

ENSAMBLADOR DESENSAMBLADOR

MSX – MSX 2

R S C II

Ramón Sala

Manual de Instrucciones

ÍNDICE

INTRODUCCIÓN.	1
-----------------------	---

APARTADO 1

INSTALACIÓN Y CARGA DEL PROGRAMA.	2
---	---

APARTADO 2

COMANDOS DEL EDITOR	3
-------------------------------	---

Comando: &B	4
Comando: &H	5
Comando: A:	6
Comando: B:	6
Comando: BA	7
Comando: BL	8
Comando: BO	9
Comando: BR	10
Comando: BU	11
Comando: CB	13
Comando: CC	14
Comando: CT	15
Comando: DE	16
Comando: EN	18
Comando: FU	20
Comando: GB	21
Comando: GC	22
Comando: GT	23
Comando: IM	24
Comando: IN	25
Comando: LE	26
Comando: LK	27
Comando: LT	29
Comando: MV	30
Comando: NU	31
Comando: PO	32
Comando: PR	33

Comando: RE	34
Comando: RN	35
Comando: SI	36
Comando: TR	40
Comando: VD	41
Comando: VE	42
Comando: VM	43

APARTADO 3

INTRODUCCIÓN DE LÍNEAS DE TEXTO	44
--	----

APARTADO 4

OTROS COMANDOS Y FUNCIONES.	45
4.1 - Pseudocomandos.	45
4.2 - Comandos del Ensamblador.	49
4.3 - Macros.	49
4.4 - Signos aritméticos.	51
4.5 - Slots/subslots.	51

APARTADO 5

INFORMACIÓN IMPORTANTE.	52
5.1 - Direcciones inferiores a &H8000	52
5.2 - Grabación y carga de programas.	53
5.3 - Consejos y notas finales.	53
5.4 - Errores	55
5.5 - Comandos autorizados para Z-80.	57
5.6 - Un programa de ejemplo.	57
5.7 - Ejemplo de macros anidadas.	59

APARTADO 6

EL PROGRAMA CONVERT	60
--------------------------------------	----

RSC SOFTWARE, le agradece su confianza y se permite felicitarle por la compra de RSC II. Por favor, rellene el siguiente cupón, dando su impresión sobre el programa, y envíelo a la dirección que abajo se indica.

Nombre: _____
Dirección: _____
Localidad: _____ D.P.: _____
Provincia: _____ Tel.: _____
Ordenador: _____ Modelo: _____

Opinión que le merece el programa RSC II:

Envíe este cupón a:

RSC SOFTWARE
Av. Princep d'Asturies, 62-64, 3r,5a.
08.012 - BARCELONA
Teléf.: (93) 218.17.52.

No dude en contactar con RSC SOFT, por carta o por teléfono, para cualquier información que necesite.

Gracias.

INTRODUCCIÓN

El *Macro-Ensamblador/Linkador* **RSC II** es un extenso y excelente programa compuesto por: un **Ensamblador** de dos pasos, un **Linkador**, un **Desensamblador** y un **Simulador/Depurador**, que puede funcionar perfectamente en cualquier ordenador MSX, con un mínimo de 64 K de memoria RAM. La amplia gama de posibilidades que permite le hace superar, ampliamente, a su predecesor, RSC, y sitúa a RSC II en la vanguardia de los programas Ensambladores de este estándar.

De entre sus innumerables mejoras e innovaciones, merece especial atención el comando **LK** (LinK=Enlazar) que permite enlazar distintos ficheros de código fuente simplificando aún más la programación. Por otra parte, también destacan los nuevos comandos del Ensamblador y del Simulador que proporcionan una inestimable ayuda a la hora de ensamblar o depurar un programa. Así pues, lea este manual detenidamente y, por favor, no dude en contactar con RSC SOFTWARE, para cualquier información y/o aclaración que necesite. Gracias.

Aunque los ordenadores MSX pertenecen a un sistema estándar, existen algunas diferencias entre ellos. Por este motivo, y con el fin de que RSC II obtenga el máximo partido de su máquina, debe instalarse previamente en ella, utilizando el programa que se acompaña en el disquete: **INSTALAR**. Este programa creará a **RSC II** adaptándolo, específicamente, a su ordenador MSX. Así pues, siga ATENTAMENTE todas las instrucciones que se indican con detalle en el apartado 1 (INSTALACIÓN Y CARGA DEL PROGRAMA) y su Macro-Ensamblador/Linkador funcionará sin problema alguno.

NOTA: *El disquete está garantizado contra todo defecto de origen.*

1. INSTALACION Y CARGA DEL PROGRAMA

Su Macro-Ensamblador RSC II no puede funcionar directamente sin una instalación previa dentro de su ordenador. Así que no se asuste, si observa que en su disquete no existe ningún programa RSC II. Siga correctamente, y paso a paso las instrucciones que siguen, para que el programa pueda ser creado.

