
# INTRODUCCION AL DRAGONDOS

por

ALAN MAYER

Manual para el Sistema Operativo del Disco Dragón

Primera impresión 1983


El autor y Dragon Data Limited agradecen al Departamento de Esta dística del University College Swansea por las facilidades y ayudas reci bidas en la preparación de este manual.

## INDICE

PAGINA

## CAPITULO 1: EL PRINCIPIO



## CAPITULO 2: SIEMPRE HAY UN FICHERO PARA TODOS



## CAPITULO 3: LECTURA Y ESCRITURA



## CAPITULO 4: MAS PODER PARA SU DRAGON

## CAPITULO 1

# EL PRINCIPIO

### Conexión del Disk Driver

Antes de accionar el interruptor de la red, que sirve tanto para el Dragón como para las unidades del disco, éstas tienen que estar dispuestas como sigue:

(a) Conecte el cable desde el disk drive al cartucho del DOS (sistema operativo del disco).

(b) Coloque el disk drive sobre una superficie horizontal próxima al Dragón, asegurándose de que el cartucho se inserte bien en la puerta de cartuchos del Dragón y de que el cable no quede tenso.

(c) Introduzca el cartucho del DOS en la puerta de cartuchos y presione hasta que se acople. Conecte el cable de forma que no se suelte accidentalmente.

(d) Conéctelo a la línea mediante el enchufe existente en la parte posterior del disk driver.

(e) El disk driver se suministra con una tarjeta en la puerta del disco. Quítela en este momento tras haber realizado las operaciones anteriores.

(f) Asegúrese de que el Dragón y la TV están conectados debidamente y pulse el interruptor que suministra energía a todas las unidades.

La pantalla de TV tiene que mostrar, primeramente, los mensajes habituales de copyright y tras varios segundos un mensaje que indica que el disk driver está conectado. (Si no lo hace, apague y verifique todas las conexiones.) El mensaje tiene que ser seguido por el OK y el cursor habituales.

### Inserción del disco

El disco adecuado es de 5 ¼ pulgadas, de una cara y de doble densidad. El DOS puede controlar hasta cuatro drives con discos de cara doble

o sencilla, pero la unidad estándar es para discos de una cara. Se pueden utilizar discos de dos caras, pero únicamente será formateada una cara. Introduzca el disco en la ranura del drive con la ranura de acero de la cabeza abierta hacia el drive, y la ranura de protección hacia la izquierda. Asegure el disco bajando el cierre (el cual se corre un poco hacia adelante cuando está asegurado). Para quitar el disco presione el cierre y saque el disco.

### ¿Qué drive?

Si hay más de un drive enlazado con el DOS es importante que cada comando se dirija al drive adecuado. El comando DRIVE puede ser utilizado para suplir la ausencia de número de drive, DEFD (defecto de número de disco). Cuando un comando no especifica el número de drive se presenta DEFD. Al principio DEFD se dirige al 1.

DRIVE *n*

establece DEFD hasta *n* ($n=1, 2, 3$ ó 4).

### Preparación de un disco nuevo

Cuando se va a utilizar por primera vez un disco hay que formatearlo, "iniciarlo" en el trabajo que va a realizar. Esto se realiza mediante el comando

DSKINIT *drive, lados, pistas*

y pulsando la tecla [ENTER]. El parámetro *drive* indica el número del lector de disco (cuando se trabaja con más de uno); *lados* es el número de caras que se deben formatear (1 ó 2), y pistas es el número de *pistas* (*tracks*) por cara (40 u 80). En ausencia de algún valor para lados y pistas se tomarán 1 y 40, respectivamente. La ausencia de especificación para *drive* es DEFD (como se estableció para DRIVE). Para el disk driver estándar del Dragón es, habitualmente, suficiente ordenar DSKINIT. El disk driver se pondrá a funcionar durante cierto tiempo, tras el cual debe aparecer "OK". Si aparece un mensaje de error, saque y vuelva a introducir el disco e inténtelo de nuevo. Fallos repetidos en el formateo pueden dañar al disco.

Tenga en cuenta que un disco solamente debe ser formateado la primera vez que se utiliza. Los demás formateos que se le hagan borran las grabaciones que pueda tener el disco.

## Video RAM

El cartucho del DOS tiene como efecto trasladar la memoria de visualización en pantalla (video RAM) hasta una página debido a que la primera página empieza en el byte 3072 en lugar del 1536. Esto puede ser causa de problemas con programas previstos para ser utilizados con cassette. Si se escriben directamente sobre el área ocupada por el DOS, el sistema operativo del disco puede ser alterado. Para restablecerlo basta con apagar el ordenador y volver a conectarlo de nuevo.

## CAPITULO 2

# SIEMPRE HAY UN FICHERO PARA TODOS

### ¿Qué hay en el disco?

El comando DIR *drive*, en el que drive es el número de un disk drive determinado, hace que se presente una lista (directorio) de todos los ficheros contenidos en el drive. También suministra información sobre los bytes disponibles para nuevos ficheros (información que también se puede alcanzar por la función FREE). En ausencia del valor del *drive* se presenta DEFD (como se dijo en DRIVE). El nombre completo del fichero es de la forma

drive: *nombre del fichero. Tipo del fichero*

donde el *nombre del fichero* es el nombre que le da el usuario (hasta 8 caracteres, empezando siempre con una letra), y el *tipo del fichero* es un código de hasta 3 caracteres para distinguir ficheros con el mismo nombre. Los siguientes códigos de 3 letras se forman automáticamente para ayudar a distinguir ficheros de Basic, código-máquina, etc.:

BAS — Programa en Basic
BAK — Fichero duplicado (Backup)
BIN — Fichero binario, es decir, en código-máquina
DAT — Fichero de datos

Al final de cada línea de la lista aparecerá un número. Es el número de bytes colocados en ese fichero.

```
DIR 2
```

da el directorio de ficheros situados en disco en el drive número 2. Si el directorio es muy largo [SHIFT] y [@] pueden servir, al igual que con el comando LIST, para hacer pausas. Pulse cualquier tecla para volver a arrancar.

### Almacenar, cargar y ejecutar (saving, loading, running)

Almacenar (saving) un programa en Basic.

Si usted tiene en memoria un programa en Basic puede almacenarlo en disco mediante el comando

```
SAVE"PROGRAMA"
```

El nombre "Programa" puede ser reemplazado por cualquier otro nombre de hasta 8 caracteres, siempre que el primero sea una letra. Cuando pulse [ENTER] el drive empezará, durante varios segundos, a funcionar y aparecerá "OK" en la pantalla de la TV. Programas que pueden tardar un par de minutos en ser almacenados en cinta, se pasan a disco en varios segundos.

Ahora ya puede utilizar el comando DIR para listar sus ficheros. El fichero PROGRAM.BAS será listado. El número al final del programa indica el número de bytes ocupados por cada fichero.

