SUBRUTINAS DE LA ROM

CAPITULO I

==========

En esta serie de artículos vamos a estudiar las subrutinas de la ROM que componen el Sistema Operativo de nuestro ordenador. Pero primero pasaremos revista a algunos conceptos básicos sobre Assembler, y al microprocesador Z-80.

==========

Jesús Alonso Rodriguez

INTRODUCCION:

Tal vez algun lector se pregunte qué utilidad tiene estudiar las subrutinas de la ROM cuando, de hecho, el ordenador sabe manejarlas muy bien en respuesta a nuestras órdenes en Basic.

En principio, se puede contestar que para programar correctamente, es necesario tener el mayor conocimiento posible del ordenador con el que se trabaja, por lo que toda informacion es poca. Pero hay una razon mas importante, si se conocen bien las subrutinas de la ROM y la forma de usarlas, se pueden hacer cosas que el Basic no permite.

Por otro lado, los programadores de código máquina encontrarán en la ROM una serie de rutinas ya hechas, que les permitiran ahorrarse trabajo, ya que en muchos casos, se podran ahorrar la construccion de una rutina propia, utilizando una de la ROM.

Finalmente, para quienes se estén introduciendo en el mundo del Código Máquina, el estudio de la ROM resultará muy interesante para aprender y depurar técnicas de programacion.

Esta serie pretende dirigirse tanto a aquellos lectores con conocimientos de programacion en Assembler, como a quienes tan solo se interesan por el Basic. Confiamos en que ambos encontrarán útiles los siguientes capítulos.

ESTRUCTURA DE LA SERIE

En cada capítulo trataremos una o varias subrutinas de la ROM, llevando un orden lógico que nos permita aproximarnos con facilidad al conocimiento del Sistema Operativo. Dado que la ROM del Spectrum se organiza como un gran número de subrutinas relacionadas entre sí, a veces será necesario hacer referencia a rutinas no estudiadas todavía, en estos casos se hará de forma que el lector comprenda la funcion de la rutina, aun cuando su estudio detallado se haga en un capítulo posterior.

Para cada rutina habrá una serie de apartados fijos, que serán: Funcion, Descripcion, Entradas alternativas, Condiciones de entrada, Registros empleados, Condiciones de salida y Posibles utilidades. Complementados con los que fueran necesarios en cada caso, así como figuras, tablas o listados, cuando se requieran para una mejor comprension.

En cuanto a la nomenclatura, utilizaremos la propia del assembler, los nombres de las variables serán los utilizados en el manual, y las etiquetas las correspondientes al código fuente del Sistema.

Los números se representarán en hexadecimal, indicándose con una "h." al final del número; a continuacion irá ese mismo número en decimal, con una "d." para indicarlo; en este segundo caso, se utilizará la notacion americana de "coma" para separar los "millares" y "punto" para los "decimales".

Por ejemplo: la direccion de memoria "cuatro mil quinientos treinta y cinco" (correspondiente a la rutina del comando NEW) se representará: 11B7h. (4,535d.)

ASSEMBLER Y CODIGO MAQUINA

Si descendemos al nivel de máquina (forma de trabajar del microprocesador) las únicas instrucciones que este entiende son números que ocupan las distintas posiciones de memoria. El Z-80 ejecuta cerca de 700 instrucciones que se codifican utilizando úno, dos o tres números comprendidos entre "cero" y FFh. (255d.), a estos números se les denomina "Código Máquina" porque es el único código que el microprocesador es capaz de entender de forma directa.

Para los humanos, sería imposible recordar 700 números y las operaciones que cada uno realiza, por lo que se han inventado los "lenguajes de programacion". Los lenguajes de programacion son sistemas de códigos intermedios entre el lenguaje humano y el de la máquina. Se dice que un lenguaje es de mas alto nivel cuanto mas se acerca al lenguaje humano, y de mas bajo nivel, cuanto mas se acerca al lenguaje de la máquina. El Basic es un lenguaje de alto nivel, mientras que el Assembler es el lenguaje de mas bajo nivel posible.

El Assembler es, simplemente, la traduccion de los números que constituyen el Código Máquina a unas palabras simbólicas llamadas "nemónicos" que a los humanos nos son mas fáciles de recordar. Por ejemplo, en Assembler, para indicarle al microprocesador que cargue el registro A con el contenido de la direccion apuntada por el registro doble HL, escribiríamos:

```
LD A,(HL)
```

El nemónico LD es abreviatura de "LoaD" (cargar) y los paréntesis que rodean a HL significan "la direccion apuntada por...". Esta instruccion se ensambla en Código Máquina como: 7Eh. (126d.). A la instruccion en Assembler se le denomina "Código Fuente" y a su codificacion, "Código Objeto". Existen programas capaces de traducir el Código Fuente a Código Objeto, estos programas se denominan "Ensambladores".

BREVE DESCRIPCION DEL Z-80:

Como el lector habrá deducido ya, la programacion en Assembler es distinta para cada microprocesador, de hecho, cada uno tiene su propio Assembler. Nosotros vamos a utilizar, logicamente, el Assembler del Z-80, por lo que será util tener un cierto conocimiento del mismo.

El microprocesador Z-80 se compone de una serie de registros que son como posiciones de memoria, pero internas al microprocesador, y con los cuales realiza este sus operaciones, tiene tambien una unidad logico-aritmética y toda la circuiteria necesaria para realizar la lectura y decodificacion de memoria y controlar las operaciones que se realizan internamente.

Permite dos modos de interrupcion: "Interrupcion no enmascarable" (NMI) con vector de interrupcion fijo (no se usa en el Spectrum) e "Interrupcion Enmascarable" (MI) con tres modos de interrupcion posibles y con posibilidad de definirle el vector de interrupcion (la MI se utiliza en el Spectrum para leer el teclado y actualizar el reloj de tiempo real). Se denomina "vector de interrupcion" a la direccion de memoria donde salta el microprocesador cuando recibe una peticion de interrupcion.

Por otro lado, el Z-80 es capaz de regenerar la RAM dinámica sin que el programador tenga que preocuparse por ello.

El Z-80 tiene 16 registros de 8 bits, que se pueden agrupar de 2 en 2 para formar registros de 16 bits, y 5 registros fijos de 16 bits con utilidades especiales. Estos últimos son:

"PC" o contador de programa
"SP" o puntero de pila
"IX" o índice x
"IY" o índice y
"IR" o vector de interrupcion y registro de regeneracion

En cuanto a los registros de 8 bits, se dividen en dos grupos de 8 registros alternativos cada uno, y se denominan: A, B, C, D, E, F, H, L, A', B', C', D', E', F', H' y L'.

El registro A se denomina "acumulador" porque es el que recibe los resultados de la mayoría de operaciones que realiza el Z-80. El registro F es el indicador de estado, y está compuesto por los "FLAGS" o indicadores de "cero", "paridad", "overflow", "suma/resta", etc. que se utilizan implicitamente en los saltos condicionales.

Los registros se pueden agrupar de la siguiente forma: AF, BC, DE, HL, A'F', B'C', D'E' y H'L'. A los cuatro últimos los denominaremos: AF', BC', DE' y HL'. El propio microprocesador se encarga de considerar los registros como sencillos o dobles, en funcion de la instruccion a la que esté respondiendo.