- 1.- No efectúe la instalación en ningún otro ordenador que no sea el suyo.
- 2.- Asegúrese de que el disquete RSC II puede ser grabado, es decir, que NO esté protegido.
- 3.- Conecte la unidad de disco (el ordenador todavía no) e introduzca el disquete RSC II.
- 4.- Conecte su ordenador MSX y mantenga pulsada la tecla **CTRL** hasta que aparezca la pantalla que indica que se encuentra en BASIC.
- 5.- Teclee: **BLOAD "INSTALAR",R** <RETURN>. Este programa efectuará la instalación y, al finalizar mostrará el mensaje: **RSC II ya está instalado.**

Observe como el programa RSC II ya existe y se halla listo para funcionar. Ahora, saque el disquete y protéjalo contra escritura para que no se borre. Su programa RSC II, funcionará perfectamente en su ordenador, aunque muy probablemente no funcione en otros ordenadores MSX. No se preocupe, ello se debe a que el programa se ha adecuado a su máquina.

A partir de ahora, para cargar el programa, deberá hacer siempre lo siguiente: Si arranca con el sistema operativo MSX-DOS, sitúese en BASIC. Elija el color de letra y fondo (comando **COLOR**) que prefiera. Si posee un MSX-2, cambie el ancho de pantalla a 80 columnas (comando **WIDTH**). Seguidamente, cargue el programa con: **BLOAD "RSCII",R** <RETURN>.

NOTA: Si en alguna ocasión, usted cambiase su ordenador por otro MSX, envíe su copia del programa a RSC SOFTWARE y le será cambiada GRATUITAMENTE.

2. COMANDOS DEL EDITOR

Una vez puesto en marcha el programa, aparecerá la siguiente pantalla:

```
Macro-Ensamblador/Linkador RSC II
Versión MSX 1.0 por Ramón Sala
(C) RSC SOFTWARE - 1990
16255 bytes libres
Vale
□
```

Para poder realizar todas las funciones de las que es capaz, RSC II dispone de una serie de comandos, a introducir en MODO DIRECTO, que serán estudiados a continuación. Observará que en su sintaxis algunos parámetros aparecen entre corchetes. Esto significa que son opcionales y, por tanto, pueden ser omitidos según el caso. Obviamente, cuando se utilicen, debe prescindirse de los corchetes. Contrariamente, aquellas partes del comando que no estén corcheteadas, son imprescindibles para el desarrollo normal del mismo.

Si usted conoce el Ensamblador/Desensamblador RSC, le será muy fácil familiarizarse con los comandos y pseudocomandos de esta nueva versión, porque siguen la misma línea. No obstante, RSC II posee algunos comandos nuevos que, junto con los antiguos, **33 en total**, se explican, detalladamente, a partir de las próximas páginas. Estúdielos a fondo y descubrirá la enorme cantidad de funciones que pueden realizarse con este programa.

ATENCIÓN: Si usted poseía el Ensamblador/Desensamblador RSC, y desea utilizar los ficheros de texto que grabó con él, en el nuevo RSC II, vea el apartado 6 para la utilización del programa **CONVERT**.

Comando: **&B**

Sintaxis: **&B([*x1*])** <RETURN>

Efectúa la conversión del número ***x1***, bien a sistema **decimal**, si *x1* es binario, o bien a sistema **binario**, si *x1* es decimal o hexadecimal, y está encerrado entre paréntesis.

Ejemplo 1: **&B01110011** <RETURN>

Indica el equivalente en sistema decimal del número binario: &B01110011.

Ejemplo 2: **&B (50000)** <RETURN>

Indica el equivalente en sistema binario del número decimal: 50000.

Ejemplo 3: **&B (&HFFFF)** <RETURN>

Indica el equivalente en sistema binario del número hexadecimal: &HFFFF.

NOTA: En todos los casos, el número ***x1*** ha de tener un máximo de 16 bits (**65535** ó **&HFFFF**).

Comando: **&H**

Sintaxis: **&H[(]x1[)]** <RETURN>

Efectúa la conversión del número **x1**, bien a sistema **decimal**, si **x1** es hexadecimal, o bien a sistema **hexadecimal**, si **x1** es decimal o binario, y se halla encerrado entre paréntesis.

Ejemplo 1: **&HFFFF** <RETURN>

Indica el equivalente en sistema decimal del número hexadecimal: &HFFFF.

Ejemplo 2: **&H (50000)** <RETURN>

Indica el equivalente en sistema hexadecimal del número decimal: 50000.

Ejemplo 3: **&H (&B01110011)** <RETURN>

Indica el equivalente en sistema hexadecimal del número binario: &H01110011.

NOTA: En todos los casos, el número **x1** ha de tener un máximo de 16 bits (**65535** ó **&HFFFF**).

Comandos: **A:** y **B:**

Sintaxis: **A:** <RETURN>

Sintaxis: **B:** <RETURN>

En caso de disponer de dos unidades de disco, este comando conecta una u otra como la unidad de disco principal (o por defecto), es decir, la unidad que actuará en los comandos de grabación o carga.