El tipo de fichero BAS se asigna por defecto. La nomenclatura BAS, BAK, BIN, DAT está prevista para ayudarle a manejarse por el camino de los ficheros y para facilitarle un sistema fácil de reproducción de programas (backup). No obstante, usted puede establecer su propio sistema y utilizarlo, por ejemplo:

SAVE"PROG.A"
o SAVE"PROG.1A"

**Toma de programas de un disco**

Si un programa en Basic, llamado "PROGRAMA", ha sido almacenado en un disco, podrá ser cargado en memoria mediante el comando

LOAD"PROGRAMA"

Si desea cargar y ejecutar un programa en Basic puede utilizar el comando RUN, que ha sido incorporado al cartucho del DOS. Sin utilizar el comando LOAD puede teclear

RUN"PROGRAMA"

y el programa en Basic será cargado y ejecutado. También se puede alcanzar el mismo resultado con el uso de los dos comandos

LOAD"PROGRAMA"
RUN

Otra forma de cargar y ejecutar un programa es el comando CHAIN. Tiene los mismos efectos que el comando RUN, excepto que las variables conservan sus valores en lugar de ponerse a cero. El comando

CHAIN"PROGRAMA"

cargará y ejecutará el programa en Basic llamado PROGRAMA, sin colocar las variables en cero (y las cadenas en vacío).

Como ejemplo del uso de RUN y de CHAIN, haga la prueba introduciendo los siguientes pequeños programas:

```
10 Z=X+Y
20 PRINT"SUMA";Z
SAVE"SUMA"
10 Z=X*Y
20 PRINT"PRODUCTO"=";Z
SAVE"PRODUCTO"
10 Z=(X+Y)/2
20 PRINT"MEDIA=";Z
SAVE"MEDIA"
10 INPUT"X";X
20 INPUT"Y";Y
SAVE"INPUT"
```

Ahora posee cuatro pequeños ficheros: SUMA.BAS, PRODUCTO.BAS, MEDIA.BAS e INPUT.BAS.

Puede utilizar

```
RUN"INPUT"
```

para asignar valores X e Y. Por ejemplo:

```
X? 10 [ENTER]
Y?  6 [ENTER]
```

Entonces el comando CHAIN puede ser utilizado para realizar la suma, producto o media.

```
CHAIN"PRODUCTO"
```

dará

```
PRODUCTO=60
```

y

```
CHAIN"MEDIA"
```

dará

```
MEDIA=8
```

y así sucesivamente.

Para dos números este método es engorroso, pero puede ser útil en programas con gran cantidad de datos.

CHAIN puede ser utilizado con un parámetro:

```
CHAIN"PROGRAMA",entrada
```

donde *entrada* es el número de línea en el que debe empezar la ejecución.

## Almacenar y cargar en Código-máquina

Se puede almacenar código-máquina (o cualquier otra serie de bytes) en disco mediante

SAVE"CODIGO",*principio,fin,entrada*

donde *principio* es la dirección inicial del código en memoria, *fin* la dirección del byte que sigue al último del código en memoria y *entrada* es el punto de entrada (en su defecto, el valor EXEC). "CODIGO" puede ser reemplazado por cualquier otro nombre lícito de archivo.

El archivo aparecerá en directorio como fichero del tipo BIN (un fichero binario). Los ficheros binarios pueden ser cargados utilizando

LOAD"CODIGO.BIN",*inicio*

donde *inicio* es la nueva dirección de comienzo. La ausencia del valor *inicio* implica la dirección de arranque del código original (cuando haya sido almacenado). La dirección ausente EXEC se calcula con relación al *inicio.*

## Duplicado (Backup) de ficheros

Cuando almacena (SAVE) un programa en Basic utilizando un nombre que ya está en el directorio, la versión vieja pasa a ser del tipo BAK y la nueva se registra como del tipo BAS. Cualquier versión previa del tipo BAK es borrada. Por ejemplo, supongamos que ya existe PROGRAMA.BAS y que va a almacenar un nuevo programa utilizando

SAVE"PROGRAMA"

Habrá dos ficheros:

PROGRAMA.BAS
PROGRAMA.BAK

PROGRAMA.BAS es la segunda (última) versión; PROGRAMA.BAK es la primera versión.

Si desea almacenar otro programa utilizando

SAVE"PROGRAM"

seguirá habiendo grabadas dos versiones, pero ahora PROGRAM.BAS será esta tercera versión y PROGRAM.BAK será la segunda versión. La primera ya no quedará grabada.

La versión backup puede ser cargada utilizando

LOAD"PROGRAM.BAK"

(o RUN, o CHAIN); así si descubre que su última versión tiene errores y desea volver a su versión backup, será posible hacerlo.

El sistema de backup para ficheros y datos binarios se opera de forma similar. Si dos o más ficheros de tipos distintos tienen el mismo nombre, los ficheros backup tendrán el mismo nombre y se superpondrán en la grabación; así solamente el fichero más reciente es el que tendrá versión backup.

### Copia y cambio de nombre de ficheros

Los ficheros pueden ser duplicados mediante el comando COPY. El comando

COPY"FICHERO1.BAS" TO "FICHERO2.BAS"

hace que FICHERO1.BAS tenga como copia FICHERO2.BAS. Ambos ficheros, entonces, existentes en el disco. Tenga en cuenta que es esencial incluir el tipo de fichero (en este caso BAS) en ambos casos. El nuevo tipo de fichero no tiene por qué ser del mismo que el viejo, pero es aconsejable llevar un sistema lógico.

Si la finalidad es solamente rebautizar el fichero (p. ej.: el original ya no se necesita), entonces se debe utilizar el comando

RENAME"FICHERO1.BAS" TO "FICHERO2.BAS"

Esto produce el mismo efecto que la orden COPY, excepto en el caso de que se borre FICHERO 1.

La orden de copia es particularmente útil cuando los ficheros deben ser transferidos de un disco a otro (cuando se dispone de más de un disk driver).

COPY"1:PROG.BAS" TO "2:PROG.BAS"

copia PROG.BAS del disco 1 en el disco 2.

Usted tiene varias posibilidades en la forma de dirigirse a los ficheros. El número del drive puede venir antes o después del nombre del fichero, siempre que se coloquen dos puntos entre ambos. Por ej.: 1:PROG.BAS y PROG.BAS:1 se refiere al mismo fichero. También se puede reemplazar el "." por "/"; así, PROG.BAS y PROG/BAS indican lo mismo.

## Administración

Si crea muchos ficheros se puede llegar a la saturación del disco. Esta eventualidad se puede prever consultando los bytes libres (utilizando bien la función FREE o el comando DIR). Si intenta grabar un fichero demasiado largo, el fichero será creado tomando todo el espacio libre del disco, pero aparecerá DF ERROR. El fichero creado puede no entrar íntegro, por lo que deberá ser borrado y vuelto a grabar en otro disco.

