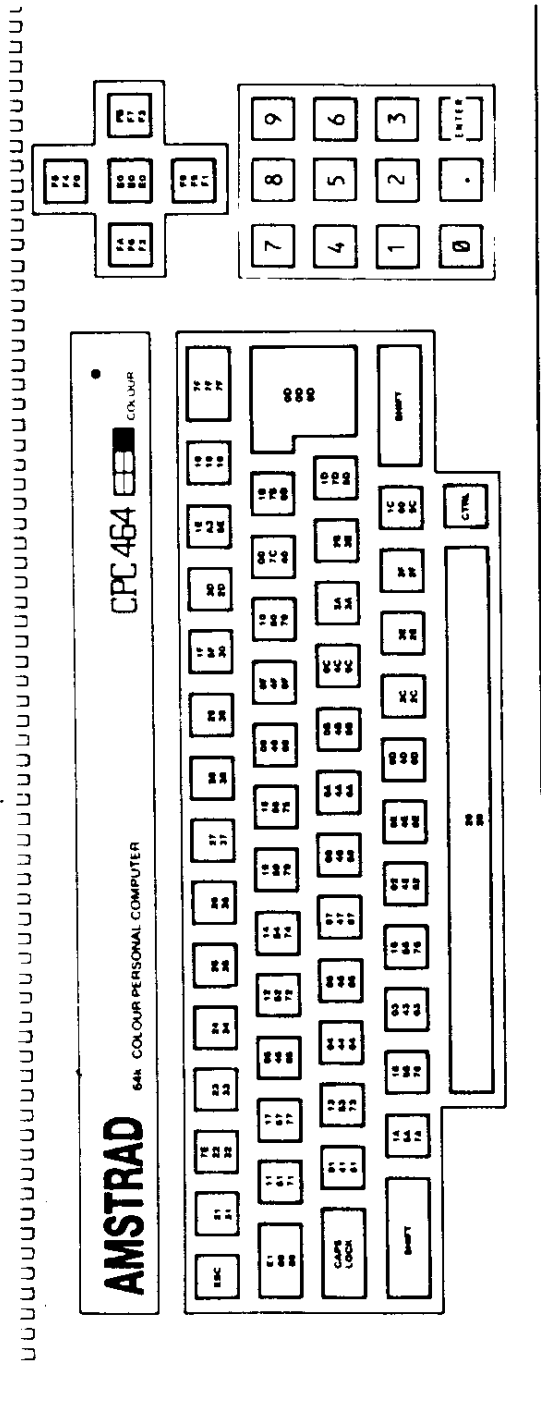


254  
&HFE  
&X11111110



255  
&HFF  
&X11111111

Valores ASCII prescritos como standard



JOYSTICK 1

26  
36

FIRE 2 FIRE 1  
12 07 06 14  
52 ← 47 → 54  
72 67 66 74

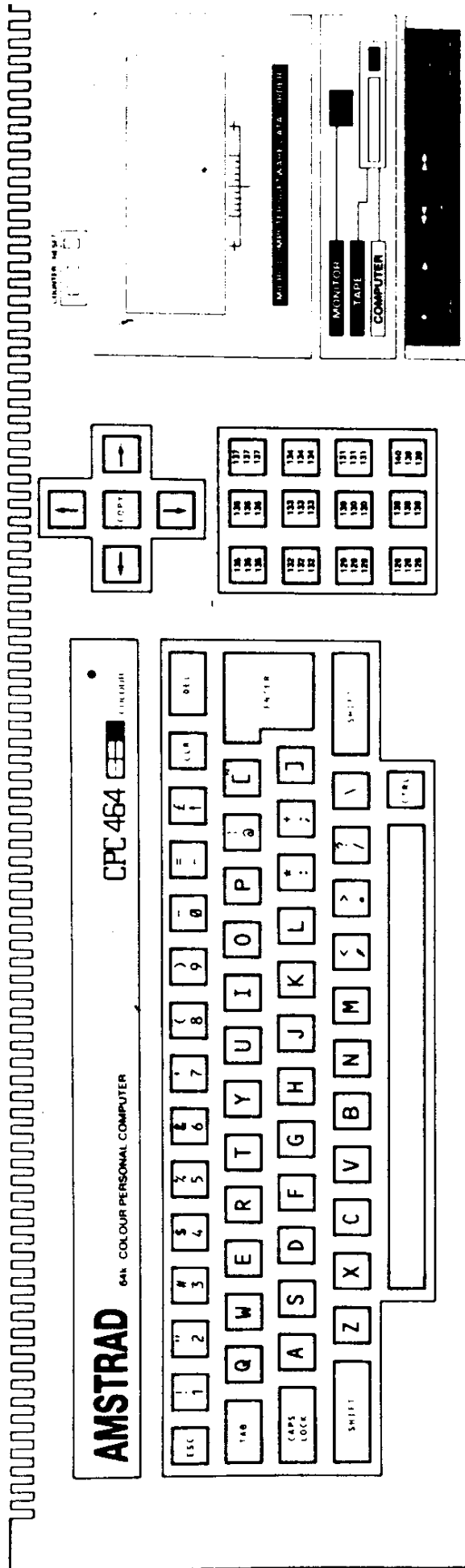
25  
35

JOYSTICK 0

0B  
0B

FIRE 2 FIRE 1  
08 58 5A 09  
08 ← 58 → 5A 09

0A  
0A

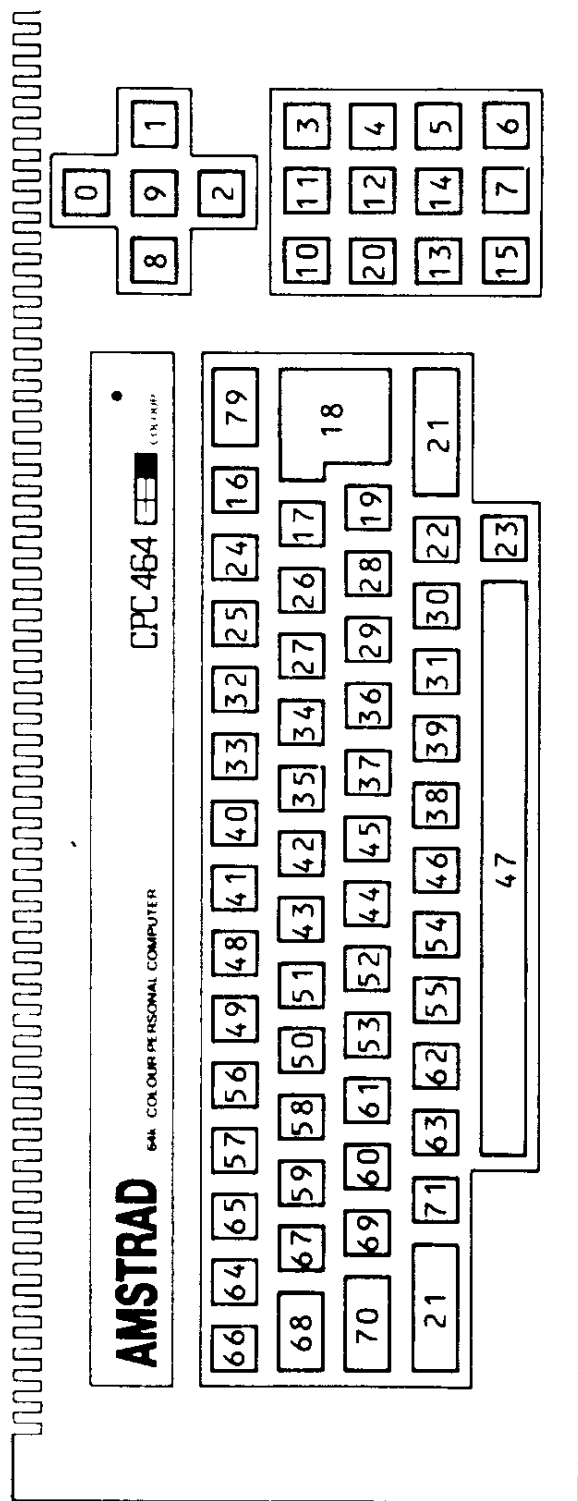


## Caracteres funcionales, standard situación y valores

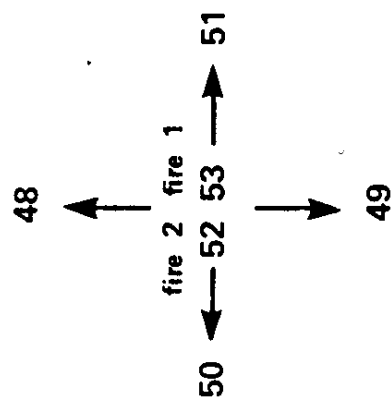
<u>exp char</u>	<u>value</u>	<u>ascii</u>
128	0	30
129	1	31
130	2	32
131	3	33
132	4	34
133	5	35
134	6	36
135	7	37
136	8	38
137	9	39
138	.	2E
139	[enter]	0D
140	run"	52,55,4E,22,0D



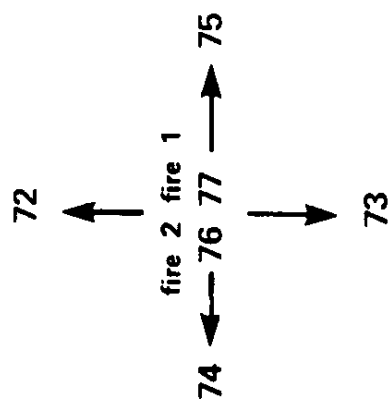
# Códigos de entrada para teclas y joysticks



## JOYSTICK 1



## JOYSTICK 0



## Apéndice IV:

### Introducción al CPC464 para expertos

El CPC464 es un ordenador personal a color de bajo coste, con una generosa, eficaz, y profunda implementación de tecnologías establecidas presentado en estilo diseñado para ofrecer un valor excepcionalmente atractivo para el coste, acoplado con una capacidad sustancial para expansión, y ser así atractivo a ambos, el principiante y el experto.

El hardware y el software ROM ha sido diseñado para proveer a los programadores principiantes y experimentados un entorno amistoso, en el que el software existente pueda ser adaptado a medida, y se pueda escribir nuevo software para sacar ventaja de las muchas facetas soportadas por el CPC464.

**Las características primordiales del sistema son:**

#### **Z80CPU**

El microprocesador más ampliamente usado en los ordenadores personales a lo ancho del mundo, y con el mejor software de base -especialmente porque el CPC464 ofrece el potencial para la implementación del CP/M. La estructura única del manejo de interrupciones, ha facultado al CPC464 para innovar el BASIC con posibilidades **AFTER** y **EVERY**, y otras facilidades de tiempo real que controla el sonido y los temporizadores.

#### **64K RAM**

Se suministra como standard una cantidad generosa de RAM, con más de 42K disponibles realmente para el usuario, gracias a la implementación en ROM de técnicas de solape al implementar el BASIC/

#### **Pantalla**

El CPC464 admite tres modos básicos de manejo en pantalla, incluyendo textos de 80 columnas, un surtido de 27 colores y resolución de hasta 640x200 pixels.

#### **Teclado Real**

Un teclado de estilo mecanográfico, con un racimo de teclas para cursor y un tablero standard separado para entrada numérica, que dobla el uso para propósitos de teclas funcionales.

**Cassette incorporado**

El almacenamiento en la cassette local se dispone como una posibilidad incorporada, permitiendo la operación con ninguna complicación adicional para el usuario con enganche, fijación de nivel, etc. Escribiendo a bien 1Kbaudios o 2Kbaudios (seleccionable por software) con velocidades de lectura automáticamente establecidas por hardware.

**BASIC**

Un BASIC standard en la industria escrito en Inglaterra, más rápido y más versátil de lo que se pudiera esperar de tales BASIC, con muchas ampliaciones para gráficos y sonido, más un extenso soporte en firmware.

**Repertorio de caracteres extendido**

Un repertorio completo de caracteres de 8 bits, incluyendo símbolos y gráficos es accesible en su mayoría vía el teclado, y usando las funciones CHR\$(n).

**Tiempo transcurrido**

Las interrupciones se general por el barrido de cuadros suministrando las facilidades para el tiempo transcurrido.

**Teclas definidas por el usuario**

Hasta 32 teclas pueden ser definidas por el usuario con sertas de 32 caracteres. La capacidad de redefinición incluye parámetros de repetición. Un repertorio completo de 255 caracteres (todo el ASCII y más de 100 extras), opcionalmente redefinibles por el usuario.

**Subrutinas**

Muchas subrutinas ensambladas están disponibles para ser citadas desde BASIC.

