



---

## **Parte 22**

### **Otros periféricos**

Temas tratados:

- Microunidades ZX
- Redes
- RS232
- Subteclado numérico

Hay muchos periféricos disponibles para conectarlos al +2. En el capítulo 10 le daremos información más amplia sobre su conexión y funcionamiento.

La microunidad ZX es un dispositivo flexible de almacenamiento masivo de datos a alta velocidad. Funcionará no sólo con SAVE, VERIFY, LOAD y MERGE, sino también con PRINT, LIST, INPUT e INKEY\$.

Una *red* es un sistema para interconectar varios ordenadores de forma que puedan no sólo comunicarse, sino también utilizar recursos comunes. Una de las aplicaciones sería, por ejemplo, tener una sola microunidad a disposición de varios ordenadores.

El interfaz RS232 es una conexión estándar que permite unir un ordenador con teclados, impresoras y otros dispositivos, aunque no hayan sido diseñados específicamente para el +2.

El subteclado numérico facilita el uso de las funciones de edición adicionales de 128 BASIC y también la introducción de datos numéricos.



---

## Parte 23

# IN y OUT

Temas tratados:

OUT  
IN

Como el lector ya sabe, podemos leer la memoria ROM y RAM y escribir en la RAM utilizando la función PEEK y la orden POKE. En realidad, al microprocesador no le importa si la memoria es ROM o RAM: sólo sabe que hay 65536 direcciones de memoria y que puede leer un byte en cada una de ellas (aunque no tenga sentido) y escribir también un byte en cada una (aunque se pierda). Análogamente, hay 65536 *puertas E/S* (puertas de entrada/salida). Estas puertas las utiliza el microprocesador para comunicarse con, por ejemplo, el teclado y la impresora, y también para controlar la memoria adicional y el chip de sonido. Algunas de ellas pueden ser controladas desde BASIC sin problemas mediante la función IN y la orden OUT, pero hay posiciones en las que usted no debe escribir desde BASIC, ya que probablemente provocaría la *caída del sistema*, lo que representaría la pérdida del programa y todos los datos.

IN es una función, como PEEK. Su forma es:

IN *dirección*

Tiene un argumento (la dirección de la puerta) y su resultado es el byte leído en esa puerta.

OUT es una sentencia, como POKE. Su forma es:

OUT *dirección,valor*

que escribe el *valor* dado en la puerta cuya *dirección* es la especificada. La interpretación que se dé a la dirección depende en gran medida del resto del ordenador. Frecuentemente muchas direcciones diferentes significan lo mismo. En el +2 lo más sensato es imaginar la dirección escrita en binario, ya que los bits individuales (cada uno de los cuales puede tener el valor 0 o 1) funcionan independientemente. Una dirección está formada por 16 bits:

A15, A14, A13, A12, A11, A10, A9, A8, A7, A6, A5, A4, A3, A2, A1, A0

A0 es el bit de las unidades, A1 el bit de los 'doses', A2 el bit de los 'cuatros', y así sucesivamente. Los bits A0, A1, A2, A3 y A4 son los importantes. Normalmente tienen el valor 1, y el hecho de que uno de ellos esté a 0 le encarga al ordenador algo específico.

---

El ordenador no puede hacer varias cosas al mismo tiempo, así que no debe estar a 0 más que uno de estos bits. Los bits A6 y A7 son ignorados, de modo que, si es usted un mago de la electrónica, puede usarlos como quiera. Las mejores direcciones son los múltiplos de 32 menos 1, de tal manera que los bits A0 a A4 estén todos a 1. Los bits A8, A9 y siguientes se usan a veces para dar información adicional; se los utiliza casi siempre para controlar la memoria adicional y el sonido.

El byte leído o escrito (byte de datos) está formado por ocho bits:

D7, D6, D5, D4, D3, D2, D1, D0

Veamos una lista de las direcciones de puerta utilizadas:

Las siguientes direcciones de entrada leen el teclado y el magnetófono. El teclado se compone de ocho medias filas de cinco teclas, a saber:

IN 65278 lee desde **MAYUSC** hasta V

IN 65022 lee desde A hasta G

IN 64510 lee desde Q hasta T

IN 63486 lee desde 1 hasta 5 (y JOYSTICK 2)

IN 61438 lee desde 0 hasta 6 (y JOYSTICK 1)

IN 57342 lee desde P hasta Y

IN 49150 lee desde **INTRO** hasta H

IN 32766 lee desde 'espacio' hasta B

(Estas direcciones son  $254 + 256 * (255 - 2 \uparrow n)$ , donde  $n$  va de 0 a 7.)

En el byte leído, los bits D0 a D4 representan las cinco teclas de la media fila en cuestión. D0 representa la tecla más externa; D4, la más próxima al centro. El bit es 0 si está pulsada la tecla, y 1 en caso contrario. D6 es controlado por el magnetófono; en ausencia de datos procedentes de la cinta, su valor es aleatorio.

Para JOYSTICK 1, el bit 0 representa disparo; el bit 1, arriba; el bit 2, abajo; el bit 3, derecha; y el bit 4, izquierda. Para JOYSTICK 2, el bit 0 representa izquierda; el bit 1, derecha; el bit 2, abajo; el bit 3, arriba; y el bit 4, disparo. En BASIC éstos se leen como las teclas de números.

La puerta de salida 254 controla el sonido (D4) y la señal de grabación hacia el magnetófono (D3) y establece el color del borde (D2, D1 y D0).

Las puertas 254, 247 y 239 se usan para controlar los dispositivos externos mencionados en la Parte 22.

La puerta 32765 gestiona la memoria adicional. La ejecución de OUT desde BASIC hacia esa puerta causará casi siempre la caída del sistema, con la consiguiente pérdida del programa y los datos. Esta puerta está descrita en la Parte 24 de este capítulo (bajo el título de 'Gestión de la memoria'). Esta puerta es de *sólo escritura*; no se puede determinar el estado actual de la paginación mediante una función IN.

---

La puerta 49149 controla los registros de datos del chip de sonido. La puerta 65533 en modo de salida escribe una dirección de registro, y en modo de entrada lee un registro. El uso cuidadoso de estos dos registros puede permitir la generación de sonidos mientras BASIC realiza otra tarea; pero hay que tener en cuenta que también controlan el RS232, el subteclado numérico y el MIDI.

Ejecute este programa para ver cómo funciona el teclado:

```
10 FOR n=0 TO 7: REM numero de media fila
20 LET a=254+256*(255-2↑n)
30 PRINT AT 0,0; IN a: GO TO 30
```

y entreténgase pulsando teclas. Cuando acabe con cada media fila, pulse **BREAK** y escriba:

```
NEXT n
```

Los *buses* de control, datos y direcciones están todos a su disposición en la parte trasera del +2, en el zócalo **EXPANSION E/S**, de modo que usted puede hacer con el +2 casi todo lo que hacía con un Z80 (aunque algunas veces el hardware del ordenador puede impedirle algo).

En el capítulo 10 damos un diagrama y una descripción de las patillas del zócalo **EXPANSION E/S**.



---

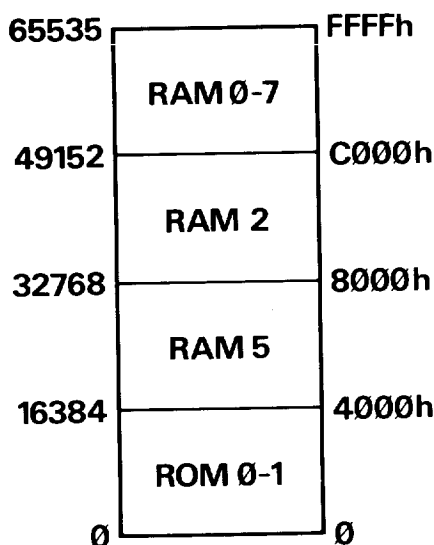
## Parte 24

# La memoria

Temas tratados:

### CLEAR

En el interior del +2 todo se almacena en forma de *bytes*, es decir, números enteros del margen de 0 a 255. Usted puede pensar que ha almacenado el precio de las verduras o la dirección de su amigo Juan, pero, de hecho, toda la información ha sido convertida en colecciones de bytes, y son los bytes lo que el ordenador ve y entiende.



Mapa de memoria del +2

Cada lugar donde puede ser almacenado un byte está caracterizado por una dirección, que es un número entre 0 y FFFFh (una h al final de los dígitos significa que el número es hexadecimal). Esto significa que una dirección puede ser almacenada en forma de dos bytes. Podemos imaginar la memoria como una larga fila de cajas numeradas, cada una de las cuales puede contener un byte. Sin embargo, no todas las cajas son iguales; del 4000h al FFFFh son cajas de RAM, lo que significa que podemos levantar la tapa y alterar el contenido. Pero del 0 al 3FFFh son cajas de ROM, las cuales tienen una tapa de cristal que no podemos abrir; lo único que podemos hacer es ver a través del



---

cristal lo que se puso en ella cuando se fabricó el ordenador. En el +2 hemos puesto más del doble de cajas que las que caben cómodamente: mientras que el procesador puede acceder directamente a 65536 bytes, el +2 tiene 131072 bytes de RAM y 32768 bytes de ROM, lo que da un total de 163840 bytes (160K). El hardware oculta este exceso de memoria al microprocesador mediante un sistema llamado *paginación*; BASIC y el microprocesador siempre «ven» la memoria como 16K de ROM y 48K de RAM.