Para evitar este problema es aconsejable borrar todos los ficheros que ya no se quieren, tanto pronto dejan de interesar. Esto se puede hacer con el comando KILL.

```
KILL"PROGRAM.BAS"
```

borrará el fichero PROGRAM.BAS. Al igual que con COPY y RENAME, hay que incluir el nombre del fichero.

## Juntemos todas las piezas

Un programa en Basic soportado en disco puede ser combinado con otro programa en Basic contenido en la memoria por medio de la orden MERGE. El programa dominante es el del disco, por lo que si coinciden los números de línea, se retendrán los del disco. El programa resultante residirá en la memoria y la versión original del disco permanecerá en él.

Por ejemplo, si el fichero FICHERO1.BAS contiene

```
10 INPUT"A";A
20 INPUT"B";B
30 SUM=A+B
40 PRINT"SUMA=";SUM
```

y el FICHERO2.BAS contiene

```
25 INPUT"C";C
30 SUM=A+B+C
```

entonces teclear

```
LOAD"FICHERO1"
MERGE"FICHERO2"
```

producirá el siguiente programa en memoria:

```
10 INPUT"A";A
20 INPUT"B";B
```

```
25 INPUT"C";C
30 SUM=A+B+C
40 PRINT"SUMA=";SUM
```

### Para su protección

Para ahorrarse el disgusto de teclear KILL"PROG.BAS" y descubrir que ha destruido dos horas de trabajo, es muy interesante aprender el uso de PROTECT.

Cualquier fichero puede ser protegido del borrado o de la sobreimpresión mediante el comando

PROTECT ON"FICHERO.BAS"

Ordenes como KILL"FICHERO.BAS" no se cumplirán a menos que se quite la protección con

PROTECT OFF"FICHERO.BAS"

Mientras está protegido un fichero aparecerá una "P" con fondo invertido en la línea del menú de ficheros donde figura el nombre de aquél.

### Una copia aislada, por si acaso

Para los que no quieren colocar todos los huevos en el mismo cesto hay un método sencillo para crear copias de seguridad (backup). Si dispone de más de un drive esto se podrá hacer de un solo golpe.

BACKUP *fuente* TO *destino, caras, pistas*

creará una copia de seguridad, pista por pista, del disco en el drive considerado *fuente* en el disco del drive considerado *destino*. Los parámetros *caras* y *pistas* son, respectivamente, el número de caras (1 ó 2) y pistas (40 u 80). Si no se figuran esos valores, se tomarán *fuente*=DEFD, *destino*=DEFD, *caras*=1, *pistas*=40.

Cuando *fuente* y *destino* son iguales (como cuando no se les pone un valor) se trata de un asunto de dos discos con un solo drive. En estas circunstancias, cuando se utiliza el comando BACKUP, el Dragón le informará alternativamente de qué disco colocar, con "INSERT SOURCE" ("INTRODUZCA FUENTE") o con "INSERT DESTINATION" ("INTRODUZCA DESTINO"), hasta que se haya transferido la totalidad del contenido.

Las pistas del menú siempre se verifican durante el BACKUP; otras pistas serán verificadas si está dada ("ON") la orden "VERIFY" (la con-

dición normal al conectar el aparato). Esta verificación puede ser suprimida mediante el comando VERIFY OFF y reactivada con VERIFY ON.

Como el comando BACKUP utiliza todo el equipo disponible, para lograr una copia más rápida es aconsejable borrar el contenido de la memoria mediante la orden NEW antes de iniciar el BACKUP.

### Varias funciones útiles

La información sobre los ficheros se da en el menú de ficheros que se visualiza mediante el comando DIR. La longitud de cada fichero y la totalidad del espacio aún libre del disco se pueden encontrar utilizando las funciones FREE y LOF.

PRINT FREE, *drive*

dará el número de bytes libres en el drive número *drive*. Si no se define el *drive*, es DEFD.

PRINT LOF"FICHERO.BAS"

dará la longitud del fichero FICHERO.BAS en bytes. El tipo de fichero (p. ej.: BAS) tiene que ser especificado.

Esto puede ser utilizado en programas, así:

```
10 X=FREE
20 Y=LOF"PROG.BAS"
30 PRINT"HAY";X;"BYTES DISPONIBLES,"
40 PRINT" Y EL PROGRAMA TOMA";Y;"BYTES."
```

## CAPITULO 3

# LECTURA Y ESCRITURA

Además de la ventaja obvia del disco sobre el cassette de la velocidad, hemos visto que cargar ficheros es más fácil gracias al menú de ficheros —no es necesario pasar a través de docenas de ficheros hasta que se encuentre el buscado— (o ¡no necesita una maleta llena de cassettes!). Esta ventaja es muy importante cuando se trata de leer y cscribir datos.

Los datos pueden ser almacenados en disco de dos formas: en ficheros, como se hace con los programas, o escribiendo directamente en un sector determinado de una pista determinada.

### Ficheros de datos

El camino más fácil para utilizar ficheros de datos es mediante los comandos FREAD y FWRITE, en su forma más simple. Para utilizarlos así no es necesario crear previamente un fichero o abrirlo (OPEN).

El programa

```
10 FWRITE"FICHERO";X,Y,Z
```

crea el fichero FICHERO.DAT (en caso de que no exista previamente) y graba en ese fichero los valores X, Y y Z. En consecuencia, el programa

```
20 FREAD"FICHERO";A
```

leerá los tres valores del "FICHERO.DAT" como un número. Esto se debe a que el comando FWRITE no escribe un finalizador detrás de cada valor, por eso se puede recurrir a una forma ampliada del comando (ver página 16). Para obviar esto se puede forzar la aparición de un finalizador de la siguiente manera:

```
10 FWRITE"FICHERO";X,",",Y,",",Z
20 FREAD"FICHERO";A,B,C
```

Con las cadenas se plantea el problema que se muestra en el siguiente programa:

```
10 X$="ESTO ES, CLARAMENTE, UNA SENTENCIA"
20 FWRITE"OUTPUT";X$
30 FREAD"OUTPUT";A$
40 PRINT A$
```

El resultado de la ejecución del programa es

```
ESTO ES
```

así la coma ha sido tomada como finalizador para la cadena.

El programa puede ser continuado:

```
50 FREAD"OUTPUT";B$,C$
60 PRINTB$:PRINTC$
```

y así dará:

```
ESTO ES
CLARAMENTE
UNA SENTENCIA
```

Para superar este problema se dispone del comando FLREAD:

```
10 X$="ESTO ES, CLARAMENTE, UNA SENTENCIA"
20 FWRITE"OUTPUT";X$
30 FLREAD"OUTPUT";A$
40 PRINT A$
```

El nuevo programa da

```
ESTO ES, CLARAMENTE, UNA SENTENCIA
```

De la forma descrita el comando FWRITE siempre empezará escribiendo al fin de fichero, extendiendo el fichero hasta donde llegue. Sin embargo, es aconsejable borrar las versiones previas de los ficheros de datos antes de volver a ejecutar los programas.