El conjunto de instrucciones del Z-80 es muy amplio y abarca: carga de registros y posiciones de memoria, búsqueda, intercambio de registros, trasferencia de bloques, operaciones aritméticas y lógicas, operaciones de entrada/salida, control CPU, saltos y bifurcaciones, etc. No podemos entrar aquí a analizar cada una detalladamente, ya que eso requeriría otra serie, pero existe abundante bibliografía sobre el tema.

LOS REINICIOS (RST) DE PAGINA CERO:

Existen una serie de instrucciones que fuerzan al microprocesador a saltar a una subrutina que empieza en una de las primeras direcciones de memoria, cada una de estas instrucciones solo ocupan un byte (frente a los tres de una llamada a subrutina) por lo que se utilizan para las subrutinas de uso mas frecuente. Se llaman: RST 0, RST 8, RST 10, RST 18, RST 20, RST 28, RST 30 y RST 38. En ellas los programadores de Sinclair han colocado las siguientes rutinas:

RST 0 : Inicializacion (START)
RST 8 : Reinicio de error (ERROR-1)
RST 10: Rutina general de salida (PRINT-OUT)
RST 18: Recoge un caracter (GET-CHAR)
RST 20: Recoge siguiente caracter (NEXT-CHAR)
RST 28: Entrada al calculador (FP-CALC)
RST 30: Hacer BC espacios (BC-SPACES)
RST 38: Rutina de la interrupcion enmascarable (MASK-INT)

Empezaremos nuestro estudio de la ROM por estas rutinas, y concretamente, en el próximo capítulo veremos las rutinas de inicializacion (START).

---------- o ----------

SUBRUTINAS DE LA ROM

CAPITULO II

==========

Una vez introducidos en materia, en este capítulo veremos la primera de las rutinas de la ROM. Se trata de la inicializacion del sistema que se ejecuta de forma automática cada vez que se conecta el ordenador.

==========

Jesús Alonso Rodriguez

FUNCION

El microprocesador comienza a trabajar desde el mismo momento que se conecta el ordenador, el circuito de AUTO-RESET se encarga de que en ese instante todos los registros estén a cero, incluido el PC (contador de programa), por lo que la rutina de inicializacion arranca desde la direccion cero.

La finalidad de esta rutina es comprobar la memoria disponible, y fijar una serie de variables y parámetros que necesitará la rutina de ejecucion principal, para empezar a trabajar.

DESCRIPCION

La mayor parte de la rutina de inicializacion se utiliza tambien cuando se ejecuta un comando NEW, esta parte se denomina "START/NEW" y comienza en la direccion 11CBh. (4,555d.). Esta rutina se comporta de forma distinta, en funcion del contenido del registro A, ("00" para START, "FF" para NEW).

Estudiaremos primero la rutina START, luego la rutina NEW, y finalmente, la START/NEW.

RUTINA "START"

Esta rutina arranca desde la direccion cero, y realiza las siguientes funciones:

- .- Desactiva la interrupcion enmascarable (MI)
- .- Pone a "00" el registro A (Acumulador), que en este caso, se usa como un FLAG para indicar a la rutina START/NEW si se trata de una inicializacion completa (RESET) o de la ejecucion de un comando NEW.
- .- Pone a FFFFh. (65,535d.) el registro doble DE, este es el valor de la máxima RAM posible.
- .- Salta a la rutina START/NEW

RUTINA "NEW"

Se trata de la rutina correspondiente a la ejecucion del comando NEW, no se ejecuta al conectar el ordenador, pero sin embargo, se la puede considerar parte de la inicializacion del sistema, de hecho, hace el papel de un "WARM RESET" (Arranque en "caliente").

Su mision es borrar el programa Basic y sus variables, e inicializar el resto de las variables del sistema, excepto RAMTOP, P_RAMT, RASP, PIP y UDG; tampoco borra los Gráficos Definidos por el Usuario ni nada que se encuentre por encima de RAMTOP. Comprueba la memoria, y no altera el valor de RAMTOP a menos que encuentre un error en una posicion de memoria.

Comienza en la direccion 11B7h. (4,535d.) y realiza las siguientes funciones:

- .- Desactiva la interrupcion enmascarable (MI)
- .- Pone a "FF" el registro A, por medio de esto se diferencia en la rutina START/NEW desde donde se entra.
- .- Copia en el registro doble DE el valor de la variable RAMTOP
- .- Salva en los registros alternativos BC', DE' y HL' las variables P_RAMT, RASP, PIP y UDG.
- .- Continua en la rutina START/NEW.

RUTINA "START/NEW"

Esta rutina es comun para el comando NEW y la inicializacion (START), con las siguientes diferencias:

Desde Start: Registro A conteniendo "00"
Registro DE conteniendo "FFFF"

Desde NEW: Registro A conteniendo "FF"
Registro DE conteniendo el valor de RAMTOP

La rutina arranca en la direccion 11CBh. (4,555d.) y se divide en varias partes que realizan las siguientes funciones:

Posicion 11CB (START/NEW)

- .- Pone blanco el color del borde
- .- Espera 24T. veces, el tiempo T. es la cuarta parte de un ciclo de instruccion del Z-80 (esto se hace ejecutando seis NOPs)

Posicion 11DA (RAM-CHECK)

- .- Chequea toda la memoria RAM desde la direccion 3FFFh. (16,383d.) hasta la direccion apuntada por el contenido del registro DE.

  Aqui hay implícita una diferencia entre la entrada desde START o desde NEW, en el primer caso chequea toda la memoria posible; mientras que en el segundo caso, solo chequea hasta el contenido de la variable RAMTOP y si no detecta ningun error, no lo cambia. Esto se hace así para no deteriorar la forma de los gráficos definidos por el usuario, que el comando NEW no tiene que inicializar.

.- Si se entra desde START crea las variables P_RAMT, UDG, RASP y PIP con los valores:

P_RAMT: Direccion del último octeto de la memoria disponible.
UDG: Contenido de P_RAMT menos A7h. (167d.)
RASP: 40h. (64d.)
PIP: Cero

y copia el formato de los caracteres de la A a la U en el espacio entre UDG y P_RAMT.

.- Crea la variable RAMTOP. En el caso de START le dá el valor UDG menos 1. En el caso de NEW, le dá el valor que tenía, a no ser que detectara un error en la memoria, en cuyo caso, le dá el valor de la direccion del octeto anterior al error. (Esto resulta muy util para permitir el volcado de software que llevan a cabo algunos interfaces durante la inicializacion).

.- Crea la variable CHARS con el valor 3C00h. (15,360d.) que es la posicion donde están definidos los caracteres menos 100h. (256d.)

.- Almacena 3Eh. (62d.) en la direccion de RAMTOP.

.- Carga el registro SP (Puntero de pila de máquina) del microprocesador con el valor de RAMTOP menos 1

.- Crea la variable ERR_SP con el valor de RAMTOP menos 3

.- Activa el modo 1 de interrupcion

.- Carga el registro IY (uno de los índices) con la direccion de la variable ERR_NR

.- Activa la interrupcion enmascarable (MI), a partir de este momento, cada 20 milisegundos se actualiza el reloj de tiempo real y se produce la lectura del teclado.