**MODOS DE PANTALLA**

Hay tres modos de operación en pantalla:

a) Normal

Modo 1: 40 columnas x 25 líneas, 4 tintas en texto  
320 x 200 pixels, direccionables individualmente en 4 colores.

b) Modo multicolor

20 columnas x 25 líneas, 16 tintas en texto  
160 x 200 pixels, individualmente direccionables en 16 colores

c) Modo de alta resolución

80 columnas x 25 líneas, 2 tintas en texto  
640 x 200 pixels, individualmente direccionables en 2 colores.

### Selección de color

(NB: A través de esta guía, 'negro' ie. luminancia nula, se considera como color para los propósitos de las siguientes descripciones).

El borde puede estipularse a CUALQUIER par de colores sin considerar el modo de pantalla: ie. flasheando; o un sólo color ie. estacionario.

El número de **INKS** disponibles depende del modo de pantalla elegido. Cada **INK** puede fijarse a un par de colores, ie. flasheando; o un sólo color ie. estacionario. El número de tintas usables en cualquier momento depende del modo de pantalla previamente definido. El **PAPER** de texto, **PEN** de texto y **PEN** de gráficos, puede fijarse a una tinta disponible.

La escritura de texto puede fijarse para que sea traslúcida u opaca: ie. bien ignorará el color del papel y sobre-escribirá los gráficos, o sobre-escribirá completamente el fondo.

### Ventanas

El usuario puede elegir hasta ocho ventanas de textos en las que se escriben caracteres, y también una ventana de gráficos en la que se pueden efectuar 'ploteado'.

Las ventanas se restauran a lo prescrito cuando se fija el modo de pantalla.

NB: Si la ventana de texto es equivalente a toda la pantalla (prescrito), se logra el despliegue rápido por hardware. Si la ventana de texto es menor que la pantalla disponible, el despliegue se consigue por software, que es correspondientemente más lento.

### Cursor

El cursor está inhibido durante los períodos en que la CPU no requiere entrada por teclado, actuando así como un prontador automático. El cursor se representa por un cuadrado de color invertido.

### Sonido polifónico

Las facilidades de sonido del CPC464 están generadas usando la pastilla generadora de sonido standard en la industrial de General Instruments familia AY8910. La pastilla opera con tres canales (voces) cada uno de los cuales puede ser independientemente fijado en tono y actitud. Se puede añadir ruido blanco como se requiera.

Los tres canales aparecen como izquierda, derecha y centro (usando la clavija de extension estéreo). El altavoz interno produce una salida combinada monoaural.

El software provee facilidades para el control de envolvente en amplitud y tono. La envolvente de amplitud interna de la pastilla generadora de sonido no se usa normalmente.

**Portal de impresora**

Está dotado de un portal de impresora paralelo compatible Centronics standard en la industria, usando la señal 'Busy' para efectuar operaciones de apretón de manos.

**Apoyo de expansión**

Muchas interfaces hardware están disponibles vía bloques de puentes o direccionamiento indirecto para proveer facilidades de expansión de software. AMSTRAD presentará, entre otros items, ductoras de disco, interfaces series con ductora de software en ROM, etc.

**Coordenadas**

El origen de texto es la esquina superior izquierda de la pantalla, y las posiciones físicas en pantalla cambian con el modo de imagen.

El origen de gráficos es la esquina inferior izquierda, y supone que la pantalla está en todo momento en el modo de alta resolución -aunque los cálculos de tintas serán efectuados correctamente en todos los modos de imagen.

**NB**

En el modo normal de pantalla, cada pixel tiene DOS direcciones horizontales, y se puede usar una u otra. En el modo multicolor, cada pixel tiene CUATRO direcciones horizontales, y cualquiera de las cuatro puede usarse para definir la posición del pixel.

El eje vertical tiene coordenadas de 0 a 399 que son divididas por dos y truncadas para dar la posición física en la gama 0 a 199. Eso asegura que las relaciones de aspecto esperadas se preservan en pantalla.

**ROMS de expansión**

Todas las ROMS ocupan los 16K de la cima de la memoria (donde reside el BASIC) y hay facilidades en el firmware para citar a 240 adicionales, 16K extra ROMS. Debe emplearse externamente una cierta cantidad de circuitos decodificadores de direcciones a la máquina base como parte del hardware de expansión ROM).

**Panorámica:**

Un breve resumen de las principales características del hardware y firmware del CPC464

**1) Hardware**

1.1) Dentro de la carcasa principal CPC464:  
Computador, teclado, cassette y altavoz. Salidas RGB y luminancia.

**1.1.1) LSI Chips**

Z80A un procesador ejecutando a 4MHz.

64K bytes de 64K x 1 RAM dinámica refrescada por accesos a la memoria de pantalla.

32 K bites de ROM conteniendo BASIC y el sistema operativo (OS).

Una ringla lógica customizada, incorpora casi toda la lógica no incluida previamente en los chips LSI; particularmente temporización, generación de color y circuitería DMA.

La pastilla controladora CRT 6845 genera señales de barrido para la RAM video.

### NB

El mapa de memoria es complejo y cambia con el modo de pantalla. El CRTC puede usarse para efectuar despliegue (lateralmente para anchura de pantalla 1/40) y arrolle (arriba y abajo cada 8 líneas de barrido) si el software lo requiere. Los parámetros dentro del CRTC seleccionan el número de líneas, la cadencia de cuadro, la anchura y posición de los bordes.

El chip generador de sonido GI AY-3-8912: 3 boces. El sonido se toma de los 3 canales y se mezcla equilibradamente para producir una salida mono en el altavoz incorporado, controlado por perilla de volumen. También hay una salida externa estéreo donde...

Izquierda=Canal A + 1/2 Canal C. Derecha=Canal B + 1/2 Canal C. Este chip también recibe información del barrido de los portales del teclado y joysticks.

Una pastilla paralelo I/O 8255 interconecta el bus al chip de sonido GI. También explora el teclado, portal de joystick, eslabones de opción y controla el cassette.

### 1.1.2) Enchufes externos

Conectores de borde PCB para expansiones de propósito general (12) y una impresora externa (12) (paralelo Centronics). Enchufes para joystick (14) (9 pines D-Tipo), salida estéreo de sonido (15), salida video (10) (RGB y sync o video compuesto o luminancia y sync).

### 1.2) Fuera de la carcasa

El sistema CPC464 viene con una elección de dos clases de entrada directa de monitor de video, incluyendo cada una, una fuente de alimentación 5V para el ordenador, diseñado para adaptarse a los standards locales de voltaje de red en el país de venta. Además, hay una PSU opcional y un modulador UHF TV, el MP1.

Se requerirán cables para conectar una impresora compatible Centronics y una unidad Hi-fi.

### 1.3) Especificación de pantalla

La pantalla opera en 16K de memoria. La máquina tiene un surtido de 27 colores que pueden ser elegidos libremente para formar una paleta. El número de tintas en la paleta depende del modo de pantalla. Ciertas tintas pueden fijarse al mismo color si se requiere. Los pixels de la pantalla se definen como motas de una tinta particular.

(Nota que el fondo, o papel, requiere una tinta de la paleta disponible). Hay un **BORDER** rodeando el área activa de imagen que puede fijarse independientemente a cualquiera de los 27 colores.

Modo	Nº de tintas	Motas verts	Motas horiz	Chars horiz
Normal	4	200	320	40
Alta Res.	1	200	640	80
Multicolor	16	200	160	20

**NB:** El ordenador producirá un efecto de escala de grises si se proyecta sobre un monitor monocromo. El orden de los colores en brillo creciente es como sigue:

TABLA MAESTRA DE COLORES

Nº Tinta	Color Tinta	Nº Tinta	Color Tinta
0	Negro	14	Azul Pastel
1	Azul	15	Naranja
2	Azul Brillante	16	Rosa
3	Rojo	17	Magenta Pastel
4	Magenta	18	Verde Brillante
5	Malva	19	Verde Marino
6	Rojo Brillante	20	Ciano Brillante
7	Púrpura	21	Verde Lima
8	Magenta Brillante	22	Verde Pastel
9	Verde	23	Ciano Pastel
10	Ciano	24	Amarillo Brillante
11	Azul Cielo	25	Amarillo Pastel
12	Amarillo	26	Blanco Brillante
13	Blanco		

#### 1.4) Mapa de Memoria

Los 64K de RAM se reparten como sigue:

Observa que parte de la ROM solapa la RAM de pantalla, por lo tanto liberando el área máxima posible para la RAM de usuario durante las operaciones BASIC.

ROM SECTION 0 <	0000H	
	3FFFH	
	4000H	
	7FFFH	
	8000H	
	BFFFH	
ROM SECTION 1 <	C000H	> RAM [Screen 3]
	FFFFH	

Cuando hay ambas, RAM y ROM, en una dirección, entonces **READING** accesa la ROM y **WRITING** accesa la RAM. Además, la sección ROM puede ser desactivada, permitiendo acceder para leer la RAM en la misma dirección.

### 1.5) Habilidad de ampliación

#### 1.5.1) ROMS colaterales

Está hecha la provisión para que puedan elegirse ROMS adicionales en lugar de cualquier parte de la ROM incorporada. El arbitraje de direcciones y la lógica de selección de bancadas estará contenida en un módulo conectado al bus de expansión, pero todas las señales requeridas están llevadas al bus de expansión.

#### 1.5.2) RAM adicional

Se puede conmutar RAM adicional en cualquier parte de la RAM incorporada. El arbitraje de direcciones y la lógica de selección de bancadas estará contenida en un módulo conectado al bus de expansión, pero todas las señales requeridas están traídas al bus de expansión. Esta memoria será sólo de lectura y un esquema especial involucrando mapas y/o se requerirá para escribir en esta rama adicional desde el ordenador.

#### 1.5.3) I/O Adicional

La mayoría de las direcciones de portales I/O están reservadas por el ordenador, y en particular las direcciones por debajo 7Fxx no serán usadas en absoluto. Lo siguiente puede ser usado por hardware externo.

**F8xx, F9xx, FАxx, FBxx**

El bus de expansión para periféricos debe decodificar las direcciones en **A0** a **A7** mientras que la dirección **A10** está baja. Los canales I/O del bus de expansión en la gama de direcciones **F800** a **FBFF** están reservadas como sigue:

#### Addresses A0-A7

00	- 7B	** Do not use **
7C	- 7F	Reserved for disk interface
80	- BB	** Do not use **
BC	- BF	Reserved for future use
C0	- DB	** Do not use **
DC	- DF	Reserved for communications interfaces
E0	- FF	Available for user peripherals

Observa que las instrucciones Z80 que colocan el registro **B** en la parte superior del bus de direcciones (**A15-A8**) debe usarse.



## 2) Teclado

Un restaurado completo se opera al pulsar conjuntamente **CTRL**, **SHIFT** y **ESC**. Las teclas que provocan caracteres visivos o movimientos de cursor repetirán automáticamente bajo control del firmware, excluyendo todas las teclas del tablero numérico.

**ESC** suspende la ejecución del programa. Seguido de otro **ESC** termina la ejecución. Seguido por cualquier otro carácter, reanuda la ejecución del programa.

**CAPS LOCK** es un basculador, operado por la tecla de enclavamiento de mayúsculas. El enclavamiento del turno mayúsculas/minúsculas es un basculador operado por **CTRL**, **CAPS LOCK** pulsadas conjuntamente.

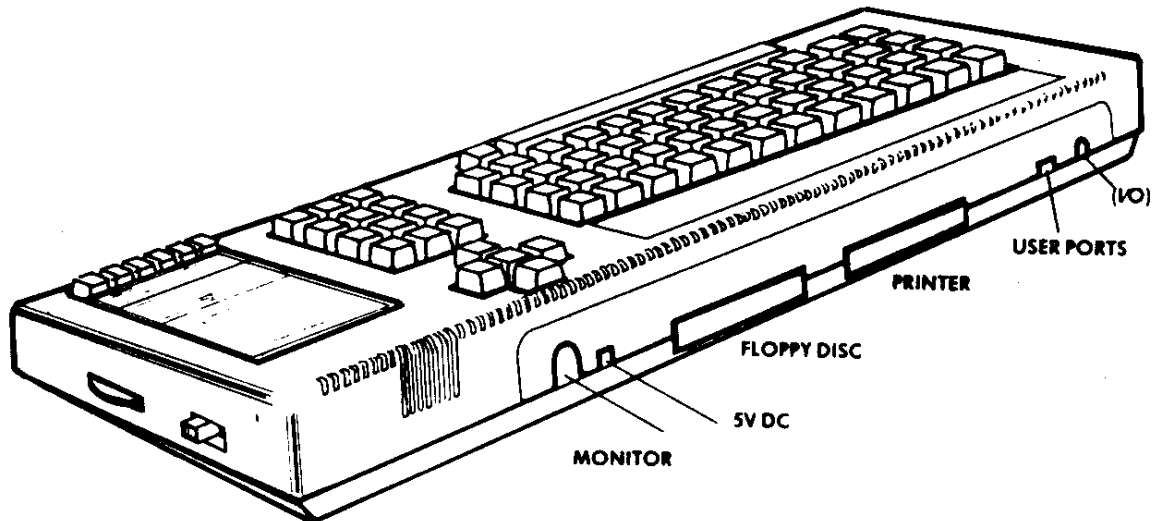
El cursor de copia se separa del cursor de entrada, operando **SHIFT** junto con las teclas de cursor. La entrada en el buzón puede conseguirse para el carácter debajo del cursor de copia, pulsando la tecla **COPY**.

