APRENDE DE TUS ERRORES

DEPURACION DE PROGRAMAS

Por muy experto que sea un programador, es difícil que un determinado programa le funcione a la primera. Es muy fácil cometer un error y nadie está libre de ello. Por esto, la última fase en el proceso de escribir un programa es la depuración, es decir, la corrección de errores. En esta serie, vamos a aprender de nuestros propios errores para corregirlos con más facilidad.

Jesús ALONSO

Tanto si escribes tu propio programa, como si lo copias de una revista, es muy fácil que cometas un error. En ocasiones habrás tecleado una «b» en lugar de una «d», o una «O» en lugar de un «0». Otras veces, te habrás olvidado de definir una variable o la habrás inicializado con un valor incorrecto. En algunos casos, el programa se detendrá con un informe de error, en otros no se detendrá, pero tampoco hará lo que esperabas de él. En cualquier situación, la corrección de errores es muy fácil si se sigue una sistemática de actuación y se tiene una cierta práctica.

En esta serie, empezaremos por estudiar la depuración de programas Basic. En este lenguaje, es muy difícil que el ordenador se quede «colgado» por un error, y siempre tendremos acceso al listado para corregirlo. Por otro lado, el propio intérprete es capaz de detectar algunos errores e informarnos de ello, lo que será de gran ayuda.

En Basic, existen tres tipos de errores: Errores de **sintaxis**, errores detectables en tiempo de **proceso** y errores **lógicos**. Veamos que son cada uno de ellos.

Errores de sintaxis

Al igual que en el lenguaje ordinario, se trata de errores en la forma de escribir las sentencias. Por ejemplo, poner **«PINT»** en lugar de **«PRINT»** o utilizar unos operandos distintos de los que requiere un determinado comando, por ejemplo, **«PLOT A$»**.

El Sistema Operativo del Spectrum no permite equivocarse en los nombres de los «tokens», ya que estos entran de una sola pulsación y, por otro lado, detecta los restantes errores de sintaxis en el mismo momento en que se introduce la línea, informándonos de ello con un signo de interrogación parpadeante **«?»**. Los errores de sintaxis no son difíciles de corregir y suelen deberse a que se ha olvidado algún operando o a que se ha escrito un carácter separador en lugar de otro (por ejemplo, un punto «.» en lugar de una coma «,»). Una causa muy frecuente de errores de sintaxis es olvidarse de cerrar las comillas de un literal, por ejemplo:

```
PRINT "PUNTUACION=;pun
```

En lugar de:

```
PRINT "PUNTUACION=";pun
```

No dedicaremos más tiempo a este tipo de errores, ya que se resuelven con una simple ojeada a la línea. En caso de duda, puede resultar útil una consulta al manual o a nuestro Curso de Basic para ver qué estructura tiene un determinado comando.

Errores detectables en tiempo de proceso

Los americanos, con su habitual pragmatismo, los denominan «Run-time errors». Se trata de aquellos errores que son detectados por el intérprete de Basic y detienen la ejecución haciendo que se presente un informe de error. Son los errores más frecuentes, y a los que más tiempo dedicaremos.

El intérprete nos informará del error a través de la última línea de la pantalla. En ella nos presenta un informe que consta de cuatro partes:

1. **Código de error:** Se trata de un número o una letra que va asociada a un determinado tipo de error.
2. **Mensaje de error:** Se trata del mensaje de error propiamente dicho. En los Spectrum de la versión española, estos mensajes están en castellano, en los de la versión inglesa, salen en inglés. En ambos casos, se trata de una frase que nos indica, de forma muy breve, lo que ha pasado.
3. **El número de línea** del programa donde el error ha sido detectado.
4. **El número de sentencia**, dentro de la línea, donde el error ha sido detectado.

Por ejemplo, en el mensaje:

```
2 Variable not found, 1230:3
```

El «2» es un código del error, la frase «Variable not found» es el mensaje de error. A continuación siempre hay una coma «,» y un espacio. Después viene el número de línea, que en nuestro casos es «1230» y finalmente, y separado por dos puntos «:», el número de comando dentro de la línea, en nuestro caso «3».

«Variable not found» significa «Variable no encontrada», por lo que el significado completo de este informe de error es que no se encontró alguna variable en la tercera sentencia de la línea 1230.

Luego veremos, uno a uno, los distintos mensajes de error. De momento, es importante tener en cuenta que el hecho de que un error se detecte en determinada línea no implica, necesariamente, que sea esa lí-

nea la causante del error, por tanto, no bastará con revisar la línea que nos indica el informe de error, sino que la mayor parte de las veces este error vendrá arrastrado desde atrás y será necesario buscar la línea que lo ha originado. Supongamos el siguiente programa:

```
100 LET d=15
110 PRINT "La variable 'b' es i
gual a ";b
```

El programa se detendrá con el informe:

```
2 Variable not found,110:1
```

El informe nos indica que no se encuentra una variable en la primera sentencia de la línea 110. La única variable que hay en esta línea es «b», y está bien escrita. El error está en la línea 100, donde hemos puesto «d» en lugar de «b». Este es uno de los muchos casos en que el error está en una línea distinta de aquella en la que se detecta.

Errores lógicos

Se denomina así a aquellos errores que no son detectados por el intérprete de Basic y, por tanto, no detienen la ejecución del programa. Sin embargo, hacen que el programa no realice lo que esperabamos de él. Son, sin duda, los más difíciles de corregir aunque, afortunadamente, los menos frecuentes. Suelen deberse a errores en el planteamiento del programa o a despistes durante la programación.

Veamos un ejemplo de error lógico: Supongamos que hemos escrito un juego en el que nuestro héroe debe enfrentarse a unos enemigos. Cada vez que matemos a un enemigo, aumentará la puntuación en 10 puntos y se escribirá la nueva puntuación resultante, en algún lugar de la pantalla. Para ello, tenemos una subrutina que se encarga de sumar los puntos y escribirlos en la pantalla, pero nos hemos olvidado de hacer un «GOSUB» a esa rutina cada vez que se elimina a un enemigo. El programa funciona correctamente, pero la puntuación permanece siempre a «cero». Existe un error: en algún lugar del programa, falta un «GOSUB», pero el intérprete es incapaz de detectarlo.

Para depurar este tipo de errores, existe, en Basic, el comando «TRON» (abreviatura de «TRace ON») que va imprimiendo en pantalla los números de línea por donde va pasando el programa. Desgraciadamente el Spectrum carece del comando «TRON», por lo que tendremos que recurrir a algunos «trucos» que más adelante explicaremos y que nos permitirán localizar con facilidad este tipo de errores.

De momento nada más, la semana próxima empezaremos a ver, uno por uno, los errores detectables en tiempo de proceso y sus posibles causas, así como la sistemática para resolverlos.

APRENDE DE TUS ERRORES

Esta semana, empezamos a ver los errores de programación que el intérprete de Basic es capaz de detectar en tiempo de proceso, es decir, mientras está ejecutando el programa.

Jesús ALONSO RODRIGUEZ

Cada vez que el intérprete de Basic ejecuta un comando, tiene que hacer una serie de operaciones. Si alguna de ellas no pudiera ser realizada porque existe un error en el programa (falta un dato, se le pide que haga algo que no puede hacer, etc.), el intérprete de Basic detiene la ejecución del programa imprimiendo, en la parte baja de la pantalla, un informe en el que nos indica qué tipo de error ha detectado y dónde.

Existen 28 mensajes de error distintos, cada uno de los cuales tiene un código que puede ser un número del 0 al 9 o una letra de la «A» a la «R». Los veremos uno a uno indicando en qué situaciones pueden producirse y cuál será la forma de actuar ante ellos.

0 OK

SIGNIFICADO: Indica que se ha terminado la ejecución de una orden sin que el intérprete encontrara ningún error. En el caso de que se presente al terminar un programa, el mensaje es una respuesta al comando «RUN» que lo puso en marcha.

CAUSA: No se ha detectado ningún error durante la ejecución del comando.

SOLUCION: Si todo ha salido correctamente, no deberemos preocuparnos. Si, por el contrario, el programa no ha hecho lo que esperábamos de él, querrá decir que nos encontramos ante un error lógico. Este tipo de errores los estudiaremos más adelante.

1 NEXT Whithout FOR

SIGNIFICADO: Indica que el intérprete ha llegado a una sentencia «NEXT» sin que exista su correspondiente sentencia «FOR».

CAUSA: La variable a que hace referencia la sentencia NEXT no ha sido definida previamente en una sentencia «FOR», pero sí existe como una variable ordinaria. Si no existiera ni como variable de control de bucle, ni como variable ordinaria, el informe de error que se obtendría sería: «2 Variable not found».

SOLUCION: Lo más probable es que el error se encuentre en la misma línea donde se ha detectado. Seguramente, nos habremos equivocado al escribir la variable después del «NEXT». Si hubiéramos definido la variable primero en un «LET» y luego la definiéramos en un «FOR», la variable sería redefinida como variable de control y el error no se produciría. Si, por el contrario, utilizamos la variable con un «LET» dentro del bucle, actuaremos sobre la variable de control, pero ésta no será redefinida, por lo que seguirá siendo identificada por la sentencia «NEXT». De hecho, una de las formas más frecuentes de salir de un bucle «FOR...NEXT» es incrementar la variable de control por encima de su límite.

Un posible error lógico se podría producir si utilizamos la misma variable en dos bucles, pero nos olvidamos de redefinirla en el segundo de ellos. Por ejemplo:

El segundo «NEXT I» sería ignorado por el ordenador, y no se imprimiría ningún mensaje de error, ya que la variable «I» existe como variable de control, pero ya salió del primer bucle con un valor superior al de su límite.

No hay que olvidar que el hecho de que el intérprete no haya pasado por la sentencia «FOR», no significa que ésta no exista. Puede haber un «GOTO» que haya traído al intérprete hasta aquí saltando por encima de la sentencia «FOR».

Queríamos poner:	Y ponemos:
FOR I=1 TO 20	FOR I=1 TO 20
.....	.....
.....	.....
NEXT I	NEXT I
.....	.....
.....	.....
FOR I=3 TO 7	.....
.....	.....
.....	NEXT I
NEXT I	

2 Variable not found

SIGNIFICADO: Indica que el intérprete no encuentra, en el área de variables, una de las que necesita utilizar para ejecutar el comando en curso.

CAUSA: Este informe de error se producirá si no se encuentra cualquier variable, ya sea de cadena, numérica, de control de bucle FOR...NEXT, o una matriz (array). La causa es siempre la misma; la variable no ha sido definida previamente, o ha sido definida y luego borrada. Hay varias formas de definir una variable, veámoslas una por una:

LET: Es el comando, por excelencia, que se utiliza para definir variables. La variable que se define de nuevo es la que va antes del signo « = ». Si esta variable ya existiera, sería reemplazada por la nueva, de hecho, pasaría a ocupar un lugar distinto en el área de variables (salvo que se trate de una matriz), el efecto es el mismo que si se borrara y se volviera a definir.

READ: Si la variable a la que se asigna el valor leído no existe, este comando la crea.

INPUT: Exactamente igual que READ; de hecho, son comandos que trabajan de igual forma, salvo que INPUT lee desde el teclado (o cualquier otra corriente abierta como entrada) y READ lee desde una sentencia DATA dentro del propio programa.

FOR: Este comando crea una nueva variable que queda identificada como variable de control de bucle. Si existiera previamente una variable ordinaria con el mismo nombre, la borra y crea una nueva variable de control con ese nombre.

DIM: En el caso de DIM se crea una variable de matriz que no será recreada por ninguno de los comandos anteriores. Una variable de matriz y otra de cual-

quier otro tipo, pueden convivir con el mismo nombre salvo que se trate de variables de cadena. Si se crea una variable con DIM, que ya existe previamente, la anterior será borrada para crear la nueva.

LOAD: Al cargar un programa, también se cargan las variables que estuvieran definidas cuando éste se salvó, por tanto, es posible usar en un programa variables que no hayan sido definidas en el mismo. Ojo a esto que es un truco de protección usado con bastante frecuencia.

Tampoco hay que olvidar que el error se puede producir tanto si la variable no ha sido definida, como si ha sido borrada. Los comandos que borran las variables son RUN y CLEAR (NEW borra las variables, pero también borra el programa). Si se ejecutara, con RUN, un programa que se cargó con variables, éstas se borrarían. Por ello, suele ser mejor ejecutar los programas desconocidos con GOTO 1 que con RUN (de hecho, la auto-ejecución con LINE funciona como un GOTO, ya que no borra las variables).

SOLUCION: Ante un error de este tipo, la sistemática de actuación es, siempre, la misma:

1. Listar la línea donde se ha producido el error.
2. Anotar todas las variables que utiliza el comando que ha producido el error (hay que tener en cuenta que en una sentencia IF... THEN, el IF es un comando, y lo que hay después del THEN es otro).
3. Hacer PRINT con todas las variables, una por una; alguna de ellas dará de nuevo el informe de error.
4. Examinar el listado de el programa hacia atrás, para seguir la pista a la variable que ha producido el error.

Veamos un ejemplo: Nuestro programa se ha detenido con el informe:

```
2 Variable not found, 1090:2
```

Listamos la línea 1090 con: LIST 1090

```
1090 IF A < 3 THEN LET
B=15*C+D*LEN A$
```

Tenemos las siguientes variables «C», «D» y «A$». Vamos a imprimirlas:

```
PRINT C
PRINT D
PRINT A$
```

En uno de estos tres comandos directos, habremos obtenido, de nuevo, el error:

```
2 Variable not found, 0:1
```

Supongamos que ha sido PRINT A$. No hay más que seguir la pista de A$ hacia atrás, para ver dónde tenía que haberse definido y no se ha hecho.

3 Subscript wrong

SIGNIFICADO: «Subíndice erróneo». Indica que el valor de una expresión (las expresiones pueden ser complejas, pero una variable o un simple número también son tratados como una expresión) que se ha utilizado como subíndice de una matriz, no concuerda con las dimensiones que le asignó su correspondiente sentencia DIM.

CAUSA: El caso en que más frecuentemente se produce este error es cuando se explora una matriz mediante un bucle FOR...NEXT, cuya variable actúa de subíndice, y se sobrepasa la longitud de una de las dimensiones.