Para inspeccionar el contenido de una caja usamos la función PEEK. Su argumento es la dirección de la caja y su resultado es el contenido. Por ejemplo, el siguiente programa escribe los primeros 21 bytes de la ROM junto con sus direcciones:

```
10 PRINT "Dirección"; TAB 10; "Byte"
20 FOR a=0 TO 20
30 PRINT a; TAB 10; PEEK a
40 NEXT a
```

Estos bytes seguramente no tendrán ningún significado para usted, pero el microprocesador los entiende como instrucciones que le dicen qué tiene que hacer.

Para cambiar el contenido de una caja (suponiendo que sea de RAM), usamos la orden POKE. Su forma es:

*POKE dirección, contenido*

donde *dirección* y *contenido* son expresiones numéricas. Por ejemplo, la orden

```
POKE 31000,57
```

da al byte de la dirección 31000 el nuevo valor 57. Escriba:

```
PRINT PEEK 31000
```

para comprobarlo. (Pruebe con otros valores para ver que no estamos haciendo trampa.) El nuevo valor debe estar entre -255 y +255; si es negativo, BASIC le suma 256.

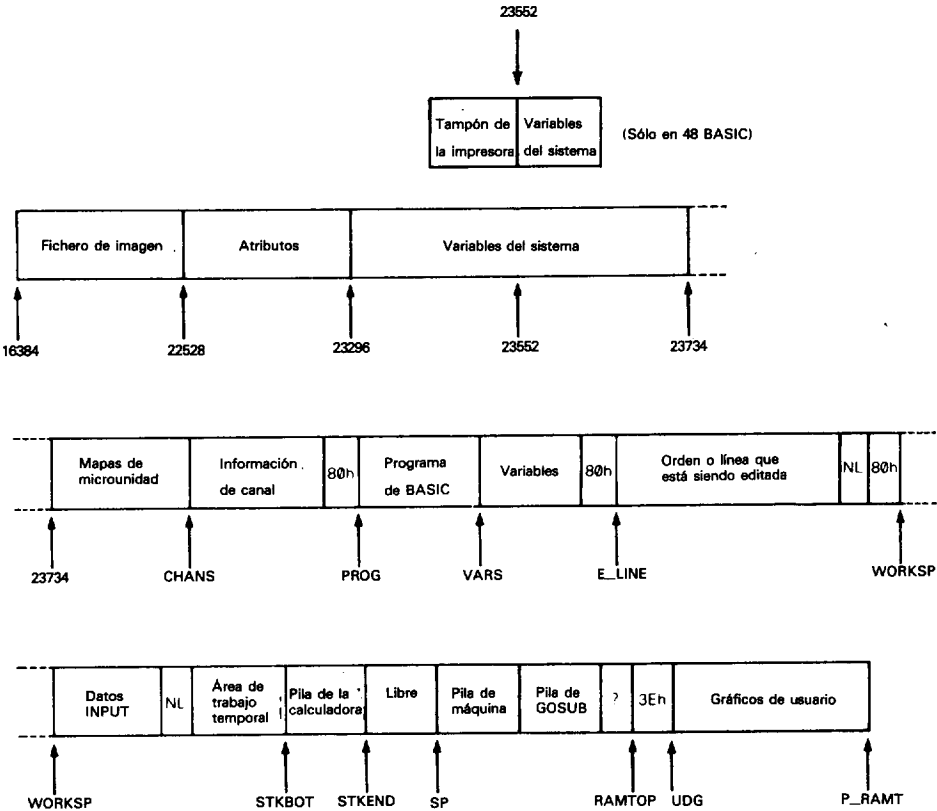
La facultad de modificar el contenido de una posición de memoria nos otorga un inmenso poder sobre el ordenador si sabemos cómo usarla, e inmensas posibilidades de destrucción si no sabemos. Es muy fácil, colocando un valor incorrecto en ciertas direcciones de memoria, perder largos programas que hemos tardado horas en transcribir. Sin embargo, afortunadamente, esto no le hará al ordenador ningún daño permanente.

Ahora veremos más detalladamente cómo se usa la RAM. No se moleste en leer esto si no está realmente interesado.

La memoria está dividida en diferentes áreas (mostradas en el diagrama siguiente), en cada una de las cuales se almacena un tipo de información diferente. El tamaño de cada área es el justo para la información que contiene en el momento; si introducimos algo más en cierta posición (por ejemplo, añadiendo una línea de programa o una variable),

el ordenador le hace hueco desplazando hacia arriba todo lo que haya por encima de esa posición. A la inversa, si borramos información, todo el resto de la memoria se des- plaza hacia abajo.

El fichero de imagen (memoria de pantalla) almacena los datos necesarios para construir la imagen de la pantalla. Está organizado de una forma más bien curiosa, así que no tiene ningún interés leerlo (PEEK) ni modificarlo (POKE). Cada posición de carácter en la pantalla consiste en una retícula de 8×8 puntos; cada punto puede ser 0 (papel) o 1 (tinta), por lo que podemos almacenar su descripción en 8 bytes (uno para cada fila). Sin embargo, estos ocho bytes no se almacenan uno al lado del otro. Las columnas co- rrespondientes en los 32 caracteres de una línea se almacenan juntas en un bloque de 32 bytes, ya que esto es lo que necesita el rayo de electrones del televisor al barrer la pantalla de izquierda a derecha. Puesto que la imagen completa consiste en 24 líneas de ocho barridos cada una, sería de esperar que los 192 bloques (24×8) de 32 bytes se almacenasen uno a continuación del otro; pues bien, no es así. Primero vienen los blo- ques superiores de las líneas 0 a 7, después los siguientes bloques de las líneas 0 a 7, y



Mapa de la memoria de BASIC

así hasta los bloques más bajos de esas mismas líneas; después se hace lo mismo con las líneas 8 a 15, y luego con las líneas 16 a 23. El resultado final de todo esto es que, si está usted acostumbrado a un ordenador en el que puede aplicar PEEK y POKE a la memoria de pantalla, más vale que empiece a habituarse a SCREEN\$ y PRINT AT (o PLOT y POINT).

Los *atributos* de cada posición de carácter son los colores y las restantes características; su formato es el que describimos a propósito de la función ATTR y *sí* son almacenados línea por línea en el orden que usted esperaría.

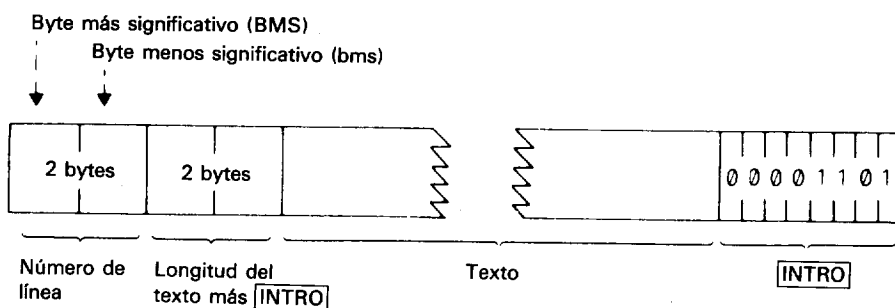
La forma en la que el ordenador organiza sus asuntos no es exactamente la misma en 48 BASIC y en 128 BASIC. El área que constituía el tampón de la impresora en 48 BASIC se utiliza para almacenar ciertas variables adicionales del sistema en 128 BASIC.

Las variables del sistema contienen diversos elementos de información que indican al ordenador en qué de estado se encuentra. La lista completa está en la Parte 26 de este capítulo; por el momento observe que algunas de ellas (CHANS, PROG, VARS, E\_LINE, etc.) contienen la dirección de los límites entre las distintas áreas de la memoria. No son variables de BASIC y por lo tanto no serán reconocidas por el +2.

Los mapas de microunidad se usan sólo cuando se tiene una microunidad conectada. Normalmente están vacíos.

La información de canal contiene datos sobre los dispositivos de entrada y salida: el teclado (junto con la pantalla inferior), la pantalla superior y la impresora.

Cada línea de programa de BASIC tiene la siguiente forma:

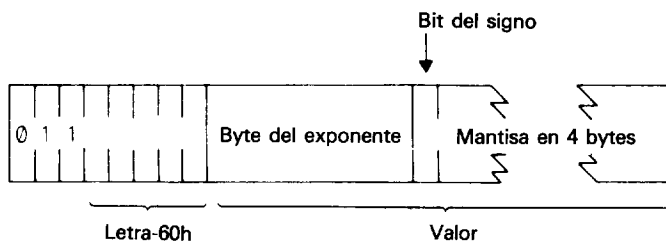


Observe que, en contraste con todos los demás casos de números de dos bytes en el Z80, aquí el número de línea se almacena con su byte más significativo primero, es decir, en el mismo orden en que nosotros los escribimos.