Para una utilización más sofisticada de los ficheros de datos, los comandos FWRITE, FREAD y FLREAD se pueden utilizar como parámetros:

FWRITE"NOMBRE", FROM *inicio*, FOR *longitud*; variables
FREAD"NOMBRE", FROM *inicio*, FOR *longitud*; variables
FLREAD"NOMBRE", FROM *inicio*, FOR *longitud*; cadenas

El parámetro FROM (*inicio*) selecciona el byte en el que se inicia el fichero. Para FWRITE las carencias son el final del fichero; para FREAD y FLREAD las carencias son el principio del fichero cuando el "read" es ejecutado por primera vez.

El parámetro FOR (*longitud*) determina la longitud de la grabación que debe ser escrita (en el caso de FWRITE) o el número de bytes que el contador debe ser avanzado (en el caso de FREAD y FLREAD). En ausencia de valor es igual a la longitud de la grabación que será procesada.

Antes de que un valor FROM pueda ser utilizado con FWRITE, la posición en el fichero (y el fichero mismo) tiene que existir. Esto puede ser logrado con el comando CREATE.

CREATE"FICHERO", *tamaño*

crea el fichero FICHERO.DAT de longitud *tamaño* (en su ausencia 0).

Lo que sigue es un ejemplo de la utilización de esos comandos:

```
10 X$="PRIMERO":X=50:Y$="SEGUNDO":Y=80
20 FWRITE"FICHERO",FOR20;X$
30 FWRITE"FICHERO",FOR20;Y$
40 FWRITE"FICHERO",FROM10;X
50 FWRITE"FICHERO",FROM30;Y
```

Siempre que FICHERO.DAT no exista previamente, la línea 20 lo creará, dándole longitud 20, con la grabación "PRIMERO" al principio. La línea 30 escribe "SEGUNDO", empezando en fin del fichero en curso y extiende el fichero 20 bytes.

La línea 40 introduce el valor de X en el byte 10.

La línea 50 introduce el valor de Y en el byte 30.

Note cómo la línea 40 lee

```
40 FWRITE"FICHERO",FROM10,FOR20;X
```

entonces la grabación "SEGUNDO" será sobrepasada con ceros.

Lo que sigue puede ser utilizado para leer la grabación:

```
60 FREAD"FICHERO";A$,A,B$,B
70 PRINTA$,A,B$,B
```

Lo que se convierte en:

```
PRIMERO          50
SEGUNDO          80
```

Otra forma de leer solamente los valores es:

```
60 FREAD"FICHERO",FROM10,FOR20;A
70 FREAD"FICHERO";B
80 PRINTA,B
```

Aquí la línea 60 lee el valor de A, empezando en el byte 10, y avanza el puntero hasta 30 (añadiendo 20 al inicio de 10), dejándole dispuesto para leer B en la línea 70.

La función LOC puede ser utilizada para determinar la posición del puntero en el fichero.

```
PRINT LOC"FICHERO"
```

da el byte próximo para ser leído en FICHERO.DAT.

### Acceso aleatorio simulado

La forma de direccionar FROM/FOR permite el almacenamiento y acceso a registros indexados con cualquier orden. La estructura de la base de datos que se expone a continuación ilustra lo anterior.

Se necesitan dos pequeños programas. El primero se utiliza para crear el fichero de datos y el segundo para tomar los datos almacenados:

```
 10 INPUT"NOMBRE";N$
 20 INPUT"NUMERO DE REGISTROS";N
 30 INPUT"LONGITUD MAXIMA DE CADA REGISTRO";L
 40 CREATE N$,N*L+20
 50 FWRITE N$, FROM 0, N
 60 FWRITE N$,FROM 10;L
 70 INPUT"NUMERO DE REGISTRO"; I
 80 PRINT"REGISTRO";I;
 90 INPUTR$
100 FWRITEN$, FROM(I-1)*L+20, FOR L; R$

SAVE"ALMACENAMIENTO"

 10 INPUT"NOMBRE";N$
 20 FREAD N$, FROM 0; N
 30 FREAD N$, FROM 10; L
 40 INPUT"NUMERO DE REGISTRO";I
 50 FLREAD N$, FROM(I-1)*L+20, FOR L; R$
 60 PRINT R$: GOTO40

SAVE"TOMA"
```

El fichero de datos se crea utilizando

```
RUN"TOMA"
```

El primer programa pide un nombre de archivo (cualquier nombre legal puede servir); luego el número máximo de registros. Como test dele el valor 5, y de longitud máxima de registro (la longitud de la cadena mayor que intente almacenar), 20 por ejemplo. Intente introducir los siguientes registros (en cualquier orden):

1 ALICIA
2 ROBERTO
3 CAROLINA
4 DAVID
5 EDUARDO

Los registros pueden ser reescritos, p. ej. puede reemplazar 4 DAVID por 4 DANI. Entonces el programa se termina con la tecla [BREAK]. Ahora los registros pueden ser recuperados utilizando

RUN"TOMA"

El nombre del fichero tiene que ser especificado de nuevo, pero se pueden leer otros parámetros del fichero. Ahora si introducimos un número de 1 a 5 recuperaremos el registro que interese.

### ¡Demasiados ficheros!

Los comandos FWRITE, FREAD y FLREAD pueden ser utilizados con diferentes ficheros simultáneamente, pero cada vez que se crea o se accede a un nuevo fichero se realiza por la izquierda. Se pueden tener abiertos hasta 10 ficheros simultáneamente. (Un fichero puede ser abierto para lectura y escritura simultáneamente, pero cuenta solamente como un solo fichero.) Cualquier intento de abrir el undécimo fichero dará TF ERROR. Todos los ficheros de un disco determinado pueden cerrarse utilizando el comando CLOSE.

Esto produce el efecto de situar los punteros (a los que se llega con la función LOC) a cero.

CLOSE, *drive*

cierra todos los ficheros del disco número *drive*. Si no se especifica el *drive* se cierran todos los ficheros de todos los discos.

### ¿Es el final?

Un problema que se presenta frecuentemente cuando se leen registros de un fichero es saber cuándo parar. Una función útil en esta situación es EOF. Si incluye la sentencia

X=EOF("FICHERO")

en su programa, normalmente X tomará valor 1. Pero si el puntero está al final de FICHERO.DAT (esto es, que no haya más registros que leer) tomará el valor 0.