.- Crea la variable CHANS con el valor ~~5C3A~~h. (~~23,610d.~~) (23734 5CB6

.- Pone valores iniciales al area que direcciona CHANS, estos valores son: F4, 09, A8, 10, 4B, F4, 09, C4, 15,53, 81, 0F, C4, 15, 52, F4, 09, C4, 15, 50, 80.

En principio hay 4 canales: "K" (Teclado y parte inferior de la pantalla), "S" (Parte superior de la pantalla), "R" (Area de trabajo) y "P" (Impresora). Para cada uno se usan 5 bytes, los dos primeros direccionan la rutina que hace el OUTPUT por ese canal, los dos segundos, la rutina que hace el INPUT, y el quinto es el código del caractér que dá nombre al canal. La forma es esta:

CANAL "K"
Output: PRINT-OUT: 09F4h. (2,548d.)
Input: KEY-INPUT: 10A8h. (4,264d.)
Código: K: 4Bh. (75d.)

```
CANAL "S"
 Output: PRINT-OUT: 09F4h. (2,548d.)
 Input:   REPORT-J: 15C4h. (5,572d)
 Código:          S: 53h. (83d.)

CANAL "R"
 Output:  ADD-CHAR: 0F81h. (3,969d.)
 Input:   REPORT-J: 15C4h. (5,572d.)
 Código:          R: 52h. (82d.)

CANAL "P"
 Output: PRINT-OUT: 09F4h. (2,548d.)
 Input:   REPORT-J: 15C4h. (5,572d.)
 Código:          P: 50h. (80d.)

MARCA DE FIN: 80h. (128d.)
```

.- Inicializa las variables DATADD, PROG, VARS, E_LINE, WORKSP, STKBOT y STKEND con los valores:

```
DATADD:   (CHANS) + 14h.
PROG:     (DATADD) + 1
VARS:     (DATADD) + 1
E_LINE:   (PROG) + 1
WORKSP:   (E_LINE) + 1
STKBOT:   (E_LINE) + 1
STKEND:   (E_LINE) + 1
```

.- Pone las marcas de FIN (80h.) antes del area de Programa y despues del area de Variables.
.- Inicializa las variables ATTR-P, ATTR-T, y BORDCR con el valor 38h. (56d.) correspondiente a: FLASH 0, BRIGHT 0, PAPER 7, INK 0.
.- Inicializa las variables REPDEL y REPPER con los valores 23h. y 05h.
.- Inicializa las variables KSTATE-0 y KSTATE-4 con el valor FFh.
.- Carga la variable STRMS con los valores: /01, 00,/06, 00,/0B, 00,/01,°00,/01,⁴00,/06,²00,/10,³00./
.- Activa el bit 1 de FLAG3 indicando "printer en uso"
.- Limpia el buffer de impresora (CALL CLEAR-PRB)
.- Inicializa la variable DF-SZ con el valor 02h.
.- Limpia la pantalla (CALL CLS)
.- Imprime el mensaje: "(c) 1982 Sinclair Research LTD" (CALL PO-MSG con A=0 y DE= 1538h.)
.- Activa bit 5 de TV-FLAG indicando que la parte baja de la pantalla requerirá ser limpiada.
.- Salta a la rutina de ejecucion principal (JR MAIN-1)

ENTRADAS ALTERNATIVAS:

En este caso, solo se puede entrar desde cero para una inicializacion total, o desde 11B7h. (4,535d.) para un NEW (RANDOMIZE USR 4535 tendrá el mismo efecto que NEW, y RANDOMIZE USR 0 tendrá el mismo efecto que desconectar y conectar el equipo, o que pulsar el RESET).

CONDICIONES DE ENTRADA:

En START y NEW no hay ninguna condicion de entrada, en START/NEW (direccion 11CBh. o 4,555d.) hay que entrar con la interrupcion desactivada y el registro A conteniendo "00" para un RESET total o "FF" para un NEW, asimismo, en caso de NEW, el registro DE deberá contener el valor de RAMTOP, y "FFFF" en el caso de un RESET total.

REGISTROS EMPLEADOS:

En este caso, se altera el contenido de todos los registros.

CONDICIONES DE SALIDA:

Se produce la inicializacion de la máquina, por lo que se pierde el control del programa, retornando a través de la "MAIN EXECUTION ROUTINE" (Rutina de Ejecucion Principal)

POSIBLES UTILIDADES:

En este caso, la utilidad mas evidente es la de inicializar el ordenador sin necesitar desconectarlo (para aquellos que no dispongan de RESET).

Hay que hacer incapié en que la forma de chequear la memoria, permite a los interfaces volcar su software durante la inicializacion. La rutina RAM-CHECK primero llena la memoria de arriba a abajo, y posteriormente, la lee de abajo a arriba. Donde lee un dato distinto del que há escrito, se detiene, y apunta la direccion inmediata inferior como valor de P_RAMT, por lo que el software colocado por el interface en la parte alta de la memoria, queda a salvo de borrados, incluso con NEW.

---------- o ----------

SUBRUTINAS DE LA ROM

CAPITULO III

==========

Una vez vistas las rutinas que permiten al sistema inicializarse y empezar a trabajar, veremos en este capítulo las restantes subrutinas de página cero.

==========

Jesús Alonso Rodriguez

RUTINAS DE PAGINA CERO:

Estas rutinas ocupan desde la posicion 0000 a la 028Dh. (653d.). En esta parte de la memoria se encuentran las rutinas correspondientes a los "reinicios" (Restart) de página cero y algunas subrutinas relacionadas con ellas, en la figura 1 se muestra un mapa de esta parte de la memoria.

Suele ser comun a todos los sistemas operativos usar las primeras posiciones de memoria para tablas de saltos, tratamiento de interrupciones, rutinas de inicializacion, etc. En general, aquellas rutinas básicas para atender al propio sistema operativo y al usuario.

En sistemas operativos no escritos para ROM, suelen encontrarse tambien en esta zona, las variables básicas tales como R_T_CLOCK (Reloj de tiempo real, en el Spectrum:"FRAMES"), Fecha, Hora, Tamaño de memoria, etc.

En el caso del Spectrum, y por estar su sistema escrito para ROM, las variables del sistema se encuentran por encima de la direccion 3FFFh. (16,383d.).

A continuacion, detallaremos todas las rutinas de esta zona.

START:

Posiciones 0000 a 0007h.

Se entra con RST 0

Esta rutina se trató ampliamente en el capítulo II dedicado a la inicializacion del sistema.

ERROR-1:

Posiciones 0008 a 000Fh. (8 a 15d.)

Se entra con RST 8 seguido de un literal con el código de error, menos uno (por ejemplo, para el error 2 "Variable not found" el literal deberá ser "1").

Por esta rutina sale siempre el intérprete de Basic, tanto si detecta un error, como si termina la ejecucion correctamente, ya que el mensaje "OK," es tratado como un mensaje mas de error.

La forma de entrar a esta rutina es mediante la instruccion "RST 8" (que se ensambla como CFh. 207d.) seguida de un literal, que es simplemente, un byte ensamblado a continuacion del código, y que indica el mensaje que hay que imprimir, por ejemplo, si se desea imprimir el mensaje "Variable not found" (código 2), la forma sería:

```
CF    RST 8      ;Llama rutina error
Ø1    DEFB +Ø1   ;Mensaje con codigo 2
```

>DESCRIPCION:

Posicion ØØØ8 (ERROR-1)

.- Pasa el valor de la variable CH-ADD a X-PTR, es la posicion donde se ha detectado el error (recuérdese que la terminacion correcta con mensaje "OK," es tratada como un error).