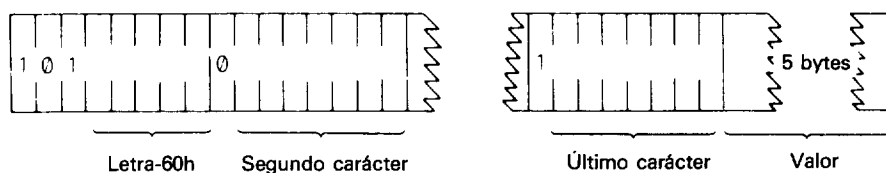
Una constante numérica del programa lleva a continuación su forma binaria, usando el carácter CHR\$ 14 seguido por cinco bytes para el número propiamente dicho.

Las variables tienen formatos diferentes para los distintos tipos. Las letras de los nombres se almacenan como si empezasen en minúscula.

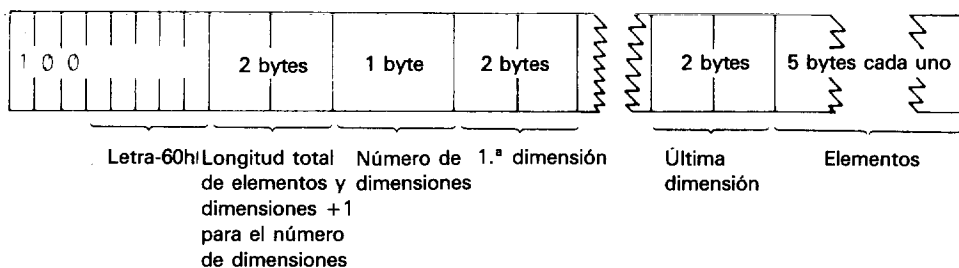
Variable numérica cuyo nombre es una sola letra:



Variable numérica cuyo nombre consiste en varias letras:



Matriz numérica:



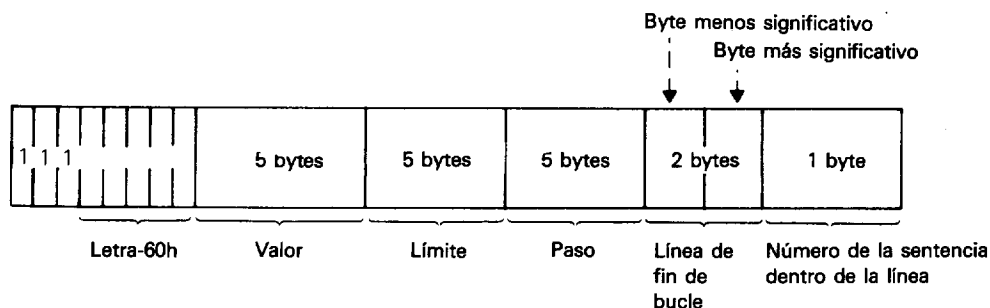
El orden de los elementos es:

- en primer lugar los elementos cuyo primer subíndice es 1
- a continuación los elementos cuyo primer subíndice es 2
- a continuación los elementos cuyo primer subíndice es 3
- ... y así sucesivamente para todos los posibles valores del primer subíndice.

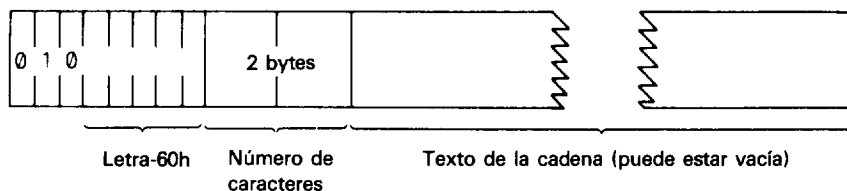
Los elementos con un primer subíndice dado se ordenan de la misma forma con respecto al segundo, y así sucesivamente hasta el último.

Por ejemplo, los elementos de la matriz *c* de 3\*6 de la Parte 12 de este capítulo se almacena en el siguiente orden: *c*(1,1) *c*(1,2) *c*(1,3) *c*(1,4) *c*(1,5) *c*(1,6) *c*(2,1) *c*(2,2)...*c*(2,6) *c*(3,1) *c*(3,2)...*c*(3,6).

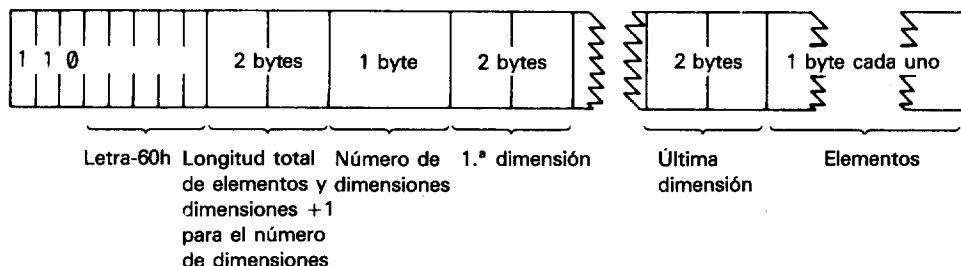
Variable de control para un bucle FOR...NEXT:



Variable literal:



Matriz literal:



---

La calculadora es la parte de BASIC que se encarga de la aritmética; los números con los que opera se guardan en su mayoría en la pila de la calculadora.

La parte libre que se sigue en el mapa de memoria es el espacio que no ha sido utilizado de momento.

La pila de máquina es la usada por el microprocesador Z80 para guardar direcciones de retorno, etc.

El funcionamiento de la pila de GO SUB está descrito en la Parte 5 de este capítulo.

El byte al que «apunta» RAMTOP tiene la dirección más alta utilizada por BASIC. Incluso NEW, que borra la RAM, no pasa de esa dirección, y por tanto no cambia los gráficos definidos por el usuario. Se puede modificar la dirección RAMTOP especificando su nuevo valor en una sentencia CLEAR:

**CLEAR *nueva RAMTOP***

cuyo efecto es el siguiente:

1. Borra todas las variables.
2. Borra el fichero de imagen.
3. Restaura la posición del cursor gráfico en el extremo inferior izquierdo de la pantalla.
4. Restaura (RESTORE) el puntero de datos.
5. Borra la pila de GO SUB e iguala su base a RAMTOP (suponiendo que ésta se encuentre entre la pila de la calculadora y el final físico de la RAM, pues de lo contrario deja RAMTOP donde estaba).

RUN ejecuta implícitamente una sentencia CLEAR, pero no modifica RAMTOP. Utilizando CLEAR de esta forma se puede, o bien desplazar RAMTOP hacia arriba para dedicar más espacio a BASIC (destruyendo los gráficos de usuario), o bien desplazarla hacia abajo para hacer más grande la zona de RAM que no es borrada por NEW.

Escriba NEW y luego CLEAR 23825 para hacerse una idea de lo que le ocurre a la máquina cuando se queda sin memoria.

Si intenta que el +2 calcule algo (escriba, por ejemplo, PRINT 1+1), verá en la pantalla el mensaje 4 Out of memory ('Memoria agotada'). Esto significa que al ordenador no le queda más espacio donde almacenar información. Si se le presenta este mensaje cuando está transcribiendo un programa largo, tendrá que vaciar un poco la memoria (por ejemplo, borrando una línea) para poder controlar el ordenador.

## **Gestión de la memoria**

Ya hemos mencionado antes que en el ordenador hay bastante más memoria que la que el microprocesador puede manejar cómodamente. Éste puede «direccionar» sólo 64K de memoria de una vez, pero la memoria adicional puede ser intercambiada a discreción

---

con esos 64K. Observe un aparato de televisión; a pesar de que sólo puede ocuparse de un canal a la vez (igual que usted), hay otros canales que pueden ser seleccionados pulsando los botones adecuados. Así, aunque hay más información de la que usted puede asimilar a un tiempo, usted puede elegir la que le interese.

Lo que ocurre en el ordenador es muy parecido. Colocando los bits adecuados en cierta puerta de E/S, la máquina puede elegir qué porciones quiere usar del total de 160K de memoria. BASIC ignora casi permanentemente toda la memoria adicional. En cambio, en los programas de juegos viene muy bien tener una RAM casi triplicada. Vuelva a echar un vistazo al mapa de memoria del principio de esta sección. Los bancos RAM 2 y RAM 5 están siempre en la posición indicada en el diagrama, y sin embargo no hay ninguna razón por la que no puedan estar en la sección conmutable (C000h a FFFFh), si bien esto tampoco tendría ninguna utilidad. Los bancos de RAM son de dos tipos: RAM 4 a RAM 7, que son compartidos (es decir, que comparten el tiempo con los circuitos de video), y RAM 0 a RAM 3, que sólo están al servicio del microprocesador. Cualquier programa de código de máquina que requiera un control muy preciso del tiempo (por ejemplo, los de música o comunicaciones) debe guardar todas las rutinas de temporización en los bancos no compartidos.