Ejemplo 1: **A:** <RETURN>

Conecta la unidad de disco: **A**, como unidad principal.

Ejemplo 2: **B:** <RETURN>

Conecta la unidad de disco: **B**, como unidad principal.

NOTA: *Estos dos comandos solamente son válidos para RSC II en versión disco.*

Comando: **BA** (BAsic)

Sintaxis: **BA** <RETURN>

Permite volver al BASIC y trabajar en ese lenguaje manteniendo a RSC II en memoria para volver a utilizarlo cuando se necesite. Si al trabajar con RSC II no se ha ensamblado en ninguna de las direcciones que afectan al programa BASIC, éste se hallará intacto en memoria.

Ejemplo: **BA** <RETURN>

Si una vez en BASIC, desea volver a RSC II, teclee lo siguiente: **DEFUSR=&HF980:X=USR(0)** <RETURN>. El programa RSC II y el código fuente (en caso de que lo hubiera) no habrán sufrido modificación alguna.

Es aconsejable grabar las direcciones comprendidas entre **&HF980** y **&HF9A8** (que contienen la rutina que permite el retorno a RSC II) puesto que se encuentran en el buffer utilizado por la instrucción del BASIC: **PLAY** y podrían ser modificadas. Por lo tanto, una vez cargado RSC II, y devuelto el control al BASIC (comando **BA**), teclee lo siguiente:

BSAVE "VOLVER", &HF980, &HF9A8 <RETURN>

Puesto que la rutina es siempre la misma, solamente será preciso grabarla una vez. De este modo, si existe alguna sospecha de que ha sido alterada, se podrá recuperar con: **BLOAD "VOLVER"** <RETURN>

NOTA: Si ha de trabajar en BASIC y al mismo tiempo utilizar el Macro-Ensamblador, debe cargar primero RSC II, devolver el control al BASIC, y empezar el trabajo en ese lenguaje. De esta manera, RSC II ya está en memoria (sin afectar para nada al BASIC) y puede ser llamado, cuando se precise, de la manera antes indicada.

Comando: **BL** (Bytes Libres)

Sintaxis: **BL** <RETURN>

El Macro/Ensamblador RSC II dispone de **16255 bytes** para la introducción de código fuente. Mediante el comando **BL** podrá saber, en todo momento, la cantidad de bytes libres que le quedan.

Ejemplo: **BL** <RETURN>

Devuelve la cantidad de memoria disponible (en bytes) que queda para la introducción de texto.

NOTA: No es posible utilizar la totalidad de memoria (**16.255 bytes**) para almacenar un fichero fuente porque el Ensamblador necesita una cierta cantidad para depositar las etiquetas. Esta cantidad, depende del tamaño (y número de etiquetas) que posea el fichero que se ensambla, y oscila entre **800 y 2000 bytes**.

Comando: **BO** (BOrrar)

Sintaxis: **BO x1[-x2]** <RETURN>

Borra el bloque de código fuente comprendido entre las líneas **x1** y **x2**. Si se omite **x2**, se borra solamente la línea **x1**.

Ejemplo 1: **BO 100** <RETURN>

Borra la línea número 100 del fichero de texto.

Ejemplo 2: **BO 50 - 150** <RETURN>

Borra el bloque de texto comprendido entre las líneas 50 y 150 (ambas inclusive).

Además del comando **BO**, es posible borrar cualquier línea, simplemente tecleando su número, seguido de <RETURN>

NOTA: Cuando alguna de las dos líneas que se indican (**x1** ó **x2**) no existe, se provoca el **error 2**.

Comando: **BR** (BoRrar)

Sintaxis: **BR "nombre.ext"** <RETURN>

Borra del disco, el fichero: **nombre.ext**. Si el fichero indicado no existe, se provoca el **error 16**.

Ejemplo: **BR "EJEMPLO.ASM"** <RETURN>

Borra el fichero que tiene el nombre: EJEMPLO.ASM

NOTA: *Este comando solamente es válido para RSC II en versión disco.*

Comando: **BU** (BUscar)

Sintaxis: **BU[#] x0,x1,x2,...xn** <RETURN>

Busca una sucesión de bytes compuesta por: **x1, x2, ..., xn** (tantos como quepan en una línea), a partir de la dirección **x0**. Cuando la encuentra (si es que existe) emite un **BEEP** y muestra la dirección en la que empieza. De lo contrario, la búsqueda finaliza al alcanzar la dirección **&HFFFF (65535)**.

Ejemplo 1: **BU &H8000, &H76** <RETURN>

Busca, a partir de la dirección **&H8000** (en el slot que está activo en ese momento), un byte que contenga el valor **&H76 (118)**.

Ejemplo 2: **BU 50000, &HCD, &H10, &H40** <RETURN>

Busca a partir de la dirección **50000** (del slot que está activo en ese momento), tres bytes consecutivos conteniendo los valores **&HCD, &H10 y &H40**.

Ejemplo 3: **BU &HC000, "A", "B"** <RETURN>

Busca, a partir de la dirección **&HC000** (en el slot que está activo en ese momento), la cadena constituida por **"AB"** (**&H41 Y &H42** respectivamente).