.- Continua en la rutina ERROR-2

Posicion Ø053h. 83d. (ERROR-2)

.- Carga en el registro "L" el código de error, definido en la posicion siguiente a la instruccion RST 8

.- Continua en la rutina ERROR-3

Posicion Ø055h. 85d. (ERROR-3)

.- Carga en la variable ERR-NR el código de error desde el registro "L".

.- Carga el registro "SP" con el contenido de la variable ERR-SP, con lo que se limpia la pila de máquina salvando solo el retorno del primer CALL que se hubiera hecho.

.- Salta con JP 16C5 a la rutina SET-STK que limpiará la pila del calculador y provocará un retorno a la direccion apuntada por el último dato de la pila de máquina, que será la direccion de retorno del primer CALL que se haya hecho, con lo que se vuelve al intérprete de Basic, que a su vez, se encargará de llamar a la rutina PO-MSG que será la que finalmente imprima el mensaje en pantalla.

Estas rutinas se estudiaran con detalle cuando se trate la rutina de ejecucion principal. El sistema operativo del Spectrum está lo suficientemente "enmarañado" como para que sea imposible seguirle la pista a cada rutina en particular, es mas facil comprenderlo todo cuando se tenga una vision global del sistema.

De momento, puede sernos util saber que podemos retornar al Basic desde C/M ejecutando un RST 8 seguido del código FFh. (255d.) para el mensaje "OK,". Esto nos permite retornar con éxito aun cuando nuestra rutina hubiera desordenado el stack de máquina, si bien se interrumpirá la ejecucion del programa Basic con el mensaje "OK,". Pruebe la siguiente rutina en código máquina:

```
E5    PUSH HL   ;Desordenamos el stack
C5    PUSH BC   ;metiendo numeros
D5    PUSH DE   ;sin significado.
CF    RST  8    ;Llamamos rutina de error
FF    DEFB #FF  ;Mensaje "Ø OK,"
```

Parece que el retorno es imposible porque hemos desordenado el stack, pero compruebe que si ejecuta "RANDOMIZE USR" a la direccion de comienzo, obtiene: "Ø OK, Ø:1"

Para quienes no dispongan de ensamblador, pueden ejecutar:

```
1Ø CLEAR 29999
2Ø FOR n=3ØØØØ TO 3ØØ4
3Ø READ a: POKE n,a
4Ø NEXT n
5Ø DATA 229,197,213,2Ø7,255
```

Y a continuacion: RANDOMIZE USR 3ØØØØ

PRINT-A-1:

Posiciones ØØ1Ø a ØØ12h. (16 a 18d.)

Se entra con RST 1Ø teniendo en el registro "A" el código del caracter a imprimir o un código de control.

Esta rutina efectua un salto a la rutina PRINT-A-2 situada en la direccion 15F2h. (5618d.) por lo que dá lo mismo hacer RST 1Ø que hacer CALL 15F2, pero en el primer caso solo se utiliza un byte. Esta es una de las rutinas de uso mas frecuente por el sistema operativo, y por esa razon se ha previsto la posibilidad de entrar con un "Restart" de página cero.

El funcionamiento se estudiará cuando se vea la rutina PRINT-A-2, pero podemos adelantar que se trata de una de las rutinas mas potentes del sistema, pudiendo manejar Tokens y códigos de control. De momento, puede hacer pruebas ejecutando RST 1Ø con diversos códigos en el acumulador; verá que con llamadas a RST 1Ø puede hacer todo lo que con PRINT.

Es importante señalar que RST 1Ø es capaz de dirigir la informacion a cualquier canal, por lo que es necesario abrir un canal antes de utilizarla, podemos anticipar que esto se hace colocando en el registro "A" el número de canal, y llamando a la rutina CHAN-OPEN mediante: CALL #16Ø1 que se ensambla: CD,Ø1,16h. (2Ø5,1,22d.)

GET-CHAR:

Posiciones ØØ18 a ØØ1Fh. (24 a 31d.)

Esta rutina comprueba el contenido de la posicion de memoria direccionada por la variable CH-ADD, si es un caracter imprimible retorna, si no incrementa la variable CH-ADD y repite la prueba hasta que encuentre un caracter imprimible.

Utiliza las subrutinas: GET-CHAR, TEST-CHAR, SKIP-OVER, NEXT-CHAR y CH-ADD+1

>DESCRIPCION:
Posicion ØØ18h. 24d. (GET-CHAR)
.- Carga en el registro "A" el contenido de la posicion direccionada por CH-ADD.
.- Continúa en TEST-CHAR

Posicion ØØ1Ch. 28d. (TEST-CHAR)
.- Efectua la rutina SKIP-OVER que devuelve el flag de acarreo activado si el caracter no es imprimible.
.- Retorna si no tiene el acarreo activado
.- Si el acarreo está activado, salta a la rutina NEXT-CHAR.

NEXT-CHAR:

Posiciones ØØ2Ø a ØØ24h. (32 a 36d.)

Apunta al siguiente caracter segun CH-ADD y retorna si es imprimible, si no sigue incrementando CH-ADD.

Se entra con RST 2Ø y un valor determinado en la variable CH-ADD. El intérprete de Basic llama repetidamente a estas dos rutinas segun se vá moviendo a lo largo de las lineas de programa, para comprobar los distintos códigos.

En realidad, cabe considerar a NEXT-CHAR como una entrada alternativa de GET-CHAR.

>DESCRIPCION:
Posicion ØØ2Øh. 32d. (NEXT-CHAR)
.- Efectua la subrutina CH-ADD+1 (incrementar CH-ADD en 1)
.- Vuelve a la posicion TEST-CHAR de la rutina anterior.

FP-CALC:

Posiciones ØØ28 a ØØ2Ah. (4Ø a 42d.)

Se entra con RST 28 seguido de una serie de literales que indican las operaciones a realizar por el calculador.

Al igual que en PRINT-A-1, se trata de el punto de entrada a otra de las mas potentes rutinas del sistema, en este caso, la rutina del calculador en coma flotante. De la misma forma que antes, se salta directamente al calculador, situado en la direccion 335Bh. (13,147d.)

El funcionamiento de esta rutina, que es tan complejo como potente, se verá con detalle cuando estudiemos las rutinas de cálculo.

BC-SPACES:

Posiciones ØØ3Ø a ØØ37h. (48 a 55d.)

Esta rutina deja un número de posiciones libres en el area de trabajo desplazando hacia arriba el resto de la memoria. Este número de posiciones viene definido por el contenido del par de registros "BC".

Se entra con RST 30 teniendo en BC el número de posiciones a dejar libres.

>DESCRIPCION:
Posicion 0030h. 48d. (BC-SPACES)

.- Salva BC en la pila de máquina (PUSH BC)

.- Carga en el par de registros HL el contenido de la variable WORK-SP que apunta al inicio del area de trabajo.

.- Salva HL en la pila de máquina (PUSH HL)

.- Salta a ejecutar la rutina RESERVE desde la que retorna al llamador. Esta rutina se verá en detalle al estudiar las rutinas de ejecucion.