El uso típico de EOF es en la carga de una matriz de datos de un fichero cuando no esté seguro del número de valores que tiene que leer.

```
10 I=0
20 FREAD"FICHERO DE DATOS"A(I):I=I+1
30 IF EOF("FICHERO DE DATOS")=1 THEN 20
```

Por supuesto, normalmente se necesita una sentencia DIM.

### Lectura y escritura sin ficheros

Cuando un disco se ha formateado está dividido en pistas y cada una de éstas en sectores. El drive normal trabaja con 40 pistas y 18 sectores por pista. Cada sector tiene 256 bytes.

Cada sector de 256 bytes puede ser accedido individualmente con los comandos SWRITE y SREAD.

Para escribir un registro en un sector, el registro está ensamblado en dos cadenas, es decir, X$ e Y$, cada una con un máximo de 128 bytes. Se debe utilizar entonces el comando

SWRITE *drive, pista, sector*, X$, Y$

donde *drive* es el número del disk driver (1-4), *pista* es el número de pista (0-39 ó 0-79) y *sector* es el número del sector (1-18 ó 1-36). X$ e Y$ pueden ser sustituidas por cualquier variable cadena, por lo que pueden contener letras, números y símbolos o estar ensambladas utilizando la función CHR$. Si una de las cadenas tiene menos de 128 bytes, el espacio reservado en el sector será llenado con caracteres CHR$(0).

Para tomar datos directamente de un sector el comando es

SREAD *drive, pista, sector*, X$, Y$

siendo el significado de los parámetros el mismo que en SWRITE.

Tanto si las variables cadena escritas en el sector ocupan la totalidad de los 128 bytes, como si no, las cadenas que se toman tendrán esa longitud. Contendrán los bytes de la cadena original seguidos por series de caracteres CHR$(0). Será preciso borrar (CLEAR) el espacio de cadenas suficiente para completar los 128 bytes cada una.

## CAPITULO 4

# MAS PODER PARA SU DRAGON

Además de los comandos específicamente relacionados con el disco, el cartucho del DOS contiene varios comandos y funciones para potenciar su intérprete Basic.

### ¿Dónde he colocado mi memoria?

Hay dos incorporaciones a la función MEM muy útiles; éstas son HIMEM y FRE$.

PRINT HIMEM

da la dirección de memoria más alta disponible en Basic. Al conectar el aparato ésta tiene el valor 32766. Si el espacio está reservado para código máquina o datos, utilizando el segundo parámetro del comando CLEAR, entonces HIMEM tomará el valor 1 menos el segundo parámetro; veamos el siguiente programa como ejemplo:

```
10 CLEAR 200, 32000
20 PRINT "MEMORIA MAXIMA EN BASIC=",HIMEM
```

da el resultado

```
MEMORIA MAXIMA EN BASIC=31999
```

```
PRINT FRE$
```

da el número de bytes libres disponibles para cadenas (al conectar el Dragón es 200). Aunque termine con "$", FRE$ es una variable numérica y no una cadena. Esto es especialmente útil para programas que requieren la introducción de cadenas a través del teclado. Cuando el espacio se vuelve escaso puede aparecer en pantalla, sugiriendo un cambio de disco. La sentencia del programa puede ser

```
IF FRE$<255 THEN PRINT "CAMBIO NECESARIO"
```

La referencia a FRE$ desencadena una "recolección de basura", es decir, el almacenamiento de las cadenas se reorganiza para hacer que el uso del espacio disponible sea lo más eficaz posible. Puede también ser utilizada durante las pausas intencionadas de los programas para evitar esas interrupciones tan molestas que algunas veces se producen en las grandes rutinas de manejo de cadenas.

## No se atasque por los errores

Siempre que el Dragón detecta un error para el programa y envía un mensaje de error. Para la mayoría de los errores es esencial parar y corregir el programa. También hay errores producidos por el operador, como teclear datos erróneos. El programa siguiente da un ejemplo sencillo:

```
10 INPUT"X";X
20 INPUT"Y";Y
30 PRINT"X DIVIDIDA ENTRE Y=",X/Y
40 GOTO 10
```

El camino obvio por el que el operador puede "romper" este programa es dándole a Y el valor 0. El programa entonces se parará con el mensaje 10 ERROR IN 30. Por supuesto, siempre podremos verificar cada valor de Y tal como haya entrado para comprobar el cero, pero se dispone de otro método más adecuado (y poderoso).

La sentencia ERROR GOTO *n* indica al Dragón que si se detecta un error se debe pasar el control a la línea número *n*. Una vez se ha hecho, hay dos funciones que ayudan a identificar el error: ERL da el número de línea en donde ha ocurrido el error, y ERR da el número de código (cada tipo de error tiene un número de código distinto).

El código para /0 es 20, por lo que se deben añadir las líneas siguientes para superar el error:

```
 5 ERROR GOTO50
50 IF ERR=20 THEN PRINT "NO SE PUEDE DIVIDIR POR
   CERO":GOTO10
60 PRINT"ERROR NUMERO";ERR;"EN LINEA";ERL
```

Ahora si introducimos 15 para la variable X y 0 para Y tendremos el mensaje "NO SE PUEDE DIVIDIR POR CERO", pero el programa continuará. Si damos a X el valor 1E50 tendremos "ERROR NUMERO 10 EN LINEA 10" (el error 10 es OV ERROR).

Al final de este manual, en el Apéndice 2, hay una lista de errores.

## Al tercer intento

Un comando que estamos seguros de que atraerá su atención es BEEP. Incluido en un programa en Basic, BEEP produce un pitido "biiip". BEEP *n* produce *n* pitidos.

### ¡Espere un momento!

Para esos momentos en los que las pantallas de texto desaparecen antes de que se hayan podido leer tenemos WAIT. WAIT *n* suspende la ejecución del programa durante *n* milisegundos. Así WAIT 1000 da un segundo, WAIT 10000 da 10 segundos, etc. Aquí va un ejemplo sencillo.

```
10 CLS
20 PRINT "DEBE TENER TIEMPO","PARA LEER ESTO"
30 WAIT 2500
40 CLS
```

### Un cambio fácil

Una función útil, especialmente en áreas como programas "de salidas", es SWAP. Permite cambiar el valor de dos variables sin el problema rutinario del almacenamiento temporal de la variable.

```
SWAP X, Y
```

da a X el valor de Y, y a Y el de X.

### Tomemos cien líneas

Supongamos que ha tecleado una línea con un número erróneo (por ejemplo, 11 en lugar de 110), lo que es un buen problema. La línea no solamente no está en su lugar, sino que, además, actúa donde no debe, y posiblemente se superpone con otra línea. Estas cosas no ocurrirán si se utiliza AUTO para la numeración automática de líneas.