El conmutador de hardware está en la dirección de E/S 7FFDh (32765 decimal). El campo de bits para esta dirección es el siguiente:

D0 a D2	Selección de RAM
D3	Selección de pantalla
D4	Selección de ROM
D5	Bloqueo de 48K

D2 a D0 crean un número de tres bits que selecciona qué RAM entra en el hueco comprendido entre C000h y FFFFh. En BASIC normalmente está seleccionado el banco RAM 0; durante la edición de usa RAM 7 para almacenar varios tampones y memorias transitorias. D3 conmuta pantallas; la pantalla 0 está en la RAM 5 (que empieza en 4000h) y es la que utiliza BASIC; la pantalla 1 está en RAM 7 (a partir de C000h) y sólo puede ser usada por programas de código de máquina. Es perfectamente factible preparar una pantalla en RAM 7 y después «desconmutarla»; de esta forma se deja los 48K libres para datos y programa. D4 determina si debe ser colocada en las direcciones 0000h a 3FFFh la ROM 0 (la ROM del editor) o la ROM 1 (la ROM de BASIC). D5 es un dispositivo de seguridad; en cuanto se pone a 1 este bit, el ordenador adopta la configuración del Spectrum de 48K estándar y bloquea todos los circuitos de paginación de memoria. A partir de ese momento no se lo puede volver a convertir en un ordenador de 128K más que apagándolo o pulsando el botón **RESET**; no obstante, el chip de sonido puede seguir siendo controlado por OUT.

---

## Parte 25

### Variables de sistema

Temas tratados:

#### POKE, PEEK

Los bytes de memoria del 23296 al 23733 están reservados para usos específicos del sistema. Hay también unas cuantas rutinas (usadas para mantener en orden la paginación de memoria) y algunas posiciones que contienen *variables de sistema*. Éstas pueden ser examinadas (con PEEK) para conocer diversos aspectos del sistema; también puede tener interés modificar algunas de ellas. En esta sección daremos la lista completa.

Hay gran diferencia, como sería de esperar, entre el área de variables de sistema de 48 BASIC y de 128 BASIC. En el modo de 48 BASIC las rutinas y variables por debajo de 23552 no existen; en su lugar hay un tampón, entre 23296 y 23552, que se usa para controlar la impresora. Ésta era una posición muy popular para pequeñas rutinas de código de máquina en el Spectrum de 48K; si usted prueba cualquiera de estas rutinas en 128 BASIC, provocará la caída del sistema con toda seguridad. Todos los programas antiguos que usan PEEK, POKE y USR deben ser ejecutados, por tanto, en 48 BASIC (no obstante, se los puede introducir en 128 BASIC y luego pasar a 48 BASIC con la orden SPECTRUM).

Las variables de sistema tienen nombres, pero es preciso no confundirlos con las palabras clave y los nombres de variable utilizados en BASIC. El ordenador no reconocerá los nombres como referencias a variables de sistema; nosotros los utilizamos solamente como nemónicos y para el ordenador no significan nada.

Las abreviaturas de la columna 1 de la tabla significan lo siguiente:

X No se debe modificar estas variables porque el sistema podría fallar.

N La modificación de las variables no tendrá efectos duraderos.

R No es una variable, sino el punto de entrada a una rutina.

El número de la columna 1 es el número de bytes de que consta la variable o rutina. En el caso de dos bytes, el primero es el menos significativo, lo contrario de lo que parecería natural. Así, para escribir el valor  $v$  en una variable de dos bytes que se encuentra en la dirección  $n$  se debe dar las órdenes:

```
POKE  $n, v - 256 * \text{INT}(v/256)$   
POKE  $n+1, \text{INT}(v/256)$ 
```

Para leer ese valor se utiliza la expresión:

```
PEEK  $n + 256 * \text{PEEK}(n+1)$ 
```



---

<b>Notas</b>	<b>Dirección</b>	<b>Nombre</b>	<b>Contenido</b>
R20	23296	SWAP	Subrutina de paginación.
R9	23316	YOUNGER	Subrutina de paginación.
R18	23325	ONERR	Subrutina de paginación.
R5	23343	PIN	Prerrutina de entrada del RS232.
R22	23348	POUT	Prerrutina de salida de claves del RS232. Puede ser modificada para evitar el filtrado de códigos de control.
R14	23370	POUT2	Prerrutina de salida de caracteres del RS232.
N2	23384	TARGET	Dirección de subrutina en ROM 1.
X2	23386	RETADDR	Dirección de retorno en ROM 0.
X1	23388	BANKM	Copia del último byte enviado al banco.
X1	23389	RAMRST	Instrucción RST8.
N1	23390	RAMERR	Número de error, ROM 1.
2	23391	BAUD	Periodo de bit del RS232 en estados T/26.
N2	23393	SERFL	Indicador de recepción del segundo carácter, y datos.
N1	23395	COL	Columna actual, desde la 1 hasta la anchura.
1	23396	WIDTH	Anchura del papel en columnas.
1	23397	TVPARS	Número de los parámetros por línea esperados por el RS232.
1	23398	FLAGS3	Diversos indicadores.
N10	23399	N STR1	Nombre de fichero.
1	23409	HD 00	Código de tipo de fichero.
2	23410	HD 0B	Longitud del bloque.
2	23412	HD 0D	Comienzo del bloque.
2	23414	HD 0F	Longitud del programa.
2	23416	HD 11	Número de línea.
1	23418	SC 00	Código de tipo de fichero, segundo grupo.
2	23419	SC 08	Longitud del bloque, segundo grupo.
2	23421	SC 0D	Comienzo del bloque, segundo grupo.
2	23423	SC 0F	Longitud del programa, segundo grupo.
X2	23425	OLDSP	SP antiguo cuando se está usando TSTACK.
X2	23427	SFNEXT	Puntero hacia la primera reseña libre del directorio.
X3	23429	SFSPACE	Número de bytes disponibles (17 bits).
N1	23432	ROW01	Indicadores e imagen de la fila 1 del teclado numérico.
N1	23433	ROW23	Imágenes de las filas 2 y 3 del teclado numérico.
N1	23434	ROW45	Imágenes de las filas 4 y 5 del teclado numérico.
X2	23435	SYNRET	Dirección de retorno para ONERR.
5	23437	LASTV	Último valor escrito por la calculadora.
2	23442	RNLINE	Línea que está siendo renumerada.
2	23444	RNFIRST	Número de línea inicial para Renumerar.
2	23446	RNSTEP	Valor del incremento para Renumerar.
N8	23448	STRIP1	Mapa de bits de la banda superior del menú.
N8	23456	STRIP2	Mapa de bits de la banda inferior del editor.

---

Notas	Dirección	Nombre	Contenido
X	23551	TSTACK	La pila temporal crece desde aquí hacia abajo.
N8	23552	KSTATE	Usada en la lectura del teclado.
N1	23560	LAST K	Última tecla pulsada.
1	23561	REPDEL	Tiempo, en cincuentavos de segundo (sesentavos en EE.UU.), que debe mantenerse pulsada una tecla antes de que se repita. Inicialmente es 35, pero se le puede dar otros valores (con POKE).
1	23562	REPPER	Pausa, en cincuentavos de segundo (sesentavos en EE.UU.), entre las repeticiones sucesivas de una tecla pulsada (inicialmente, 5).
N2	23563	DEFADD	Si se está evaluando una función definida por el usuario, dirección de los argumentos; de lo contrario, 0.
N1	23565	K DATA	Segundo byte de los controles de color introducidos por el teclado.
N2	23566	TVDATA	Almacena bytes de los controles de color, AT y TAB que van al televisor.
X38	23568	STRMS	Direcciones de los canales conectados a las salidas.
2	23606	CHARS	256 menos que la dirección del juego de caracteres (que empieza con el espacio y continúa hasta el símbolo de copyright). Normalmente en ROM, pero lo puede copiar en la RAM y hacer que CHARS apunte hacia él.
1	23608	RASP	Duración del pitido de aviso.
1	23609	PIP	Duración del 'click' del teclado.
1	23610	ERR NR	1 menos que el código del informe. Empieza en 255 (para -1), de forma que PEEK 23610 da 255.
X1	23611	FLAGS	Diversos indicadores que controlan el sistema BASIC.
X1	23612	TVFLAG	Indicadores asociados con el televisor.
X2	23613	ERR SP	Dirección del elemento de la pila de máquina que hay que usar como retorno de error.
N2	23615	LIST SP	Dirección de la dirección de retorno desde un listado automático.
N1	23617	MODE	Especifica cursor K, L, C, E o G.
2	23618	NEWPPC	Línea a la que hay que saltar.
1	23620	NSPPC	Número de sentencia dentro de una línea al que hay que saltar. La modificación de NEWPPC primero y de NSPPC después fuerza el salto a una sentencia específica dentro de una línea.
2	23621	PPC	Número de línea de la sentencia que está siendo ejecutada.
1	23623	SUBPPC	Número, dentro de la línea, de la sentencia que está siendo ejecutada.