Otra posible causa es que no se haya redimensionado la matriz cuando debería haberse hecho. De todas formas, la matriz tiene que existir, ya que de lo contrario, se obtendría el informe: 2 Variable not found.

Por último, cabe la posibilidad de que nos hayamos equivocado al dimensionar la matriz o (más probablemente), al escribir el nombre de la variable que actúa de subíndice. Supongamos que queríamos poner: A$(b,c) y hemos puesto A$(d,c). En el caso de que «d» llegue a un valor superior al máximo que puede adoptar «b», se produciría este error. Por otro lado, es lo mejor que podría ocurrir, ya que si «d» nunca llegase a sobrepasar el máximo valor de «b», nos encontraríamos ante un error lógico que nos costaría bastante detectar.

SOLUCION: En primer lugar, deberemos listar la línea donde se ha detectado el error, para comprobar si nos hemos equivocado en el nombre de la variable que actúa como subíndice, o si se trata de un error al escribir un número. Si no es ninguna de estas causas, buscaremos la línea donde hemos dimensionado la matriz y veremos cuál es el valor máximo para cada una de sus dimensiones. A continuación, imprimiremos el contenido de cada variable que actúe como subíndice, y ya sabremos cuál es la que está produciendo el error. En este punto, no hay más que seguir la pista, hacia atrás, a esa variable, para ver dónde adquiere un valor incorrecto.

4 Out of memory

SIGNIFICADO: «No hay suficiente memoria». Quiere decir que la zona de memoria denominada «espacio de reserva», se ha agotado y las restantes áreas de memoria no pueden expandirse más a costa de ella. Por tanto, no hay suficiente memoria para lo que se pretende hacer.

CAUSA: El intérprete de Basic tiene que expandir un área determinada de memoria cada vez que va a guardar un dato en ella, o a crear un espacio de trabajo. Cada expansión de memoria se hace a costa del área de reserva, y de cada vez, se comprueba si hay suficiente sitio. Caso de no haberlo, se produce este error.

Una causa muy frecuente, suele ser por intentar correr, en un Spectrum de 16 K, un programa que está escrito para correr con 48 K.

Otra posibilidad, es que se provoque una expansión indefinida de la pila de GOSUB por un proceso recursivo mal definido; por ejemplo, una subrutina que estuviera continuamente llamándose a sí misma. Este caso puede ilustrarse con el siguiente ejemplo:

```
100 GOSUB 100
```

Por lo demás, el error puede producirse con casi cualquier operación del ordenador, sobre todo, si el programa ocupa mucho sitio y se está trabajando al límite de la memoria o si se tiene la RAMTOP demasiado baja.

SOLUCION: Si está copiando o cargando un programa y su ordenador es de 16 K, compruebe que el programa sea, efectivamente, para 16 K. Si, por el contrario, su ordenador es de 48 K, conviene que lea el valor de la variable P-RAMT con la siguiente sentencia:

```
PRINT PEEK
23732+256*PEEK 23733
```

Podría ocurrir que tuviera una avería de memoria y el ordenador no la estuviera utilizando en su totalidad. En un Spectrum de 48 K «sano», esta sentencia tiene que imprimir el número: 65535 (El interface de impresora de MHT baja este número aproximadamente 1 K).

Otra posibilidad es que el error se deba a un mal planteamiento del programa, quizá está pretendiendo almacenar demasiados datos. Compruebe si ha bajado demasiado la RAMTOP, o si ha dimensionado una matriz demasiado grande.

Si todo lo anterior falla y es poca la memoria que necesitamos, cabe recurrir a alguna de las técnicas de optimización de memoria; por ejemplo, poner «SGN PI» donde ponga el número «1» o «NOT PI» donde ponga el número «0». En el caso de cualquier otro número, se puede encerrar entre comillas y anteponerle un «VAL», por ejemplo, la sentencia:

```
PRINT AT 20,0;A$
```

Se podrían convertir en:

```
PRINT AT VAL «20»,
NOT PI;A$
```

Pasando de ocupar 21 bytes a ocupar 13. Si se hace esto en todo el programa, puede reducirse considerablemente su ocupación de memoria.

APRENDE DE TUS ERRORES

El informe «Out of screen» no siempre indica lo que parece. Para interpretarlo es necesario conocer como está estructurada la pantalla en el Basic del Spectrum.

Jesús ALONSO RODRÍGUEZ

5 Out of screen

SIGNIFICADO: «Fuera de pantalla». Indica que al intentar imprimir en uno de los canales de pantalla, la expansión de éste exige quitarle el sitio al otro.

CAUSA: Podría parecer que una instrucción como:

```
PRINT AT 12,33;A$
```

daría dar lugar a este mensaje. Lo cierto es que esta instrucción provocaría el mensaje «B Integer out of range». Hemos intentado imprimir fuera de la pantalla y, sin embargo, no ha aparecido ese informe. Entonces, ¿cuándo se obtiene el «Out of screen»?

El Spectrum maneja, en realidad, dos pantallas. Una de ellas es la «parte alta» que consta, normalmente, de 22 líneas y se direcciona por el canal «S» (normalmente unido a la corriente #2). La otra es la «parte baja» que consta, normalmente, de dos líneas y se direcciona por el canar «K» (normalmente unido a las corrientes #1 y #0).

Por defecto, el ordenador dirige las instrucciones PRINT por la corriente #2 y las hace ir a parar a la parte alta de la pantalla, pero también podemos hacer que se dirijan por la corriente #1, por ejemplo:

```
PRINT #1;A$
```

En este caso, imprimiríamos en la parte baja. De igual forma, el comando INPUT se dirige a la parte baja, aunque en este caso, no podemos hacer INPUT #2 ya que el canal «S» no puede utilizarse como entrada.

Cuando imprimimos en la parte baja (ya sea con INPUT o con PRINT #1), ésta se expande a costa de la parte alta. Puede ver el efecto con el programa:

```
100 BORDER 5
110 FOR I=1 TO 10
120 PRINT #1; «PARTE BAJA»
130 NEXT I
```

Si la expansión de este canal amenazara con cubrir completamente la parte alta, obtendríamos el mensaje «Out of screen». Se puede visualizar cambiando la línea 110 por:

```
110 FOR I=1 TO 30
```

La variable del Sistema DF-SZ (dirección 23659) contiene el número de líneas asignado a la parte baja de la pantalla. Normalmente es 2, pero podemos cambiarlo para que sea, por ejemplo, 1 (este cambio se puede hacer dentro del programa con POKE 23659,1, pero si se detuviera el programa, el ordenador se «colgaría», ya que el Sistema Operativo no podría presentar el informe correspondiente). La parte alta de la pantalla contiene, siempre, un número de líneas que es 24 menos las que tenga la parte baja. Normalmente, la parte baja tiene 2, así que la parte alta tendrá 22 (de la 0 a la 21) si intentáramos imprimir en la 23.ª línea, la parte alta intentaría expandirse amenazando con cubrir la parte baja, y obtendríamos, de nuevo, el mensaje «Out of screen». Sin embargo, si intentáramos imprimir en la 24.ª, el mensaje que obtendríamos será «Integer out of range» en vez de «Out of screen», ya que la sentencia PRINT AT no admite un número mayor de 22.

Si nos salimos por la derecha, no se produce el «Out of screen», sino el «Integer out of range».

SOLUCION: Este error se tiene que producir en una sentencia INPUT o en una PRINT. En el primer caso, el texto que hemos puesto como salida de INPUT es más largo de 736 caracteres, por ejemplo:

```
INPUT (A$);B
```

Donde LEN A$ sea mayor de 736. Si la sentencia que ha provocado el error es de este tipo, se debe comprobar la longitud de la variable de salida (en nuestro caso A$).

Si el error se ha producido en una sentencia PRINT, lo más probable es que exista una variable, que controle la línea de impresión, a la que no se le ha puesto límite, por ejemplo:

```
PRINT AT a,b;A$
```

Produciría este error si «a» llega a valer 22. La solución está en poner un límite a la varible, por ejemplo:

```
IF a=22 THEN LET a=21
```

Aunque, de nuevo, no estaría de más comprobar si hemos escrito un nombre de variable equivocado que también existe y vale 22.

6 Number too big

SIGNIFICADO: «Número demasiado grande». Indica que alguna operación aritmética ha llevado a obtener un resultado superior al máximo número con el que puede operar el Spectrum. Este número máximo es 10 elevado a 38, es decir, un «1» seguido de 38 ceros.

CAUSA: La causa más frecuente de que se obtenga este error, es cuando se intenta dividir por «0». Supongamos que tenemos una línea de tipo: LET a = b/c. Esta línea produciría el error «Number too big» en caso de que «c» llegara a valer «0». Por ello, es interesante colocar un «filtro» antes de este tipo de líneas, que las salte si el denominador vale «0». Otra posible causa es que estemos realizando una operación cuyo resultado sea superior a 1e38 (10 elevado a 38). Es difícil encontrar operaciones que den resultados tan altos, pero un ejemplo podría ser el intentar hallar la factorial de un número mayor de 33 (n! // n > 33).

SOLUCIÓN: Lo primero que hay que ver es si el error se nos ha producido en un cociente. En ese caso habrá que comprobar si el denominador vale «0» y, si es así, seguirle la pista hacia atrás para ver dónde adquiere ese valor, o colocar un «filtro» antes de la línea de la forma vista anteriormente. En caso de que el error no se haya producido en un cociente, leeremos el valor de todas las variables que entren en la expresión para ver cuál de ellas tiene un valor desmesuradamente grande o pequeño que nos provoque este resultado. En algunos casos puede que tengamos que replantear el problema, porque estemos intentando hacer algo que el Spectrum no pueda realizar.

APRENDE DE TUS ERRORES

Esta semana veremos tres errores que, si bien no se presentan con frecuencia, sí resulta útil conocerlos ya que nos indican algunas características del funcionamiento del ordenador que, habitualmente pasan desapercibidas.

Jesús ALONSO RODRÍGUEZ

7 RETURN without GOSUB

SIGNIFICADO: «RETURN sin GOSUB». Indica que el intérprete se ha encontrado un comando «RETURN», pero no sabe a dónde retornar, ya que no ha habido un «GOSUB» previo que lo enviara aquí. El hecho de que el error sólo pueda ser producido por el comando «RETURN» puede hacer pensar que su solución es fácil. Nada más alejado de la realidad. En algunos casos puede ser uno de los errores más difíciles de resolver.

CAUSA: Cada vez que el intérprete se encuentra con una sentencia «GOSUB», mete la dirección de la sentencia siguiente en una pila denominada «Pila de GOSUB» y salta a la línea que se le indique. Posteriormente, al encontrar una sentencia «RETURN», lee de la pila la dirección a donde debe retornar, que será la sentencia siguiente al «GOSUB» que le trajo a esta rutina. Como la pila de GOSUB es del tipo «Último en entrar-Primero en salir», permite la anidación de subrutinas, de forma que siemore se retornará a la sentencia siguiente a la última llamada. Pero, ¿que ocurre si se encuentra un «RETURN» si que haya habido una llamada previa con «GOSUB»? En ese caso, cuando el intérprete vaya a buscar la dirección de retorno en la pila, la encontrará vacía y no sabrá sónde retornar, por lo que detendrá la ejecución del programa e imprimirá el mensaje: «RETURN without GOSUB». La causa más frecuente es que se haya saltado con «GOTO» en medio de una subrutina. Hay que tener en cuenta que si hay varias subrutinas anidadas, el error puede no ser detectado inmediatamente, puesto que la pila de GOSUB no estará vacía; sin embargo, el error hará que la pila tenga un elemento de menos y los retornos no se harán en el orden correcto. En cualquier caso, el informe «RETURN without GOSUB» se producirá cuando el programa intente realizar el último retorno, a menos que la alteración producida en la secuencia de retornos, provoque algún otro error con anterioridad.

SOLUCIÓN: En general, se trata de un error sumamente difícil de localizar sin ayuda de un trazador (el comando «TRON» comentado en el primer capítulo y del que carece el Spectrum), por lo que lo más indicado es hacer un organigrama (diagrama de flujo) del programa y ver si la codificación que hemos realizado sigue fielmente los pasos indicados por el organigrama. Este tipo de errores suelen ocurrir en programas con una secuencia de ejecución muy «enmarañada», por lo que el mejor consejo que podemos dar al programador es que escriba sus programas con una estructura lo más clara posible y evite el uso de «miles de GOTOs» (uno de los principales «vicios» que se adquieren aprendiendo a programar en Basic) que hagan difícil seguirle la pista. Si se sigue este consejo, se facilitará mucho la depuiración de cualquier error.

8 End of file

SIGNIFICADO: «fin de fichero». Indica que se ha intentado leer más allá del final de un fichero de acceso secuencial en Microdrive, RS-232 o ZX-Net. Es decir, se han intentado leer más datos que los que contenía el fichero.

CAUSA: este error sólo puede presentarse si se tiene conectado el Interface 1, ya que es el único dispositivo de Sinclair para Spectrum que permite manejar ficheros de acceso aleatorio. Cuando se abre uno de estos ficheros y se asigna a una corriente (por ejemplo: con OPEN #5 ...) todo lo que se escriba por esa corriente (PRINT #5 ...) irá a parar al fichero. Asimismo, lo que se lea de esa corriente (INPUT #5) será leído del fichero. Si cuando se intenta abrir el fichero, éste no existe, se abrirá para escritura, en caso contrario, se abrirá para lectura. Pues bien, si con un fichero abierto para lectura, ejecutamos más INPUTs que los PRINTs que se ejecutaron al crearlo, no habrá datos que leer y el programa se detendrá con este error.

SOLUCIÓN: lo más probable es que se haya utilizado un bucle para escribir los datos en el fichero y otro para leerlos. Convendría comprobar si ambos bucles tienen el mismo número de interacciones. Por otro lado, puede ocurrir que se intente leer, por segunda vez, un fichero sin haberlo cerrado y vuelto a abrir, éste sería otro posible motivo que provocaría el error «End of file».

9 STOP Statement

SIGNIFICADO: «sentencia STOP». Indica que el intérprete ha detenido la ejecución porque se ha encontrado con una sentencia «STOP» que así se lo ordena.