AUTO *inicio, incremento*

le da números de línea empezando por el *inicio* y pasando a la siguiente con el *incremento* determinado. Usted teclea las líneas finalizándolas con la tecla [ENTER]; entonces para volver al modo normal basta teclear [ENTER]. AUTO se superpone a lo que esté en memoria con números de línea coincidentes; no borra (NEW) el programa anterior, por lo que se puede utilizar para insertar líneas. Si no se dan valores a *inicio* y a *incremento* se tomarán, respectivamente, 100 y 10.

El ejemplo siguiente muestra cómo se puede utilizar AUTO:

```
AUTO100,100
OK
100 PRINT"PRIMERA PAGINA"
200 PRINT"SEGUNDA PAGINA"
300 PRINT"TERCERA PAGINA"
400 [ENTER]
```

Hemos decidido en este punto que deberíamos haber utilizado varias sentencias CLS:

```
AUTO50,100
OK
50 CLS
150 CLS
250 CLS
350 CLS
450 [ENTER]
```

Pero ¡la velocidad de lectura tiene sus límites! Debemos insertar algunas pausas:

```
AUTO120,100
OK
120 WAIT2500
220 WAIT2500
320 WAIT2500
420 [ENTER]
```

Esto producirá el efecto buscado.

## Otro DOS

Para cargar otro sistema operativo del disco está el comando BOOT. Carga el sistema en memoria en el byte 9728. El comando para su ejecución es entonces EXEC 9730.

El comando BOOT toma la forma siguiente:

BOOT*n*

donde *n* es el número del disk drive (en su ausencia DEFD).

# APENDICE 1
# COMANDOS DEL DRAGONDOS

## AUTO

La orden AUTO proporciona automáticamente números de línea para un programa.

AUTO
produce líneas empezando por 100 e incrementándose en pasos de 10.

AUTO 50,5
genera líneas empezando por 50 e incrementándose en pasos de 5.

La forma AUTO termina presionando [ENTER] inmediatamente después de aparecer el número de una línea.

AUTO sobreescribe líneas que tienen números inadecuados, pero deja otros sin cambios.

## BACKUP

La orden BACKUP crea una copia backup de un disco completo (1 ó 2 caras).

BACKUP 30 TO 4, 2, 80
crea una copia del disco 3 en el disco 4 (2 caras, 80 pistas).

Si los discos de origen y de destino van a usar el mismo drive, su Dragón le dará instrucciones para insertar el disco fuente y el de destino alternativamente.

BACKUP
crea una copia de un disco de origen en un disco de destino utilizando solamente el drive DEFD, 1 cara, 40 pistas.

## BEEP

La orden BEEP da un sonido de aviso, "beep" (bip).

BEEP
da 1 bip.

BEEP 20
da 20 bips.

## BOOT

La orden BOOT almacena un sistema operativo de un disco en la memoria, empezando en 9728.

BOOT
almacena el sistema del disco DEFD.

BOOT 2
almacena el sistema del disco número 2.

El sistema se ejecuta utilizando EXEC9730.

## CHAIN

La orden CHAIN se puede utilizar para cargar y ejecutar un programa, preservando los valores de las variables. Esto es útil cuando más de un programa se va a poner en funcionamiento en el mismo conjunto de datos.

CHAIN"PROG2"
carga PROG2.BAS desde el drive DEFD y comienza la ejecución en la primera sentencia sin inicializar las variables a cero, ni la cadena en nulo.

CHAIN"PROG3",50
almacena el PROG3.BAS de la transmisión DEFD y comienza la ejecución en la línea 50, sin inicializar las variables a cero ni las cadenas en nulo.

CHAIN"2:PROG"
realiza en la operación CHAIN en el programa PROG.BAS del disco en el drive número 2.

Cuando se usa la orden CHAIN con programas que utilizan cadenas se recomienda forzar una colección de estériles al principio de cada programa CHAINed (encadenado). Esto se puede conseguir incluyendo una llamada a la función FRE$ como la primera sentencia del programa. Por ejemplo:

```
10 ZZ = FRE$ 'COLECCION FORZADA DE ESTERILES'
20 REM RESTO DE PROGRAMAS ENCADENADOS
   SIGUIENTES
```

## CLOSE

La orden CLOSE se utiliza para cerrar ficheros que se han abierto por órdenes como FWRITE, FREAD y FLREAD. Un máximo de 10 ficheros se puede abrir en cualquier momento de forma que algunas veces es necesario cerrar ficheros antes de abrir otros. Cuando se cierra un fichero, su indicador de lectura (accedido por la función LOC) está colocado en cero.

CLOSE
cierra todos los ficheros de todos los discos.

CLOSE 2
cierra todos los ficheros del disco en el drive 2.

## COPY

La orden COPY se utiliza para hacer copias de ficheros. La copia y el original pueden ser del mismo disco o de diferentes discos (si usted tiene más de un drive).

COPY"ORIGINAL.BAS"TO"NEWFILE.BAS"
hace a NEWFILE.BAS una copia de ORIGINAL.BAS (en el disco en la transmisión DEFD). El mecanografiado de archivo (ej.: BAS) se tiene que incluir en la especificación de ambos ficheros.

COPY"1:FILE.BAK"TO"2:BACKUP.AAA"
hace el fichero BACKUP.AAA en el disco 2 una copia de FILE.BAK en el disco 1.

## CREATE

La orden CREATE se puede utilizar para crear un archivo de datos.

CREATE"DATAFILE"
crea DATAFILE.DAT con longitud de 0 bytes.

CREATE"DATAFILE",80
crea DATAFILE.DAT con longitud de 80 bytes.

## DIR

La orden DIR hace listas de los ficheros de un disco y da el número de espacios asignados a cada fichero y el número de espacios libres del disco.

DIR
hace una lista de ficheros en el disco del drive número DEFD (como colocado para DRIVE).

DIR 2
hace una lista de los ficheros del disco en el drive número 2. Un menú típico de archivos es como sigue:

```
DIR
BASPROG         .BAK    1855
MACHCODE        .BAK    3809
BASPROG         .BAS    1855
DATAFILE        .DAT    3000
MACHCODE        .BIN    1009
163072 FREE BYTES
```

En este menú, BASPROG.BAK es un backup del programa en Basic. BASPROG.BAS, MACHCODE.BAK es un backup del fichero binario MACHCODE.BIN, y DATAFILE.DAT es un fichero de datos.

## DRIVE

La orden DRIVE selecciona el número de drive que falte, DEFD. Todas las órdenes que no especifican una transmisión particular se les asignará la transmisión DEFD hasta que se use otra vez DRIVE. Recién conectado, DEFD está colocada en 1.

DRIVE 3
coloca DEFD en 3.

## DSKINIT

La orden DSKINIT se usa para formatear un disco nuevo y establecer un menú de ficheros. No se puede aplicar ninguna otra orden a un disco hasta que éste está formateado.