Cuando utilice la opción **[#]**, siga las instrucciones dadas para la misma en el apartado 4.5

Ejemplo 4: **BU# &H3F00, &HCD, "A", &H76** <RETURN>

Al haber utilizado la opción **[#]** se habrá indicado el slot donde se ha de efectuar la búsqueda. Supo-

niendo que usted haya respondido: **3-2**, busca en el **subslot 2** del **slot 3**, un bloque de tres bytes consecutivos conteniendo: **&HCD**, "A" (**&H41**) y **&H76**.

NOTA: *Este comando muestra en todo momento, la dirección que se está comprobando. La búsqueda puede ser interrumpida pulsando <CTRL+STOP>.*

Comando: **CB** (Cargar Bytes)

Sintaxis: **CB "nombre[.ext]"** <RETURN>

Carga el programa en lenguaje máquina que tenga el nombre indicado entre comillas. Cuando finaliza la carga, se muestran tres direcciones en hexadecimal que son, respectivamente, dirección **inicial**, **final** y de arranque o **ejecución** del programa cargado. Si desea cargar un programa desde el **casete** (únicamente cuando se utiliza unidad de disco), añada la letra **"C"** al comando (**CBC**).

Ejemplo 1: **CB "EJEMPLO"** <RETURN>

Carga el programa que tiene el nombre EJEMPLO.BIN

Ejemplo 2: **CBC "PRUEBA"** <RETURN>

Carga del **casete** el programa que tiene el nombre PRUEBA.

En ambos casos, cuando el fichero solicitado no se encuentra, se provoca el **error 16**.

El comando **CB**, cargará todos los programas que se hayan grabado con **GB**, o desde BASIC con **BSAVE**.

NOTA: Cuando se omite la extensión, el programa la coloca automáticamente (**BIN** en este caso. Para más información, véase el apartado 5.2).

Comando: **CC** (Cargar COM)

Sintaxis: **CC "nombre[.ext]"** <RETURN>

Carga el programa con el nombre indicado entre comillas (y con extensión **COM** o **SYS**) y lo deposita a partir de la dirección **&H8100**. Al final de la carga se muestran las direcciones inicial y final (en hexadecimal) donde se ha depositado el programa.

Ejemplo 1: **CC "COMMAND.COM"** <RETURN>

Carga el programa con el nombre de COMMAND.COM.

Ejemplo 2: **CC "MSXDOS.SYS"** <RETURN>

Carga el programa con el nombre de MSXDOS.SYS.

Ejemplo 3: **CC "EJEMPLO"** <RETURN>

Carga el programa con el nombre de EJEMPLO.COM.

Este comando únicamente será válido para RSC II en versión disco.

NOTA: Cuando se omite la extensión (ejemplo 3) ésta es colocada por el programa automáticamente (en este caso **COM**. Véase apartado 5.2).

Comando: **CT** (Cargar Texto)

Sintaxis: **CT "nombre[.ext]"** <RETURN>

Carga el fichero de texto con el nombre que se indica entre comillas. En caso de haber algún fichero en memoria al utilizar este comando, el que se carga lo hará a continuación de aquel y se reenumerará todo el programa. De este modo, es posible unir ficheros distintos y convertirlos en uno solo. Si no se dispone de suficiente memoria para alojar al programa solicitado, se provoca el **error 6**.

Ejemplo 1: **CT "EJEMPLO.RSC"** <RETURN>

Carga en memoria el fichero de texto con el nombre EJEMPLO.RSC.

Ejemplo 2: **CT "EJEMPLO"** <RETURN>

Carga en memoria el fichero de texto con el nombre EJEMPLO.**ASM**.

Cuando se omite la extensión, el programa la coloca automáticamente (**ASM** en este caso. Para más información, véase el apartado 5.2).

NOTA: El comando **CT** cargará ÚNICA y EXCLUSIVAMENTE los ficheros de texto grabados con **GT**

Comando: **DE** (DEsensamblar)

Sintaxis: **DE[#] x1-x2[,I][,G]** <RETURN>

Desensambla (convierte en código fuente) el bloque de memoria comprendido entre las direcciones: **x1** y **x2** (ambas inclusive). Al utilizar este comando, aparece el siguiente mensaje:

Modo (H/D):

que solicita si el listado debe hacerse en Hexadecimal o Decimal. Responda **H** ó **D**, según prefiera, y pulse <RETURN>. Seguidamente, aparece el mensaje:

DEFB:

que solicita las direcciones que no deben ser tratadas como código objeto, sino como zona de datos. Si hay alguna zona que deba tratarse de esta manera, introduzca las direcciones inicial y final separadas por un guión. De lo contrario, simplemente pulse <RETURN>.

Ejemplo 1: **DE &H8000 - &HA000** <RETURN>

Desensambla la parte de la memoria comprendida entre las direcciones **&H8000** y **&HA000**.