<b>Notas</b>	<b>Dirección</b>	<b>Nombre</b>	<b>Contenido</b>
1	23624	BORDCR	Color del borde multiplicado por 8; también contiene los atributos usados normalmente para la parte inferior de la pantalla.
2	23625	EPPC	Número de la línea actual (con el cursor de programa).
X2	23627	VAR5	Dirección de variables.
N2	23629	DEST	Dirección de la variable en asignación.
X2	23631	CHANS	Dirección de los datos del canal.
X2	23633	CURCHL	Dirección de la información que está siendo usada para entrada y salida.
X2	23635	PROG	Dirección del programa de BASIC.
X2	23637	NXTLIN	Dirección de la siguiente línea del programa.
X2	23639	DATADD	Dirección del terminador del último elemento de DATA.
X2	23641	E LINE	Dirección de la orden que está siendo escrita en el teclado.
2	23643	K CUR	Dirección del cursor.
X2	23645	CH ADD	Dirección del próximo carácter que debe ser interpretado (el carácter que sigue al argumento de PEEK, o el código de 'línea nueva' al final de una sentencia POKE).
2	23647	X PTR	Dirección del carácter que está detrás del marcador [?].
X2	23649	WORKSP	Dirección de área de trabajo temporal.
X2	23651	STKBOT	Dirección del extremo inferior de la pila de la calculadora.
X2	23653	STKEND	Dirección del comienzo del espacio libre.
N1	23655	BREG	Registro <i>b</i> de la calculadora.
N2	23656	MEM	Dirección del área usada para memoria de la calculadora. (Generalmente MEMBOT, pero no siempre.)
1	23658	FLAGS2	Otros indicadores.
X1	23659	DF SZ	Número de líneas (incluida una en blanco) de que consta la pantalla inferior.
2	23660	STOP	El número de la primera línea en listados automáticos.
2	23662	OLDPPC	Número de la línea a la cual salta CONTINUE.
1	23664	OSPCC	Número de la sentencia, dentro de la línea, a la cual salta CONTINUE.
N1	23665	FLAGX	Diversos indicadores.
N2	23666	STRLEN	Longitud de la cadena que está siendo asignada.
N2	23668	T ADDR	Dirección del siguiente elemento en la tabla de sintaxis (muy improbable que sea útil).
2	23670	SEED	Semilla para RND. Ésta es la variable establecida por RANDOMIZE.

Notas	Dirección	Nombre	Contenido
3	23672	FRAMES	Contador de barridos del cuadro en 3 bytes (primero el byte menos significativo); se incrementa cada 20 ms. (Véase Parte 18 de este capítulo.)
2	23675	UDG	Dirección del primer gráfico definido por el usuario. Usted puede cambiarla, por ejemplo, para ahorrar espacio disponiendo de menos gráficos definibles.
1	23677	COORDS	x-coordenada del último punto dibujado.
1	23678		y-coordenada del último punto dibujado.
1	23679	P POSN	33 menos número de columna de la posición de la cabeza impresora.
1	23680	PR CC	Byte menos significativo de la dirección de la próxima posición en la que debe escribir LPRINT (en el tampón de la impresora).
1	23681		No utilizada.
2	23682	ECHO E	33 menos el número de columna y 24 menos número de fila (pantalla inferior) del final del tampón de entrada.
2	23684	DF CC	Dirección en el fichero de imagen de la posición de escritura.
2	23686	DF CCL	Como DF CC, pero para la pantalla inferior.
X1	23688	S POSN	33 menos número de columna de la posición de escritura.
X1	23689		24 menos número de fila de la posición de escritura.
X2	23690	SPOSNL	Como S POSN, pero para la pantalla inferior.
1	23692	SCR CT	Cuenta los desplazamientos de la pantalla. Es siempre 1 más que el número de desplazamientos que serán realizados antes de emitir la pregunta scroll?. Si usted modifica este número introduciendo un valor mayor que 1, la pantalla se desplazará sin pedirle permiso.
1	23693	ATTR P	Colores permanentes actuales, etc. (establecidos por las instrucciones de color).
1	23694	MASK P	Usada para colores transparentes, etc. Cualquier bit que sea 1 indica que el tributo correspondiente no se toma de ATTRP, sino de lo que ya está en la pantalla.
N1	23695	ATTR T	Colores temporales actuales, etc. (establecidos por cláusulas de color).
N1	23696	MASK T	Como MASK P, pero temporal.
1	23697	P FLAG	Otros indicadores.
N30	23698	MEMBOT	Área de memoria de la calculadora, usada para almacenar números que no conviene introducir en la pila de la calculadora.

---

Notas	Dirección	Nombre	Contenido
2	23728		No utilizada.
2	23730	RAMTOP	Dirección del último byte del área de sistema de BASIC.
2	23732	P-RAMT	Dirección del último byte de la RAM física.

### Ejercicio

1. El siguiente programa muestra 22 bytes del área de variables (de KSCAN en adelante):

```
10 FOR n=0 TO 21
20 PRINT PEEK (PEEK 23627+256*PEEK 23628+n)
30 NEXT N
```

Trate de identificar la variable de control n a la vista de las descripciones anteriores. Ahora cambie la línea 20 por

```
20 PRINT PEEK (23755+n)
```

El programa muestra así los primeros 22 bytes del área de programa. Trate de correlacionarlos con el propio programa.

---

## Parte 26

# Utilización del código de máquina

Temas tratados:

### USR con argumento numérico

Esta sección está escrita para quienes ya conocen el *código de máquina* del Z80, es decir, el juego de instrucciones a las que responde el microprocesador Z80. Si usted no sabe nada sobre código de máquina y tiene interés en aprenderlo, puede leer alguno de los muchos libros que se han escrito sobre el tema; debería adquirir uno que se titulase "Código de máquina (o lenguaje ensamblador) en el Z80 para el principiante absoluto"; y si en él se menciona el +2 o los otros ordenadores ZX Spectrum, tanto mejor.

Los programas de código de máquina se escriben normalmente en *lenguaje ensamblador*, el cual, aun siendo bastante críptico, con la práctica no resulta muy difícil de comprender. (Puede ver la lista de las instrucciones del lenguaje ensamblador en la Parte 27 de este capítulo.) Sin embargo, para ejecutar en el +2 un programa escrito en lenguaje ensamblador es necesario convertir esas instrucciones en una serie de bytes, que constituyen el programa en código de máquina. Esa conversión puede realizarla el propio ordenador mediante un programa que se llama *ensamblador*. En el +2 no hay ningún ensamblador incorporado, pero usted puede adquirirlo en una cinta. A falta de ensamblador, puede hacer la conversión usted mismo, pero, en la práctica, esto sólo es posible con programas muy cortos.

Tomemos como ejemplo el programa:

```
ld bc, 99
ret
```

que carga el par de registros bc con el número 99. Al ensamblarlo, este programa se traduce en los cuatro bytes siguientes: 1, 99, 0 (que significan ld bc, 99) y 201 (que es ret). (Si consulta los códigos 1 y 201 en la tabla de la Parte 27, verá que el 1 corresponde a ld bc, NN, donde NN representa cualquier número de dos bytes, y que el 201 corresponde a ret).

Una vez convertido el programa de lenguaje ensamblador a código de máquina, el siguiente paso es introducirlo en el ordenador (un ensamblador probablemente lo haría de forma automática). Hay que empezar por decidir el lugar de la memoria en el que queremos situarlo (lo más conveniente es reservarle un espacio entre el área de BASIC y la de los gráficos definidos por el usuario).

La orden

```
CLEAR 65267
```

---

reserva un espacio de 100 bytes a partir de la dirección 65268. Para introducir el programa de código de máquina, podríamos usar un programa de BASIC de este estilo:

```
10 LET a=65268
20 READ n: POKE a,n
30 LET a=a+1: GO TO 20
40 DATA 1,99,0,201
```

(Este programa se detiene y emite el mensaje E Out of DATA ('Datos agotados') cuando termina de escribir en la memoria los cuatro bytes especificados.)

Para ejecutar el programa de código de máquina se utiliza la función USR (pero ahora con argumento numérico; en concreto, la dirección inicial del programa). El resultado de USR es el valor que queda en el par de registros bc al volver del programa en código de máquina, así que si escribimos

```
PRINT USR 65268
```

obtendremos la respuesta 99.

La dirección de retorno a BASIC es 'apilada' de la manera usual, así que el retorno se lleva a cabo mediante una instrucción ret del Z80. No se debe usar los registros iy e i en rutinas de código de máquina que vayan a utilizar el mecanismo de interrupciones de BASIC. Tampoco se debe cargar i con valores comprendidos entre 40h y 7Fh (aunque no se vaya a usar nunca IM 2). Los valores entre C0h y FFh para i también deberían ser eludidos si se va a insertar la memoria compartida (RAM 4 o RAM 7) entre C000h y FFFFh. Esto se debe a una interacción entre el controlador de video y el mecanismo de refresco del Z80, que puede ocasionar la caída del sistema, la corrupción de la pantalla u otros efectos indeseables. Por tanto, sólo se debe dirigir las interrupciones IM 2 hacia direcciones comprendidas entre 8000h y BFFFh, a menos que se esté muy seguro de la estructura del mapa de memoria.