CAUSA: no se trata, propiamente, de un error. Si el intérprete ha encontrado una sentencia «STOP» es porque el programador la ha puesto ahí por alguna razón. Normalmente, se pone una sentencia «STOP» durante el proceso de depuración de un programa, para detenerlo en ese punto y comprobar el estado de algunas variables. Se puede reanudar la ejecución del programa con el comando «CONTINUE» que empieza ejecutando la sentencia siguiente al «STOP».

SOLUCIÓN: cuando el programa ha puesto ahí la sentencia «STOP», será por alguna razón. La solución evidente para que el programa no se detenga es editar el listado y quitar la sentencia «STOP». Puede ocurrir, no obstante, que el programa se detenga sin que estuviera previsto, es decir, que alcance la sentencia «STOP» cuando no debiera haberla alcanzado. En ese caso, nos encontraríamos ante un «error lógico» y, como de costumbre en este tipo de errores, lo mejor es trazar un organigrama y comprobar si el programa que hemos escrito lo sigue fielmente. Siempre es más sencillo detectar los errores lógicos sobre un organigrama que sobre un listado.

APRENDE DE TUS ERRORES

Al igual que cualquier calculadora, el Spectrum tiene una serie de restricciones matemáticas. Veremos qué ocurre cuando estas restricciones no se respetan.

Jesús ALONSO RODRÍGUEZ

A Invalid argument

SIGNIFICADO: «argumento inválido». Indica que el argumento de una función no permite, por alguna razón, que ésta se ejecute.

CAUSA: normalmente, este error se produce al intentar aplicar las funciones «SQR» o «LN» sobre un argumento negativo, «ASN» o «ACS» sobre un argumento mayor de «1» o «USR» sobre una cadena cuya primera letra sea mayor de «u». También ocurrirá si se intenta elevar un número negativo a cualquier exponente. Esto último se debe a que la potenciación se realiza multiplicando el exponente por el logaritmo de la base y hallando el antilogaritmo; por tanto, la base no puede ser negativa.

SOLUCIÓN: una situación muy frecuente que da lugar a este error es intentar hallar la raíz cuadrada de un número negativo. Para evitarlo, se puede sustituir la función «SQR» por las dos funciones «SQR ABS» con lo que tendremos la seguridad de que el argumento de «SQR» no sea negativo. No obstante, tenga en cuenta que la raíz cuadrada de un número negativo está fuera del campo de los números reales. Debería tomar esto en cuenta si escribe un programa, por ejemplo, para resolver ecuaciones de segundo grado. Lo mismo vale para «LN» (aunque en este caso, el logaritmo de un número negativo no tiene sentido ni siquiera con número imaginario). En cuanto a «ASN» y «ACS», lo mejor es colocar un «filtro» que compruebe si el número sobre el que vamos a aplicar las funciones es mayor de «1». En el caso de «USR», no es fácil que se produzca el error, pero no estará de más comprobar sobre qué cadena estamos aplicando la función, no sea que nos hayamos confundido en el nombre de la variable. Para la potenciación, podemos comprobar el signo de la base, si es negativo y el exponente es impar, el resultado será también negativo, pero deberemos cambiar de signo la base antes de la potenciación y volver a cambiar de signo el resultado después de ésta. Aunque aritméticamente es correcto realizar una potenciación con base negativa, lo cierto es que ningún ordenador lo permite, de hecho, tampoco lo permite ninguna calculadora. La razón es que, tanto en un ordenador como en una calculadora, la exponenciación no se hace multiplicando la base por sí misma tantas veces como indique el exponente; éste es el método que empleamos para hacerlo «a mano», pero para una máquina no resulta rentable. Lo que hace el ordenador es hallar el logaritmo neperiano de la base (mediante un desarrollo en serie), multiplicarlo por el exponente y hallar el antilogaritmo del resultado. Dado que es imposible hallar el logaritmo de un número negativo, el ordenador no permite operaciones de exponenciación en las que la base sea negativa. Por otro lado, nada se opone a que el exponente sea negativo, ya que el ordenador no tendrá problema en hallar su anti-logaritmo.

B Integer out of range

SIGNIFICADO: «entero fuera de rango». Significa que un número requerido como parámetro de una instrucción se encuentra fuera del rango que admite ese parámetro.

CAUSA: existe un gran número de instrucciones que requieren un parámetro numérico entero (en caso de que el parámetro dado por el usuario no sea entero, se redondea al entero más próximo). En muchas de ellas, el número debe estar dentro de un cierto margen. Si el parámetro dado por el usuario rebasara este margen, el intérprete se detendría con el error «Integer out of range». Veámoslo con un ejemplo: El comando «POKE» requiere dos parámetros, el primero de ellos es la dirección de memoria donde se va a «POKEar» el segundo. Esta dirección tiene que estar, necesariamente, comprendida entre «0» y «65535», ya que no existen direcciones de memoria fuera de este rango.

Una causa frecuente de este error es dar unas coordenadas de pantalla incorrectas. El error se producirá con un número de columna superior a 31 o con un número de línea superior a 22. Como ya vimos en un capítulo anterior, la impresión en la línea 22 no produce este error, sino el «Out of screen», salvo que, por programa, se hubiera ampliado la parte superior a 23 líneas.

SOLUCIÓN: si el error se ha detectado en una sentencia del tipo: «PRINT AT...», es muy probable que exista una variable que controle la posición de impresión, y cuyo margen haya resultado excedido. Cuando el usuario mueve algún elemento por la pantalla usando el teclado (o un joystick), se deben poner límites a las variables que actúen como coordenadas. Puede ocurrir que el límite no se haya puesto, que se haya puesto de forma incorrecta, o que un error lógico haya traído al intérprete hasta aquí, saltándose el límite. Todos éstos serán extremos a comprobar para resolver un error de este tipo.

En cualquier otro caso, lo primero que hay que comprobar es que la variable que actúa como parámetro esté correctamente escrita. Si es así, o si el parámetro no es una variable, sino un número, habrá que comprobar cuál es su valor y cuál el rango permitido. A continuación, se seguirá la pista, hacia atrás, de la variable para ver dónde adquiere el valor que la coloca fuera de rango. Una solución rápida (aunque bastante «chapucera») es colocar un «filtro» antes del comando en cuestión, de forma que la variable que da problemas nunca llegue a tener un valor fuera de rango. Veámoslo con un ejemplo: supongamos que la variable se llama «A» y el máximo valor permitido es «B», el filtro podría ser:

```
IF A > B THEN LET A = B
```

Este sistema puede dar resultado en algunos casos, aunque lo mejor es encontrar la causa del error en vez de poner «parches».

APRENDE DE TUS ERRORES

Hace algunos meses, nos escribía una joven lectora preguntando si «Nonsense in Basic» quería decir «No digas tonterías en Basic». En cierto sentido no andaba descaminada, ya que éste es el informe que obtendremos si le pedimos al intérprete que haga lo que, para él, representa «una tontería».

Jesús ALONSO RODRÍGUEZ

C Nonsense in Basic

SIGNIFICADO: «Sin sentido en Basic». Indica que el intérprete se ha encontrado con alguna instrucción que no puede interpretar. La construcción puede ser una expresión inevaluable o una sentencia inejecutable y que se encuentre ahí a pesar del análisis sintáctico del editor.

CAUSA: La causa más frecuente es el haber aplicado la función «VAL» sobre una cadena cuyo contenido no es susceptible de evaluacion matemática. Hay que tener en cuenta, no obstante, que si el contenido de la cadena representa un posible nombre válido de variable, el intérprete lo tomará así, y emitirá el informe «Variable not found» que puede provocar confusión en el usuario.

Otra posible causa de este error es que el programa haya resultado corrompido, por ejemplo, por haber «POKEado» en una dirección que se encuentre en medio del listado. En general, el intérprete emite este informe cuando, por cualquier causa, no entiende el listado.

SOLUCIÓN: Este error presenta una particularidad que lo hace difícil de corregir. El informe se presenta como una respuesta al comando directo «RUN», «GOTO» o «LOAD» que puso en marcha el programa, por tanto, el número de línea será siempre «0» y nos resultará bastante difícil saber en qué lugar del programa se ha detectado.

Lo primero es comprobar si el programa ha sido corrompido, para lo cual habrá que revisarlo detenidamente. Si no es este el caso, habrá que buscar todas las sentencias «VAL», ya que una de ellas tiene que ser la responsable del error. Antes de cada una, colocaremos un «STOP» que nos detenga la ejecución de forma temporal, para poder imprimir el contenido de la cadena sobre la que se va a aplicar «VAL» y ver si es evaluable antes de teclear «CONTINÚE» para seguir ejecutando (estas detenciones temporales se conocen como «break-points» y son muy útiles en la depuración de errores lógicos). Una vez encontrada cuál es la cadena responsable del error, no hay más que seguirle la pista hacia atrás para ver en qué punto adquiere un valor que la haga inevaluable.

D BREAK-CONT repeats

SIGNIFICADO: «BREAK-CONT repite.» Indica que se ha pulsado la tecla «BREAK» durante la ejecución de un comando. Normalmente, la tecla «BREAK» es muestreada después de ejecutar cada comando, y si está pulsada, se detiene la ejecución con el mensaje «BREAK into program». No obstante, hay algunos comandos que pueden dejar «colgado» al ordenador esperando la respuesta de un periférico. Estos comandos son: LOAD, SAVE, VERIFY, MERGE, LPRINT, LLIST y COPY. En este caso, la tecla «BREAK» se muestrea continuamente durante la ejecución del comando y, si se pulsa, se detiene con este mensaje. También ocurre si se responde que «NO» a la pregunta «Scroll?».

Normalmente, cuando se produce un «BREAK», si luego se pulsa «CONTINÚE», la ejecución se reanuda en el comando siguiente a aquel que se había terminado de ejecutar cuando se detectó el BREAK. Sin embargo, en el caso de este informe, si se pulsa «CONTINÚE», se repite el comando que se estaba ejecutando al pulsar «BREAK». Este es el significado del mensaje «CONT repeats».

CAUSA: En este caso, la causa no es otra que la intervención del usuario.

SOLUCIÓN: Si se desea reanudar la ejecución, bastará dar el comando «CONTINÚE» directamente. La sentencia que se estaba ejecutando al pulsar «BREAK» volverá a repetirse.

E Out of DATA

SIGNIFICADO: «Se acabaron los DATAs». Indica que una sentencia READ ha intentado leer más allá de las sentencias DATA disponibles. En un programa, no se pueden ejecutar más sentencias READ que el número de datos que existan en sentencias DATA. A menos que se haga un RESTORE que vuelva a colocar el puntero de datos al principio.

Podemos imaginar la lista de datos en sentencias DATA como un fichero secuencial interno al programa. Por tanto, este informe viene a indicarnos que se ha rebasado el final del fichero.

CAUSA: La causa más frecuente de este error es haber olvidado incluir una sentencia RESTORE antes de volver a leer, por segunda vez, una lista de datos ya leída con anterioridad.

Otra posible causa es haber omitido una coma de separación entre números dentro de alguna sentencia DATA, con lo que dos de los datos han sido tomados como uno sólo y, lógicamente, al final faltará un dato.

Por último, puede ocurrir que, en un despiste, hayamos puesto menos datos que los que luego se van a leer, o que estemos ejecutando más sentencias READ de las debidas.

SOLUCIÓN: Si existe una lista de datos que deba ser leída más de una vez, el primer paso es revisar el listado para asegurarse de que se hace «RESTORE» antes de empezar a leerla de cada vez. Tampoco está de más revisar todas las sentencias DATA por si se ha omitido alguna coma y, de paso, contar el número de datos que tenemos para ver si es el correcto.

Si todo lo anterior falla, se puede seguir «a ojo» la ejecución sobre el listado y contar cuántas veces se ejecuta una sentencia READ. Puede ocurrir que la sentencia se encuentre dentro de un bucle que se ejecute más veces de las previstas, por ejemplo, porque exista un error en el límite de la variable de control del mismo.

Por último, no hay que descartar la posibilidad de que exista un error lógico en el flujo del programa que lo dirija a una sentencia READ cuando todos los datos hayan sido leídos.

Si no se puede prefijar la longitud de la lista de datos (supongamos una aplicación en la que los DATAs se «MERGEan» con el programa principal), es posible establecer un indicador de «fin de lista». Supongamos, por ejemplo, que tenemos que leer una lista, de datos numéricos, de longitud variable y ninguno de los cuales es 0. Podemos colocar un 0 al final de la lista, y hacer la lectura de la siguiente forma:

READ a
IF NOT a THEN GOTO (salir del bucle)

Con lo que nunca se nos producirá el error **«Out of DATA»**. (La sentencia «IF NOT a THEN...» es equivalente a «IF a=0 THEN...», pero ocupa menos memoria y es más rápida.)

APRENDE DE TUS ERRORES

Esta semana veremos algunos errores que, a pesar de no presentarse con frecuencia, resultan sencillos de corregir.

Jesús ALONSO RODRÍGUEZ

F Invalid file name

SIGNIFICADO: «Nombre de fichero no válido». Indica que se ha especificado un nombre de fichero no válido en un comando «SAVE» dirigido al cassette. La razón suele ser por tener más de 10 caracteres o ser una cadena vacía.

CAUSA: Con este mensaje de error, nos encontramos ante una de las imprecisiones del Sistema Operativo. En la configuración básica, todo funciona correctamente y obtendremos este mensaje siempre que hagamos un SAVE con un nombre de fichero no válido. Sin embargo, las cosas dejan de funcionar bien si conectamos el Interface-1. En este caso, las situaciones que antes nos producían un «Invalid file name», ahora nos producirán un «Nonsense in Basic», es decir, siempre que hagamos un SAVE dirigido al cassette y con nombre de fichero no válido.

En el caso de un SAVE o un LOAD dirigido al microdrive, ZX-NET, etc. con un nombre de fichero vacío o de más de 10 caracteres, el informe que se obtendrá será: «Invalid name», que es un informe propio del Interface 1.

En conclusión, quien utilice el Spectrum con un Interface 1 conectado, jamás obtendrá este informe. En su lugar, obtendrá un «Nonsense in Basic» aunque, esta vez, con número de línea y sentencia... ¡todo un detalle! En cambio, quien utilice la configuración básica, obtendrá este mensaje siempre que intente hacer SAVE con un nombre de fichero incorrecto.