Las teclas de cursor están pensadas para permitir la edición del buzón de entrada, que puede ocupar unas cuantas líneas de pantalla. Las teclas de cursor pueden usarse para colocar el comienzo de la imposición por teclado en cualquier posición de pantalla anterior a cualquier ingreso por teclado que se esté recibiendo. Una vez que se ha recibido cualquier ingreso por teclado se prefija la posición de pantalla. El nuevo texto ingresado sobrescribirá cualquier contenido existente en la pantalla en esa posición.

**DEL** es supresión retrocediendo y **CLR** es supresión avanzando.

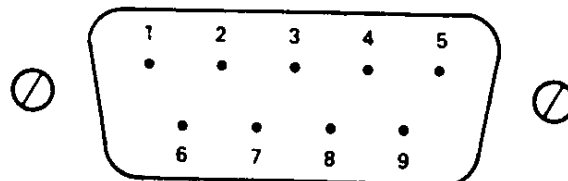
# Apéndice V:

## Conexión del panel posterior al CPC464



PADDLE PORT CONNECTOR (9 PIN D)

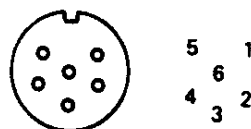
VIEWED FROM REAR



PIN 1	UP	PIN 6	FIRE 2
PIN 2	DOWN	PIN 7	FIRE 1
PIN 3	LEFT	PIN 8	COMMON
PIN 4	RIGHT	PIN 9	COM 2
PIN 5	SPARE		

VIDEO OUTPUT CONNECTOR (6 PIN DIN)

VIEWED FROM REAR



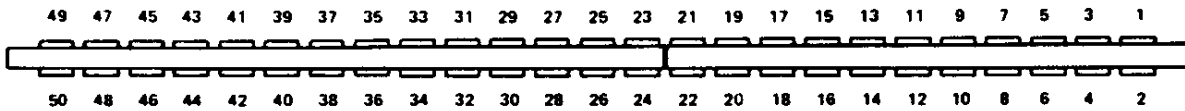
PIN 1	RED	PIN 4	SYNC
PIN 2	GREEN	PIN 5	GND
PIN 3	BLUE	PIN 6	LUM

# INTERCONEXIONES DE USUARIO

## EXPANSION PORT

50 WAY 0.1 EDGE CONNECTOR

VIEWED FROM REAR

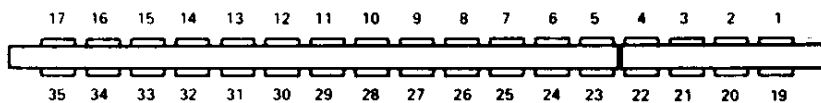


PIN 1	SOUND	PIN 18	A0	PIN 35	INT
PIN 2	GND	PIN 19	D7	PIN 36	NMI
PIN 3	A15	PIN 20	D6	PIN 37	BUSRD
PIN 4	A14	PIN 21	D5	PIN 38	BUSAK
PIN 5	A13	PIN 22	D4	PIN 39	READY
PIN 6	A12	PIN 23	D3	PIN 40	BUS RESET
PIN 7	A11	PIN 24	D2	PIN 41	RESET
PIN 8	A10	PIN 25	D1	PIN 42	ROMEN
PIN 9	A9	PIN 26	D0	PIN 43	ROMDIS
PIN 10	A8	PIN 27	+5v	PIN 44	RAMRD
PIN 11	A7	PIN 28	MREQ	PIN 45	RAMDIS
PIN 12	A6	PIN 29	M1	PIN 46	CURSOR
PIN 13	A5	PIN 30	RFSH	PIN 47	L. PEN
PIN 14	A4	PIN 31	IORQ	PIN 48	EXP
PIN 15	A3	PIN 32	RD	PIN 49	GND
PIN 16	A2	PIN 33	WR	PIN 50	
PIN 17	A1	PIN 34	HALT		

## PRINTER PORT

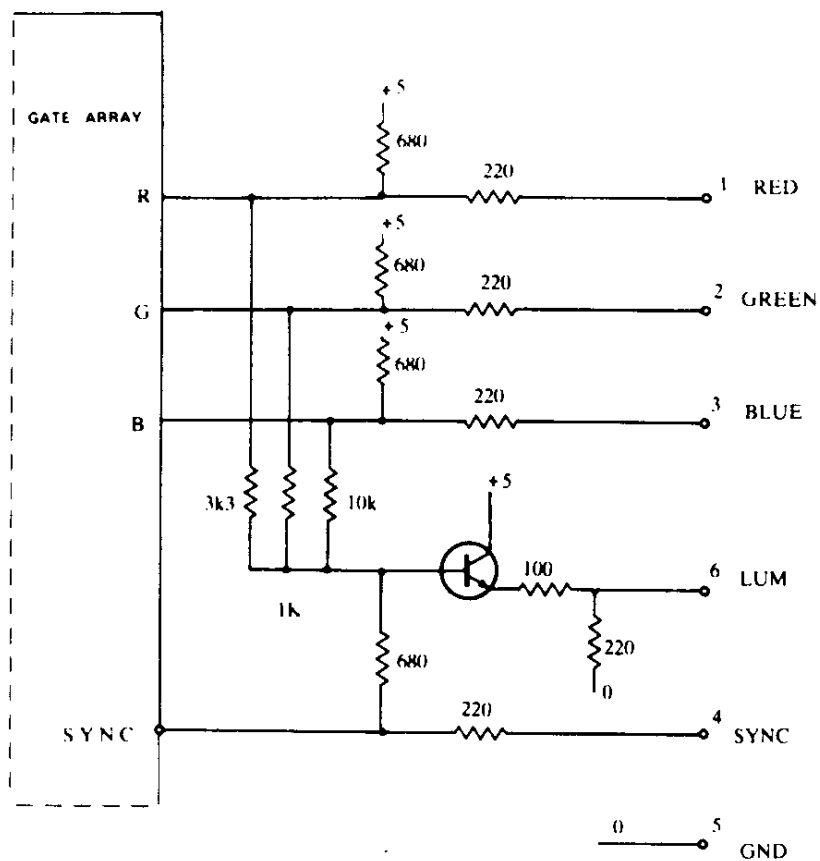
34 WAY 0.1 EDGE CONNECTOR

VIEWED FROM REAR

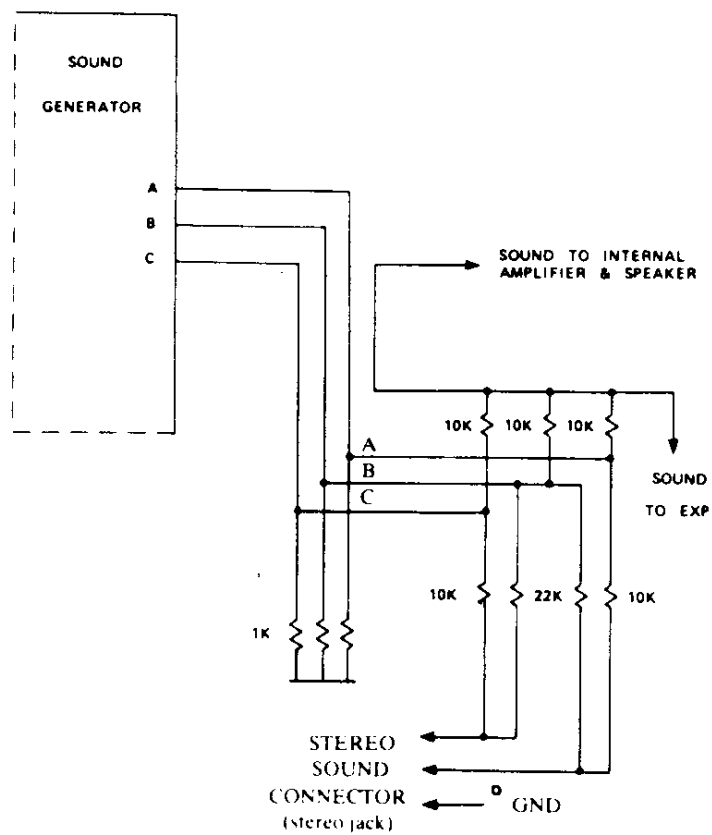


PIN 1	STROBE	PIN 19	GND
PIN 2	D0	PIN 20	GND
PIN 3	D1	PIN 21	GND
PIN 4	D2	PIN 22	GND
PIN 5	D3	PIN 23	GND
PIN 6	D4	PIN 24	GND
PIN 7	D5	PIN 25	GND
PIN 8	D6	PIN 26	GND
PIN 9	D7	PIN 27	GND
PIN 11	BUSY	PIN 28	GND
PIN 14	GND	PIN 33	GND
PIN 16	GND	All other pins NC	

# INTERCONEXIONES DE USUARIO



VIDEO  
OUTPUT  
CONNECTOR  
(6PIN DIN)



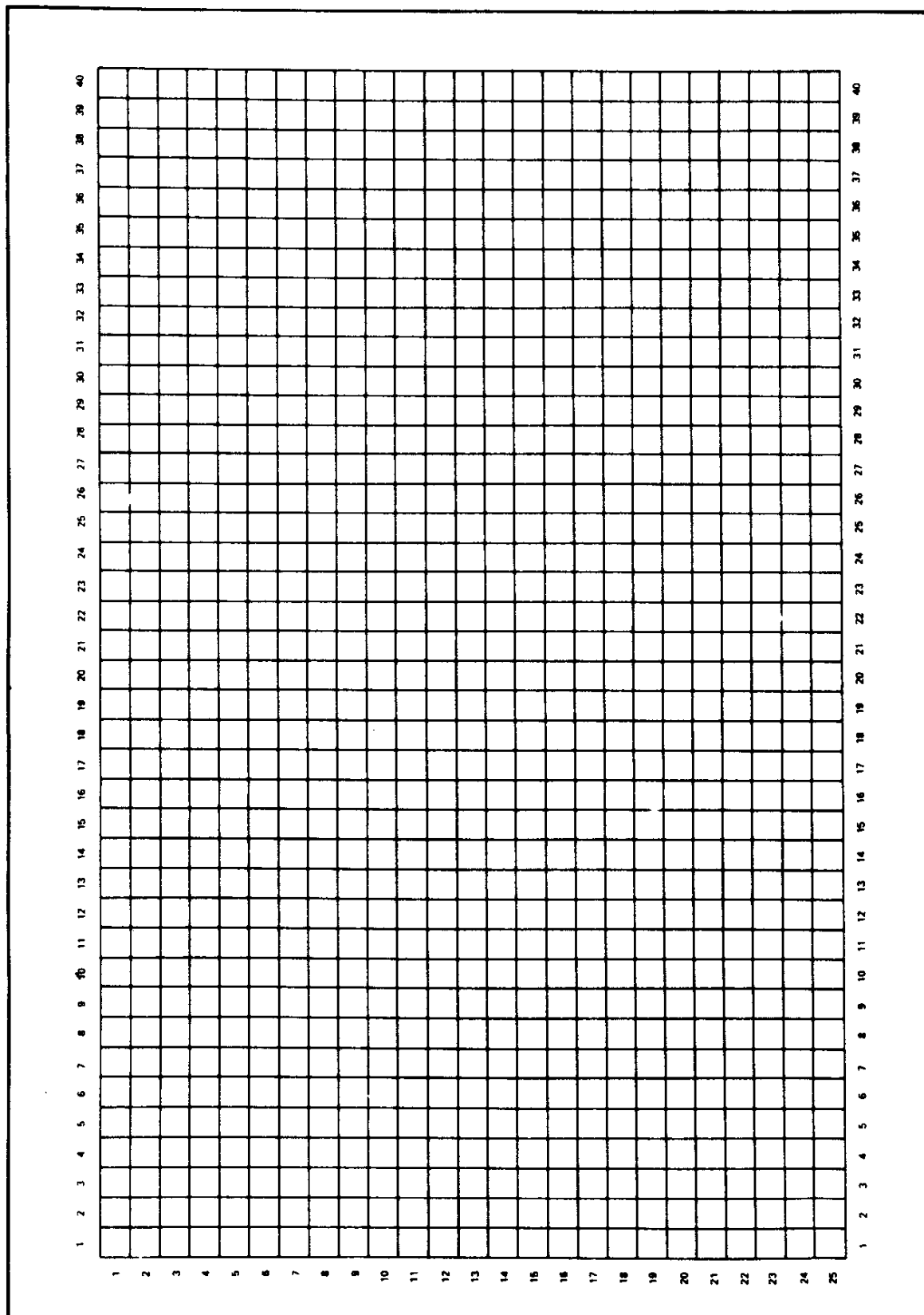
## Apéndice VI:

**Retícula de TEXTO y VENTANA**  
**MODO 0      20 columnas**

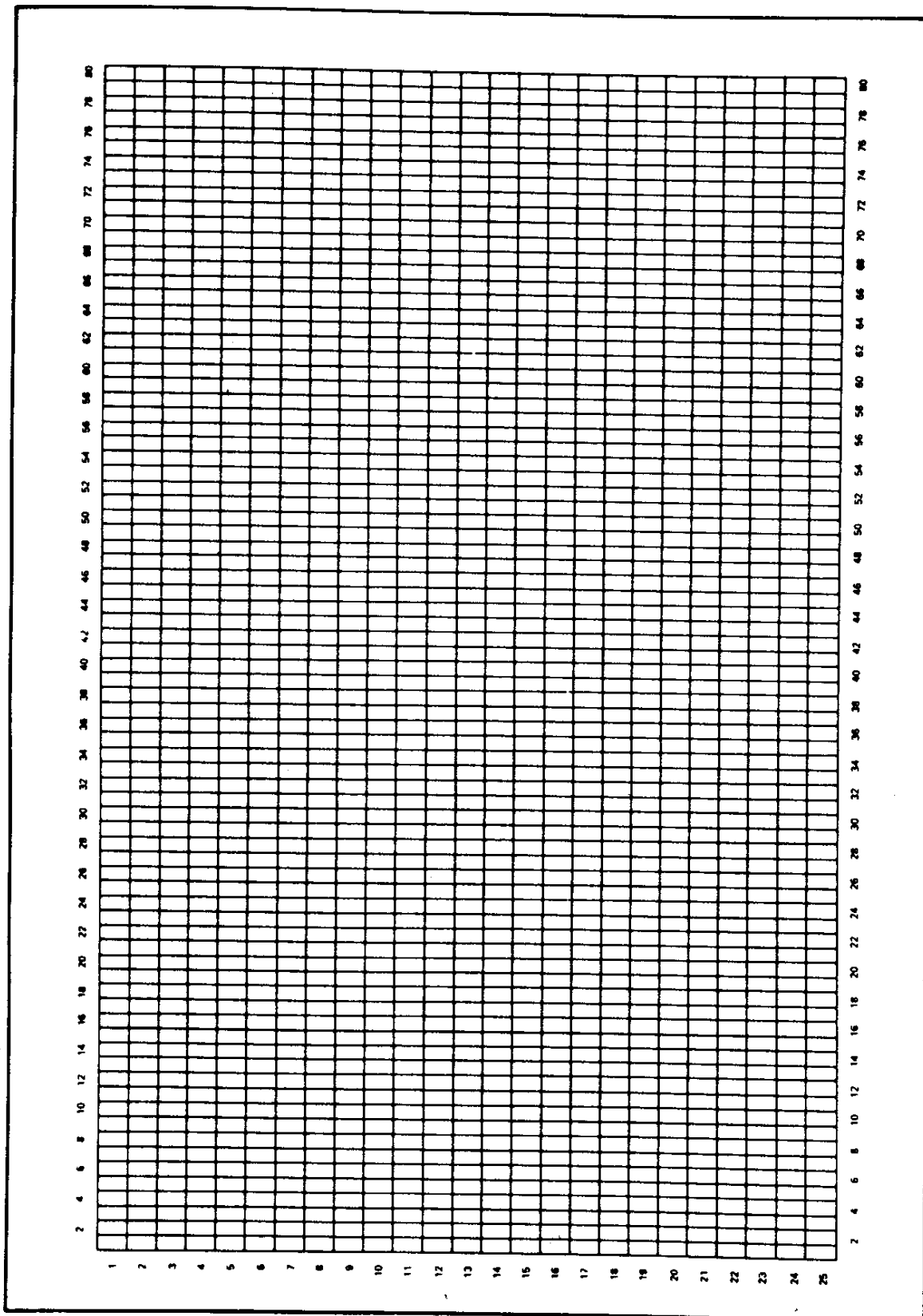
[illegible]

# Retícula de TEXTO y VENTANA

## MODO 1 40 columnas



# Retícula de TEXTO y VENTANA MODO 2 80 columnas



## Apéndice VII:

### Notas y Períodos de Tono

La tabla que sigue da los ajustes recomendados de períodos de tono para notas en la escala usual de pentagrama para la gama completa de 8 octavas.

La frecuencia producida no es exactamente la frecuencia requerida porque el ajuste de período tiene que ser un entero. El error relativo es el cociente de la diferencia entre las frecuencias requerida y real dividido por la frecuencia requerida, i.e: (REQUERIDA-ACTUAL)/(REQUERIDA).

NOTE	FREQUENCY	PERIOD	RELATIVE ERROR
C	32.703	3822	-0.007%
C#	34.648	3608	+0.007%
D	36.708	3405	-0.007%
D#	38.891	3214	-0.004%
E	41.203	3034	+0.009%
F	43.654	2863	-0.016%
F#	46.249	2703	+0.009%
G	48.999	2551	-0.002%
G#	51.913	2408	+0.005%
A	55.000	2273	+0.012%
A#	58.270	2145	-0.008%
B	61.735	2025	+0.011%

Octave -3

NOTE	FREQUENCY	PERIOD	RELATIVE ERROR
C	65.406	1911	-0.007%
C#	69.296	1804	+0.007%
D	73.416	1703	+0.022%
D#	77.782	1607	-0.004%
E	82.407	1517	+0.009%
F	87.307	1432	+0.019%
F#	92.499	1351	-0.028%
G	97.999	1276	+0.037%
G#	103.826	1204	+0.005%
A	110.000	1136	-0.032%
A#	116.541	1073	+0.039%
B	123.471	1012	-0.038%

Octave -2



NOTE	FREQUENCY	PERIOD	RELATIVE ERROR	
C	130.813	956	+0.046%	Octave -1
C#	138.591	902	+0.007%	
D	146.832	851	-0.037%	
D#	155.564	804	+0.058%	
E	164.814	758	-0.057%	
F	174.614	716	+0.019%	
F#	184.997	676	+0.046%	
G	195.998	638	+0.037%	
G#	207.652	602	+0.005%	
A	220.000	568	-0.032%	
A#	233.082	536	-0.055%	
B	246.942	506	-0.038%	
NOTE	FREQUENCY	PERIOD	RELATIVE ERROR	
C	261.626	478	+0.046%	Middle C
C#	277.183	451	+0.007%	
D	293.665	426	+0.081%	
D#	311.127	402	+0.058%	
E	329.628	379	-0.057%	Octave 0
F	349.228	358	+0.019%	
F#	369.994	338	+0.046%	
G	391.995	319	+0.037%	
G#	415.305	301	+0.005%	
A	440.000	284	-0.032%	International A
A#	466.164	268	-0.055%	
B	493.883	253	-0.038%	
NOTE	FREQUENCY	PERIOD	RELATIVE ERROR	
C	523.251	239	+0.046%	Octave 1
C#	554.365	225	-0.215%	
D	587.330	213	+0.081%	
D#	622.254	201	+0.058%	
E	659.255	190	+0.206%	
F	698.457	179	+0.019%	
F#	739.989	169	+0.046%	
G	783.991	159	-0.277%	
G#	830.609	150	-0.328%	
A	880.000	142	-0.032%	
A#	932.328	134	-0.055%	
B	987.767	127	+0.356%	

NOTE	FREQUENCY	PERIOD	RELATIVE ERROR
------	-----------	--------	----------------

C	1046.502	119	-0.374%
C#	1108.731	113	+0.229%
D	1174.659	106	-0.390%
D#	1244.508	100	-0.441%
E	1318.510	95	+0.206%
F	1396.913	89	-0.543%
F#	1479.978	84	-0.548%
G	1567.982	80	+0.350%
G#	1661.219	75	-0.328%
A	1760.000	71	-0.032%
A#	1864.655	67	-0.055%
B	1975.533	63	-0.435%

Octave 2

NOTE	FREQUENCY	PERIOD	RELATIVE ERROR
------	-----------	--------	----------------

C	2093.004	60	+0.462%
C#	2217.461	56	-0.662%
D	2349.318	53	-0.390%
D#	2489.016	50	-0.441%
E	2637.021	47	-0.855%
F	2793.826	45	+0.574%
F#	2959.955	42	-0.548%
G	3135.963	40	+0.350%
G#	3322.438	38	+0.992%
A	3520.000	36	+1.357%
A#	3729.310	34	+1.417%
B	3951.066	32	+1.134%

Octave 3

NOTE	FREQUENCY	PERIOD	RELATIVE ERROR
------	-----------	--------	----------------

C	4186.009	30	+0.462%
C#	4434.922	28	-0.662%
D	4698.636	27	+1.469%
D#	4978.032	25	-0.441%
E	5274.041	24	+1.246%
F	5587.652	22	-1.685%
F#	5919.911	21	-0.548%
G	6271.927	20	+0.350%
G#	6644.875	19	+0.992%
A	7040.000	18	+1.357%
A#	7458.621	17	+1.417%
B	7902.133	16	+1.134%

Octave 4

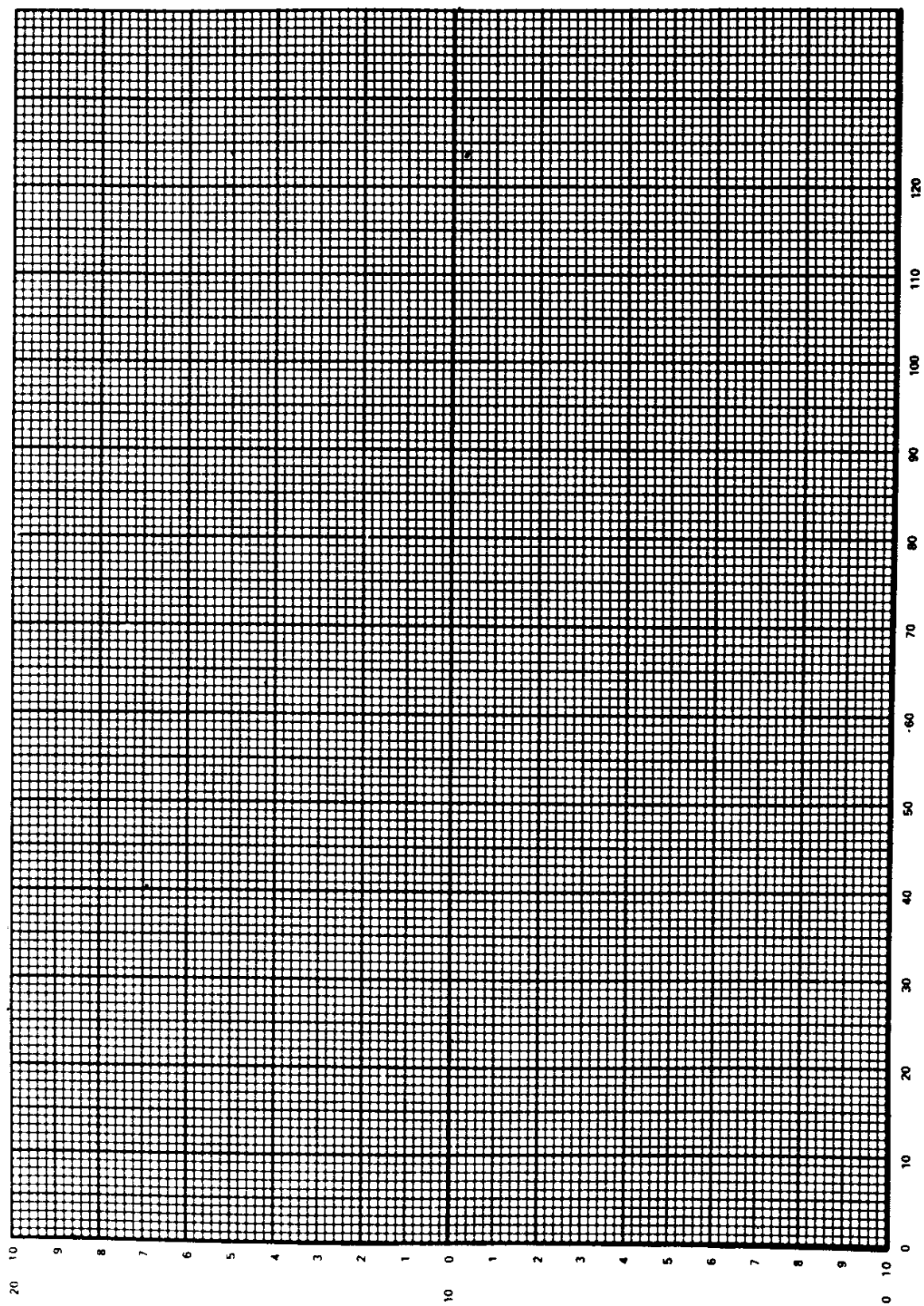
Estos valores están todos calculados a partir de la A internacional:

$$\text{FREQUENCY} = 440 * (2^{\uparrow (\text{OCTAVE} + (10 - N) / 12)})$$

$$\text{PERIOD} = \text{ROUND}(125000 / \text{FREQUENCY})$$

donde N es 1 para C, 2 para C#, 3 para D, etc.