Cuando utilice la opción **[#]**, siga las instrucciones dadas para la misma en el apartado 4.5

El comando **DE** dispone además de la opción **[,G]** que permite grabar el código fuente que genera como un fichero de texto (con el mismo formato que lo hace el comando **GT**). De este modo, el fichero así creado, podrá cargarse en cualquier momento con **CT**.

Ejemplo 2: **DE &H100 - &H1A00, G** <RETURN>

Desensambla el bloque de memoria comprendido entre las direcciones &H100 y &H1A00, grabándolo como un fichero de texto.

Además de todos los mensajes vistos anteriormente, con la opción [**G**], aparecerá el siguiente:

Nombre:

que solicita el nombre que ha de darle a la grabación. Introdúzcaselo, encerrado entre comillas (si se omite la extensión, ésta será **ASM**). Por ejemplo "PRUEBA" Y PULSE <RETURN>.

En ese momento, deberá tener a punto el disco o el casete.

La opción [**G**], es la ÚNICA que deposita el código fuente generado (antes de grabarlo) en la parte de la memoria utilizada por el Ensamblador para almacenar el código fuente y, por tanto, borrará cualquier fichero que allí haya. De haber alguno, debe grabarlo previamente si no quiere perderlo.

Cuando se utiliza la opción [**I**], el listado se efectúa por impresora.

NOTA: Cuando el código fuente que se genera es demasiado largo para caber en una sola grabación, el programa efectuará todas las que sean necesarias e irá solicitando el nombre de las mismas. Entre cada grabación, aparecerá la última dirección desensamblada que se ha grabado, con su correspondiente código objeto y código fuente.

Comando: **EN** (ENsamblar)

Sintaxis: **EN [x1]** <RETURN>

Ensambla el código fuente (lo convierte en lenguaje máquina) empezando por la línea **x1**. Si se omite **x1**, se empieza a ensamblar desde la primera línea. Cuando no hay ningún fichero fuente en memoria, el comando queda sin efecto pero no provoca error.

Ejemplo 1: **EN** <RETURN>

Ensambla el fichero fuente que hay en memoria, empezando por la primera línea del mismo.

Ejemplo 2: **EN 100** <RETURN>

Ensambla el fichero fuente que hay en memoria, solamente a partir de la línea 100.

En ambos casos, aparecerá el siguiente mensaje:

No. de Opción:

que solicita el tipo de ensamblado que debe hacer. Se dispone de las siguientes opciones:

0 = (o simplemente pulsando <RETURN>). Ensambla el fichero sin mostrar ningún listado (opción rápida)

1 = Ensambla el código fuente mostrando el listado ensamblador por la pantalla.

2 = Igual que la opción 1 pero mandando el listado a la **impresora**.

3 = Ensambla el código fuente mostrando un listado ensamblador por la pantalla, y listando, al final, todas las etiquetas y macros.

4 = Igual que la opción 3 pero mandando el listado a la **impresora**.

En todas estas opciones, el Ensamblador vigila que el código objeto nunca llegue a alcanzar, o sobrepasar, la zona de memoria utilizada por el **SP** (cuya dirección depende de diversos factores). De todas formas, si desea ensamblar en alguna dirección superior, súmele 10 al número de opción. Es decir, las opciones **10, 11, 12, 13 y 14**, realizan la misma función, respectivamente, que la **0, 1, 2, 3 y 4** pero **sin comprobar donde se deposita el código objeto**. Si añade una "**M**" a la opción (p.e.: **1M, 12M**, etc.), se listará también, el contenido de las **macros** al llamarlas.

Al final del ensamblado, si no ha habido errores, se mostrará una lista de las etiquetas ausentes (o no definidas). Cuando el fichero ensamblado, contenga el programa completo, NO deberá haber ninguna etiqueta sin definir. Por contra, si debe linkarse con otros, es lógico que aparezcan ausentes, las etiquetas que, encontrándose en otros ficheros son llamadas desde el que se ensambla.

IMPORTANTE: Vea el apartado 5.1 para el ensamblado de direcciones inferiores a **&H8000**.

NOTA: Con el fin de poder almacenar las etiquetas, no olvide dejar entre **800** y **2000** bytes libres (según las etiquetas y tamaño que tenga el fichero).

Comando: **FU** (FUnciones S/N)

Sintaxis: **FU x** <RETURN>

Hace aparecer (cuando **x = S**), o desaparecer, (cuando **x = N**) la última línea de la pantalla que muestra el contenido de las teclas de función.

Ejemplo 1: **FU N** <RETURN>

Desaparece la última línea de la pantalla que contiene las teclas de función.

Ejemplo 2: **FU S** <RETURN>

Aparece la última línea de la pantalla con el contenido de las teclas de función.

Comando: **GB** (Grabar Bytes)

Sintaxis: **GB "nombre[.ext]",x1,x2,[x3]**

Graba el programa en código máquina, con el nombre que se indica entre comillas, donde **x1** será la dirección **inicial**, **x2** la **final** y **x3** la de **ejecución**. Si se omite x3, se tomará como dirección de ejecución la misma que la inicial. Si quiere grabar en el **casete** (solamente cuando se utiliza unidad de disco), añada una letra "**C**" al comando (**GB**C****).