SOLUCIÓN: Dado que este error sólo se puede producir en un comando SAVE, su solución es bien sencilla. Basta con buscar el parámetro de SAVE que ha producido el error (normalmente será una variable de cadena) y seguirle la pista hacia atrás hasta descubrir dónde y por qué adquiere un contenido no válido.

G No room for line

SIGNIFICADO: «No hay sitio para la línea». Indica que la memoria asignada al Basic está llena y no hay sitio para almacenar la línea que se intenta introducir.

CAUSA: Este informe se produce en el momento de pulsar «ENTER» para introducir una línea. Hay dos posibles causas: una, que la RAMTOP esté demasiado baja; otra, que el programa Basic sea demasiado largo.

No siempre que nos quedemos sin memoria se presentará este mensaje. Sólo en el caso de que la memoria que quede sea suficiente para construir la línea en el área de edición, pero insuficiente para copiarla en el área de programa. Sin embargo, puede ocurrir que la memoria que queda no dé ni para construir la línea en el área de edición, en cuyo caso, el ordenador emitirá el famoso pitido de alarma.

SOLUCIÓN: Si el informe se presenta por tener la RAMTOP demasiado baja, lo más fácil es subirla, almacenando menos datos por encima de RAMTOP. Por el contrario, si el informe se debe a que el programa es demasiado largo, no habrá más remedio que recurrir a alguna de las técnicas de ahorro de memoria; por ejemplo, cambiar los «unos» por SGN PI, los «ceros» por NOT PI, los «treses» por INT PI, y así sucesivamente. Otra posibilidad es replantear el programa utilizando más subrutinas y evitando la codificación repetitiva de procesos similares. No obstante, hay que tener mucho cuidado con los programas que ocupen la memoria casi por completo, ya que son muy propensos a producir informes del tipo «Out of memory».

H Stop in INPUT

SIGNIFICADO: «STOP en un INPUT». Indica que se ha encontrado un código de «STOP» en la entrada procedente de una sentencia INPUT.

CAUSA: En el caso de un INPUT numérico, se producirá el informe siempre que se haya tecleado el comando STOP como parte de la entrada. Sin embargo, en un INPUT de cadena, sólo se producirá el error si el STOP está fuera de las comillas, ya que si está dentro, se tomará como un carácter más de la cadena. En un INPUT LINE no hay comillas y el STOP nunca puede estar fuera, no obstante, se producirá el error si se teclea el código 10, es decir, «cursor abajo».

SOLUCIÓN: Dado que es un error provocado intencionadamente por el usuario, no tiene sentido hablar de solución en este caso. No obstante, podemos profundizar un poco en el significado de todo esto. El hecho de que un programa pueda ser detenido tecleando «STOP» en un INPUT, hace más difícil su protección; veamos por qué:

Una técnica utilizada con mucha frecuencia para proteger un programa en Basic es cerrar la parte inferior de la pantalla con un POKE 23659,0 de forma que si el usuario hace BREAK, el ordenador se «cuelga» al no tener sitio donde imprimir el informe. Ahora bien, el INPUT va por la parte inferior, así que no hay más remedio que abrirla con POKE 23659,2 antes del INPUT y volverla a cerrar después. Por tanto, el programa queda desprotegido durante toda la ejecución del INPUT y expuesto a ser detenido con un STOP.

I FOR without NEXT

SIGNIFICADO: «FOR sin NEXT». Indica que se ha encontrado una sentencia FOR cuyos parámetros hacen que no haya que ejecutarla ninguna vez (por ejemplo, FOR n = 1 TO 0), y el intérprete no encuentra la correspondiente sentencia NEXT para saltar a ella.

CAUSA: Ya el hecho de que se encuentre un bucle FOR, que no ha de ejecutarse ninguna vez, resulta sospechoso, pero el hecho de que, además, no exista su correspondiente NEXT lo convierte en un error muy infrecuente. En realidad, sólo podría producirse como resultado de un error lógico que trastocara totalmente el orden de ejecución del programa o porque el programador haya equivocado el nombre de la variable de control del bucle. Esta última será, sin duda, la causa más frecuente de este tipo de error.

SOLUCIÓN: Aparte del posible error lógico, que no estaría de más buscar con la ayuda de un organigrama, conviene comprobar la variable de control del bucle que figura en la sentencia FOR y la que figura en la sentencia NEXT, ya que lo más posible es que no sean iguales y por ello se haya producido el error. En este caso, ya podremos alegrarnos de que el límite establecido haya llevado a un número cero de iteraciones, ya que de lo contrario, el error hubiera sido muy difícil de detectar.

No es frecuente definir bucles FOR - NEXT que no hayan de ejecutarse ninguna vez, por tanto, la aparición de este mensaje suele implicar la existencia de dos errores, uno en la variable de control y otro en el límite de ésta.

APRENDE DE TUS ERRORES

Esta semana pasaremos revista a una serie de informes de error que poco tienen que ver entre sí. También veremos uno de los pocos errores que se pueden producir mientras listamos un programa.

Jesús ALONSO RODRÍGUEZ

J Invalid I/O device

SIGNIFICADO: «Dispositivo de entrada/salida no válido». Indica que, para una operación de entrada o de salida, se ha especificado un número de corriente que tiene asignado un canal que no permite tal operación. Por ejemplo, intentar una entrada por un canal de salida o viceversa.

CAUSA: Este error es más frecuente que se presente si se está utilizando el Interface-1, no obstante, también es posible obtenerlo en la configuración básica. Por ejemplo, un INPUT #2 produciría este error, ya que la corriente #2 está asignada al canal «S» que sirve para salida pero no para entrada (a menos que, con un OPEN, se hubiera reasignado a otro canal).

La causa más probable de presentación de este error es que se haya omitido reasignar una corriente que previamente estaba asignada a un canal que no permitía realizar la operación que ha producido el error. Puede deberse, también, a un error de concepción del programa o a un error lógico que provoque el que una sentencia se alcance antes de lo debido.

SOLUCIÓN: Lo mejor es empezar por preguntarse qué se está pretendiendo hacer y si el Sistema Operativo lo permite. Por ejemplo, si intentamos hacer un INPUT en la parte alta de la pantalla, obtendremos este informe, ya que la parte alta corresponde al canal «S» que es sólo de salida.

Si lo que pretendemos hacer es «legal», lo siguiente es preguntarnos qué corrientes necesitamos tener abiertas y a qué canales deberán estar asignadas. A continuación sólo resta seguir el flujo del programa hacia atrás para ver si, efectivamente, tenemos las corrientes asignadas a los canales correctos (quien no disponga de Interface-1 es probable que no tenga muy claro el asunto de las corrientes y los canales, no se preocupe, es muy improbable que obtenga este informe de error).

K Invalid colour

SIGNIFICADO: «Color no válido». Indica que una cadena que se iba a imprimir en pantalla contenía un especificador de color seguido de un parámetro no válido. Por ejemplo, BRIGHT 3 o PAPER 15.

CAUSA: Este error se puede producir tanto en tiempo de ejecución, como al listar un programa. Empecemos por el primer caso. Si el error se ha producido mientras se estaba ejecutando el programa, indicará que alguno de los parámetros de color que se especificaban en la sentencia correspondiente no era válido. Sin embargo, con frecuencia se produce este error mientras se está listando un programa. En este caso existen dos posibles causas. Primera: que el listado se haya corrompido y contenga caracteres sin sentido, que son interpretados como controles erróneos de color. Segunda: que exista una línea REM conteniendo datos o Código Máquina que, a la hora de imprimir, son interpretados como controles de color.

SOLUCIÓN: Si el error se ha producido durante la ejecución, revisar la sentencia donde se ha detectado (la que indica el informe) y comprobar el valor de los parámetros que asignan los controles de color. Por ejemplo: supongamos que el informe obtenido es:

```
K Invalid colour, 1230,1
```

Listamos la línea 1230 y vemos que es:

```
1230 PRINT PAPER P;
INK t; a$
```

Deberemos comprobar los valores de las variables «p» y «t» que actúan como parámetros de los controles de color PAPER e INK para comprobar que sus contenidos correspondan a colores válidos. Tampoco estaría de más comprobar el contenido de «a$» no sea que el error venga de ahí (hay que tener en cuenta que el error puede ser producido por cualquier cosa que se envíe a la pantalla).

Por el contrario, si el error se ha producido mientras se estaba listando un programa, habrá que revisar qué contiene la última línea que aparezca en el listado, ya que ésta será la que contenga los datos que han provocado el error. Si se tratase de una línea REM conteniendo Código Máquina, lo mejor será ingnorarla y continuar el listado a partir de la línea siguiente.

L BREAK into program

SIGNIFICADO: «BREAK en el programa». Indica que se ha pulsado la tecla BREAK mientras se estaba ejecutando el programa.

CAUSA: Como es lógico, la causa no puede ser otra que la intervención del usuario. No hay que olvidar que este error se produce tanto si se pulsa la tecla «BREAK» del Plus, como si se pulsa «CAPS SHIFT» + «SPACE».

SOLUCIÓN: Se puede continuar la ejecución del programa con el comando CONTINUE. La ejecución se reanudará a partir de la sentencia siguiente a la indicada en el informe, a diferencia de lo que ocurre con el informe «BREAK CONT repeats» que repite la sentencia en curso.

M RAMTOP no good

SIGNIFICADO: «RAMTOP incorrecto». Indica que un comando CLEAR ha especificado un valor para RAMTOP que no es viable, ya que si se bajase a la dirección indicada, no dejaría suficiente espacio para el Basic.

CAUSA: El error puede ser producido tanto por un valor de RAMTOP demasiado bajo, como por uno superior al de P-RAMT. El segundo caso podría ser síntoma de una avería de memoria, aunque no hay que olvidar que algunos interfaces (por ejemplo, el Centronics de Indescomp/MHT) bajan la P-RAMT. En el primer caso, el número especificado después del CLEAR cae dentro de la zona actualmente ocupada por el Basic; el CLEAR se ejecuta, pero la RAMTOP se deja como estaba.

SOLUCIÓN: Evidentemente, la solución consiste en especificar un valor diferente de RAMTOP, aunque habrá que tener cuidado, ya que si el valor fuera próximo, se podría obtener pronto un «Out of memory».

APRENDE DE TUS ERRORES

Esta semana terminaremos de ver los informes de error en tiempo de proceso, que se pueden obtener en la configuración básica.

Jesús ALONSO RODRÍGUEZ

N Statement Lost

SIGNIFICADO: «sentencia perdida». Se produce este error cuando un comando RETURN, NEXT o CONTINUE intenta saltar a una sentencia que ya no existe.

CAUSA: no es frecuente que se produzca este error, pero el S.O. lo tiene previsto para evitar un «cuelgue» al procesar alguno de estos comandos. Para que se produzca este error es necesario que la sentencia existiera previamente y luego dejar de existir. Es una situación que, prácticamente, sólo puede darse si se ha detenido el programa y se ha vuelto a reanudar tras borrar alguna línea.

SOLUCIÓN: la solución, en este extraño caso, es ver a qué sentencia se está intentando saltar y por qué no existe.

O Invalid Stream

SIGNIFICADO: «corriente no válida». Indica que se ha intentado hacer una entrada o una salida a través de una corriente que no ha sido abierta y que, por tanto, no tiene ningún canal asignado.

CAUSA: este informe se producirá en sentencias PRINT, INPUT, INKEY$, LIST, LPRINT y LLIST que se dirijan a corrientes no abiertas. En la configuración básica, suelen utilizarse las corrientes #1, #2 y #3 que son las que mantiene abiertas el Sistema Operativo, y a las que se dirigen, por defecto, los comandos INPUT, PRINT y LPRINT respectivamente. A lo sumo, se hará algún PRINT #1; para escribir en la parte inferior de la pantalla, pero no se utilizan otras corrientes, por lo que la aparición de este informe indicaría un error en la propia sentencia donde se ha detectado que ha intentado acceder a una corriente que no es ninguna de las permitidas. La cosa cambia si se utiliza el Interface-1. Este dispositivo permite abrir otras corrientes y asignarlas a otros canales, por lo que la aparición de este informe podría indicar que nos hemos olvidado de abrir una corriente en la que luego hemos intentado imprimir o leer datos.

SOLUCIÓN: si estamos trabajando con la configuración básica, el error tiene que estar en la sentencia que indica el informe, así que bastará con revisarla para ver a qué corriente nos estamos intentando dirigir. Por el contrario, si tenemos el Interface-1 conectado, convendrá comprobar también si la corriente ha sido abierta previamente, para lo cual no hay más remedio que seguir hacia atrás el flujo del programa. Por último, cabe la posibilidad de que estemos corriendo, en un Spectrum sin Interface-1, un programa preparado para correr con este interface conectado. Aunque, en este caso, lo más probable es que obtengamos un «Nonsense in Basic», mucho antes.

P FN without DEF

SIGNIFICADO: «FN sin DEF». Indica que se ha intentado utilizar una función definible por el usuario que no ha sido previamente definida.

CAUSA: el informe puede ser debido tanto a un error en el nombre de la función al invocarla, como a un error al definirla o, simplemente, a que nos hemos olvidado de definir la función en algún lugar del programa.

SOLUCIÓN: una inspección de la línea donde se ha detectado el error (la indicada en el informe), nos revelará si nos hemos equivocado en el nombre de la función. Caso contrario, será que no la hemos definido. La definición de una función puede estar en cualquier lugar del programa, ya que el intérprete la buscará por todo él. No obstante, esta búsqueda se realiza empezando por el principio del programa y hacia el final, así que ahorraremos bastante tiempo si colocamos todas las definiciones de funciones al principio del programa. Otro tanto se puede decir, por cierto, de las subrutinas que se usen con más frecuencia.

Q Parameter error

SIGNIFICADO: «error de parámetros». Indica que, al intentar invocar una función definida por el usuario, hemos cometido algún error al especificar los argumentos de ésta.

CAUSA: el error tiene que consistir o bien en que hemos especificado un número de argumentos que no se corresponde con los que especificamos al definir la función, o bien en que hemos puesto una variable de cadena donde debería ir una numérica o viceversa. De nuevo, el error puede estar en la línea donde se invoca la función (la que aparece en el informe) o en la línea donde se define.