Retícula para música/envolventes



## Apéndice VIII:

### Códigos de error y palabras reservadas

#### Números y mensajes de error

Cuando BASIC encuentra una instrucción, palabra o variable en una línea del programa que no puede interpretar ni ejecutar, detendrá el programa y mostrará un **mensaje de error**. La forma del mensaje indicará generalmente lo que está equivocado, y algunas veces si el error es un error tipográfico al escribir el programa, el BASIC pasará automáticamente al modo edición, mostrando la línea donde ha detectado el error.

El mensaje más popular entre los usuarios descuidados es el 'Error sintáctico' (número 2) y el BASIC lo muestra junto con la línea que deberá EDITAR si detectó el error al ejecutar el programa. En el modo directo, cuando le impones comandos, simplemente declara que ha detectado un error, y presupone que la última línea tecleada es visible y por tanto puedes examinarla en busca del problema.

Si has incluido al comienzo del programa la instrucción **ON ERROR GOTO** para que SEGUN error VAYA..., a un determinado número de línea en que comienza el tratamiento de errores, dará un salto a ese párrafo inmediatamente que detecte el error, y antes de sacar sus mensajes prescritos. En el ejemplo, el ordenador saltará a la línea 1000 al detectar un error:

```
10 ON ERROR GOTO 1000
```

program

```
1000 PRINT CHR$(7) :MODE 2:INK 1,0: INK 0,9: CLS :LIST
```

Y en este caso, ante el error, el CPC464 emitirá un pitido, limpiará lo que haya en pantalla, y cambiará a la combinación de colores mencionada y en el modo de imagen de 80 columnas, y luego listará el programa para que puedas examinarlo. Si el error es un error sintáctico, aparecerá al pie del listado la línea en que lo detectó, esperando que la corrijas (edites) aunque el mensaje habitual

#### Syntax Error

será suprimido. (Recuerda colocar una instrucción **END** en la línea inmediatamente anterior a la 1000, si deseas conservar lo que tengas puesto en pantalla cuando el programa haya sido ejecutado normalmente).

El BASIC no producirá mensajes de error cuando las líneas de programa ingresadas por teclado son válidas; por lo tanto, debe suponerse que siempre que ocurre un error, puede ser RASTREADO siguiendo el hilo del programa, y habitualmente guiándose por los mensajes producidos durante el proceso de DEPURACION del mismo. Como en muchas otras situaciones, aprenderás más y mejor analizando tus equivocaciones, y abusando del hecho de que el CPC464 es el más tolerante de los 'tutores'. Te cansarás de ensayar mucho antes de que el CPC464 muestre signos de impaciencia.

Todos los mensajes de error generador por BASIC, están enumerados en este Apéndice, ordenador por el número de error pertinente. También te presentamos una breve descripción de las posibles causas.

### 1 Unexpected NEXT

Ha encontrado una instrucción que le manda dar OTRA posible ronda de un bucle, 'inesperadamente' porque no estaba repitiendo nada; o la variable de control mencionada en la instrucción NEXT no concuerda con la mencionada en la instrucción FOR.

### 2 Syntax Error

Normalmente alguna palabra clave está mal deletreada, o no hay igual número de paréntesis de cierre que paréntesis de apertura en una expresión.

### 3 Unexpected RETURN

Se ha encontrado con una instrucción en que le pides VUELVA 'inesperada', porque no se había desviado en ningún momento anterior.

### 4 DATA exhausted

Le mandas que 'apunte' el siguiente DATO y ya está 'exhausta'... la serie de constantes que mencionas en el programa.

### 5 Improper argument

Tiene un significado general que indica que el valor del ARGUMENTO de una función es 'impropio', o que el **parámetro** mencionado en una instrucción 'no es válido'.

### 6 Overflow

El resultado de una operación aritmética hace que se REBASE la gama de valores permitida a la variable. Puede corresponder a una variable real, en cuyo caso, en esa operación se ha producido un valor mayor en valor absoluto que  $1.7E-38$  (aprox.). También puede ser el resultado de un intento fallido de cambiar un número real a un número entero, cuando la parte entera del número real se sale fuera de la gama -32768 a +32767.

### 7 Memory full

La memoria está 'llena'. El programa es demasiado largo, o simplemente hay demasiadas variables, o innumerables 'desvíos' y 'bucles'.

El comando **MEMORY** dará este error si se intenta estipular un valor demasiado bajo como 'cima' de la memoria reservada a BASIC; o bien un valor imposiblemente elevado. Recuerda además, que al abrir un fichero se ocupa un determinado espacio en memoria para ser usado como **buzón** donde alojar los registros procedentes o con destino al fichero, y eso restringe la memoria disponible.

### 8 Line does not exist

Mencionas un número de línea que no existe en el programa.

### 9 Subscript out of range

Uno de los SUFIJOS ('suscritos' o 'subíndices') usado para seleccionar un elemento de una **tabla**, está 'fuera de gama', por demasiado grande o demasiado pequeño.

### 10 Array already dimensioned

La **tabla** (RINGLA) que mencionas 'ya está dimensionada'. Debes previamente hacer que LIBRE el espacio pertinente. (ERASE).

### 11 Division by zero

Con esos valores resulta una 'división por cero'.

### 12 Invalid direct command

Le estás dando un comando, usando una palabra **clave** que sólo admite como **instrucción**, ie. precedida de un número de línea.

### 13 Type mismatch

'Clase discordancia'. Das como valor de una variable **literal** una constante **numérica**, o viceversa.

### 14 String space full

Está 'lleno el espacio' que internamente reserva para las 'sartas' de caracteres (LITEROS ENTRECOMILLADOS). Ya no dispone de más espacio, incluso aunque haya hecho 'recogida de basura'. (Supresión de los viejos valores de las variables literales).

### 15 String too long

'SARTA demasiado larga', pues sobrepasa de 255 caracteres. Puede ser resultado del empalme de datos litéricos.

#### 16 String expression too complex

Cuando las 'expresiones literales son demasiado complejas' porque pueden dar lugar a demasiados valores intermedios, que exceden de un límite razonable.

#### 17 Cannot CONTinue

'No puede continuar'. Observa que el comando CONT es para hacer que SIGA un programa que se ha visto interrumpido por encontrar una instrucción STOP, que le hace que PARE; porque has pulsado ESC ESC para ESCAPAR; pero no para cuando ha detectado un error y EDITAS esa instrucción y luego quieres que continúe. En esa situación es imposible que SIGA con el programa en el punto en que se interrumpió.

#### 18 Unknown user function

Esa función definible por el usuario es 'desconocida'. Necesitas avisarle previamente para que la USE.

#### 19 RESUME missing

NO es que 'falte' el resumen. Lo que SI falta es la instrucción para que REANUDE en algún punto el programa después de haberse visto "atrapado" en un error, que el propio programa se encarga de tratar.

#### 20 Unexpected RESUME

NO es un 'inesperado' resumen. SI es que no puedes decirle 'inesperadamente' que REANUDE cuando no se ha visto 'atrapado' en ningún error. Coloca una instrucción para que PARE o TERMINE antes de que llegue al párrafo pertinente.

#### 21 Direct command found

Sólo puede aparecer después de CARGAR un programa desde el cassette, y es porque ha 'encontrado' un COMANDO (directo) y no una instrucción (con número de línea delante).

#### 22 Operand missing

En esa expresión 'falta' algún operando.

#### 23 Line too long

'Línea demasiado larga'.

#### 24 EOF met

Intentas tomar información del cassette, y te has 'topado' con el final del fichero.

### 25 File type error

'Error en la CLASE (tipo) de fichero'. El fichero que hay en la cinta no es de la clase adecuada. Observa que el comando **OPENIN** sólo está preparado para ABRIR como fichero de ENTRADA, ficheros con textos en caracteres ASCII; y que los comandos **LOAD**, **RUN**, sólo están preparados para ficheros con programas que previamente has hecho que **GUARDE** mediante el comando **SAVE**.

### 26 NEXT missing

No puede encontrar la instrucción que le mande hacer OTRA ronda, y estando dentro de un bucle.

### 27 File already open

Ese 'fichero ya está abierto'. Quizás convenga decirle que lo **CIERRE**, mediante la instrucción **CLOSED**.

### 28 Unknown command

Comando 'desconocido'.

### 29 WEND missing

No puede encontrar la instrucción **WEND** que siempre tiene que haber en correspondencia con cada instrucción **WHILE**.

### 30 Unexpected WEND

Encontró una instrucción **WEND** sin estar dentro de un bucle **WHILE**.

## Palabras clave en el BASIC AMSTRAD

Las siguientes, son las palabras clave en BASIC, y están **reservadas** para significar exactamente lo que hemos descrito en esta guía.

**ABS, AFTER, AND, ASC, ATN, AUTO  
BIN\$, BORDER**

**CALL, CAT, CHAIN, CHR\$, CINT, CLEAR, CLG, CLOSEIN,  
CLOSEOUT, CLS, CONT, COS, CREAL**

**DATA, DEF, DEFINIT, DEFREAL, DEFSTR, DEG, DELETE,  
DI, DIM, DRAW, DRAWR**

**EDIT, EI, ELSE, END, ENT, ENV, EOF, ERASE, ERL, ERR,  
ERROR, EVERY, EXP**

**FIX, FN, FOR, FRE**



GOSUB, GOTO

HEX\$, HIMEN

IF, INK, INKEY, INKEY\$, INP, INPUT, INSTR, INT

JOY

KEY

LEFT\$, LEN, LET, LINE, LIST, LOAD, LOCATE, LOG, LOG10,  
LOWER\$

MAX, MEMORY, MERGE, MID\$, MIN, MOD, MODE, MOVE, MOVER  
NEXT, NEW, NOT

ON, ON BREAK, ON ERROR GOTO, ON SQ, OPENIN, OPENOUT, OR,  
ORIGIN, OUT

PAPER, PEEK, PEN, PI, PLOT, PLOTR, POJE, POS, PRINT

RAD, RANDOMIZE, READ, RELEASE, REM, REMAIN, RENUM,  
RESTORE, RESUME, RETURN, RIGH\$, REOD, ROUND, RUN

SAVE, SGN, SIN, SOUND, SPACE\$, SPC, SPEED, SQ, SQR,  
STEP, STOP, STR\$, STRING\$, SWAP, SYMBOL

TAB, TAG, TAGOFF, TAN, TEST, TESTR, THEN, TIME, TO,  
TROFF, TRON

UNT, UPPER\$, USING  
VAL, VPOS

WAIT, WEND, WHILE, WIDTH, WINDOW, WRITE

XOR, XPOS

YPOS

ZONE

## GLOSARIO DE TERMINOS

Algunos términos usados comúnmente en el mundo informático, explicados para los usuarios del CPC464...

### Acoplador

### Acoustic Couper

También se le llama Modem Acústico, y es un dispositivo electrónico en el que se puede apoyar el micrófono y el audífono del aparato de abonado, y que adapta las señales del ordenador a las señales necesarias en la red telefónica. De esta forma, el ordenador se pone en comunicación con sistemas de información pública (PRESTEL, VIDEOTEXT...) o con otros usuarios para intercambiar información, programas, etc.

### Acumulador

### Accumulator

Un circuito electrónico incorporado en el microprocesador, o un lugar específico de la memoria, por el que pasa la inmensa mayoría del tráfico de datos y operandos, y donde se acumula el resultado de las operaciones efectuadas. Se usa constantemente al programar en código máquina, pero en BASIC es como si no existiese.