Hay una serie de problemas típicos que se presentan sistemáticamente cuando se programa en código de máquina un sistema de conmutación de bancos como el del +2. Si a usted le ocurre lo mismo, compruebe que su pila no está siendo conmutada durante las interrupciones, y que su rutina de interrupción está siempre donde debe estar (es conveniente inhibir las interrupciones durante las operaciones de paginación). También es recomendable que guarde una copia del registro de bancos actual en algún lugar no paginado de la RAM, ya que la puerta es de sólo escritura. BASIC y el editor usan la variable de sistema BANK M.

El programa de código de máquina se puede grabar con toda facilidad:

```
SAVE "nombre" CODE 65268,4
```

---

No hay ninguna manera de grabar el programa de forma que se ejecute automáticamente a sí mismo una vez que haya sido cargado. No obstante, esto se remedia empleando un pequeño programa de BASIC:

```
10 LOAD "" CODE 65268,4  
20 PRINT USR 65268
```

que debe ser grabado en la cinta inmediatamente antes que el código de máquina. El procedimiento sería grabar este programa con la orden:

```
SAVE "cargador" LINE 0
```

y luego grabar el código de máquina con:

```
SAVE "codigo m" CODE 65268,4
```

Hecho esto, se puede ejecutar el código de máquina desde BASIC con sólo dar la orden:

```
LOAD "cargador"
```

que carga y ejecuta automáticamente el programa de BASIC, el cual a su vez carga y ejecuta el código de máquina.





## Parte 27

# Juego de caracteres del Spectrum

Temas tratados:

Códigos de control

Caracteres

Nemónicos de ensamblador en el Z80
















La tabla siguiente da la lista completa del juego de caracteres del Spectrum, con sus códigos en versión decimal y hexadecimal. Por otra parte, si se considera esos códigos como instrucciones de código de máquina del Z80, entonces las columnas de la derecha dan los nemónicos correspondientes en lenguaje ensamblador. Tenga en cuenta que algunas instrucciones del Z80 empiezan por CBh o EDh; éstas aparecen en las dos columnas de la derecha.

Cuando un carácter es distinto en las dos versiones de BASIC (48k y 128k), el carácter correspondiente a 48 BASIC se escribe entre paréntesis a continuación del de 128 BASIC.

Código	Carácter	Hex	Ensamblador	Tras CBh	Tras EDh
0	no utilizado	00	nop	ob	nc
1	no utilizado	01	ld bc,NN	rlc c	
2	no utilizado	02	ld(bc),a	rlc d	
3	no utilizado	03	inc bc	rlc e	
4	no utilizado	04	inc b	rlc h	
5	no utilizado	05	dec b	rlc l	
6	coma de PRINT	06	ld b,N	rlc(hl)	
7	EDIT	07	rlca	rlc a	
8	cursor a la izquierda, ←	08	ex af,af	rrc b	
9	cursor a la derecha, →	09	add hl,bc	rrc c	
10	cursor abajo, ↓	0A	ld a,(bc)	rrc d	
11	cursor arriba, ↑	0B	dec bc	rrc e	
12	BORR	0C	inc c	rrc h	
13	INTRO	0D	dec c	rrc l	
14	número	0E	ld c,N	rrc (hl)	
15	no utilizado	0F	rrca	rrc a	
16	control de INK	10	djnz DIS	rl b	
17	control de PAPER	11	ld de,NN	rl c	
18	control de FLASH	12	ld (de),a	rl d	
19	control de BRIGHT	13	inc de	rl e	

Código	Carácter	Hex	Ensamblador	Tras CBh	Tras EDh
20	control de INVERSE	14	inc d	rl h	
21	control de OVER	15	dec d	rl l	
22	control de AT	16	ld d,N	rl (hl)	
23	control de TAB	17	rla	rl a	
24	no utilizado	18	jr DIS	rr b	
25	no utilizado	19	add hl,de	rr c	
26	no utilizado	1A	ld a,(de)	rr d	
27	no utilizado	1B	dec de	rr e	
28	no utilizado	1C	inc e	rr h	
29	no utilizado	1D	dec e	rr l	
30	no utilizado	1E	ld e,N	rr (hl)	
31	no utilizado	1F	rra	rr a	
32	espacio	20	jr nz,DIS	sla b	
33	!	21	ld hl,NN	sla c	
34	"	22	ld(NN),hl	sla d	
35	#	23	inc hl	sla e	
36	\$	24	inc h	sla h	
37	%	25	dec h	sla l	
38	&	26	ld h,N	sla (hl)	
39	'	27	daa	sla a	
40	(	28	jr z,DIS	sra b	
41	)	29	add hl,hl	sra c	
42	*	2A	ld hl,(NN)	sra d	
43	+	2B	dec hl	sra e	
44	,	2C	inc l	sra h	
45	-	2D	dec l	sra l	
46	.	2E	ld l,N	sra (hl)	
47	/	2F	cpl	sra a	
48	0	30	jr nc,DIS		
49	1	31	ld sp,NN		
50	2	32	ld (NN),a		
51	3	33	inc sp		
52	4	34	inc (hl)		
53	5	35	dec (hl)		
54	6	36	ld (hl),N		
55	7	37	scf		
56	8	38	jr c,DIS	srl b	
57	9	39	add hl,sp	srl c	
58	:	3A	ld a,(NN)	srl d	
59	;	3B	dec sp	srl e	
60	<	3C	inc a	srl h	

Código	Carácter	Hex	Ensamblador	Tras CBh	Tras EDh
61	=	3D	dec a	srl l	
62	>	3E	ld a,N	srl (hl)	
63	?	3F	ccf	srl a	
64	@	40	ld b,b	bit 0,b	in b,(c)
65	A	41	ld b,c	bit 0,c	out (c),b
66	B	42	ld b,d	bit 0,d	sbc hl,bc
67	C	43	ld b,e	bit 0,e	ld (NN),bc
68	D	44	ld b,h	bit 0,h	neg
69	E	45	ld b,l	bit 0,l	retn
70	F	46	ld b,(hl)	bit 0,(hl)	im 0
71	G	47	ld b,a	bit 0,a	ld i,a
72	H	48	ld c,b	bit 1,b	in c,(c)
73	I	49	ld c,c	bit 1,c	out (c),c
74	J	4A	ld c,d	bit 1,d	adc hl,bc
75	K	4B	ld c,e	bit 1,e	ld bc,(NN)
76	L	4C	ld c,h	bit 1,h	
77	M	4D	ld c,l	bit 1,l	reti
78	N	4E	ld c,(hl)	bit 1,(hl)	
79	O	4F	ld c,a	bit 1,a	ld r,a
80	P	50	ld d,b	bit 2,b	in d,(c)
81	Q	51	ld d,c	bit 2,c	out (c),d
82	R	52	ld d,d	bit 2,d	sbc hl,de
83	S	53	ld d,e	bit 2,e	ld (NN),de
84	T	54	ld d,h	bit 2,h	
85	U	55	ld d,l	bit 2,l	
86	V	56	ld d,(hl)	bit 2,(hl)	im 1
87	W	57	ld d,a	bit 2,a	ld a,i
88	X	58	ld e,b	bit 3,b	in e,(c)
89	Y	59	ld e,c	bit 3,c	out (c),e
90	Z	5A	ld e,d	bit 3,d	adc hl,de
91	i	5B	ld e,e	bit 3,e	ld de,(NN)
92	Ñ	5C	ld e,h	bit 3,h	
93	¿	5D	ld e,l	bit 3,l	
94	↑	5E	ld e,(hl)	bit 3,(hl)	im 2
95	—	5F	ld e,a	bit 3,a	ld a,r
96	Pt	60	ld h,b	bit 4,b	in h,(c)
97	a	61	ld h,c	bit 4,c	out (c),h
98	b	62	ld h,d	bit 4,d	sbc hl,hl
99	c	63	ld h,e	bit 4,e	ld (NN),hl
100	d	64	ld h,h	bit 4,h	
101	e	65	ld h,l	bit 4,l	