SOLUCIÓN: podemos empezar por comprobar la sentencia que indica el informe para ver si hemos invocado la función correctamente. Si el error no estuviera aquí, habrá que revisar la línea donde se define la función, ya que el error puede venir arrastrado desde allí.

R Tape loading error

SIGNIFICADO: «error de carga desde cinta». Indica que durante la carga de un bloque de datos (programas, bytes, etc.), se ha producido algún tipo de error o que, durante la verificación, alguno de los datos almacenados en cinta es distinto del que le corresponde en la memoria del ordenador.

CAUSA: hay varias circunstancias que puden dar lugar a este error:

1. Si se interrumpe el envío de datos desde el cassette (por ejemplo, porque se detenga éste).

2. Si se produce una alteración significativa en la frecuencia o intensidad de la señal recibida (cabeza de cassette mal alineada o sucia, cinta de baja calidad, grabación o reproducción realizadas a bajo volumen, etc.).

3. Si alguno de los datos recibidos es distinto del que le corresponde en la memoria (sólo en verificación).

4. Si el byte final de comprobación que se ha recibido no concuerda con el que ha calcu-

lado el ordenador (el byte de comprobación es el resultado de realizar una operación XOR entre todos los bytes que le preceden en el bloque).

En cualquiera de estos casos, la aparición del informe de error nos indicará que la operación de cassette no se ha realizado con éxito.

SOLUCIÓN: si el error se ha producido durante una verificación, lo mejor es volver a salvar el bloque correspondiente y volverlo a verificar. Si se ha producido durante la carga, no hay más remedio que volver a empezar de nuevo. Dada la poca fiabilidad del sistema de almacenamiento en cassette, es conveniente verificar cada bloque que se salve y guardar, al menos, dos copias de cualquier cosa importante, ya sean programas o datos.

Ante una presentación frecuente de este tipo de errores, conviene revisar el cassette. Los pasos a seguir son los siguientes:

1. Asegurarse de que el cassette está funcionando a la tensión correcta (si está trabajando con pilas, asegurarse de que éstas no están gastadas).

2. Limpiar los cabezales, el capstán (eje de arrastre) y el rodillo presor con un algodón empapado en alcohol iso-propílico (en su defecto, se puede emplear alcohol etílico, aunque tiene el inconveniente de provocar oxidación).

3. Alinear la cabeza reproductora utilizando un pequeño destornillador y una cassette donde se haya grabado una señal de frecuencia media. El ajuste se hace girando uno de los dos tornillos que sujetan la cabeza (el que tiene un muelle debajo) mientras se escucha la señal grabada. El punto de ajuste óptimo será aquel donde la señal se escuche con una mayor claridad. En el comercio venden cintas especiales para ajustar la cabeza del cassette sirviéndose de un programa de ordenador. Este ajuste es más crítico en los aparatos estereofónicos, por lo que no recomendamos su empleo para almacenar datos de ordenador. En su lugar, dan muy buenos resultados los dictáfonos tipo «periodista».

4. En ocasiones, puede ser necesario reajustar la velocidad del cassette actuando sobre una resistencia ajustable que suele estar ubicada en la placa de circuito impreso encargada de regular la velocidad del motor. Este ajuste se puede hacer «a oído» utilizando una cinta que contenga algún tema musical con el que estemos familiarizados. La precisión de este sistema suele ser suficiente para las necesidades del ordenador; no obstante, quien desee afinar más este ajuste, puede utilizar como «estroboscopio» las bandas producidas en la pantalla por el tono guía que precede a cada bloque. Con una velocidad correcta, las bandas deben desplazarse lentamente hacia arriba. Por supuesto, la cinta que se utilice deberá haber sido grabada en un cassette cuya velocidad sea correcta.

Si no se dispone de otro cassette para referencia, o se duda de su velocidad, se puede seguir el siguiente procedimiento: en primer lugar, grabar en la cinta un tono de aproximadamente 1.000 Hz y de una duración de un minuto. A continuación, extraer un trozo de cinta de unos 60 cm de la parte donde se ha grabado el tono. Acercar un imán a la cinta en dos lugares que disten entre sí 47,5 cm; esto creará dos silencios separados 10 segundos entre sí. Por último, reproducir este trozo de cinta y ajustar la velocidad hasta que la separación entre silencios sea de exactamente 10 segundos.

APRENDE DE TUS ERRORES

En este capítulo, empezaremos a ver los informes de error que pueden producirse cuando se tiene el Interface-1 conectado.

Jesús ALONSO RODRÍGUEZ

Los informes del Interface-1

Cuando se conecta un Interface-1 a un Spectrum, se añaden una serie de comandos nuevos y algunas variaciones en la sintaxis de los ya existentes. Por ello, se pueden producir una serie de informes nuevos que indicarán errores en la ejecución de estos comandos.

Los informes del Interface-1 corresponden a rutinas de la ROM que incorpora este dispositivo, habitualmente denominada: «Shadow ROM» (ROM Sombra) dado que se pagina, como una sombra, sobre las 8.192 primeras direcciones de la ROM del ordenador, es decir, sobre los 8 primeros Ks.

Estos informes carecen de código, es decir, no se nos presentan precedidos por una letra o un número como ocurría en la configuración básica. Sin embargo, seguirán presentándose seguidos del número de línea y sentencia donde el error haya sido detectado.

Empezaremos por ver dos informes que no figuran en el manual y, a continuación, iremos viendo los restantes uno a uno como hemos venido haciendo hasta ahora.

Program Finished

SIGNIFICADO: «Programa terminado». Indica que se ha intentado ejecutar más allá del programa existente en memoria.

CAUSA: este informe sustituye, en algunos casos, al «0 OK» de la versión básica y contribuye a dar más coherencia al funcionamiento del ordenador. Si se ejecuta un comando directo cuya ejecución termine con éxito, aunque lleve envuelta la ejecución de un programa, obtendremos el informe «0 OK» como era de esperar. No obstante, si intentamos saltar a un número de línea superior al más alto existente, obtendremos el informe «Program Finished» que nos dará más pistas sobre la situación real que el «0 OK» que obtendríamos en la configuración básica.

Hay otra circunstancia en la que también obtendremos este informe, aunque requiere un estudio algo más detallado. Casi todos los usuarios de Interface-1 saben que, si se teclea el comando RUN inmediatamente después de encender el ordenador o de hacer NEW, el Microdrive se pondrá en marcha para buscar un fichero que se denomine «run» y que contenga un programa en Basic que será cargado y ejecutado automáticamente. Esto permite una forma de auto-ejecución de la que están dotados casi todos los ordenadores, pero que faltaba en el Spectrum. Ahora bien, esto ocurre al teclear RUN nada más encender el ordenador; sin embargo, si se ha hecho alguna operación previa con él y no hay ningún programa en memoria, se obtendrá «Program Finished». Para entender por qué ocurre esto, es necesario echar un vistazo al modo en que funciona el Interface-1.

En el momento de encender el ordenador, la inicialización se lleva a cabo como si el Interface-1 no estuviera conectado, de hecho, el ordenador ignora este dispositivo hasta el momento mismo en que lo utiliza por primera vez. Cada vez que se hace una llamada al Interface-1, éste comprueba si sus variables propias han sido insertadas previamente en memoria; si no es así, las inserta, alterando el mapa de memoria original del ordenador. Por tanto, estas variables se insertarán la primera vez que se haga una llamada al Interface-1 y permanecerán insertadas desde ese momento en adelante. Pero, ¿cuándo se hace una llamada a este dispositivo? En primer lugar, cada vez que se ejecuta uno de los nuevos comandos tales como CAT, ERASE, etc., pero también —y esto es muy importante— cada vez que se imprime un informe de error, cualquiera que éste sea, incluido el «0 OK». La razón de que se haga así, la veremos al estudiar el siguiente informe, de momento, nos basta con tener en cuenta que casi cualquier cosa que hagamos con el ordenador después de encenderlo o hacer NEW, hará que se inserten las variables del Interface-1.

Cuando se teclea el comando RUN sin que haya ningún programa en memoria, el Interface-1 comprueba si sus variables han sido ya insertadas y, de ser así, emite el informe «Program Finished». Esta es la segunda circunstancia a la que hacíamos referencia antes, que provoca la aparición de este informe.

SOLUCIÓN: si el informe se presenta con número de línea «0», indicará que se ha hecho un RUN sin ningún programa en memoria pero habiendo ejecutado alguna operación anteriormente. En este caso, la solución es hacer un RESET o un NEW y volver a teclear el comando. Esta vez, el Microdrive sí funcionará.

Por el contrario, si el informe se obtuviera con un número de línea determinado, indicará que se ha hecho un GOTO a un número de línea superior a todos los existentes. En este caso, lo mejor es revisar la línea que se indica en el informe para ver a qué línea se está pretendiendo saltar y por qué ésta no existe. Hay que tener en cuenta que si la ejecución llevara a un punto a partir del cual no hubiera más sentencias (no con un GOTO, sino siguiendo el flujo normal del programa), lo que se obtendría sería un «0 OK» en respuesta al comando RUN que lanzó la ejecución, pero con indicación del número de la última línea ejecutada.

Hook code error

SIGNIFICADO: «Código de enganche erróneo». Indica que el Interface-1 ha recibido un código de enganche para el cual no tiene operación asignada. Este error se producirá con códigos comprendidos entre 51 y 254, ambos inclusive.

CAUSA: para comprender este informe es necesario saber, primero, qué es un «código de enganche», así que..., vamos a ello:

Cuando se diseñó el Spectrum, se preveía la posibilidad de dotarlo, posteriormente, con un dispositivo que permitiese el uso de algún sistema de almacenamiento masivo y ciertas posibilidades de comunicación. Ese dispositivo vendría a ser el Interface-1, pero entonces, nadie sabía en qué iba a consistir ni cómo iba a funcionar, así que se incluyeron los comandos para utilizarlo en la ROM del ordenador, pero no se

previó ninguna rutina que ejecutara estos comandos. Si se intentan utilizar sin tener conectado el interface, se producirá un error de sintaxis que no dejará subir la línea. Si se ejecuta un programa cargado desde cinta que los contenga, se producirá el error «Nonsense in Basic».

Con el Spectrum ya en la calle, se abordó el diseño del Interface-1. Tenía que ser compatible con la ROM ya existente (no se podía pedir al usuario que cambiara la ROM) y funcionar con los comandos que ésta ya tenía. Pero estos comandos provocaban errores de sintaxis, así que se optó por una ingeniosa solución: cada vez que se produjera un error en el funcionamiento de la ROM principal, entraría a funcionar la «Shadow ROM» para gestionar el error «a su manera». Si se tratara de un error normal, el control sería devuelto a la ROM principal para que lo gestionase en la forma acostumbrada. Ahora bien, si el error era producido por el uso de algùno de los comandos del Interface-1, entonces sería este dispositivo el que asumiría el control para proceder a la ejecución (o análisis sintáctico en tiempo de edición) del comando.

Por otro lado, se pretendía que la ROM no ocupase más de 8 Ks, por lo que había que evitar la repetición de rutinas que ya estuviesen en la ROM principal (aunque éstas fuesen necesarias para la ejecución de los nuevos comandos). El resultado fue que, cada vez que se ejecuta uno de estos comandos, el control es pasado alternativamente a la ROM principal y a la «Shadow ROM». Fue necesario, por tanto, preveer alguna forma de comunicación entre las rutinas de los dos dispositivos. Por otro lado, había que permitir que los programas escritos en Código Máquina pudieran tener acceso a las rutinas de la «Shadow ROM». Para ello, se inventaron los «códigos de enganche».

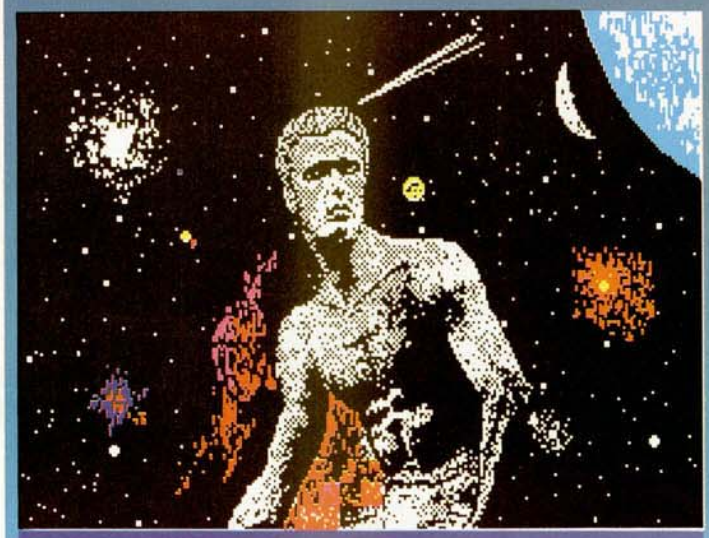
Cada vez que se produce un error en tiempo de proceso, el Sistema Operativo salta a la dirección 0008 de la ROM principal, con un código colocado en la dirección siguiente a aquella que provoca el salto. Este código indica a la rutina de gestión de errores cuál es el informe que ha de presentar en pantalla. Para presentar el informe «0 OK», el código es «255», para el «1 NEX without FOR», el código es «0», para el «2 Variable not found» el código es «1»; y así sucesivamente, el código es siempre uno menos que el presentado en el informe (en Código Máquina, 0 menos 1 es igual a 255); por tanto, el código para presentar el informe «R Tape loading error» será «27». Si se presentara un código superior a «27» e inferior a «52», el Interface-1 entendería que se trata de un «código de enganche» que le indica que realice alguna operación; así, por ejemplo, el código «34» significa «abrir un fichero», el «33» significa «poner en marcha el motor de un Microdrive», así para cada uno. Los códigos superiores a «51» no tienen operación asignada y provocan, por tanto, la emisión del informe «Hook code error».

Ésta es la única causa que puede producir este error, por tanto, es imposible que se produzca durante la ejecución de un programa que sólo trabaje en Basic. Para que el programa produjera este error, tendría que tener, necesariamente, alguna llamada a una rutina en Código Máquina.