### Alfanumérico/Alfamérico

### Alphanumeric/Alphameric

No usado en BASIC. Sí en otros lenguajes donde es atributo de un dato cuando puede incluir caracteres literales (alfabéticos) y numerales. El equivalente en BASIC son los "líteros".

### Algebra de Boole

### Boolean algebra

Estructura de matemáticas con proposiciones y relaciones lógicas que sólo pueden tener dos respuestas: cierto o falso. En estrecha relación con el sistema binario, simbolizando con 1 y 0 respectivamente.

### Algoritmo

### Algorithm

De la misma raíz árabe que 'guarismos', designa la secuencia de operaciones aritmético-lógicas que detalla el método a seguir para ejecutar una determinada tarea.

### AMSOFT

### AMSOFT

División especializada de AMSTRAD, suministrando programas, periféricos, y publicaciones primordialmente para el CPC464 y sus muchas aplicaciones.

### Analfabetismo moderno

### Illiteracy

En breve plazo se considerará oficialmente analfabeto a aquél que no tenga conocimientos informáticos.

**Animación****Animation**

Tomado de los 'dibujos animados'. La animación por ordenador está basada en el movimiento de figuras y gráficos para simular movimientos 'vivos'.

**Apretón (de manos)****Handshaking**

Protocolo electrónico entre dos ordenadores, o entre ordenador y periférico, antes del intercambio de información entre ellos.

**Argumento****Argument**

Variable independiente de una función matemática (v.g. en la expresión  $z=x+\lg(y)$  la  $y$  es el argumento de la función logarítmica que interviene).

**Arquitectura****Architecture**

Estrategia en la construcción de un sistema informático, organizando las relaciones y conexiones entre los diferentes equipos que lo componen. También y en especial la estructura interna del ordenador con sus circuitos procesadores, memorizadores, canalizadores, etc.

**Arranque/'Botadura'****Booting/Bootstrapping**

Puesta en marcha de la máquina. En especial cuando hay un pequeño programa grabado permanente e indeleblemente en la máquina, encargado de IMPLANTAR en memoria el sistema operativo del ordenador residente hasta entonces en un medio auxiliar de almacenamiento.

**ASCII****American Standard Code for Information Interchange**

Convenio Internacional que establece la equivalencia entre códigos binarios y caracteres visivos y de control empleados en el intercambio de información entre equipos informáticos.

**Aviso****Prompt**

Símbolo o mensaje típico -la **divisa**- que caracteriza a un ordenador, e indica al usuario que está preparado para recibir sus comandos e instrucciones. En el CPC464 es la palabra **Ready (Dispuesto)**.

**Aleatorio****Random****Azar**

Fenómeno que no es predecible ni repetible. En los ordenadores suelen existir generadores de números aleatorios que realmente producen una secuencia de números reales tan grande que es prácticamente imposible de apreciar la falsedad (los puristas le llaman **pseudo-aleatorio**).

**Barrido en rastrillo****Raster scan**

Exposición de imágenes visuales mediante varios haces proyectores que la recorren según líneas paralelas (en lugar del barrido habitual con un solo haz en zig-zag).

**Base**

**Base**

Cardinal del conjunto de símbolos sobre el que se constituye un sistema de numeración. El sistema binario tiene base 2 (y emplea 2 símbolos); el decimal tiene base 10 (y 10 símbolos); y el hexadecimal base 16 (y 16 símbolos).

**Base/Banco de datos**

**Database**

Conjunto organizado de datos, de diversas clases y heterogéneos, y accesibles por diversos conceptos, utilizados por varios ordenadores locales o remotos.

**BASIC**

**BASIC**

Beginners All-purpose Symbolic Instruction Code. Lenguaje usado por casi todos los ordenadores personales, pensado en su origen para PRINCIPIANTES y ampliamente desarrollado por su fácil aprendizaje y su capacidad de preparar programas y comprobar su funcionamiento a medida que se van perfeccionando. En la actualidad está completamente evolucionado y además de principiantes es usado por programadores experimentados en todas las partes del mundo.

**Baudio**

**Baud**

Unidad en un principio telegráfica para transferencia de señales binarias, expresada en intervalos por segundo. (En las transmisiones habituales coincide con **bits** por segundo).

**BCD**

**Binary Coded Decimal**

Sistema de representación de los diez símbolos (dígitos) del sistema de numeración decimal en grupos de cuatro símbolos binarios.

**Binario**

**Binary**

(Vea **Base**). Sistema de numeración de base 2.

**Bico**

**Byte**

**Bito**

**Bit**

**Bitio**

**Bit**

Cualquiera de los dos símbolos (0 y 1) usado en el sistema de numeración binario se denomina **bito**. La unidad de información-incertidumbre de un suceso (relacionada con el logaritmo de la probabilidad del mismo) es el **bitio**, cuando el logaritmo se toma en base 2.

El grupo de bits tratados a la vez por los circuitos electrónicos de los equipos informáticos se denomina **bico**. En los ordenadores personales con microprocesadores de OCHO BIToS es exactamente equivalente a octeto.

**Bit más significativo****MSB**

En un número binario el bit anterior del número (BAN) y que por tanto ocupa la posición extremo-izquierda. Su **peso** es de  $2^7=128$  (con bicos de 8 bits) y  $2^{15}=32768$  (en 'bicos' direccionales de 16 bits).

**Bit menos significativo****LSB**

En un número binario en bit ulterior del número (BUN) y que por tanto ocupa la posición extremo-derecha. Su **peso** es de  $2^0=1$  (siempre).

**Bit interpretado (calibrado en binario)****Bit significant**

Dato cuyo significado sólo se obtiene hallando el equivalente binario e interpretando los valores según las posiciones en que aparezcan los unos y ceros.

**Bus****Bus**

Haz de cables que recorre la totalidad de los circuitos electrónicos de un equipo de proceso de datos, portando tensiones de alimentación y señales de información de las que cada circuito usa las que le corresponde. En el CPC464 dicho bus o haz es accesible en el más grande de los dos conectores impresos en la tarjeta situada en la parte posterior de la carcasa.

**CAD****CAD**

Habitualmente interacción entre la potencia de cómputo y la capacidad gráfica para constituir un tablero electrónico de dibujo.

**CAE****CAE**

Enseñanza asistida por ordenador, modismo de los muchos que se añaden cada día, y que es difícil de separar del adiestramiento ayudado por ordenador (CAI) y del aprendizaje apoyado por ordenador (CAL).

**Captura de datos****Data capture**

Término que describe la obtención y colecta de datos procedentes de cualquier fuente externa que son introducidos de alguna manera hacia el ordenador central.

**Carácter****Character**

Cualquier símbolo con representación gráfica o que ejerza una determinada acción siempre y cuando tenga asociado intrínsecamente una representación numérica interna.

En el CPC464 se utilizan los 256 posibles con bicos de 8 bits.

**Carácter gráfico****Graphic character**

Carácter de **tipo** visivo que no corresponde a las letras o signos habituales y que puede estar previamente grabado o ser diseñado por el usuario.

**Cartucho  
Cassette****Cartridge  
Cassette**

Cajita de innumerables formas que se inserta en un alojamiento adecuado. En informática se suele reservar el término 'cartuchos' para los que incluyen pastillas de semiconductores grabadas en forma permanente, indeleble (o deletable) con programas usualmente de juegos, vocabulario, etc. Y se reserva el término 'cassette' para los que incluyen como soporte de información cinta magnética trasladable entre los carretes en los dos sentidos.

**Cauce/Chorro****Stream**

También canal, flujo, ruta... El conducto por donde se **encauza** la información transferida entre los diversos órganos que integran un ordenador. En ambos extremos del cauce suele haber un circuito de almacenamiento temporal - un buzón cuando los ritmos de emisión de un equipo no coincide con el ritmo de admisión del otro.

**Celdilla/Célula****Character cell**

En relación con la memoria se usa más a menudo el término celdilla para designar el sitio donde está alojado un carácter.

En relación con la pantalla, se emplea más habitualmente el término 'célula' para designar la posición ocupada por un carácter. En el CPC464 está realmente formada por 64 motas dispuestas regularmente en 8 trazos horizontales con 8 motas por trazo. La iluminación selectiva de las motas en uno u otro color determina el **tipo** del símbolo que aparece.

**Chip****Chip**

Nombre popular (aunque muchos los llamamos '**pastillas**' y '**cucarachas**' y los puristas 'circuitos integrados en una sola oblea de silicio') del microprocesador, las memorias, y demás...

**Circuitalia****Hardware**

Todo los aparatos, equipos, cables, que puedes tocar y son 'duros'. Constituyen el soporte físico de todo lo demás.

**Circuito integrado****Integrated Circuit**

Componente electrónico que incorpora centenares de diodos, transistores, resistencias, etc. constituyendo todo un circuito especializado en una función determinada.

**Código****Code**

Convenio generalmente numérico para representar información estableciendo relaciones biunívocas con los símbolos visibles en pantalla, impresora, teclado.

**Código barrado****Bar code**

Mire cualquier paquete de jabón y tendrá un ejemplo.

**Código máquina****Machine Code**

Lenguaje de programación que puede ser '**entendido**' directamente por el microprocesador y los circuitos internos del ordenador (es una forma de hablar, sólo entienden de electrónica...).

**Comando****Command**

Orden o mandato dado directamente al ordenador que por supuesto, inmediatamente obedece, a no ser que cometas un error de tecleado.

**Compilador****Compiler**

Programa complejo que convierte y traduce programas escritos en lenguajes de 'alto nivel' para personas, en programas escritos en lenguajes de 'bajo nivel' para máquinas, consiguiendo así una mayor rapidez de ejecución.

**Conjunto****Array**

Mejor otros nombres como **ringlas**, tablas, arreglos, matrices, etc. para designar colecciones de datos homogéneos dispuestos regularmente a los que se les asigna un nombre genérico y cuyos elementos pueden ser individualmente seleccionados usando un **sufijo** que guarda relación con la posición que ocupa dicho elemento.

**Conjunto de instrucciones****Instruction set**

Mejor **repertorio**; para designar las operaciones y acciones individuales admitidas en un determinado lenguaje de programación o en un determinado ordenador, constituyendo las palabras **clave** o reservadas.

Los programas se escriben como series ordenadas de estas instrucciones, y el programa **interpretador** se encarga de examinarlas, y desglosarlas en las correspondientes series de operaciones que deben efectuarse internamente por los órganos circuitales del equipo, en particular el microprocesador.

**Conversión A/D****Analogic/Digital Conversion**

Las magnitudes cuyo cambio entre un punto inicial y uno final ocurre gradualmente y no de manera infinitesimalmente continua se denominan digitales o discretas. Los ordenadores son mecanismos discretos y digitales (aunque los hay analógicos) y por tanto, sus cambios de estado se producen a impulsos. La mayoría del mundo natural está basado en principios analógicos (a nivel perceptible claro está) por lo que en los ordenadores es necesario efectuar una conversión analógico-digital a la entrada para que el dato pase por sus portales de entrada.

**Copia impresa****Hard Copy**

Listado en papel de un programa, texto, o imagen gráfica que aparece en pantalla. Su equivalente en un medio magnético o electrónico de almacenamiento se conoce como copia grabada (soft copy).

**CP/M****CP/M**

Marca comercial de Digital Research para el sistema operativo más ampliamente difundido en los ordenadores personales con microprocesadores de 8 bits.

**CPU****Central Processing Unit**

Unidad Procesadora Central. Es el órgano circuital encargado de recabar las instrucciones del programa de la memoria, examinarlas e interpretar su significado, y ejecutar o controlar su ejecución por otros órganos circuitales.

**Cursor****Cursor**

Marca que 'corre' por toda la pantalla indicando la posición en que aparecerá el siguiente carácter.

**Cursor de gráficos****Graphics cursor**

Similar al cursor de textos, pero habitualmente 'corre sin que se le vea', pero capaz de situar los gráficos usando coordenadas más precisas. No confundir con caracteres gráficos.

**Depuración****Debugging**

La actividad de quitar las 'pifias' de un programa, mediante combinación de métodos científicos, prácticos y demás.

**Desrolla/Despliega****Scrolling**

Desplazamiento de la página de texto visible en pantalla hacia cualquiera de los costados o hacia la parte superior o inferior. El cursor no se desplaza de su posición relativa, sino con toda la ventana de texto.

**Desrollador****Mouse**

Artilugio en forma de rodillo o bola que logra el desrolle en pantalla por el simple rodamiento sobre su alojamiento en el teclado.

**Documentalea****Paperware**

El conjunto de impresos, manuales, listados... que utilizan o generan los procesos informáticos.