MASK-INT:

Posiciones 0038 a 0052h. (56 a 82d.)

Se trata de la rutina a la que salta el microprocesador cada vez que recibe una peticion de interrupcion enmascarable. Esto ocurre cada 20 milisegundos.

Esta rutina actualiza la variable FRAMES (reloj de tiempo real del Spectrum) y lee el teclado.

Se puede evitar que se ejecute esta rutina si se deshabilita la interrupcion con la instruccion DI (se ensambla como F3h. 243d.). La interrupcion se puede rehabilitar con la instruccion EI (se ensambla como FBh. 251d.).

El propio sistema desabilita la interrupcion cuando se inicializa, ejecuta el comando BEEP o realiza alguna operacion con el cassette, por tanto, durante estos intervalos no se actualiza el reloj de tiempo real (FRAMES).

>DESCRIPCION:
Posicion 0038h. 56d. (MASK-INT)

.- Salva los registros AF y HL

.- Incrementa la variable FRAMES

.- Continua en KEY-INT

Posicion 0048h. 72d. (KEY-INT)

.- Salva los registros BC y DE

.- Ejecuta la subrutina KEYBOARD en la direccion 02FBh. (763d.) que busca en el teclado el código de la última tecla pulsada, si es que la hay.

.- Recupera los registros DE, BC, HL y AF

.- Habilita la interrupcion enmascarable

.- Retorna al punto donde ocurrio la interrupcion.

RESET:

Posiciones 0066 a 0073h. (102 a 115d.)

Se trata de la rutina de servicio a la interrupcion no enmascarable, no se usa en el modelo standard del Spectrum, pero el código permite una reposicion del sistema si se dá una activacion de la linea NMI, ya que en ese caso, el Z-80 saltaría a la posicion 0066.

En esta rutina se puede observar hasta qué punto el software está sujeto a consideraciones de tipo comercial. Si se hubiese escrito la rutina de forma que provocase un salto al sistema operativo (por ejemplo, un RST 8 con literal FFh.), se podría salir de un programa comercial y recobrar el control sin perder la informacion almacenada en memoria, lo que dificultaría enormemente la proteccion de software. Los técnicos de Sinclair, conscientes de que un ordenador se vende en gran medida, en funcion de los programas disponibles para él, prefirieron en este caso, jugar la baza a favor de los fabricantes de software, y nos quitaron a los usuarios la posibilidad de instalar facilmente un RESET "en caliente".

>DESCRIPCION:
Posicion ØØ66h. 102d. (RESET)
.- Salva los registros AF y HL
.- Mira si es "cero" la variable NMIADD.
.- Si es "cero" salta a la rutina START (Direccion ØØØØ).
.- En caso contrario, recupera los registros AF y HL y retorna al punto donde se produjo la interrupcion.

CH-ADD+1:

Posiciones ØØ74 a ØØ7Ch. (116 a 124d.)
Se trata de una subrutina de GET-CHAR. Incrementa el contenido de la variable CH-ADD y retorna con el contenido de la direccion apuntada por esta variable en el registro "A".

>DESCRIPCION:
Hay tres entradas posibles:
Posicion ØØ74h. 116d. (CH-ADD+1)
.- Carga en HL el valor de CH-ADD
.- Continúa en TEMP-PTR1

Posicion ØØ77h. 119d. (TEMP-PTR1)
.- Incrementa HL
.- Continúa en TEMP-PTR2

Posicion ØØ78h. 128d. (TEMP-PTR2)
.- Carga CH-ADD con el valor de HL
.- Carga en el registro A el contenido de la direccion apuntada por HL.
.- Retorna

SKIP-OVER:

Posiciones ØØ7D a ØØ94h. (125 a 148d.)
Se trata de otra subrutina de GET-CHAR. Comprueba si el valor del registro A corresponde a un caracter imprimible. Algunos códigos hacen que CH-ADD se incremente en uno o dos.

>DESCRIPCION:
Posicion ØØ7Dh. 125d. (SKIP-OVER)

.- Retorna sin acarreo si A >= 2Øh.
.- Retorna sin acarreo si A = ØDh. (Código de ENTER)
.- Retorna con acarreo si A está entre ØØ y ØFh.
.- Retorna con acarreo si A está entre 18 y 1Fh. (Códigos no utilizados).
.- Incrementa HL
.- Retorna con acarreo y CH-ADD incrementado en uno si A es igual a 1Øh. o a 15h.
.- Incrementa HL
.- Retorna con acarreo y CH-ADD incrementado en dos si A es igual a 16h. o 17h. (AT o TAB).

Una vez vistas las subrutinas de página "cero", en el siguiente capítulo veremos una serie de tablas que utiliza el sistema para expandir Tokens y leer el teclado, así como un listado completo en Assembler de las rutinas vistas hasta ahora.

---------- o ----------

FIGURA 1

MAPA DE LAS RUTINAS DE PAGINA CERO

0000h. (0d.)	START	Rutina de inicializacion del sistema operativo.
0008h. (8d.)	ERROR-1	Rutina para imprimir mensajes en la parte inferior de la pantalla.
0010h. (16d.)	PRINT-A-1	Rutina general de salida de datos por cualquier canal.
0013h. (19d.)	------	Posiciones no usadas (Definidas con valor FFh.)
0018h. (24d.)	GET-CHAR	Rutina general para comprobar caracteres.
0020h. (32d.)	NEXT-CHAR	Entrada alternativa de GET-CHAR.
0025h. (37d.)	------	Posiciones no usadas (Definidas con valor FFh.)
0028h. (40d.)	FP-CALC	Punto de entrada a las rutinas de calculo en coma flotante.
002Bh. (43d.)	------	Posiciones no usadas (Definidas con valor FFh.)
0030h. (48d.)	BC-SPACES	Rutina para hacer sitio en el area de trabajo.
0038h. (56d.)	MASK-INT	Rutina de servicio a la interrupcion enmascarable.
0053h. (83d.)	ERROR-2	Continuacion de ERROR-1.
005Fh. (95d.)	------	Posiciones no usadas (Definidas con valor FFh.)
0066h. (102d.)	RESET	Rutina de servicio a la interrupcion no enmascarable.
0074h. (116d.)	CH-ADD+1	Subrutina de GET-CHAR.
007Dh. (125d.)	SKIP-OVER	Subrutina de GET-CHAR.
0095h. (149d.)	COMIENZO DE LAS TABLAS.	

SUBRUTINAS DE LA ROM

CAPITULO IV

==========

En este cuarto capítulo de la serie, veremos las tablas que utiliza el Sistema para expandir los "tokens" y leer el teclado, así como los listados en Assembler y Código Máquina de las rutinas vistas hasta ahora.

==========

Jesús Alonso Rodriguez

LA TABLA DE TOKENS:

A lo largo de la ROM del Spectrum, no solo hay rutinas del Sistema, en algunos lugares encontramos tablas de datos de las que se sirve para determinados cometidos.

La primera de ellas es la "Tabla de Tokens". En el Spectrum los programas en Basic se almacenan "tokenizados", es decir, cada comando o funcion Basic tiene su código correspondiente al igual que los caracteres normales, esto se hace así con la finalidad de ahorrar memoria. Por ejemplo, el comando PRINT ocuparía expandido 5 bytes (los códigos de la "P", "R", "I", "N" y "T") mientras que "tokenizado" ocupa un solo byte con el código F5h. (245d.).