DSKINIT
formateará una cara del disco en el drive número DEFD (1, excepto que sea alterado por DRIVE), estableciendo 40 pistas. Esta orden puede tardar alrededor de un minuto en ejecutarse.

DSKINIT 3, 2, 80
formateará ambas caras del disco en el drive 3, estableciendo 80 pistas por cara.

## EOF

La función EOF se utiliza para comprobar si el indicador de lectura está al final de un fichero (ej.: si todas las grabaciones se han leído). La sentencia del programa

```
X=EOF("FILE")
```

dará a X el valor 0 si el indicador está al final de FILE.DAT, y el valor 1 en el resto.

Note que para esta función se requieren los paréntesis.

## ERL

Cuando ocurre un error, la función ERL da el número de línea en el que ha ocurrido el error.

Después de cualquier error en la línea 50,

```
PRINT ERL
```

volverá al 50.

## ERR

Cuando hay un error, la función ERR da el número de código del error.

Después de un IO ERROR(input/output)

```
PRINT ERR
```

dará valor 42 (el código para IO).

Los códigos de error están en una lista al final de este Manual, en el Apéndice 2.

## ERROR GOTO

La orden ERROR GOTO dirige el control a una línea particular si un error es detectado posteriormente.

Después de que la orden

```
ERRORGOTO5000
```

ha sido ejecutada; si se detecta cualquier error, el control pasará a la línea 5000. Si se ejecuta otra sentencia ERROR GOTO, esto se superpondrá con la instrucción previa.

## FLREAD

La orden FLREAD se utiliza para leer una cadena de un fichero. Las comas y los dos puntos no se consideran como terminadores.

FLREAD"FILE";(cadena)
lee una cadena de FILE.DAT, empezando en la posición del indicador de lectura para ese fichero (inicialmente al principio del fichero). El indicador de lectura es actualizado en el byte siguiente a la cadena leída.

FLREAD"FILE",FROM100,FOR60;cadena
lee una cadena del FILE.DAT, comenzando en el byte 100. Después de leerlo, el puntero de lectura avanza al espacio 160 (100 + 60).

Cualquier fichero al que se ha accedido por FLREAD está "abierto" por la izquierda. Se pueden abrir hasta un máximo de 10 ficheros.

## FREAD

La orden FREAD se utiliza para leer grabaciones de un fichero.

FREAD"FILE",(lista de variables)
lee la lista de variables de FILE.DAT, empezando en la posición del indicador de lectura para ese fichero (inicialmente al principio del fichero). El indicador de lectura se sitúa en el byte siguiente a la última grabación leída.

FREAD"FILE",FROM 30,FOR 50; lista de variables
lee la lista de variables de FILE.DAT, empezando en el byte 30.
Después de la lectura, el indicador de lectura ha avanzado al espacio 80 (30 + 50).

Las cadenas terminan con comas y dos puntos, al igual que los caracteres "end of line" (fin de línea).

Cualquier fichero al que se accede por FREAD está "open" (abierto) por la izquierda. Se pueden abrir hasta un máximo de 10 ficheros.

## FREE

La función FREE da el número de bytes libres en un disco.

PRINT FREE
da el número de bytes libres en el disco en el drive número DEFD.

PRINT FREE2
da el número de bytes libres en el disco del drive número 2.

FREE puede incluirse en programas. Ej.:

```
X=FREE1
```

## FRE$

La función FRE$ proporciona el número de bytes disponibles para ser utilizados por cadenas. Si no se trata de una cadena, tiene valor numérico. La utilización de FRE$ obliga a utilizar datos de relleno ("garbage collection").

PRINT FRE$

facilita el número total de bytes libres para ser asignados a cadenas, después de que las cadenas ya presentes hayan sido almacenadas correctamente.

## FWRITE

La orden FWRITE se utiliza para escribir grabaciones en un fichero.

FWRITE"FILE", lista de variables
escribe la lista de variables en FILE.DAT. Si FILE.DAT no existe ya, se crea, y la escritura empieza desde el principio. Si existe, la escritura empieza al final del archivo.

FWRITE"FILE",FROM 10, FOR 30; lista de variables
escribe la lista de variables en FILE.DAT, empezando en el byte 10 y extiende la longitud de la grabación a 30 bytes. El fichero debe existir ya y tener al menos 9 bytes para hacer posible empezar en el byte 10.

Cualquier fichero al que se ha accedido por FWRITE está "abierto" por la izquierda. En cualquier momento dado se pueden abrir un máximo de 10 ficheros.

## HIMEM

La función HIMEM da la posición más alta en memoria disponible en Basic. Esto se altera por el segundo parámetro de una sentencia CLEAR.

PRINT HIMEM
volverá a 32766, a menos que se haya usado CLEAR para reservar espacio al final de la RAM.

## KILL

La orden KILL se utiliza para suprimir ficheros de un menú, descargando el espacio del disco para otros ficheros.

KILL"PROG.BAS"
suprime el archivo PROG.BAS del disco en el drive número DEFD. El mecanografiado del fichero (ej.: BAS) debe incluirse en la especificación.

KILL"2:CODE.BIN"
suprime el fichero CODE.BIN del disco en el drive número 2.

## LOAD

La orden LOAD se puede utilizar para trasladar archivos en Basic o Binarios del disco a la memoria.

LOAD"BPROG"
cargará el programa en Basic del fichero BPROG.BAS del drive DEFD a la memoria.

LOAD"CODE.BIN"
cargará el fichero en código binario CODE.BIN en el mismo área de memoria en la que se grabó originalmente, y situará la dirección ausente EXEC en el punto de entrada especificado cuando se grabó el fichero.

LOAD"MACH.BIN",1000
cargará el fichero en código binario MACH.BIN en la memoria, empezando en la dirección 1000. Si se salvó originalmente MACH usando

SAVE"MACH",comienzo,final,entrada

entonces la dirección ausente EXEC se situará en 1000 + entrada + comienzo.
Note que este es un convenio diferente (y más flexible) que el utilizado por CLOADM.

LOAD"PROG.BAK"
cargará el archivo "PROG.BAK". Si este es el backup de un programa en Basic, se almacenará en la posición normal del Basic; si es un fichero binario se cargará como el CODE.BIN de arriba.

LOAD"2: PROG"
cargará PROG.BAS del disco en el drive 2.

## LOC

La función LOC se usa para encontrar la posición del puntero en un fichero.

LOC"DATAFILE"
Da el número del byte siguiente para ser leído en DATAFILE.DAT. Para una orden posterior "read" será el parámetro ausente "FROM".

## LOF

La función LOF da la longitud de un fichero en bytes.

PRINTLOF"CODE.BIN"
da la longitud de CODE.BIN en bytes.

LOF se puede utilizar en programas. Ej.:

```
X=LOF"PROG.BAS"
```

El mecanografiado del fichero (ej.: BIN o BAS) se tiene que incluir.