Ejemplo 1: **GB "EJEMPLO.RSC", &H8010, &HC0B0, &HC000**

Graba el programa contenido entre las direcciones: **&H8010** y **&HC0B0**, con el nombre de **EJEMPLO.RSC**, poniendo como dirección de ejecución la **&HC000**.

Ejemplo 2: **GB "EJEMPLO", &H8010, &HC0B0**

Graba el programa contenido entre las direcciones: **&H8010** y **&HC0B0**, con el nombre de **EJEMPLO.BIN**, poniendo como dirección de ejecución la **&H8010**.

Ejemplo 3: **GB**C** "PRUEBA", &H8010, &HC0B0**

Graba el programa igual que en el ejemplo anterior pero en el **casete** y con el nombre de **PRUEBA**.

Todos los programas grabados con el comando **GB**, se pueden cargar desde el **BASIC** con **BLOAD**.

NOTA: Cuando se omite la extensión, el programa la coloca automáticamente (**BIN** en este caso. Para mas información, véase el apartado 5.2).

Comando: **GC** (Grabar COM)

Sintaxis: **GC "nombre[.ext]",x1,x2**

Graba el programa en código máquina, con el nombre que se indica entre comillas, comprendido entre la dirección **x1** y **x2**. A diferencia de **GB**, este comando graba los programas en formato **COM** (o **SYS**) para que puedan ser cargados desde el **D.O.S.**

Ejemplo 1: **GC "EJEMPLO", &H8100, &HA000**

Graba el programa contenido entre las direcciones: **&H8100** y **&HA000**, con el nombre de **EJEMPLO.COM**

Ejemplo 2: **GC "MSXDOS.SYS", &H8100, &HA000**

Graba el programa contenido entre las direcciones: **&H8100** y **&HA000**, con el nombre de **MSXDOS.SYS**

ATENCIÓN: Lea el apartado 5.1 para la grabación de programas en formato **COM** (direcciones inferiores a **&H8000**).

Este comando solamente será válido para RSC II en versión disco.

NOTA: Cuando se omite la extensión, el programa la coloca automáticamente (**COM** en este caso. Para mas información, véase el apartado 5.2).

Comando: **GT** (Grabar Texto)

Sintaxis: **GT "nombre[.ext]"[,x1][-x2]**

Graba el bloque de texto que comprenden las líneas **x1** y la **x2** (ambas inclusive), con el nombre que se indica entre comillas. Si se omite **x2**, se graba la línea **x1** solamente. Si se omiten **x1** y **x2**, se graba todo el fichero. De no haber ningún fichero en memoria, el comando queda sin efecto.

Ejemplo 1: **GT "EJEMPLO", 100** <RETURN>

Graba la línea 100 del fichero que hay en memoria, con el nombre de EJEMPLO.ASM.

Ejemplo 2: **GT "EJEMPLO", 100 - 500** <RETURN>

Graba el bloque de texto comprendido entre las líneas 100 y 500 (ambas inclusive), del fichero que hay en memoria, con el nombre de EJEMPLO.ASM.

Ejemplo 3: **GT "EJEMPLO"** <RETURN>

Graba todo el fichero de texto que hay en memoria, con el nombre de EJEMPLO.ASM.

NOTA: Cuando se omite la extensión, el programa la coloca automáticamente (**ASM** en este caso. Para mas información, véase el apartado 5.2).

Comando: **IM** (IMpresora)

Sintaxis: **IM x1** <RETURN>

Indica el número de líneas por página a la hora de hacer listados por **impresora**. Cuando **x1=0**, las líneas se imprimen una tras otra sin hacer saltos de página (esta opción, es la que se encuentra conectada al arrancar el programa).

Ejemplo 1: **IM 62** <RETURN>

Indica al programa, que efectúe un salto de página cada 62 líneas.

Ejemplo 2: **IM 0** <RETURN>

Desconecta la opción IM y no se producen saltos de página.

NOTA: *Este comando actúa correctamente siempre que la longitud de una línea en el listado, no sea mayor que la longitud de una línea de la impresora.*

Comando: **IN** (INTroducir)

Sintaxis: **IN [x1][,x2]** <RETURN>

Numera automáticamente las líneas de texto facilitando su introducción, donde **x1** es el primer número de línea y **x2** el incremento. Cuando x1 y x2 son omitidos, su valor por defecto es 10.

Ejemplo 1: **IN** <RETURN>

Da a la primera línea el número 10, incrementando las siguientes de 10 en 10 (10, 20, 30, etc.).

Ejemplo 2: **IN 100** <RETURN>

Da a la primera línea el número 100, incrementando las siguientes de 10 en 10 (100, 110, 120, etc.).

Ejemplo 3: **IN 100, 2** <RETURN>

Da a la primera línea el número 100, incrementando las siguientes de 2 en 2 (100, 102, 104, etc.).