Cuando el ordenador nos lista un programa, expande los "tokens" para que podamos leerlos. Este proceso se realiza con la ayuda de la "Tabla de Tokens" que se encuentra entre las direcciones ØØ95h. y Ø2Ø4h. (149d. y 516d.). En esta tabla los códigos de los caracteres que componen los "tokens" se encuentran todos seguidos, el ordenador sabe donde termina cada "token" porque su último código tiene a "1" el bit de mas peso. Por ejemplo, PRINT no se escribe 5Ø 52 49 4E 54, sino 5Ø 52 49 4E D4 es decir, se ha cambiado el código 54 de la "T" por D4 para indicar que es el último caracter del "token". Cuando el Sistema tiene que buscar un "token" en la tabla, le resta A4h. (164d.) a su código, y entra en la tabla con ese número, cada vez que se encuentra un código con el bit de mas peso a "1", resta "1" del código del "token", cuando este se hace "cero", el siguiente "token" es el que está buscando.

En la TABLA I se han representado los caracteres finales de cada "token" entre comillas, para indicar que el bit de mas peso de su código se encuentra a "1".

LAS TABLAS DEL TECLADO:

Una de las características mas particulares del Spectrum es la versatilidad de su teclado, ya que permite la introduccion de letras, símbolos, tokens y códigos de control segun el "modo" en que se esté trabajando y las teclas de SHIFT que se utilicen. La decodificacion de estas teclas se lleva a cabo mediante una serie de tablas, utilizándose una u otra en funcion del "modo" y de las teclas de SHIFT.

En la TABLA II se muestran las distintas tablas que utiliza el Spectrum para decodificar el teclado. No hay tabla para las letras minúsculas, ya que estas se consiguen simplemente, sumando 32d. al código de las mayúsculas.

Al igual que en la TABLA I, se ha representado a la derecha del "punto y coma" el caracter, token o funcion que corresponde a cada código de la izquierda.

En los capítulos siguientes, veremos las rutinas que utiliza el Spectrum para "barrer" y decodificar el teclado.

LOS LISTADOS:

En las FIGURAS 1 y 2 se han representado los listados de las rutinas vistas hasta ahora, la primera columna corresponde a las direcciones, la segunda al código objeto o Código Máquina, la tercera son las etiquetas a las que se hace referencia en el texto, y la cuarta es el listado del código fuente escrito en lenguaje Assembler.

Todos los números estan en hexadecimal y al final de cada listado hay una tabla con las direcciones de todas las variables del sistema empleadas. Para los que esten empezando a aprender Código Máquina puede resultar muy util observar la correspondencia entre cada instruccion de Assembler y su código objeto correspondiente.

La FIGURA 1 muestra el listado de todas las rutinas de página cero y sus subrutinas auxiliares, mientras que la FIGURA 2 muestra la rutina de inicializacion del sistema, cuya mayor parte es comun a la ejecucion del comando NEW.

El pseudonemónico DEFB (DEFine Byte) se utiliza para definir el contenido de una direccion de memoria que no se corresponde con ninguna instruccion del código fuente, y EQU significa que la etiqueta que le precede es EQUivalente al número que le sigue.

Las posiciones no utilizadas estan definidas a +FFh. esto se debe a que cuando se programa una ROM, en principio todas las posiciones contienen +FFh. por tanto, las que no se programan se quedan con este valor.

---------- o ----------

FIGURA 1

REINICIOS DE PAGINA CERO

```
0000 F3       START      DI
0001 AF                  XOR A
0002 11FFFF              LD DE,+FFFF
0005 C3CB11              JP 11CB ;START/NEW
0008 2A5D5C   ERROR-1    LD HL,(CH-ADD)
001B 225F5C              LD (X-PTR),HL
001E 1843                JR 0053 ;ERROR-2
0010 C3F215   PRINT-A-1  JP 15F2 ;PRINT-A-2
0013 FFFFFFFF            DEFB +FF,+FF,+FF,+FF
0017 FF                  DEFB +FF
0018 2A5D5C   GET-CHAR   LD HL,(CH-ADD)
001B 7E                  LD A,(HL)
001C CD7D00   TEST-CHAR  CALL 007D ;SKIP-OVER
001F D0                  RET NC
0020 CD7400   NEXT-CHAR  CALL 0074 ;CH-ADD+1
0023 18F7                JR 001C ;TEST-CHAR
0025 FFFFFF              DEFB +FF,+FF,+FF
0028 C35B33   FP-CALC    JP 335B ;CALCULATE
002B FFFFFFFF            DEFB +FF,+FF,+FF,+FF
002F FF                  DEFB +FF
0030 C5       BC-SPACES  PUSH BC
0031 2A615C              LD HL,(WORKSP)
0034 E5                  PUSH HL
0035 C39E16              JP 169E ;RESERVE
0038 F5       MASK-INT   PUSH AF
0039 E5                  PUSH HL
003A 2A785C              LD HL,(FRAMES)
003D 23                  INC HL
003E 22785C              LD (FRAMES),HL
0041 7C                  LD A,H
0042 B5                  OR L
0043 2003                JR NZ,0048 ;KEY-INT
0045 FD3440              INC (IY+40) ;FRAMES-3
0048 C5       KEY-INT    PUSH BC
0049 D5                  PUSH DE
004A CDBF02              CALL 02BF ;KEYBOARD
004D D1                  POP DE
004E C1                  POP BC
004F E1                  POP HL
0050 F1                  POP AF
0051 FB                  EI
0052 C9                  RET
0053 E1       ERROR-2    POP HL
0054 6E                  LD L,(HL)
0055 FD7500              LD (IY+00),L ;ERR-NR
0058 ED7B3D5C            LD SP,(ERR-SP)
005C C3C516              JP 16C5 ;SET-STK
005F FFFFFFFF            DEFB +FF,+FF,+FF,+FF
0063 FFFFFF              DEFB +FF,+FF,+FF
0066 F5       RESET      PUSH AF
0067 E5                  PUSH HL
0068 2AB05C              LD HL,(NMIADD)
006B 7C                  LD A,H
006C B5                  OR L
006D 2001                JR NZ,0070 ;NO-RESET
006F E9                  JP (HL)
0070 E1       NO-RESET   POP HL
0071 F1                  POP AF
0072 ED45                RETN
0074 2A5D5C   CH-ADD+1   LD HL,(CH-ADD)
0077 23       TEMP-PTR1  INC HL
0078 225D5C   TEMP-PTR2  LD (CH-ADD),HL
007B 7E                  LD A,(HL)
007C C9                  RET
007D FE21     SKIP-OVER  CP +21
007F D0                  RET NC
0080 FE0D                CP +0D
0082 C8                  RET Z
0083 FE10                CP +10
0085 D8                  RET C
0086 FE18                CP +18
0088 3F                  CCF
0089 D8                  RET C
008A 23                  INC HL
008B FE16                CP +16
008D 3801                JR C,0090 ;SKIPS
008F 23                  INC HL
0090 37       SKIPS      SCF
0091 225D5C              LD (CH-ADD),HL
0094 C9                  RET
5C16          WORKSP     EQU +5C16
5C3D          ERR-SP     EQU +5C3D
5C5D          CH-ADD     EQU +5C5D
5C5F          X-PTR      EQU +5C5F
5C78          FRAMES     EQU +5C78
5CB0          NMIADD     EQU +5CB0
```