SOLUCIÓN: Dado que el error sólo puede ser producido por una rutina en C/M, habrá que ver qué rutina se estaba ejecutando cuando se produjo y revisarla para comprobar en qué punto se hace una llamada a la dirección 0008 con un código incorrecto. Desgraciadamente, para depurar un programa en Código Máquina no contamos con la ayuda del Basic, por lo que la tarea será considerablemente más difícil.

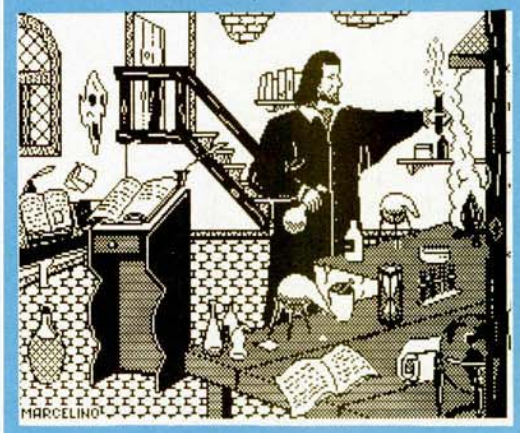
PIXEL A PIXEL

Este continúa siendo el rincón reservado para mostraros semanalmente los trabajos que quedaron clasificados entre los 100 primeros puestos de nuestro 1.er Concurso de «Diseño gráfico por ordenador».



Emilio Rueda de la Puerta (Málaga). N.º 25. 33 puntos.

Luis Bajo Jiménez (Palma de Mallorca). N.º 26. 32 puntos.



Marcelino Castaño Ortiz (Sevilla). N.º 78. 27 puntos.

APRENDE DE TUS ERRORES

Esta semana veremos tres errores del Interface-1 relacionados con el empleo del Microdrive.

Jesús ALONSO RODRÍGUEZ

Code error

SIGNIFICADO: «Error en CODE». Indica que el área especificada después de «CODE», en una sentencia «LOAD», es menor que el tamaño del bloque que se está pretendiendo cargar.

CAUSA: El Microdrive no hace la carga de forma tan «despreocupada» como el cassette. Por el contrario, comprueba si el bloque que se va a cargar cabe en el área específica; de no ser así, emite este informe. Por supuesto, sólo se presentará en sentencias «LOAD» referidas a bloques de bytes, y eso sólo si se especifica el área de destino. Si no se especificase ésta, se tomaría la que figure en la cabecera del bloque a cargar.

La causa de este error puede ser tanto una especificación errónea del fichero (queríamos cargar uno, pero hemos dado el nombre de otro), como una confusión en su supuesta longitud, sin olvidar —como siempre— la posibilidad de haber equivocado el nombre de una variable que constituya uno de los parámetros de «CODE».

SOLUCIÓN: En principio, lo mejor es listar la sentencia donde se ha detectado el error para ver qué nombre de fichero estamos dando y qué longitud le estamos suponiendo. En caso de que alguno de estos parámetros venga dado por una variable, convendría seguirle la pista hacia atrás. Si es necesario, puede servir de ayuda comprobar la parte del programa donde se almacena el fichero, para comprobar si se le dio una longitud errónea al crearlo.

Drive "write" protected

SIGNIFICADO: «Unidad protegida de escritura». Indica que se ha intentado escribir información en un cartucho de Microdrive que tiene partida la lengüeta de plástico que lo protege de borrados accidentales.

CAUSA: El error se puede producir tanto con un comando «SAVE» como con un «OPEN» a un fichero inexistente (con lo que es abierto para escritura). También puede ocurrir que tengamos un cartucho con la lengüeta rota al que hayamos puesto una «pegatina» de papel para habilitarlo, pero que esta «pegatina» esté floja, rota o desprendida.

SOLUCIÓN: Es posible que no fuera éste el cartucho sobre el que quisiéramos escribir; la solución en este caso es sacarlo e insertar otro. Si a pesar de todo queremos escribir en él, se puede anular la protección colocando una «pegatina» de papel en el lugar anteriormente ocupado por la lengüeta de plástico (de forma similar a como se hace con una cassette).

Cabe otra posibilidad que no está de más considerar. En algunas aplicaciones, se utilizan dos unidades de Microdrive. Una de ellas contiene un cartucho protegido de escritura con trozos de programa que se cargan sólo cuando se van a ejecutar. El otro contiene los datos propiamente dichos. Si al ir a guardar un dato nos dirigiéramos —por error— al cartucho que contiene el programa, también se produciría este informe.

File not found

SIGNIFICADO: «Fichero no encontrado». Indica que el Microdrive es incapaz de cargar o verificar un determinado fichero, bien porque no existe, bien porque no se encuentra alguno de sus sectores.

CAUSA: Este error es, sin duda, el que con más frecuencia se produce en las operaciones con Microdrive. No siempre indica lo que parece; por tanto, es imprescindible hacer un pequeño repaso a la forma en que trabaja el Microdrive para comprender perfectamente las posibles causas de este informe.

El Microdrive es un dispositivo formateado, lo cual quiere decir que la cinta se encuentra dividida en sectores individualmente identificables por el sistema operativo. Cada sector se compone de dos bloques seguidos de un «silencio» cada uno de ellos. En primer lugar, está el bloque de «cabecera» de 1,25 milisegundos de duración, que contiene 27 bytes con el siguiente significado: 10 bytes a "0" y dos a «255» para sincronizar el inicio de la lectura; 1 byte de «flag»; 1 byte con el número de sector; 2 bytes no utilizados; 10 bytes con el nombre del cartucho y un byte de «checksum», que es el XOR de los 14 anteriores. A continuación viene un «silencio» (o espacio en blanco) de 3,75 milisegundos de duración, seguido del bloque de datos, de 25 milisegundos de duración, que contiene 540 bytes en el siguiente significado: 10 bytes a «0» y 2 a «255» para inicio de lectura; 1 byte de «flag»; 1 byte con el número de registro; 2 bytes con la longitud del registro (normalmente será 512, excepto en el último registro de cada fichero); 10 bytes con el nombre del fichero al que corresponde el registro; 1 byte de «cheksum» que contiene el XOR de los 14 anteriores; 512 bytes de datos (los datos propiamente dichos del fichero) y, por último, 1 byte de «cheksum» que contiene el XOR de los 512 bytes anteriores. A continuación de este bloque y antes del siguiente sector, hay un «silencio» de 7 milisegundos de duración.

La duración total de un sector, con silencios incluidos, es de 37 milisegundos. Contiene un total de 567 bytes, de los que sólo son utilizables para datos 512 (de los 37 milisegundos, tan sólo son utilizados 26,25 para información, si suprimimos los silencios, lo que nos da una velocidad de lectura de nada menos que 172.800 baudios o bits por segundo; esta alta velocidad se obtiene gracias al uso de una cabeza con dos entrehierros que almacena los bits en dos pistas de forma alternativa, lo que permite un mayor «empaquetado» de la información).

El número de sectores que caben en un cartucho de Microdrive depende de la longitud de la cinta y del número de áreas inservibles que ésta tenga, pero

suele oscilar entre los 180 y 190 sectores, lo que nos da una capacidad total de datos comprendida entre 90 y 95 K. Los sectores de la cinta van pasando ante la cabeza grabadora/lectora en orden descendente según su número, de forma que el sector de mayor número pasa inmediatamente después del primero.

Cada vez que se almacena un fichero en el Microdrive, se fragmenta en registros de 512 bytes, cada uno de los cuales se almacena en un sector. Pero, y esto es muy importante, los sectores no son nunca consecutivos. Si es el primer fichero de un cartucho, se dejan aproximadamente 8 sectores vacíos entre cada dos llenos. Estos sectores serán rellenados, posteriormente, con registros correspondientes a otros ficheros; cuando el cartucho ya tiene varios ficheros, se van buscando sectores libres donde ir colocando cada registro. El resultado es que, tras unas cuantas operaciones de escritura, lectura y borrado, la información en el cartucho se encuentra totalmente desordenada y sólo alguien tan «inteligente» como nuestro Spectrum es capaz de encontrar en cada caso lo que está buscando.

Lo cierto es que la tarea no es fácil y, en algunas ocasiones, hasta nuestro querido ordenador se ve incapacitado para realizarla y «arroja la toalla» con un rotundo "File not found". No es que el fichero no exista, sino que la información en el cartucho ha llegado a un grado tal de desorden que el Sistema Operativo es incapaz de encontrar alguno de los registros que componen el fichero. Parece mentira que esto ocurra, pero se trata de uno de los fallos más graves del Microdrive. Afortunadamente, no es frecuente; nosotros, por propia y sufrida experiencia, hemos comprobado que suele pasar cuando los últimos sectores de un fichero se almacenan a bastante distancia de los primeros, lo que se puede observar por la cadencia de parpadeos en el borde de la pantalla durante la operación «SAVE».

La causa de este informe será, la mayor parte de las veces, un cartucho con la información muy desordenada. Aunque no hay que olvidar que también se puede producir este error si alguno de los sectores ha resultado corrompido por apagar o encender el ordenador con el cartucho insertado.

SOLUCIÓN: Por supuesto, ante la aparición de este informe conviene comprobar si son correctos tanto el nombre de fichero como el cartucho que está insertado. No obstante, la solución a estos errores pasa por una sistemática de actuación que los prevenga y que se podría resumir en los siguientes puntos:

1.° Verificar cualquier cosa que se almacene en Microdrive. Si un cartucho está tan desordenado que el fichero no se vaya a poder leer, tampoco se podrá verificar, lo que nos dará la posibilidad de almacenarlo de nuevo. Ante una verificación fallida por dos o tres veces consecutivas, se recomienda utilizar otro cartucho. En estos casos, el fallo en la verificación vendrá indicado por un "File not found" en lugar de por el «Verification has failed» que veremos más adelante.

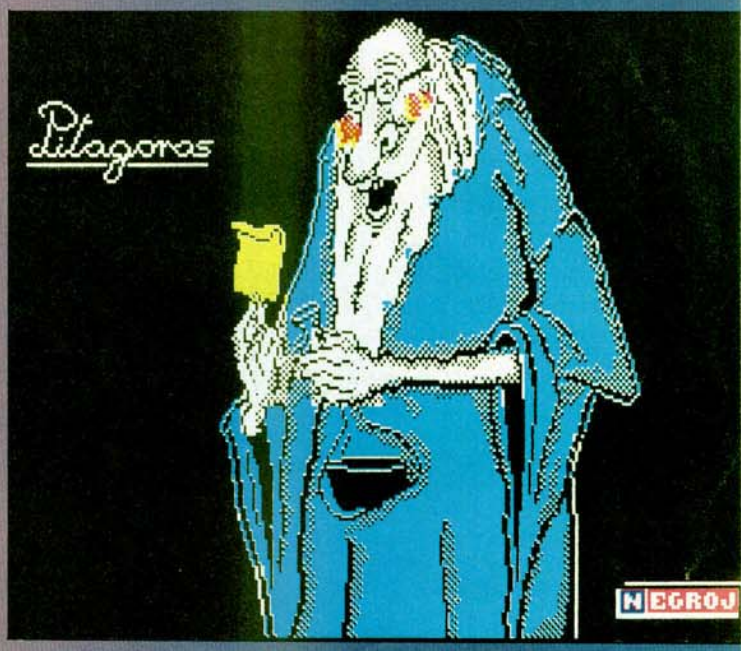
2.° Evitar la realización de borrados y reescrituras frecuentes en un mismo cartucho, ya que estas operaciones son las que más desordenan la información.

3.° Guardar por duplicado cualquier dato que consideremos importante. El perder el trabajo de varios días por culpa de un cartucho que se niega a cargar es una de las situaciones más desesperantes que conocemos (lo decimos por propia experiencia).

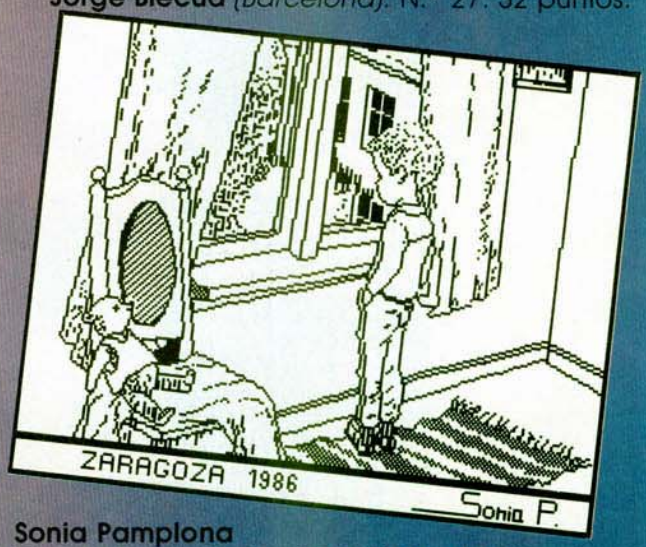
Si se siguen estas recomendaciones, es difícil que perdamos un fichero. No obstante, no hay que olvidar que el Microdrive es un sistema de almacenamiento poco fiable si se lo compara con un disco.

PIXEL A PIXEL

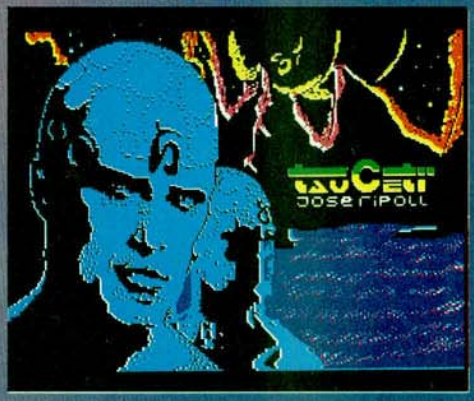
Este continúa siendo el rincón reservado para mostraros semanalmente los trabajos que quedaron clasificados entre los 100 primeros puestos de nuestro 1.er Concurso de «Diseño gráfico por ordenador».



Jorge Blecua *(Barcelona)*. N.° 27. 32 puntos.



Sonia Pamplona Roche *(Zaragoza)*. N.° 76. 27 puntos.



José Ripoll Campos *(Madrid)*. N.° 77. 27 puntos.