**Editar****Editor de plana****Editor de línea****Edit****Screen editor****Line editor**

En el mundo informático **editar** es primordialmente enmendar, corregir, arreglar... las instrucciones de un programa. Dependiendo de las facilidades del sistema se puede hacer desplazando el cursor por todas las partes de la pantalla, o es preciso designar el número de línea que se desea cambiar. En el CPC464 tienes la posibilidad de tratar con ambos 'editores'.

**Error sintáctico****Syntax error**

Popularísimo mensaje de error recibido cuando se infringe una o muchas de las reglas gramaticales que están convenidas.

**Ex-línea****Off line**

Estado en que se encuentra un periférico del ordenador cuando no está ligado directamente al ordenador, o no es accesible en ese momento por alguna causa.

**Generador de sonido****Sound generator**

El conjunto de circuitos internos del sistema capaz de crear ruidos y sonidos en base a los parámetros y señales enviadas por el microprocesador.

**Impresora****Printer**

Uno de los primeros periféricos que adquieres.

**Ingeniería de computación****Software engineering**

Una expresión grandiosa en inglés para indicar programación y que no suena también en otros idiomas.

**In-línea****On-line**

Estado en que se encuentra un periférico cuando sí está ligado directamente al ordenador y además es accesible en ese momento.

**Interface/Interfase****Interface**

Frontera donde se 'encaran' las señales de dos equipos cualesquiera, o en donde tiene lugar el cambio de una etapa a otra en un proceso. Es el **nexo** de unión. Se suele hablar de interface hombre-máquina para designar la manera en que se relacionan y los medios utilizados.

**Interface serie****Serial interface**

Modo de acoplamiento entre dos equipos que se transfieren información, en que se descompone el dato a nivel de bit y se envían sucesivamente a través de un conductor dichos bits. Con los ordenadores personales casi siempre hace referencia a la norma RS232, aunque existen otras.

**Interface paralelo****Parallel interface**

Modo de acoplamiento entre dos equipos que se transfieren información, en que se envían simultáneamente por varios conductores los bits que componen un dato. La transferencia es mucho más rápida que con una interface serie.

El CPC464 lleva incorporados los circuitos necesarios para poder conectar directamente impresoras con esta clase de interface.

**Lápiz luminoso****Light pen**

Mecanismo con dicha forma que permite ingresar datos expuestos en pantalla señalándolos simplemente con el extremo puntiagudo. También hay versiones que permiten leer directamente códigos de barras, caracteres ópticos y hasta letra de imprenta normal.

Es algo así como el famoso dedo de E.T.

**Lazo/Bucle****Loop**

Situación empleada en un programa para hacer que el ordenador se dedique a dar rondas y más rondas a la misma serie de instrucciones (muchas veces sin que pretendiéramos eso...).

**Legible por máquina****Machine readable**

Un soporte de información que puede ser inmediatamente admitido en un ordenador sin ningún tecleo adicional.

**Lenguaje de bajo nivel****Low-level language**

Se aplica a aquellos lenguajes de programación, tal y como el de ensamblaje con un microprocesador dado, en que cada una de las instrucciones escritas por el programador es casi un reflejo exacto de las que acepta el microprocesador.

**Lisp****LISt Processor language**

Procesador de **listas**; es otro de los lenguajes de 'alto nivel'. Especializado en manejar textos y sertas de caracteres.

**Lógica (electrónica)****Logic**

Nombre genérico de los componentes electrónicos que llevan a cabo operaciones de lógica aritmética y lógica comparativa, y a partir de los cuales se construyen los órganos circuitales del ordenador.

**Logo****Logo**

Otro lenguaje de alto nivel simple de aprender y orientado hacia el desarrollo de gráficos, empleando preferentemente técnicas recursivas. Se está popularizando en las escuelas infantiles como ayuda a la enseñanza de informática.

**LSI****Large Scale Integration**

Pastillas en que el número de elementos de lógica electrónica incorporados es **grande**, por supuesto mayor que en los **MediumSI**, que a su vez es mayor que los **SmallSI**, en que es **pequeño** (que son cientos).

**Mapa de memoria****Memory map**

Diagrama mostrando el reparto de memoria en diversas áreas reservadas para funciones determinadas, tablas de variables, programa ejecutándose, interpretador BASIC, etc.

**Matriz****Matrix**

Véase **Conjunto**. Usado además de en matemáticas, para designar el conjunto de agujas que forma el cabezal de escritura en las impresoras matriciales. (Aunque la mayoría de ellas sólo poseen un par de hileras como mucho).

**MEL****RAM**

Designa con mayor propiedad a la **Memoria de Escritura y Lectura** que posee el ordenador, ya que el término original inglés Random Access Memory puede dar lugar a confusiones. Véase **MUL**.

**MEMOrandum****REM**

Lo que sirve de recordatorio. En el BASIC la palabra clave es REM.

**Memoria****Memory**

El conjunto de circuitos electrónicos que organizados a manera de 'panel de celdillas' permite al procesador señalar la **dirección** de cada celdilla y alojar o extraer información de ella. Véase **MEL** y **MUL**.

**Menú****Menu**

La 'carta' con que un programa te ofrece los 'platos' que puede servir.

**Microprocesador****Microprocessor**

El causante de la popularización del ordenador y el circuito integrado encargado de controlar todos los demás que componen el ordenador.

**Modem****Modem**

Acrónimo de **modulador** y **demodulador**, que son los procesos a efectuar para pasar de un medio de transmisión a otro, v.g. del ordenador a la línea telefónica, y viceversa.

**Modo de gráficos**

**Graphics mode**

Antiguamente, los ordenadores necesitaban ser modificados y ampliados para poder realizar dibujos y pintar figuras en la pantalla. Los ordenadores personales, actualmente pueden (unos más y otros menos) mezclar textos, números y gráficos sin más que pulsar una tecla o darles un comando.

**Modulador UHF**

**UHF modulator**

Equipo mediante el cual las señales de **video** del ordenador son transformadas para adecuarlas a las normas establecidas para la transmisión en Ultra High Frequency en televisión.

**Monitor**

**Monitor**

Debía reservarse para designar el equipo que muestra la información: la pantalla; pero también se usa para designar el programa que permite controlar y observar la operación del ordenador trabajando en código máquina.

**Mota**

**Pixel**

La mínima área accesible de la pantalla que puede ser iluminada en uno u otro color. (Pixel es acrónimo de picture element).

**MUL**

**ROM**

Memoria Unicamente de Lectura. MUL designa con mayor propiedad a la memoria del ordenador que viene grabada de fábrica. El término original inglés Read Only Memory, es perfectamente válido, pero da la casualidad que la ROM es una memoria de acceso **Random** también.

**Notación decimal**

**Decimal notation**

**Notación hexadecimal**

**Hexadecimal notation**

Aunque el fundamento de los ordenadores es la notación binaria, también se usa el sistema en base 10 con sus dígitos 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 y todo lo demás.

Todo número puede expresarse también en cualquier otra base y la más útil en los ordenadores personales es la base 16, que en el CPC464 se puede usar directamente mediante el prefijo & o &H. (Y la binaria también, mediante el prefijo &X).

**Notación polaca inversa**

**Reverse polish notation**

Método usado internamente para las operaciones aritméticas y que algunos fabricantes propugnan con sus calculadoras. Es polaca por su origen, e inversa porque los signos de operación se colocan después de los operandos. Extremadamente útil.

**Números reales****Real numbers**

Conjunto de números formados por los enteros positivos y negativos, los racionales (3, 4) y los irracionales ( $\pi$ , e) el problema es que la llamada parte decimal de un número puede confundirte porque se refiere realmente a la parte fraccionaria (la que está detrás de la coma (hispano-parlantes) y detrás del punto (anglo-parlantes)).

**OCR****Optical Character Recognition**

Un medio de leer caracteres impresos o manuscritos con una lectora óptica para transferirlos a un soporte de información legible por la máquina.

**Octal****Octal**

Sistema numérico de base 8, con 8 símbolos posibles (0 a 7) que en binario se representa por 3 bits.

**PASCAL****PASCAL**

Lenguaje de alto nivel que empieza a ser asequible en los ordenadores personales.

**Periférico****Peripheral**

Primitivamente todos los equipos situados en la **periferia** del ordenador central como consolas, impresoras, ductoras de cinta y de disco, etc. Actualmente, sólo están en la periferia desde un punto de vista conceptual, pero materialmente pueden formar parte del ordenador, en especial teclados y ductoras de cinta para mejor aprovechamiento.

**Pinzar****PEEK**

Acción de sacar de su celdilla de memoria el contenido para aprovecharlo, simplemente **fisgar**, o examinar cómo se puede copiar un programa.

**Portabilidad****Portability**

Atributo para designar la mayor o menor facilidad que tiene un programa para ser traspasado de un sistema a otro.

**Portal****Port**

También 'puerto' y 'compuerta'. Un sitio directamente direccionable por el procesador y similar por tanto, a una celdilla de memoria en alguna manera, que permite la entrada y salida de datos del mundo exterior.

**Portor****Sprite**

No es un refresco. Sino el mecanismo por el que se **llevan** las figuras por toda la pantalla de un lado a otro. Y no hay 'espíritus', 'duendes', ni 'meigas' que valgan; es necesario trabajar bastante.

**Programa**

**Program**

Serie detallada y ordenada exactamente que gobierna el funcionamiento de todo ordenador.

**Programa de aplicación**

**Applications program**

Un programa con una labor específica que realizar, concretamente para realizar un proceso determinado y en contraposición con los utensilios de programación y operación, tales como programas compiladores, interpretadores, gestores de impresoras, ductoras de cassette, etc. que realizan tareas generales y ayudan a la programación y operatoria de un sistema.

**Programación estructurada**

**Structured programming**

Métodos y técnicas de programación sistemática partiendo de un análisis de la labor general y desglosándola en las tareas que la componen.

**Programalia**

**Software**

Todo lo referente al aspecto **logical** del funcionamiento de los ordenadores. Véase **Circuitalia**.

**PROM**

**ProgrammableROM**

Memoria Unicamente de Lectura de la que puede borrarse la información y volverse a "escribir" utilizando equipos relativamente simples y baratos.

**Puertas**

**Gates**

Componentes electrónicos que realizan las funciones lógicas primarias tales como YLIación (AND), OLIación (OR), OLEación (XOR), NegaciOn (NO)... No confundir con los **portales**.

**Punzar**

**POKE**

Acción de meter **-hincar-** en una determinada celdilla de memoria un determinado valor. Véase **Pinzar**.

**Reconocimiento de la voz**

**Speech recognition**

Investigaciones y estudios para que sirva directamente el habla como entrada de datos al sistema ordenador; que ya empiezan a traducirse en equipos comerciales.

**Recursión**

**Recursion**

Variante fundamental de los bucles e iteraciones usadas en programación que viene a consistir en algo así como lo que siempre se prohibía.

Lo definido no puede entrar en la definición.

**Redes****Networks**

Dos o más ordenadores estrechamente enlazados para el intercambio de datos e informaciones.

**Registro****Register**

Con lenguajes de alto nivel es una **ficha** de un fichero.  
En la máquina circuitos especiales para manipulación de los octetos e incluidos normalmente dentro del propio microprocesador. Véase **Acumulador**.

**Resolución****Resolution**

La capacidad para discriminar las motas y trazos componentes de una figura. Algunas veces aplicado **impropiamente** a la 'precisión' con que los números se manejan internamente.

**Ruido****Noise**

Además del significado normal, hace referencia a las posibilidades de algunos ordenadores para simular explosiones, bombazos, disparos, etc. y algunos, el CPC464 entre ellos, son expertos.

**Rutina****Routine**

**PARRAFO** de un programa que realiza una tarea concreta y determinada y que puede existir como 'módulo' incorporable en diferentes programas. Suele emplearse para aquellas tareas 'rutinarias' que se ejecutan constantemente en un programa.

**Salida****Output**

Todo lo que el ordenador **pone fuera** de sus dominios.

**Sarta****String**

También 'cadena', 'ristra', 'letrero'... En BASIC el concepto es fundamental y representa series de caracteres cualesquiera que se "entrecomillan" para indicar que han de tomarse letra a letra y no ser interpretados en absoluto. Por contraposición a **número**, también se les llama **lítero**.

**Sentencia****Statement**

Término cayendo en desuso por razones obvias. Se prefiere 'declaraciones' cuando son instrucciones en el programa que **enuncian** la clase de variables a usar, las tablas y ficheros, etc.

**Separador****Separator**

Suelen ser los **signos** de puntuación habituales , . ; : ... que permiten separar cláusulas dentro de una frase o datos en una lista.