## MERGE

La orden MERGE se utiliza para fusionar un fichero de un disco con un programa en Basic en memoria. El fichero del disco no es afectado. El resultado en la memoria es un programa que contiene todos los números de línea del fichero y los números de línea originalmente en la memoria que no estaban en el fichero. Ej.: el fichero y el programa original se mezclan, pero si hay algún número de línea en común, son las versiones de archivo las retenidas.

MERGE"PROG"
une el programa en Basic PROG.BAS, en el disco DEFD, con cualquier programa en la memoria, sobreescribiendo donde los números de línea coinciden.

MERGE sólo puede usarse con ficheros en Basic.

## PROTECT

La orden PROTECT sitúa un fichero en una categoría "protegida" de forma que no sea sobreescrito o borrado.

PROTECT ON"PROG.BAS"
hace que el PROG.BAS sea protegido, de forma que KILL"PROG.BAS" y SAVE"PROG" conducen a PT ERROR.

PROTECT OFF"PROG.BAS"
quita la protección.

Aparecerá una "P" en carácter invertido en pantalla al final de los ficheros protegidos, en el fichero.

## RENAME

La orden RENAME se utiliza para cambiar el nombre de un fichero en un disco, o para trasladar un fichero de disco a disco.

RENAME"OLDNAME.BIN"TO"NEWNAME.BIN"
da al fichero OLDNAME.BIN el nuevo nombre NEWNAME.BIN. Note que el mecanografiado de fichero (en este caso BIN) se debe incluir en la especificación de ambos nombres.

## RUN

La orden RUN se utiliza para trasladar un fichero en Basic del disco a la memoria y ejecutarlo, empezando por la primera sentencia.

RUN"PROG1"
carga PROG.1 BAS del drive DEFD y RUN (ejecuta) el programa.

RUN"3:FILE.BAK"
carga el programa en Basic de backup FILE.BAK del drive número 3 y la ejecuta (RUN).

## SAVE

La orden SAVE se puede utilizar para transferir programas en Basic o en código máquina (u otras secuencias binarias) de la memoria al disco.

SAVE"PROG."
creará un fichero PROG.BAS en el drive DEFD que contiene el programa en Basic en curso en la memoria. Si un fichero PROG.BAS ya existe, será transferido al fichero PROG.BAK. Si un fichero PROG.BAK ya existe, será sobreescrito.

SAVE"CODE",2000,5001,4000
creará un fichero CODE.BIN conteniendo la secuencia de bytes desde la dirección de comienzo 2000 a la dirección de final 5000. Cuando este fichero es cargado, la dirección EXEC se colocará en 4000. Si un fichero CODE.BIN ya existe, será transferido a CODE.BAK, etc.

El número de drive del disco se puede especificar poniendo *drive* delante del nombre del fichero.
SAVE"2: PROG"
salvará PROG.BAS en el drive 2.

## SREAD

La orden SREAD se utiliza para leer la grabación contenida en un solo sector de un disco. La grabación es de 256 bytes y se lee en dos cadenas. Ej.: A$ y B$, ambos de longitud de 128 bytes.

SREAD 2,6,10,A$,B$
lee la grabación de 256 bytes contenida en el décimo sector de la sexta pista del drive de disco 2, situando los primeros 128 bytes en A$ y los últimos 128 bytes en B$. Incluso si están formados mayormente por caracteres CHR$ (0), ambos A$ y B$ serán de longitud 128 bytes.

## SWAP

La función SWAP cambia los valores de dos variables.

SWAPX,Y
da a X el valor de Y y a Y el valor de X.

## SWRITE

La orden SWRITE se usa para escribir una grabación en un sencillo sector de un disco. La grabación se monta en dos cadenas (ej.: X$ e Y$), cada una de longitud máxima de 128 bytes.

SWRITE 3, 7, 9, X$, Y$

escribe las grabaciones contenidas en X$ e Y$ al noveno sector de la séptima pista del disco en el drive 3. Si cualquiera X$ o Y$ es menor de 128 bytes, el espacio extra se llenará con valores CHR$ (0).

## VERIFY

Las órdenes VERIFY ON y VERIFY OFF inician y finalizan, respectivamente, el procedimiento de comprobación. Este está normalmente disponible (con la potencia encendida) y actúa una verificación en la pista del menú.

## WAIT

La orden WAIT suspende la ejecución del programa, produciendo una pausa fácilmente controlable.

WAIT5000
hace pausa de 5000 milisegundos (5 segundos).

# APENDICE 2
# CODIGOS DE ERROR

## CODIGOS DE ERROR

| | | |
|---|---|---|
| 0 | NF | NEXT sin FOR. |
| 2 | SN | Error de sintaxis. |
| 4 | RG | RETURN sin GOTO. |
| 6 | OD | READ, sin datos. |
| 8 | FC | Llamada ilegal de función. |
| 10 | OV | Exceso de carga. |
| 12 | OM | Sin memoria. |
| 14 | UL | Línea no definida. |
| 16 | BS | Indice erróneo. |
| 18 | DD | Redimensión de la matriz. |
| 20 | /0 | División por cero. |
| 22 | ID | Sentencia directa errónea. |
| 24 | TM | Mecanografiado erróneo |
| 26 | OS | Sin memoria para cadenas. |
| 28 | LS | Cadena demasiado larga. |
| 30 | ST | Fórmula para cadena demasiado compleja. |
| 32 | CN | No se puede continuar. |
| 34 | UF | |
| 36 | FD | Fichero erróneo. |
| 38 | AO | Archivo ya abierto. |
| 40 | DN | Número del drive. |
| 42 | IO | Error de entrada/salida. |
| 44 | FM | Fichero en modo erróneo. |
| 46 | NO | Fichero sin abrir. |
| 48 | IE | Entrada pasado EDF. |
| 50 | DS | Sentencia directa. |
| 128 | NR | No preparado. |
| 134 | RT | Mecanografiado de grabación. |
| 136 | RF | Grabación no encontrada. |
| 138 | CC | Redundancia cíclica. |
| 140 | LD | Datos perdidos. |
| 142 | BT | |
| 144 | IV | Menú inválido. |
| 146 | FD | Menú lleno. |
| 148 | DF | Disco lleno. |

| | | |
|---|---|---|
| 150 | FS | Fichero Spee |
| 152 | PT | Protección. |
| 154 | PE | Leer pasado EOF. |
| 156 | FF | Fichero no encontrado. |
| 158 | FE | Fichero que existe. |
| 160 | NE | No existente. |
| 162 | TF | Demasiadas aperturas. |
| 164 | PR | Parámetro. |
| 166 | ?? | |

Eurohard

Españoleto, 25
Teléfs. (91) 410 30 64 - 410 31 96 - 410 34 98
Télex 45845 ICSG E
28010 MADRID