APRENDE DE TUS ERRORES

Esta semana veremos tres informes relacionados con el empleo del Microdrive, que suelen presentarse con bastante frecuencia.

Jesús ALONSO RODRÍGUEZ

Merge error

SIGNIFICADO: «Error en MERGE». Indica que se ha producido un error al utilizar el comando MERGE y, por alguna razón, no es posible mezclar el bloque pedido con el programa que se encuentra en memoria.

CAUSA: existen dos posibles causas que pueden dar lugar a este error. La primera es que el fichero especificado en el comando MERGE no sea un programa Basic sino un bloque de bytes, una matriz o un fichero secuencial. En este caso, lógicamente, es imposible fusionar el fichero pedido con el programa.

La segunda causa que puede dar lugar a este error es que el fichero sea un programa Basic pero esté grabado con autoejecución, es decir, con LINE. Cuando utilizábamos el cassette, podíamos evitar la autoejecución de un programa grabado con LINE si tecleábamos MERGE en lugar de LOAD; de esta forma, se podía acometer la desprotección de cualquier programa. Al diseñar el Sistema Operativo de Microdrive, el equipo de Sinclair jugó una baza a favor de los productores de software cerrando la puerta del MERGE para dificultar la desprotección del software grabado en Microdrive (de hecho lo consiguieron, un programa grabado en Microdrive es mucho más difícil de desproteger que uno grabado en cassette). El resultado que se pretendía lograr era incentivar la producción de software para este dispositivo a pesar del mayor coste del soporte. Este objetivo, sin embargo, no se puede decir que lo consiguieran.

SOLUCIÓN: conviene comprobar, en primer lugar, si lo que estamos intentando «MERGEar» es un programa en Basic. Si es así, querrá decir que está grabado con autoejecución (lo que se puede comprobar haciendo LOAD) y no podremos «MERGEarlo». Cuando se graba un programa propio, protegido en Microdrive y se hace con LINE, es conveniente guardar una copia desprotegida y almacenada sin LINE en otro cartucho para facilitar las modificaciones posteriores, ya que es muy difícil —prácticamente imposible— desproteger un programa grabado en Microdrive. (En realidad no es imposible, pero ha-

cerlo requiere unos conocimientos de Código Máquina y un dominio del Sistema Operativo del Microdrive, que se encuentran fuera del alcance del 99 por 100 de los usuarios.)

Microdrive Full

SIGNIFICADO: «Microdrive lleno». El mensaje no puede ser más explícito. Evidentemente, indica que el cartucho que se está usando, está totalmente lleno de información y no queda sitio en él para almacenar el fichero.

CAUSA: este informe se presentará al hacer un SAVE sobre un cartucho que no tenga suficientes sectores libres para almacenar el fichero que se está pretendiendo guardar. Pero también se presenta cuando tenemos abierto un canal de salida en Microdrive y ejecutamos un PRINT por ese canal, habiendo agotado todos los sectores libres. En este caso, el fichero queda sin cerrar y no se pueden leer los datos que se hayan metido hasta ahora. Se puede borrar con ERASE, pero tardará bastante ya que el Microdrive lee cinco veces todo el cartucho para asegurarse de que el fichero no está cerrado y, sin embargo, habrá que repetir el programa que se estaba ejecutando, para volver a almacenar los datos que se hubieran guardado hasta entonces. En ocasiones, se perderá una valiosa información, por lo que es recomendable asegurarse de que el fichero nos va a caber, antes de abrirlo y empezar a meter datos. El número de Ks libres se obtiene al ejecutar un comando CAT.

SOLUCIÓN: evidentemente, si el cartucho está lleno, no hay más remedio que cambiarlo por otro. No obstante, no es conveniente agotar al máximo la capacidad de almacenamiento de un cartucho, ya que se podrían producir los problemas que vimos al estudiar el informe «File not found».

Microdrive not present

SIGNIFICADO: «Microdrive no presente». Indica que no hay un cartucho válido insertado en la unidad de Microdrive especificada.

CAUSA: este informe se puede producir por tres razones. En primer lugar, si se especifica un número de unidad que no está conectada. El Interface-1 permite conectar un máximo de 8 unidades de Microdrive que quedarán numeradas de 1 a 8 y de derecha a izquierda, siendo la unidad número 1, la más cercana al ordenador. Si se especificase, por ejemplo, «3» como número de unidad y sólo se tuvieran dos conectadas, se produciría este error.

La segunda causa posible es que la unidad esté conectada pero no tenga ningún cartucho insertado. Por último, se producirá, también, este error si la unidad está concectada, tiene un cartucho insertado, pero éste no está formateado.

SOLUCIÓN: se puede empezar por revisar la línea indicada en el informe para comprobar a qué unidad se está intentando acceder. Una vez visto esto, comprobaremos si la unidad existe y si tiene cartucho insertado, de ser así, la razón será que el cartucho no está formateado (tal vez hayamos metido uno equivocado) por lo que la solución será formatearlo o cambiarlo por otro.

APRENDE DE TUS ERRORES

En este capítulo, veremos una serie de informes del Interface-1 que se producen cuando olvidamos alguno de los parámetros de una sentencia.

Jesús ALONSO RODRÍGUEZ

Missing baud rate

SIGNIFICADO: «falta el número de baudios». Indica que se ha olvidado especificar la velocidad de transmisión —en baudios— al ejecutar una sentencia FORMAT dirigida a la conexión RS-232.

CAUSA: por la conexión RS-232 se transmite en serie. Esto quiere decir que los bits que componen cada byte se mandan uno a continuación de otro. La velocidad a la que salen o entran los bits —número de bits por segundo— se mide en «baudios» (unidad de medida que lleva este nombre en honor del ingeniero francés Emile Baudot, uno de los pioneros de la telegrafía). Los distintos periféricos que se pueden conectar con el Spectrum vía RS-232 pueden utilizar diferentes velocidades de transmisión. Para permitir la sincronización del ordenador con el periférico a él conectado, es posible ajustar la velocidad de transmisión a cualquiera de los valores normalizados internacionalmente. Estos valores son: 50, 110, 300, 600, 1200, 2400, 4800, 9600 y 19200 baudios. La elección de una u otra velocidad no responde a un capricho. Utilizando una impresora muy rápida, tal vez no se presenten problemas trabajando a 19200 baudios, sin embargo, si se están mandando datos por línea telefónica mediante un MODEM, no conviene utilizar velocidades superiores a los 1200 baudios y, en cualquier caso, el porcentaje de errores aumentará en relación directa con la velocidad.

Antes de utilizar un canal asociado al RS-232 hay que fijar la velocidad de transmisión que utilizará ese canal, mediante una sentencia FORMAT; por ejemplo:

```
FORMAT "t";19200
```

Fijará una velocidad de 19200 baudios para el canal "t". Si al hacer esto, se olvida incluir la velocidad de transmisión, aparecerá este mensaje para indicarlo.

SOLUCIÓN: se trata de un error poco frecuente que, además, estará localizado en la línea indicada por el informe.

Missing drive number

SIGNIFICADO: «falta el número de microdrive». Indica que se ha olvidado especificar el número de microdrive en una sentencia dirigida a este

dispositivo.

CAUSA: dado que el Spectrum puede manejar hasta ocho unidades de microdrive, será necesario especificar a cuál de ellas nos queremos dirigir. Por ejemplo, en la sentencia:

SAVE *''m'';3;''pepe''

Estaremos especificando que el programa ''pepe'' debe salvarse en la unidad de microdrive número 3, es decir, la tercera de las conectadas empezando a contar por la derecha. Si olvidáramos especificar este número, se produciría el informe.

SOLUCIÓN: al igual que en el caso anterior, el problema deberá estar en la línea indicada en el informe por lo que será ésta la línea a revisar.

Missing name

SIGNIFICADO: «falta el nombre». Indica que se ha olvidado especificar el nombre del fichero en una operación dirigida al microdrive, RS-232 o red de área local.

CAUSA: este informe se producirá siempre que realicemos una operación SAVE, LOAD, VERIFY, MERGE, OPEN (en microdrive), ERASE, FORMAT (en microdrive) o MOVE (en microdrive), sin especificar un nombre de fichero (en el caso de FORMAT no es un nombre de fichero lo que se especifica, sino el nombre del cartucho).

SOLUCIÓN: de nuevo, bastará con revisar la línea indicada en el informe, ya que ella será la causante del error.

Missing station number

SIGNIFICADO: «falta el número de estación». Indica que se ha olvidado especificar el número de estación en una sentencia dirigida a la red de área local.

CAUSA: este informe se producirá siempre que se ejecute una instrucción dirigida a la red de área local sin especificar un número de estación. Recuerde que si quiere enviar a todas las estaciones o recibir de todas las estaciones, deberá utilizar el número de estación ''0''.

SOLUCIÓN: al igual que en los casos anteriores, el error debe residir en la línea donde ha sido detectado, es decir, en la indicada por el informe.

Reading a "write" file

SIGNIFICADO: «leyendo un fichero de escritura». Indica que se ha intentado leer un fichero que se encuentra abierto para escritura, no para lectura.

CAUSA: cuando se abre un fichero en microdrive (con el comando OPEN), el sistema operativo comprueba si existe ya un fichero con ese nombre, en cuyo caso, lo abre para lectura. Si no existiera un fichero con ese nombre, abre uno nuevo para escritura. Por tanto, es el ordenador quien decide en qué modo abre un fichero (esta «chapuza» sólo la hace el Spectrum). De alguna forma, el usuario debe saber si un fichero existe o no antes de abrirlo, para saber en qué modo se le va a abrir. Lo que nos indica, en realidad, la aparición de este informe es que el fichero del que estamos intentando leer no existía previamente. La causa suele ser debida, en la mayoría de los casos, a un error en el nombre del fichero.

SOLUCIÓN: podemos empezar por comprobar la línea indicada en el informe. Si estamos utilizando una variable como nombre de fichero, habrá que seguirle la pista hacia atrás para ver en qué punto tomó un valor equivocado. En cualquier caso, habrá que cerrar el fichero que hemos abierto por error y borrarlo para evitar la acumulación de «morralla» en el cartucho.

APRENDE DE TUS ERRORES

Esta semana terminaremos de ver los informes de error en tiempo de proceso, asociados con la utilización del Interface-1.

Jesús ALONSO RODRÍGUEZ

Stream already open

SIGNIFICADO: «Corriente ya abierta». Indica que se ha intentado abrir una corriente que ya estaba abierta.

CAUSA: antes de reasignar una corriente a un canal distinto de aquél al que se asignó en la última sentencia OPEN, hay que cerrarla con CLOSE. Esto no reza, lógicamente, para las corrientes #0, #1, #2 y #3 cuando se encuentren asociadas a los canales «K», «K», «S» y «P» respectivamente, ya que éstos son los canales que asumen, por defecto, si intentamos cerrarlas. En cualquier otro caso, si hacemos un OPEN a una corriente que ya está abierta, obtendremos este mensaje de error.

SOLUCIÓN: salvo que se haya equivocado en el número de corriente, podría decirse que se trata de un error de tipo «lógico». En primer lugar, conviene comprobar la línea indicada en el informe para asegurarse de que estamos dando bien el número de corriente. Si esto es así, habrá que revisar el flujo del programa para ver dónde teníamos que haber cerrado la corriente y no lo hemos hecho. Podría ocurrir, también, que hayamos intentado abrir la misma corriente por segunda vez sin que esto tuviera que ser así; por ejemplo, si la sentencia OPEN se encuentra dentro de un bucle de iteración y su correspondiente sentencia CLOSE está fuera. Éste sería uno de los pocos «errores lógicos» que, sin embargo, se pueden detectar en tiempo de ejecución.

Verification has failed

SIGNIFICADO: «Ha fallado la verificación». Indica que, durante un proceso de verificación, los datos recibidos no concuerdan con los almacenados en memoria.

CAUSA: si el informe se produce mientras se está verificando una transmisión por red local o RS-232, lo más probable es que la causa resida en uno o más bits alterados por la presencia de ruido o parásitos en la línea. El error se ha tenido que producir durante la verificación; ya que, de haberse producido durante la transmisión, hubiera sido detectado en el ordenador de destino. No obstante, si al repetir la verificación se volviera a producir error, sería conveniente repetir la transmisión. Cuando se trabaja con la conexión RS-232, hay que tener en cuenta que el porcentaje de errores crece exponencialmente al aumentar la velocidad de transmisión.

Por el contrario, si el informe se produce al verificar un fichero almacenado en microdrive, lo más probable es que la cinta tenga algún sector defectuoso (no todo él, sino una parte, ya que el fichero ha podido leerse por completo). Si el error se presenta con todos los cartuchos, puede ser debido a suciedad o mal contacto en las conexiones del microdrive.

SOLUCIÓN: si el error se ha producido durante una transmisión por red local o RS-232, repetir la verificación. Si volvie-

ra a fallar, volver a enviar el fichero. Si se producen frecuentes errores, conviene revisar las conexiones o seleccionar una velocidad de transmisión menor.

Si el error se ha producido trabajando con el microdrive, volver a almacenar el fichero. Si persisten los errores con determinado cartucho, lo mejor es volverlo a formatear (salvando previamente la información que contenga) para evitar más problemas. Si no se quiere formatear el cartucho, se puede dejar el fichero que ocupa el sector problemático, y almacenarlo de nuevo con otro nombre; de esta forma, el fichero antiguo nos ocupa el sector defectuoso y evitamos que nos vuelva a dar problemas. Por supuesto, este método no resulta rentable si se trata de un fichero largo, ya que serán muchos los sectores anulados.

Ante una presentación frecuente de este tipo de problemas con diversos cartuchos, es aconsejable limpiar los contactos del microdrive y el Interface-1, así como la cabeza grabadora-lectora de aquél (lo cierto, es que la cabeza se ensucia poquísimo y rara vez será necesario limpiarla); esto se puede hacer muy bien, frotando con un algodón empapado en alcohol iso-propílico y asegurándose de que no queden hilillos.

Writing to a "read" file

SIGNIFICADO: «Escribiendo en un fichero de lectura». Indica que se ha intentado escribir en un fichero que ya existía y que, por tanto, se había abierto para lectura.