**Operador**

**Operator**

A parte de tú mismo cuando estás ante la máquina, se refiere a los **signos** que indican la clase de operación a efectuar + - \* / ^ ( )...

**Simulación**

**Simulation**

Una técnica para remedar situaciones de la vida real usando el ordenador, en sistemas útiles para el aprendizaje práctico o análisis y previsiones de futuro.

**Sintetizador de voz**

**Speech synthesis**

Estudios, investigaciones y equipos que permiten generar los fonemas, palabras y frases del habla humana.

**Sistema operativo**

**Operating system**

Conjunto de programas residentes permanentemente en memoria o implantados desde la memoria auxiliar, encargados de las labores 'domésticas' del sistema del que viene a ser **gobernador**.

**Sobre-escribir**

**Overwrite**

No es reescribir sino más bien escribir encima de algo, borrando o no lo que hubiera.

**Subrutina**

**Subroutine**

Véase Rutina.

**Tabla de verdad**

**Truth table**

Diagrama que muestra en un solo bloque los operandos y los resultados de una operación lógica, y a pesar del nombre, se suele usar un 1 para indicar cierto y un 0 para indicar falso.

**Tablero gráfico**

**Graphics tablet**

Artefacto más o menos rectangular con una amplia superficie plana para que lo dibujado sobre el mismo sea directamente transferido al ordenador.

**Tecla programable**

**Soft key**

**Tecla funcional**

**Function key**

Una tecla a la que se le puede asociar una función a voluntad. El CPC464 dispone de una amplia variedad, y con capacidad para enviar hasta 32 caracteres con una sola pulsación.

**Teclado**

**Keyboard**

**Teclado numérico**

**Numeric keypad**

Lo que estás pulsando continuamente y que para mayor facilidad en el CPC464 se presenta en dos grupos separados, después de haber tenido en cuenta todos los factores ergonómicos.



**Teclado QWERTY****QWERTY Keyboard**

Observa el de una máquina de escribir y verás que al lado de la Q está la W, luego la E, luego la R y la T, y le sigue la Y, y si no, no es QWERTY.

**Terminal inteligente****Intelligent terminal**

Equipo donde además de satisfacer las necesidades de entrada y salida de datos, dispone también de capacidad de proceso propia.

**Terminal tonto****Dumb terminal**

Simplemente no tiene la 'mollera' que tiene el anterior.

**Tiempo real****Real time**

Cuando el ordenador proporciona respuestas a nuestras entradas de manera que podemos inmediatamente tomar las decisiones pertinentes, decimos que el ordenador trabaja en tiempo real. Si tenemos que aguardar a que concluya el proceso que está efectuando para poder actuar, es que trabaja en diferido.

**Transferencia descendente****Download**

En una red de ordenadores es la transferencia desde el ordenador central hacia los periféricos. Cuando la transferencia es en sentido contrario, es ascendente (Upload).

**Trazadora****Plotter**

Una clase especial de impresora que desplaza el papel en todos los sentidos y suele usar pluma o chorros de tinta en lugar de agujas de impacto. Usada primordialmente para dibujos técnicos.

**Unidad aritmeticológica****ALU**

Incorporada en el microprocesador generalmente y a veces como circuito electrónico separado, es el especialista en realizar dichas operaciones, recibiendo los operandos del microprocesador y suministrando los resultados de la operación.

**Utensilio (de programación)****Utility**

Programa y conjunto de programas que se usa frecuentemente para llevar a cabo tareas comunes en todos los ordenadores como copias ficheros, mantener catálogos, etc.

**Variable****Variable**

El elemento fundamental manejado por un programa que corresponde a un sitio de la memoria identificado por un 'nombre concreto' y donde se aloja el dato asociado.

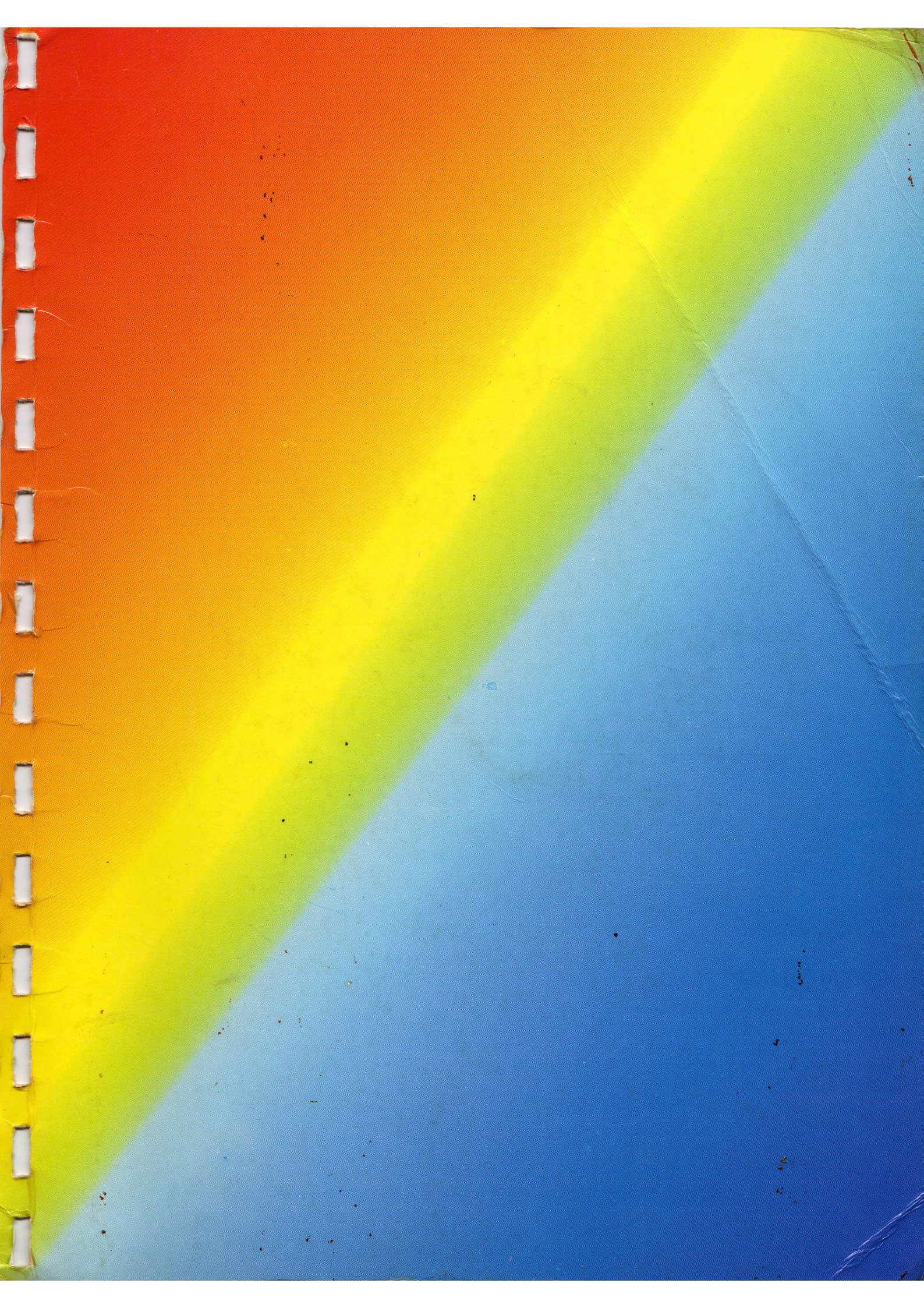
ABS 8.4  
AFTER 8.5; 10.1  
AND 4.18  
ASC 8.4  
ASCII 1.7; A.3.1  
ATN 8.5  
AUTO 4.19; 8.5  
BIN\$ 8.5  
BORDER F3.3; 4.15; 8.6  
CALL 8.6  
CAPS LOCK F2.3  
CAT 8.6  
CHAIN , CHAIN MERGE 8.7  
CHR\$ F3.9; 1.8  
CINT 8.8  
CLEAR 8.8  
CLG 8.9  
CLOSEIN 8.9  
CLOSEOUT 8.9  
CLR F2.3  
CLS F2.5; 8.9  
CONT 4.7; 8.10  
COPY F2.11; 1.19  
COS 8.10  
CREAL 8.11  
CTRL 1.9; 9.2  
DATA 4.18; 8.11  
DEF FN 8.12  
DEFINT, DEFREAL, DEFSTR 8.12  
DEG 8.12  
DEL F2.1  
DELETE 8.13  
DI 8.13  
DIM 4.15; 8.14  
DRAW F3.13; 8.15  
DRAWR F3.13; 8.15  
EDIT F2.10; 1.19; 8.15  
EI 8.16  
END 8.16  
ENT F3.20; 8.16  
ENTER F2.1; 1.10  
ENV F3.19; 8.18  
EOF 8.20  
ERASE 8.20  
ERL, ERR 8.20  
ERROR 8.21; A.8  
ESC F2.4  
EVERY 8.21; 10.3

EXP 8.22  
FIX 8.22  
FOR F2.13; 8.22  
FRE 8.23  
GOSUB F3.16; 8.23  
GOTO F2.7; 8.23  
HEX\$ 8.24  
HIMEM 8.24  
IF F2.12; 4.13; 8.24  
INK F3.5; 8.25  
INKEY 8.25  
INKEY\$ 8.26  
INP 8.26  
INPUT F2.8; 8.27  
INSTR 8.28  
INT 8.28  
JOY 8.28  
KEY 8.29  
KEY DEF 8.29  
LEFT\$ 8.30  
LEN 8.30  
LET 8.31  
LINE INPUT 8.31  
LIST F2.7; 1.16; 8.31  
LOAD 8.32  
LOCATE F3.9; 4.12; 8.32  
LOG 8.33  
LOG10 8.33  
LOWER\$ 8.34  
MAX 8.34  
MEMORY 8.34  
MERGE 8.35  
MID\$ 8.35  
MIN 8.36  
MOD 4.2  
MODE F3.1; 8.36  
MOVE 8.36  
MOVER 8.37  
NEW F3.14; 8.37  
NEXT F2.13; 8.38  
NOT 4.22  
ON BREAK GOSUB 8.39  
ON BREAK STOP 8.39  
ON ERROR GOTO 8.40; A.8  
ON GOSUB ON GOTO 8.38  
ON SQ GOSUB 6.12; 8.40  
OPENIN 8.41  
OPNEOUT 8.41  
ORIGIN F3.15; 8.42

OUT 8.42  
PAPER F3.2; 8.43  
PEEK 8.43  
PEN F3.2; 8.44  
PI 8.48  
PLOT F3.12; 8.45  
PLOTTR 8.45  
POKE 8.46  
POS 8.46  
PRINT F2.6; 8.47  
RAD 8.47  
RAM A.4  
RANDOMIZE 8.47  
READ 8.48  
RELEASE 8.48  
REM 8.48  
REMAIN 8.49  
RENUM 8.49  
RESTORE 8.50  
RESUME 8.50  
RETURN F3.16; 8.51  
RIGHT\$ 8.51  
RND 8.51  
ROUND 8.52  
RUN F2.6; 8.52  
SAVE F1.12; 8.53  
SGN 8.54  
SHIFT F2.1  
SIN 8.54  
SOUND F3.17; 6.1; 8.54; A.3; A.7  
SPACE\$ 8.55  
SPC 8.69  
SPEED INK 8.55  
SPEED KEY 8.56  
SPEED WRITE 8.56  
SQ 6.12; 8.57  
SQR 8.57  
STEP F2.13  
STOP 8.58  
STR\$ 8.58  
STRING\$ 8.58  
SYMBOL 8.59  
SYMBOL AFTER 8.59  
TAB 3.8; 8.70  
TAG 8.60  
TAGOFF 8.60  
TAN 8.61  
TEST 8.61  
TESTR 8.61

THEN F2.12;  
TIME 8.62  
TO F2.13  
TRON, TROFF 8.62  
UNT 8.62  
UPPER\$ 8.63  
USER PORT A.5  
USING 3.8; 8.71  
VAL 8.63  
VPOS 8.64  
WAIT 8.64  
WEND 8.65  
WHILE 8.65  
WIDTH 8.66  
WINDOW 5.12; 8.66  
WRITE 8.67  
XOR 4.22  
XPOS 8.67  
ZONE 3.8; 8.68







**EL MATERIAL EXHIBIDO  
ES SOLO PARA USO  
EDUCATIVO, NO COMERCIAL**

ENCICLOPEDIA DE  
REFERENCIA | EDUCATIVA

# RETROTECNIA

PARA ENTENDER LA EVOLUCION DE LOS ORDENADORES A TRAVES DEL TIEMPO