Código	Carácter	Hex	Ensamblador	Tras CBh	Tras EDh
102	f	66	ld h,(hl)	bit 4,(hl)	
103	g	67	ld h,a	bit 4,a	rrd
104	h	68	ld l,b	bit 5,b	in l,(c)
105	i	69	ld l,c	bit 5,c	out (C),l
106	j	6A	ld l,d	bit 5,d	adc hl,hl
107	k	6B	ld l,e	bit 5,e	ld hl,(NN)
108	l	6C	ld l,h	bit 5,h	
109	m	6D	ld l,l	bit 5,l	
110	n	6E	ld l,(hl)	bit 5,(hl)	
111	o	6F	ld l,a	bit 5,a	rld
112	p	70	ld (hl),b	bit 6,b	in f,(c)
113	q	71	ld (hl),c	bit 6,c	
114	r	72	ld (hl),d	bit 6,d	sbc hl,sp
115	s	73	ld (hl),e	bit 6,e	ld (NN),sp
116	t	74	ld (hl),h	bit 6,h	
117	u	75	ld (hl),l	bit 6,l	
118	v	76	halt	bit 6,(hl)	
119	w	77	ld (hl),a	bit 6,a	
120	x	78	ld a,b	bit 7,b	in a,(c)
121	y	79	ld a,c	bit 7,c	out (c),a
122	z	7A	ld a,d	bit 7,d	adc hl,sp
123	{	7B	ld a,e	bit 7,e	ld sp,(NN)
124	ñ	7C	ld a,h	bit 7,h	
125	}	7D	ld a,l	bit 7,l	
126	~	7E	ld a,(hl)	bit 7,(hl)	
127	©	7F	ld a,a	bit 7,a	
128		80	add a,b	res 0,b	
129		81	add a,c	res 0,c	
130		82	add a,d	res 0,d	
131		83	add a,e	res 0,e	
132		84	add a,h	res 0,h	
133		85	add a,l	res 0,l	
134		86	add a,(hl)	res 0,(hl)	
135		87	add a,a	res 0,a	
136		88	adc a,b	res 1,b	
137		89	adc a,c	res 1,c	
138		8A	adc a,d	res 1,d	
139		8B	adc a,e	res 1,e	
140		8C	adc a,h	res 1,h	
141		8D	adc a,l	res 1,l	
142		8E	adc a,(hl)	res 1,(hl)	

Código	Carácter	Hex	Ensamblador	Tras CBh	Tras EDh
143	■	8F	adc a,a	res 1,a	
144	(a)	90	sub b	res 2,b	
145	(b)	91	sub c	res 2,c	
146	(c)	92	sub d	res 2,d	
147	(d)	93	sub e	res 2,e	
148	(e)	94	sub h	res 2,h	
149	(f)	95	sub l	res 2,l	
150	(g)	96	sub (hl)	res 2,(hl)	
151	(h)	97	sub a	res 2,a	
152	(i)	98	sbc a,b	res 3,b	
153	(j)	99	sbc a,c	res 3,c	
154	(k)	9A	sbc a,d	res 3,d	
155	(l)	9B	sbc a,e	res 3,e	
156	(m)	9C	sbc a,h	res 3,h	
157	(n)	9D	sbc a,l	res 3,l	
158	(o)	9E	sbc a,(hl)	res 3,(hl)	
159	(p)	9F	sbc a,a	res 3,a	
160	(q)	A0	and b	res 4,b	ldi
161	(r)	A1	and c	res 4,c	cpi
162	(s)	A2	and d	res 4,d	ini
163	SPECTRUM (t)	A3	and e	res 4,e	outi
164	PLAY (u) ┘	A4	and h	res 4,h	
165	RND	A5	and l	res 4,l	
166	INKEY\$	A6	and (hl)	res 4,(hl)	
167	PI	A7	and a	res 4,a	
168	FN	A8	xor b	res 5,b	ldd
169	POINT	A9	xor c	res 5,c	cpd
170	SCREEN\$	AA	xor d	res 5,d	ind
171	ATTR	AB	xor e	res 5,e	outd
172	AT	AC	xor h	res 5,h	
173	TAB	AD	xor l	res 5,l	
174	VAL\$	AE	xor (hl)	res 5,(hl)	
175	CODE	AF	xor a	res 5,a	
176	VAL	B0	or b	res 6,b	ldir
177	LEN	B1	or c	res 6,c	cpir
178	SIN	B2	or d	res 6,d	inir
179	COS	B3	or e	res 6,e	otir
180	TAN	B4	or h	res 6,h	
181	ASN	B5	or l	res 6,l	
182	ACS	B6	or (hl)	res 6,(hl)	
183	ATN	B7	or a	res 6,a	

Código	Carácter	Hex	Ensamblador	Tras CBh	Tras EDh
184	LN	B8	cp b	res 7,b	lddr
185	EXP	B9	cp c	res 7,c	cpdr
186	INT	BA	cp d	res 7,d	indr
187	SQR	BB	cp e	res 7,e	otdr
188	SGN	BC	cp h	res 7,h	
189	ABS	BD	cp l	res 7,l	
190	PEEK	BE	cp (hl)	res 7,(hl)	
191	IN	BF	cp a	res 7,a	
192	USR	C0	ret nz	set 0,b	
193	STR\$	C1	pop bc	set 0,c	
194	CHR\$	C2	jp nz,NN	set 0,d	
195	NOT	C3	jp NN	set 0,e	
196	BIN	C4	call nz,NN	set 0,h	
197	OR	C5	push bc	set 0,l	
198	AND	C6	add a,N	set 0,(hl)	
199	<=	C7	rst 0	set 0,a	
200	>=	C8	ret z	set 1,b	
201	<>	C9	ret	set 1,c	
202	LINE	CA	jp z,NN	set 1,d	
203	THEN	CB		set 1,e	
204	TO	CC	call z,NN	set 1,h	
205	STEP	CD	call NN	set 1,l	
206	DEF FN	CE	adc a,N	set 1,(hl)	
207	CAT	CF	rst 8	set 1,a	
208	FORMAT	D0	pop de	set 2,b	
209	MOVE	D1	pop de	set 2,c	
210	ERASE	D2	jp nc,NN	set 2,d	
211	OPEN #	D3	out (N),a	set 2,e	
212	CLOSE #	D4	call nc,NN	set 2,h	
213	MERGE	D5	push de	set 2,l	
214	VERIFY	D6	sub N	set 2,(hl)	
215	BEEP	D7	rst 16	set 2,a	
216	CIRCLE	D8	ret c	set 3,b	
217	INK	D9	exx	set 3,c	
218	PAPER	DA	jp c,NN	set 3,d	
219	FLASH	DB	in a,(N)	set 3,e	
220	BRIGHT	DC	call c,NN	set 3,h	
221	INVERSE	DD	Prefijo de instrucciones que afectan a ix	set 3,l	
222	OVER	DE	sbc a,N	set 3,(hl)	

Código	Carácter	Hex	Ensamblador	Tras CBh	Tras EDh
223	OUT	DF	rst 24	set 3,a	
224	LPRINT	E0	ret po	set 4,b	
225	LLIST	E1	pop hl	set 4,c	
226	STOP	E2	jp po,NN	set 4,d	
227	READ	E3	ex (sp),hl	set 4,e	
228	DATA	E4	call po,NN	set 4,h	
229	RESTORE	E5	push hl	set 4,l	
230	NEW	E6	and N	set 4,(hl)	
231	BORDER	E7	rst 32	set 4,a	
232	CONTINUE	E8	ret pe	set 5,b	
233	DIM	E9	jp (hl)	set 5,c	
234	REM	EA	jp pe,NN	set 5,d	
235	FOR	EB	ex de,hl	set 5,e	
236	GOTO	EC	call pe,NN	set 5,h	
237	GO SUB	ED		set 5,l	
238	INPUT	EE	xor N	set 5,(hl)	
239	LOAD	EF	rst 40	set 5,a	
240	LIST	F0	ret p	set 6,b	
241	LET	F1	pop af	set 6,c	
242	PAUSE	F2	jp p,NN	set 6,d	
243	NEXT	F3	di	set 6,e	
244	POKE	F4	call p,NN	set 6,h	
245	PRINT	F5	push af	set 6,l	
246	PLOT	F6	or N	set 6,(hl)	
247	RUN	F7	rst 48	set 6,a	
248	SAVE	F8	ret m	set 7,b	
249	RANDOMIZE	F9	ld sp,hl	set 7,c	
250	IF	FA	jp m,NN	set 7,d	
251	CLS	FB	ei	set 7,e	
252	DRAW	FC	call m,NN	set 7,h	
253	CLEAR	FD	Prefijo de instrucciones que afectan a iy	set 7,l	
254	RETURN	FE	cp N	set 7,(hl)	
255	COPY	FF	rst 56	set 7,a	



1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

---

## Parte 28

# Mensajes

Temas tratados:

Informes emitidos por la pantalla  
Mensajes de error  
Mensajes  
CONTINUE

En la última línea de la pantalla aparece un mensaje cada vez que el +2 termina de ejecutar alguna orden en BASIC. Su misión es explicar a qué se debe la detención, ya sea por razones naturales o porque ha habido un error.

El informe consiste en un código (número o letra), un breve texto que explica lo que ha ocurrido y el número de línea (y el número de sentencia dentro de esa línea) en la que se ha detenido. Cuando se trata de una orden directa, el número de línea que muestra es el 0. Dentro de una línea, la sentencia 1 es la que se encuentra al principio, la 2 es la que va a continuación del primer signo de dos puntos (o de THEN), etc.