CAUSA: se trata de una consecuencia de la forma un tanto «chapucera» que tiene el Interface-1 de manejar los ficheros secuenciales. En realidad, lo que nos indica este informe es que hemos intentado abrir para escritura un fichero que ya existía. A diferencia de los restantes ordenadores, el Spectrum no permite «sobre-escribir» el contenido de un fichero, por lo que la única forma de alterar alguno de sus datos será: leer el fichero y cargarlo en memoria, alterar los datos precisos, borrar el fichero antiguo del cartucho y volverlo a almacenar.

También puede ser causa del error el haber intentado escribir en un fichero que habíamos abierto con intención de leer en él y a sabiendas de que ya existía; pero parece bastante improbable que alguien cometa tal error.

SOLUCIÓN: como de costumbre, empezamos por revisar la línea indicada en el informe para ver si nos hemos «armado un lío» y hemos confundido el nombre del fichero con el de uno que ya existe, o hemos equivocado el número de corriente (por ejemplo, si estamos leyendo de uno y escribiendo en otro).

Si lo que estamos intentando hacer es sobre-escribir un fichero, no habrá más remedio que borrarlo antes.

Wrong file type

SIGNIFICADO: «Tipo de fichero equivocado». Indica que hemos intentado hacer una operación con un fichero de un tipo que no la admite.

CAUSA: el microdrive maneja cuatro tipos de ficheros:

Programas: cuando se almacena un programa Basic con SAVE.

Bloques de bytes: se crean con SAVE... CODE.

Datos: son los que se crean con SAVE... DATA.

Ficheros secuenciales: son los que se crean con el empleo de OPEN y PRINT #...

Hay operaciones que sólo pueden hacerse con ficheros de determinado tipo, por ejemplo, no se puede hacer un MOVE con un fichero de programas. Por el contrario, no se puede hacer LOAD a un fichero secuencial, ni LOAD... CODE a uno de datos. En cualquier caso, el fichero indicado deberá ser del tipo que permita la operación que se va a realizar.

SOLUCIÓN: conviene revisar la línea indicada en el informe para ver qué operación estamos intentando hacer. Al mismo tiempo, habrá que comprobar la parte del programa que creó el fichero para ver de qué tipo es. Una consulta al manual nos aclarará qué operaciones están permitidas con cada tipo de fichero; aunque, en la mayoría de los casos, el error se verá claramente por simple «sentido común».

APRENDE DE TUS ERRORES

Jesús ALONSO RODRÍGUEZ

En este último capítulo de la serie sobre depuración de programas, veremos cómo procesar los errores automáticamente, así como la forma de producir, intencionadamente, un determinado error.

Procesamiento de errores

La mayor parte de los intérpretes de Basic tienen una instrucción que se llama: ON ERROR GOTO..., y que sirve para que el intérprete salte a un número de línea determinado cuando se produzca un error, en lugar de detener la ejecución del programa. El Spectrum carece de esta instrucción, pero podemos imitar su funcionamiento mediante una sencilla rutina en Código Máquina. Veamos, detenidamente lo que es un «ON ERROR GOTO».

En algunos casos, se pueden presentar situaciones que son de error para el sistema e impiden al intérprete seguir con la ejecución del programa; pero que, sin embargo, no se deben a errores de programación, sino a situaciones que han sido previstas por el programador, pero ante las que se encuentra incapacitado de resolver dado que el intérprete las resuelve por su cuenta, deteniendo la ejecución. Pensemos, por ejemplo, en que el usuario apriete la tecla «BREAK», o en que le pidamos que introduzca, con INPUT, una expresión a evaluar con VAL y nos meta una inevaluable que nos provoque un «Nonsense in Basic»; otra posibilidad es que escribamos un programa pensado para funcionar con Microdrive, pero que también pueda ser utilizado en un ordenador que carezca de Interface-1; en este caso, sería más útil que, cuando el usuario seleccione una opción de Microdrive, el programa le sacara un mensaje diciendo algo como: «Opción inválida, no hay Microdrive», en lugar de detenerse con el absurdo «Nonsense in Basic». También, es posible que se quiera dar la posibilidad de retornar al menú principal desde cualquier lugar del programa mediante la pulsación de «BREAK». En todos estos casos, necesitamos una rutina que detecte la que es una situación de error para el sistema, y salte a algún lugar del programa para hacer algo que nosotros hayamos previsto.

Ni que decir tiene, que este método no sirve para resolver errores de programación, es más, no conviene incorporarlo hasta que el programa haya sido depurado por completo, ya que el proceso de depuración se complica muchísimo si el ordenador empieza a hacer «cosas raras» en lugar de detenerse con el informe de error.

En la «Microficha R-1» se publicó una rutina que simulaba un «ON ERROR GOTO»; por si algún lector no pudo adquirir el ejemplar en su día, la volvemos a publicar (ver LISTADO 1) y explicamos su funcionamiento. El LISTADO 2 es la misma rutina en formato de Cargador Universal de Código Máquina (DUMP en 60000).

En las líneas 60 a 150, fijamos la línea a la que queremos saltar en caso de error (habrá una instrucción Basic que comunique los datos a esta rutina, luego la veremos) e inicializamos el contenido del elemento de la pila que contiene la dirección de retorno en caso de error, para que el retorno se produzca a la línea 180 de nuestra rutina. A partir de esta línea, comprobamos si el error es «0 OK», «End of file» o «STOP Statement» ya que procesar estos errores podría provocar un «cuelgue» del ordenador. Si no es ninguno de ellos, se pasa la ejecución a la línea del programa Basic cuyo número está contenido en la variable «GOTOL» (dirección 23728). Para activar la rutina, tecleamos: RANDOMIZE línea + USR 60000 donde «línea» es el número de la línea donde queremos que se transfiera el control en caso de error. Veamos un ejemplo: queremos que, en caso de error, el control sea transferido a la línea 9000; esto sería equivalente a un:

ON ERROR GOTO 9000

Y nosotros lo hacemos con:

RANDOMIZE 9000 + USR 60000

Una vez en esta línea, podemos saber qué tipo de error se ha producido leyendo su código de la dirección 23681 para lo que podemos utilizar una sentencia como la siguiente:

LET err = PEEK 23681

En la variable «err» tendremos el código de error (1 para «NEXT without FOR», 2 para «Variable not found», etc.). Podemos comprobar este código y actuar en consecuencia según qué tipo de error se haya producido. Desgraciada-

LISTADO 1

Rutina para simular un ON ERROR GOTO.

```
10 ; * ON ERROR GOTO *
20 ;
30 ;
40         ORG   60000          ;Rutina reubicable.
50 ;
60 ERR0    LD    HL,ERR1-ERR0 ;Long. de la rutina.
70         ADD   HL,BC          ;Calcula dir. ERR1.
80         EX    DE,HL          ;La transfiere a DE.
90         LD    HL,(ERRSP)     ;La guarda en ERRSP.
100        LD    (HL),E         ;Dirección de salto
110        INC   HL             ;  en caso de error.
120        LD    (HL),D         ;
130        CALL  FINT2          ;Lee nº línea de STACK.
140        LD    (GOTOL),BC     ;Lo guarda en 23728.
150        RET                  ;Vuelve al Basic.
160 ;
170 ;
180 ERR1   DEC   SP             ;Decrementa la pila.
190        DEC   SP             ;
200        LD    A,(IY+0)       ;Carga cód. de error.
210        INC   A              ;Lo incrementa.
220        CP    0              ;
230        JR    Z,CONT         ;Salta si es "0 OK".
240        CP    8              ;Salta si es "8 End
250        JR    Z,CONT         ;  of file".
260        CP    9              ;Salta si es "9 STOP
270        JR    Z,CONT         ;  Statement".
280        LD    (ERRNR2),A     ;Guarda cód. de error.
290        LD    (IY+0),255     ;Carga error "0 OK".
300        LD    HL,(GOTOL)     ;Número de línea a
310        LD    (NEWPPC),HL    ;  saltar.
320        XOR   A              ;Limpia acumulador.
330        LD    (IY+10),A      ;Indica primer comando.
340        SET   7,(IY+1)       ;Marca ejecutando Basic.
350        JP    STMTR1         ;Entra en ejecución.
360 ;
370 ;
380 CONT   INC   SP             ;Restablece la pila.
390        INC   SP             ;
400        JP    MAIN4          ;Detiene ejecución e
410 ;                              imprime el error.
420 ERRSP  EQU   23613          ;Puntero de ret. en error.
430 GOTOL  EQU   23728          ;Línea a la que saltar.
440 NEWPPC EQU   23618          ;Línea donde inicia ejec.
450 FINT2  EQU   7833           ;Lee num. del STACK.
460 STMTR1 EQU   7037           ;Ejecuta proxima instrucc.
470 MAIN4  EQU   4867           ;Imprime informe de error.
480 ERRNR2 EQU   23681          ;Código del error producido.
```

LISTADO 2

```
1 2113000 9EB2A3D5C7323 641
2 72CD991EED43B05CC93B 1334
3 3BFD7E003CFE002820FE 1078
4 08281CFE09281832815C 674
5 FD3600FF2AB05C22425C 1064
6 AFFD770AFDCB01FEC37D 1588
7 1B3333C3031300000000 346
```

LISTADO 2:

DUMP en 60000
N.° de bytes = 66

mente, la rutina no almacena en ningún sitio el número de la linea donde se ha producido el error; tal vez alguno de nuestros lectores sea capaz de modificarla para que lo haga.

Simulación de errores

Teniendo una rutina que nos gestione los errores, no estaría mal tener algún método para forzar la producción de un determinado error, nada más fácil. También, necesitaremos una rutina en Código Máquina; aunque, en este caso, sólo tendrá dos bytes de longitud. El primero será 207 (código de operación de «RST 8») y el segundo será el código de error menos uno; en caso de querer producir el error «0 OK» el segundo byte deberá ser 255. Podemos almacenar los bytes en cualquier lugar donde no estorben al Basic. En el modelo de 48 K es posible colocarlos en el buffer de impresora (23296 y 23297); en el modelo de 128 K es preferible colocarlos por encima de la RAMTOP. Veamos un ejemplo para producir el error: «2 Variable not found»; colcaremos los datos por encima de RAMTOP en las direcciones 65534 y 65535:

POKE 65534,207
POKE 65535,1
RANDOMIZE USR 65534

Pruebe «POKEando» distintos valores en 65535 para producir distintos informes de error. Este método puede servirle tanto para forzar un salto a la rutina de manejo de errores como para detener la ejecución con un determinado mensaje.

Si «POKEa» el código del error menos 1 en la dirección 23610 (Variable del sistema «ERR-NR») podrá hacer que el programa presente un mensaje distinto de «0 OK» cuando termine de ejecutarse con lo que, también, podrá «engañar» a la rutina de manejo de errores.

Por útlimo, todo lo explicado en este capítulo vale para los errores de la configuración básica (códigos 0 al 27), pero no funcionará para los errores del Interface-1, ya que éstos carecen de código.

Aquí damos por terminada la serie sobre depuración de errores. Esperamos que haya servido para que ninguno de nuestros lectores se quede parado sin saber qué hacer ante la presentación de un mensaje de error.

ACLARACIÓN

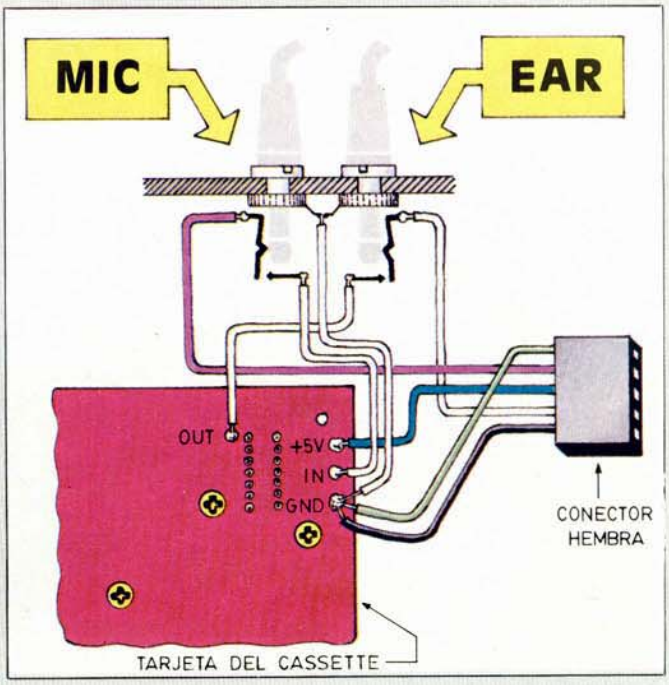
CONEXIONES PARA OTROS CASSETTES EN EL SPECTRUM +2

Primitivo de FRANCISCO

En el artículo sobre conexiones para otros cassettes en el Spectrum +2 se deslizó un pequeño error en la **figura 3** de la página 26, de MICROHOBBY 114. Están invertidos los títulos de **MIC** y **EAR** que son, sin duda, los que dan lugar al problema que han detectado algunos de nuestros lectores. En las pruebas que nosotros hicimos en nuestro ordenador no se presentó en absoluto ningún inconveniente, funcionando prácticamente según lo previsto; naturalmente que en cableado no existía el error mencionado de **MIC** y **EAR.**

No obstante, hay que tener presente que esta adaptación para otros cassettes se ha realizado de una forma sencilla, lo cual conlleva ciertos condicionamientos. En primer lugar, para el correcto funcionamiento hay que emplear un cassette de calidad que posibilite entregar cierto nivel de señal a la entrada del ordenador EAR (nuestras pruebas las hicimos con dos cassettes marca Sony y Sanyo). Por tanto, es necesario emplear un buen cassette y poner su volumen al máximo. También es aconsejable no tener conectados al tiempo dos cables apantallados entre ordenador y cassette a fin de evitar oscilaciones parásitas que perturbarían la señal.

Siguiendo estas recomendaciones todo funcionará correctamente como en las clásicas tomas de MIC y EAR de los modelos precedentes.



Disposición correcta de los conectores EAR y MIC una vez realizada la corrección en la figura. Los que hayan intentado cargar programas por la salida MIC en vez de por la entrada EAR es muy posible que lo hayan conseguido... Eso sí, con el volumen al máximo.