Además del comando IN, se puede introducir una línea directamente, simplemente con teclear su número, seguido del texto correspondiente, y <RETURN>.

NOTA: Dentro del modo IN, las líneas que ya están ocupadas, aparecerán seguidas de un asterisco (*). Para salir de IN, pulse <CTRL+C> ó <CTRL+STOP>.

Comando: **LE** (Listar Etiquetas)

Sintaxis: **LE[,I]** <RETURN>

Lista todas las etiquetas y macros que se producen después de un ensamblado (o linkado) con sus valores en decimal y hexadecimal. Las etiquetas permanecerán en memoria, hasta que se produzca la mas mínima variación del código fuente (cargar un fichero o modificar, añadir o borrar una línea). En caso de no haber ni etiquetas ni macros, el comando queda sin efecto pero no provoca error.

Ejemplo: **LE** <RETURN>

Lista todas las etiquetas y macros con sus respectivos valores en decimal y hexadecimal.

Cuando se utiliza la opción **[,I]**, el listado se efectúa por **impresora**.

NOTA: Aunque las macros son listadas junto con las etiquetas, se distinguen de éstas porque no llevan los dos puntos finales. Puesto que el valor de una macro no puede ser fijo, ya que varía cada vez que se llama, el valor que aparece asociado a ella es, simplemente, la dirección que ocupa dentro del fichero de texto.

Comando: **LK** (LinKar)

Sintaxis: **LK "nombre[.ext]",["nombre[.ext]"],...**

Linka los ficheros de texto encerrados entre comillas y separados por comas (si omite la extensión, ésta será **ASM**).

Por motivos de compatibilidad con todas las máquinas MSX, el comando LK dispone de **16 K** de memoria para almacenar las etiquetas de todos los ficheros fuente que se linkan. Por tanto, es posible linkar tantos ficheros como se quiera, siempre y mientras que el espacio lo permita. Aún así, con 16 K, pueden linkarse de 15 a 20 ficheros fuente (valor más que respetable) que contengan una cantidad de etiquetas "normal". No obstante, si se diera el caso de que hay tantas etiquetas que se rebasa la memoria disponible, se provocará el **error 6**.

El comando **LK**, funciona igual que **EN** y, además de la información de éste, solicita la siguiente:

Bytes Macros:

para que se le indique la cantidad de bytes que ha de reservar para las macro. Si los ficheros a linkar, no contienen macro alguna, pulse <RETURN>. De lo contrario, introduzca una cantidad de bytes que sea proporcional al número de macros que haya (para calcular la cantidad de bytes que ocupa una línea, cuente los caracteres que la componen, excepto el número de la línea y los espacios en blanco, y súmele 6).

Ejemplo: **LK "PARTE1", "PARTE2", "PARTE3", "PARTE4"**

Linka los ficheros fuente: PARTE1.ASM, PARTE2.ASM,

PARTE3.ASM y PARTE4.ASM

Al contrario que **EN**, **LK** interrumpe el linkado mostrando el **error 3**, si en alguno de los ficheros se llama a una etiqueta que no se ha definido en ninguno de ellos.

Puesto que, por cuestiones de espacio, no es posible colocar todos los ficheros en memoria, **LK** dispone de un buffer para ir depositando, por partes, los ficheros que debe enlazar. En la primera pasada, actúa de la misma forma que el comando **EN**. Por el contrario durante el segundo paso, carga (en el buffer) pequeñas partes de código fuente y las ensambla, por lo que, aparte de los **16K** para almacenar las etiquetas, no hay ningún límite en la cantidad de ficheros a linkar.

El resultado de todo ello, es que **LK** está constantemente accediendo al disco (o al casete) y, aún dependiendo de la velocidad de acceso de la unidad de disco, el linkado se efectúa prácticamente a la misma velocidad que el ensamblado.

Consecuentemente, es aconsejable que todos los ficheros que se hayan de linkar de una vez (en el ejemplo: PARTE1, PARTE2, etc.), se encuentren en el mismo disquete. No obstante, si no es así, el programa se encarga de dar el aviso correspondiente:

Cambie el disco y pulse una tecla.

pudiendo efectuar el cambio de disquete, sin interrumpir el linkado, tantas veces como sea preciso.

NOTA: *Cuando se omite la extensión, el programa la coloca automáticamente (**ASM** en este caso. Para mas información, véase el apartado 5.2).*

Comando: **LT** (Listar Texto)

Sintaxis: **LT [x1][-][x2][,I]** <RETURN>

Lista la parte del fichero fuente que hay en memoria, comprendida entre las líneas **x1** y/o **x2**, ambas inclusive. Si se omiten **x1** y **x2**, se listará el fichero en su totalidad. En caso de estar la memoria vacía, el comando queda sin efecto y no se provoca error.

Ejemplo 1: **LT 100 - 500** <RETURN>

Lista desde la línea 100 hasta la línea 500 (ambas inclusive).

Ejemplo 2: **LT 100-** <RETURN>

Lista desde la línea 100 (inclusive) hasta la última línea del fichero.