El comportamiento de CONTINUE depende en gran medida de los mensajes. Normalmente CONTINUE va a la línea y sentencia indicadas en el último informe, pero hay excepciones en los mensajes de códigos 0, 9 y D.

La siguiente tabla muestra todos los informes e indica en qué circunstancias pueden aparecer; con esta información se puede consultar la Parte 30. Por ejemplo, el error A Invalid argument (Argumento no válido) puede presentarse en SQR, IN, ACS y ASN, y la descripción que se da de estas palabras clave en la Parte 30 indica exactamente qué argumentos no son válidos para ellas.

---

Código	Texto y significado	Situación
0	OK  Tarea terminada con éxito, o intento de saltar a un número de línea mayor que cualquiera de los existentes. Este mensaje no cambia la línea ni la sentencia a la que salta CONTINUE.	Cualquiera

---

<b>Código</b>	<b>Texto y significado</b>	<b>Situación</b>
1	<p>NEXT without FOR</p> <p>NEXT sin FOR</p> <p>La variable de control no existe (no ha sido establecida por una sentencia FOR), pero hay una variable ordinaria con el mismo nombre.</p>	NEXT
2	<p>Variable not found</p> <p>Variable no encontrada</p> <p>En el caso de una variable sencilla, ocurre si se intenta utilizarla antes de definirla (con LET, READ o INPUT), o de cargarla desde la cinta, o de definirla en una sentencia FOR. En el caso de una variable indexada, ocurre si se intenta utilizarla antes de dimensionar la matriz (DIM) o de cargarla desde la cinta.</p>	Cualquiera
3	<p>Subscript wrong</p> <p>Subíndice incorrecto</p> <p>Un subíndice es mayor que la dimensión de la matriz, o el número de subíndices no se corresponde con el de dimensiones de la matriz. Si el subíndice es negativo o mayor que 65535, se produce el error B.</p>	Variables indexadas, subcadenas
4	<p>Out of memory</p> <p>Memoria agotada</p> <p>No hay espacio suficiente en la memoria del ordenador para lo que se pretende hacer. Si el ordenador parece ser incapaz de salir de esta situación, tendrá que borrar la línea de órdenes actual pulsando <b>BORR</b> y luego borrar una o dos líneas del programa (con la intención de reintroducirlas más tarde).</p>	LET, INPUT, FOR, DIM, GO SUB, LOAD, MERGE. A veces durante la evaluación de una expresión

Código	Texto y significado	Situación
5	Out of screen Fuera de la pantalla Una sentencia INPUT ha tratado de generar más de 23 líneas en la pantalla inferior. También se produce con PRINT AT 22,xx.	INPUT, PRINT AT
6	Number too big Número demasiado grande Los cálculos han desembocado en un número superior a aproximadamente $10^{38}$ .	Cualquier cálculo aritmético
7	RETURN without GO SUB RETURN sin GO SUB Hay un RETURN al que no corresponde ningún GO SUB.	RETURN
8	End of file Fin de fichero.	Operaciones con microunidades, etc.
9	STOP statement Sentencia STOP Después de este mensaje, CONTINUE no repetirá la sentencia STOP, sino que continuará a partir de la sentencia siguiente.	STOP
A	Invalid argument Argumento no válido El argumento que se ha puesto en una función no es adecuado.	SQR, LN, ASN, USR (con argumento literal)

Código	Texto y significado	Situación
B	<p>Interger out of range</p> <p>Entero fuera de margen</p> <p>Cuando una sentencia requiere un entero, el argumento de punto flotante es redondeado al entero más próximo. Este error se produce si el resultado de la operación de redondeo es inadecuado.</p> <p>En relación con las matrices, véase también el error 3.</p>	<p>RUN, RANDOMIZE, POKE, DIM, GOTO, GO SUB, LIST, LLIST, PAUSE, PLOT, CHR\$, PEEK, USR (con argumento numérico)</p> <p>Acceso a matrices</p>
C	<p>Nonsense in BASIC</p> <p>Absurdo en BASIC</p> <p>El texto del argumento (literal) no constituye una expresión válida. También se produce cuando el argumento de una función o de una orden es escandalosamente erróneo.</p>	VAL, VAL\$
D	<p>BREAK – CONT repeats</p> <p>BREAK – CONT para repetir</p> <p>Se ha pulsado <b>BREAK</b> durante alguna operación con algún periférico. El comportamiento de CONTINUE tras este informe es normal en el sentido de que repite la sentencia. Compárelo con el informe L.</p>	<p>LOAD, SAVE, VERIFY, MERGE.</p> <p>También cuando el ordenador pregunta scroll? y se responde pulsando N, <b>BREAK</b> o la barra espaciadora</p>
E	<p>Out of DATA</p> <p>Datos agotados</p> <p>Se ha tratado de leer más allá del final de la lista DATA.</p>	READ
F	<p>Invalid file name</p> <p>Nombre de fichero no válido</p> <p>El nombre especificado tras SAVE consta de más de 10 caracteres o es la cadena vacía.</p>	SAVE

Código	Texto y significado	Situación
G	No room for line No queda espacio para la línea No queda espacio en la memoria para acomodar la nueva línea de programa.	Introducción de una línea de programa
H	STOP in INPUT STOP en INPUT Algún dato introducido en INPUT comenzó con STOP. A diferencia del caso del informe 9, después de H la orden CONTINUE se comporta normalmente, repitiendo la sentencia INPUT.	INPUT
I	FOR without NEXT FOR sin NEXT Se ha establecido un bucle que no tiene que ser ejecutado ninguna vez (por ejemplo, FOR n=1 TO 0) y BASIC no ha encontrado el NEXT correspondiente.	FOR
J	Invalid I/O device Dispositivo de E/S no válido	Operaciones con microunidades, etc.
K	Invalid colour Color no válido El número especificado no es correcto.	INK, PAPER, BORDE, FLASH, BRIGHT, INVERSE, OVER; también después de uno de los correspondientes caracteres de control
L	BREAK into program Programa interrumpido Se ha pulsado <b>BREAK</b> . La posible pulsación de esta tecla se explora entre cada dos sentencias. Los nú-	Cualquiera

---

Código	Texto y significado	Situación
	meros de línea y sentencia del informe se refieren a la sentencia anterior a la pulsación de <b>BREAK</b> , pero <b>CONTINUE</b> pasa a la sentencia siguiente, de forma que no repite ninguna.	
M	RAMTOP no good RAMTOP no válido  El número especificado para RAMTOP es demasiado grande o demasiado pequeño.	CLEAR; posiblemente también en RUN
N	Statement lost Sentencia perdida  Salto a una sentencia que ya no existe.	RETURN, NEXT, CONTINUE
O	Invalid Stream Canal no válido	Operaciones con microunidades, etc.
P	FN without DEF FN sin DEF  Se ha intentado usar una función de usuario que no está definida en el programa.	FN7
Q	Parameter error Error en parámetro  Número de argumentos incorrecto, o bien uno de ellos es de tipo incorrecto (cadena en vez de número, o vice versa).	FN

---

Código	Texto y significado	Situación
R	Tape loading error Error de carga de cinta  Se ha encontrado el fichero en la cinta, pero por alguna razón no ha sido posible leerlo o verificarlo.	VERIFY, LOAD, MERGE
a	MERGE error Error en MERGE  MERGE ! no ha podido ser ejecutada por alguna razón (tamaño o tipo de fichero incorrectos).	MERGE !
b	Wrong file type Fichero de tipo incorrecto  Se ha especificado un fichero de tipo inadecuado para una operación con el disco de silicio; por ejemplo, un fichero CODE en LOAD ! " <i>nombre</i> ".	MERGE !, LOAD !
c	CODE error Error en CODE  El tamaño del fichero encontrado es tal que se sobrepasaría el límite de la memoria.	LOAD ! <i>fichero</i> CODE
d	Too many brackets Demasiados paréntesis  Demasiados paréntesis en una frase repetida en uno de los argumentos.	PLAY
e	File already exists Ya existe el fichero  Ya existe en el disco de silicio un fichero con ese mismo nombre.	SAVE !



---

---

Código	Texto y significado	Situación
f	Invalid name Nombre no válido El nombre de fichero especificado es la cadena vacía o tiene más de diez caracteres.	ERASE I
h	File does not exist No existe ese fichero En el disco de silicio no hay ningún fichero con ese nombre.	LOAD I, ERASE I
i	Invalid device Dispositivo no válido El nombre del dispositivo que se ha puesto en la orden FORMAT no existe o no corresponde a un dispositivo físico.	FORMAT
j	Invalid baud rate Velocidad de transmisión no válida Se ha especificado velocidad cero para la puerta RS232.	FORMAT
k	Invalid note name Nombre de nota no válido PLAY ha encontrado una nota o una orden que no reconoce, o una orden escrita en minúsculas.	PLAY
l	Number too big Número demasiado grande El parámetro de una orden de PLAY es un orden de magnitud demasiado grande.	PLAY

---