FIGURA 2

RUTINA DE INICIALIZACION

```
11B7 F3        NEW        DI
11B8 3EFF                 LD A,+FF
11BA ED5BB25C             LD DE,(RAMTOP)
11BE D9                   EXX
11BF ED4BB45C             LD BC,(P-RAMT)
11C3 ED5B385C             LD DE,(RASP/PIP)
11C7 2A7B5C               LD HL,(UDG)
11CA D9                   EXX
11CB 47        START/NEW  LD B,A
11CC 3E07                 LD A,+07
11CE D3FE                 OUT (+FE),A
11D0 3E3F                 LD A,+3F
11D2 ED47                 LD I,A
11D4 00                   NOP
11D5 00                   NOP
11D6 00                   NOP
11D7 00                   NOP
11D8 00                   NOP
11D9 00                   NOP
11DA 62        RAM-CHECK  LD H,D
11DB 6B                   LD L,E
11DC 3602      RAM-FILL   LD (HL),+02
11DE 2B                   DEC HL
11DF BC                   CP H
11E0 20FA                 JR NZ,11DC ;RAM-FILL
11E2 A7        RAM-READ   AND A
11E3 ED52                 SBC HL,DE
11E5 19                   ADD HL,DE
11E6 23                   INC HL
11E7 3006                 JR NC,11EF ;RAM-DONE
11E9 35                   DEC (HL)
11EA 2803                 JR Z,11EF ;RAM-DONE
11EC 35                   DEC (HL)
11ED 28F3                 JR Z,11E2 ;RAM-READ
11EF 2B        RAM-DONE   DEC HL
11F0 D9                   EXX
11F1 ED43BA5C             LD (P-RAMT),BC
11F5 ED53385C             LD (RASP/PIP),DE
11F9 227B5C               LD (UDG),HL
11FC D9                   EXX
11FD 04                   INC B
11FE 2819                 JR Z,1219 ;RAM-SET
1200 22B45C               LD (P-RAMT),HL
1203 11AF3E               LD DE,+3EAF
1206 01A800               LD BC,+00A8
1209 EB                   EX DE,HL
120A EDB8                 LDDR
120C EB                   EX DE,HL
120D 23                   INC HL
120E 227B5C               LD (UDG),HL
1211 2B                   DEC HL
1212 014000               LD BC,+0040
1215 ED43385C             LD (RASP/PIP),BC
1219 22B25C               LD (RAMTOP),HL
121C 21003C               LD HL,+3C00
121F 22365C               LD (CHARS),HL
1222 2AB25C               LD HL,(RAMTOP)
1225 363E                 LD (HL),+3E
1227 2B                   DEC HL
1228 F9                   LD SP,HL
1229 2B                   DEC HL
122A 2B                   DEC HL
122B 223D5C               LD (ERR-SP),HL
122E ED56                 IM 1
1230 FD213A5C             LD IY,+5C3A
1234 FB                   EI
1235 21B65C               LD HL,+5CB6
1238 224F5C               LD (CHANS),HL
123B 11AF15               LD DE,+15AF
123E 011500               LD BC,+0015
1241 EB                   EXX DE,HL
1242 EDB0                 LDIR
1244 EB                   EXX DE,HL
1245 2B                   DEC HL
1246 22575C               LD (DATADD),HL
1249 23                   INC HL
124A 22535C               LD (PROG),HL
124D 224B5C               LD (VARS),HL
1250 3680                 LD (HL),+80
1252 23                   INC HL
1253 22595C               LD (E-LINE),HL
1256 360D                 LD (HL),+0D
1258 23                   INC HL
1259 3680                 LD (HL),+80
125B 23                   INC HL
125C 22615C               LD (WORKSP),HL
125F 22635C               LD (STKBOT),HL
1262 22655C               LD (STKEND),HL
1265 3E38                 LD A,+38
1267 328D5C               LD (ATTR-P),A
126A 328F5C               LD (ATTR-T),A
126D 32485C               LD (BORDCR),A
1270 212305               LD HL,+0523
1273 22095C               LD (REPDEL),HL
1276 FD35C6               DEC (IY+C6) ;KSTATE-0
1279 FD35CA               DEC (IY+CA) ;KSTATE-4
127C 21C615               LD HL,+15C6
```

```
127F 1110 5C              LD DE,+5C10
1282 010E 00              LD BC,+000E
1285 EDB0                 LDIR
1287 FDCB 01CE            SET 1,(IY+1) ;FLAGS
128B CDDF 0E              CALL 0EDF ;CLEAR-PRB
128E FD36 3102            LD (IY+31),+02 ;DF-SZ
1292 CD6B 0D              CALL 0D6B ;CLS
1295 AF                   XOR A
1296 113815               LD DE,+1538
1299 CD0A 0C              CALL 0C0A ;PO-MSG
129C FDCB 02EE            SET 5,(IY+2) ;TV-FLAG
12A0 1807                 JR 12A9 ;MAIN-1
5C09          REPDEL      EQU +5C09
5C38          RAS/PIP     EQU +5C38
5C36          CHARS       EQU +5C36
5C3A          ERR-NR      EQU +5C3A
5C3D          ERR-SP      EQU +5C3D
5C48          BORDCR      EQU +5C48
5C4B          VARS        EQU +5C4B
5C4F          CHANS       EQU +5C4F
5C53          PROG        EQU +5C53
5C57          DATADD      EQU +5C57
5C59          E-LINE      EQU +5C59
5C61          WORKSP      EQU +5C61
5C63          STKBOT      EQU +5C63
5C65          STKEND      EQU +5C65
5C7B          UDG         EQU +5C7B
5C8D          ATTR-P      EQU +5C8D
5C8F          ATTR-T      EQU +5C8F
5CB2          RAMTOP      EQU +5CB2
5CB4          P-RAMT      EQU +5CB4
================================================
```

TABLA I

TABLA DE TOKENS

```
0090  --  --  --  --  --  BF  52  4E  ;  -  -  -  -  -  - "?" R  N
0098  C4  49  4E  4B  45  59  A4  50  ; "D" I  N  K  E  Y "$" P
00A0  C9  46  CE  50  4F  49  4E  D4  ; "I" F "N" P  O  I  N "T"
00A8  53  43  52  45  45  4E  A4  41  ;  S  C  R  E  E  N "$" A
00B0  54  54  D2  41  D4  54  41  C2  ;  T  T "R" A "T" T  A "B"
00B8  56  41  4C  A4  43  4F  44  C5  ;  V  A  L "$" C  O  D "E"
00C0  56  41  CC  4C  45  CE  53  49  ;  V  A "L" L  E "N" S  I
00C8  CE  43  4F  D3  54  41  CE  41  ; "N" C  O "S" T  A "N" A
00D0  53  CE  41  43  D3  41  54  CE  ;  S "N" A  C "S" A  T "N"
00D8  4C  CE  45  58  D0  49  4E  D4  ;  L "N" E  X "P" I  N "T"
00E0  53  51  D2  53  47  CE  41  42  ;  S  Q "R" S  G "N" A  B
00E8  D3  50  45  45  CB  49  CE  55  ; "S" P  E  E "K" I "N" U
00F0  53  D2  53  54  52  A4  43  48  ;  S "R" S  T  R "$" C  H
00F8  52  A4  4E  4F  D4  42  49  CE  ;  R "$" N  O "T" B  I "N"
0100  4F  D2  41  4E  C4  3C  BD  3E  ;  O "R" A  N "D" < "=" >
0108  BD  3C  BE  4C  49  4E  C5  54  ; "=" < ">" L  I  N "E" T
0110  48  45  CE  54  CF  53  54  45  ;  H  E "N" T "O" S  T  E
0118  D0  44  45  46  20  46  CE  43  ; "P" D  E  F     F "N" C
0120  41  D4  46  4F  52  4D  41  D4  ;  A "T" F  O  R  M  A "T"
0128  4D  4F  56  C5  45  52  41  53  ;  M  O  V "E" E  R  A  S
0130  C5  4F  50  45  4E  20  A3  43  ; "E" O  P  E  N     "#" C
0138  4C  4F  53  45  20  A3  4D  45  ;  L  O  S  E     "#" M  E
0140  52  47  C5  56  45  52  49  46  ;  R  G "E" V  E  R  I  F
0148  D9  42  45  45  D0  43  49  52  ; "Y" B  E  E "P" C  I  R
0150  43  4C  C5  49  4E  CB  50  41  ;  C  L "E" I  N "K" P  A
0158  50  45  D2  46  4C  41  53  C8  ;  P  E "R" F  L  A  S "H"
0160  42  52  49  47  48  D4  49  4E  ;  B  R  I  G  H "T" I  N
0168  56  45  52  53  C5  4F  56  45  ;  V  E  R  S "E" O  V  E
0170  D2  4F  55  D4  4C  50  52  49  ; "R" O  U "T" L  P  R  I
0178  4E  D4  4C  4C  49  53  D4  53  ;  N "T" L  L  I  S "T" S
0180  54  4F  D0  52  45  41  C4  44  ;  T  O "P" R  E  A "D" D
0188  41  54  C1  52  45  53  54  4F  ;  A  T "A" R  E  S  T  O
0190  52  C5  4E  45  D7  42  4F  52  ;  R "E" N  E "W" B  O  R
0198  44  45  D2  43  4F  4E  54  49  ;  D  E "R" C  O  N  T  I
01A0  4E  55  C5  44  49  CD  52  45  ;  N  U "E" D  I "M" R  E
01A8  CD  46  4F  D2  47  4F  20  54  ; "M" F  O "R" G  O     T
01B0  CF  47  4F  20  53  55  C2  49  ; "O" G  O     S  U "B" I
01B8  4E  50  55  D4  4C  4F  41  C4  ;  N  P  U "T" L  O  A "D"
01C0  4C  49  53  D4  4C  45  D4  50  ;  L  I  S "T" L  E "T" P
01C8  41  55  53  C5  4E  45  58  D4  ;  A  U  S "E" N  E  X "T"
01D0  50  4F  4B  C5  50  52  49  4E  ;  P  O  K "E" P  R  I  N
01D8  D4  50  4C  4F  D4  52  55  CE  ; "T" P  L  O "T" R  U "N"
01E0  53  41  56  C5  52  41  4E  44  ;  S  A  V "E" R  A  N  D
01E8  4F  4D  49  5A  C5  49  C6  43  ;  O  M  I  Z "E" I "F" C
01F0  4C  D3  44  52  41  D7  43  4C  ;  L "S" D  R  A "W" C  L
01F8  45  41  D2  52  45  54  55  52  ;  E  A "R" R  E  T  U  R
0200  CE  43  4F  50  D9  --  --  --  ; "N" C  O  P "Y" -  -  -
```

TABLA II

TABLAS DE TECLADO

TABLA PRINCIPAL, Modo "L" y CAPS SHIFT

```
0200  --  --  --  --  --  42  48  59  ;  -  -  -  -  -  B  H  Y
0208  36  35  54  47  56  4E  4A  55  ;  6  5  T  G  V  N  J  U
0210  37  34  52  46  43  4D  4B  49  ;  7  4  R  F  C  M  K  I
0218  38  33  45  44  58  0E  4C  4F  ;  8  3  E  D  X S/S L  O
0220  39  32  57  53  5A  20  0D  50  ;  9  2  W  S  Z SP ENT P
0228  30  31  51  41  --  --  --  --  ;  0  1  Q  A  -  -  -  -
```

MODO EXTENDIDO, Teclas de letras sin SHIFT

```
022C  E3  C4  E0  E4  ;  READ     BIN      LPRINT   DATA
0230  B4  BC  BD  BB  ;  TAN      SGN      ABS      SQR
0234  AF  B0  B1  C0  ;  CODE     VAL      LEN      USR
0238  A7  A6  BE  AD  ;  PI       INKEY$   PEEK     TAB
023C  B2  BA  E5  A5  ;  SIN      INT      RESTORE  RND
0240  C2  E1  B3  B9  ;  CHR$     LLIST    COS      EXP
0244  C1  B8  --  --  ;  STR$     LN       --       --
```

MODO EXTENDIDO, Teclas de letras y cualquier SHIFT

```
0244  --  --  7E  DC  ;  --       --       ~        BRIGHT
0248  DA  5C  B7  7B  ;  PAPER    \        ATN      {
024C  7D  D8  BF  AE  ;  }        CIRCLE   IN       VAL$
0250  AA  AB  DD  DE  ;  SCREEN$  ATTR     INVERSE  OVER
0254  DF  7F  B5  D6  ;  OUT      (c)      ASN      VERIFY
0258  7C  D5  5D  DB  ;  |        MERGE    [        FLASH
025C  B6  D9  5B  D7  ;  ACS      INK      ]        BEEP
```

CODIGOS DE CONTROL, Teclas numericas y CAPS SHIFT

```
0260  0C  07  06  04  ;  DELETE   EDIT     C.LOCK   TRUE VIDEO
0264  05  08  0A  0B  ;  IN.VIDEO Curs.IZQ Curs.ABA Curs.ARRI
0268  09  0F  --  --  ;  Curs.DER GRAPHICS --       --
```

CODIGOS DE SIMBOLOS, Teclas de letras y SIMBOL SHIFT

```
0268  --  --  E2  2A  ;  --       --       STOP     *
026C  3F  CD  C8  CC  ;  ?        STEP     >=       TO
0270  CB  5E  AC  2D  ;  THEN     ^        AT       -
0274  2B  3D  2E  2C  ;  +        =        .        ,
0278  3B  22  C7  3C  ;  ;        "        <=       <
027C  C3  3E  C5  2F  ;  NOT      >        OR       /
0280  C9  60  C6  3A  ;  <>       &        AND      :
```

MODO EXTENDIDO, Teclas numericas y SIMBOL SHIFT

```
0284  D0  CE  A8  CA  ;  FORMAT   DEF FN   FN       LINE
0288  D3  D4  D1  D2  ;  OPEN #   CLOSE #  MOVE     ERASE
028C  A9  CF  --  --  ;  POINT    CAT      --       --
```