Ejemplo 3: **LT 100** <RETURN>

Lista únicamente la línea 100.

Ejemplo 4: **LT -100** <RETURN>

Lista desde la primera línea del fichero hasta la línea 100 (inclusive).

NOTA: Si se añade la opción **[,I]**, en todos los casos, el correspondiente listado se efectúa por impresora.

Comando: **MV** (MoVer)

Sintaxis: **MV x1,x2,x3** <RETURN>

Mueve el bloque del fichero de texto que hay en la memoria, comprendido entre las líneas **x1** y **x2** (ambas inclusive) y lo deposita inmediatamente a continuación de la línea **x3**. Seguidamente, se renumera todo el fichero. Si no hay código fuente en memoria, o bien alguno de los números de línea indicados no existe, se provoca el **error 2**.

Ejemplo 1: **MV 100, 500, 1000** <RETURN>

Mueve el bloque de texto comprendido entre las líneas 100 y 500, y lo deposita a continuación de la línea 1000.

Ejemplo 2: **MV 500, 1000, 100** <RETURN>

Mueve el bloque de texto comprendido entre las líneas 500 y 1000 y lo deposita a continuación de la línea 100.

Comando: **NU** (NUEvo)

Sintaxis: **NU** <RETURN>

Al finalizar la introducción de un fichero de texto y después de grabarlo con el comando **GT**, el comando **NU** borra por completo los **16.255 bytes** de que dispone el Ensamblador. De esta manera, la memoria queda inicializada para recibir un fichero nuevo.

Si la memoria ya está vacía, este comando no tiene efecto alguno y no provoca error.

Ejemplo: **NU** <RETURN>

Borra los 16.255 bytes de memoria del Ensamblador.

NOTA: No olvide grabar el fichero que tiene en memoria antes de utilizar el comando **NU**, ya que después de su ejecución, no se puede recuperar lo que había en ella.

Comando: **PO** (POkear)

Sintaxis: **PO x1,x2** <RETURN>

Coloca el valor **x2** (máximo **255**) en la dirección de memoria **x1** (máximo **65.535**). Si uno (o ambos) de los dos parámetros excede de los valores indicados, se provocará el **error 5**.

Ejemplo 1: **PO &H8000, &H76** <RETURN>

Coloca el valor **&H76** en la dirección de memoria nº **&H8000** (32768)

Ejemplo 2: **PO 50000, "A"** <RETURN>

Coloca el valor ASCII de la letra A (**&H41** ó 65) en la dirección de memoria nº 50000.

Comando: **PR** (PRobar)

Sintaxis: **PR[#] x1** <RETURN>

Ejecuta un programa en código máquina, donde **x1** es la dirección de arranque. Lógicamente, el código objeto debe encontrarse en memoria, ya sea cargándolo (comando **CB**) o ensamblando ficheros fuente desde RSC II.

Ejemplo: **PR &H8800** <RETURN>

Pone en marcha un programa, cuya dirección de ejecución es **&H8800**.

Para poder ejecutar todo tipo de programas, el comando **PR** configura la memoria de esta manera:

Páginas 0 y 1 = ROM (BIOS y ROM BASIC)

Páginas 2 y 3 = RAM accesible al BASIC

lo cual permite poner en marcha los programas (comerciales o no) que se carguen desde BASIC.

La opción **[#]**, no modifica la configuración de los slots y se utiliza para probar rutinas o programas que se hallan en otro slot, o subslot. Cuando utilice esta opción, siga las instrucciones dadas para la opción **[#]** en el apartado 4.5

NOTA: Si el programa arrancado con **PR** en cualquiera de sus modalidades, no modifica el **SP** y finaliza con un **RET**, el editor de RSC II volverá a tomar el control.

Comando: **RE** (REnumerar)

Sintaxis: **RE [x1][,x2][,x3]** <RETURN>

Renombra líneas de texto, donde **x1** es el **nuevo** número de línea. **x2** el **antiguo** número de línea y **x3** el **incremento**. El valor por defecto de los parámetros es: **x2=1, x1=10, x3=10**. Cuando alguna línea de las que se indican (x1, x2, x3) no existe, se provoca el **error 2**.

Ejemplo 1: **RE** <RETURN>

Renombra todo el fichero, dando a la primera línea el número 10 e incrementando las siguientes de 10 en 10 (10, 20, 30, etc.).

Ejemplo 2: **RE 100** <RETURN>

Renombra todo el fichero, dando a la primera línea el número 100 e incrementando las siguientes de 10 en 10 (100, 110, 120, etc.).

Ejemplo 3: **RE 1000,100** <RETURN>

Renombra el fichero solamente a partir de la línea 100, que pasa a ser la 1000, e incrementa las siguientes de 10 en 10 (1000, 1010, 1020, etc.).

Ejemplo 4: **RE 1000, 100, 2** <RETURN>

Renombra el fichero solamente a partir de la línea 100, que pasa a ser la 1000, e incrementa las siguientes de 2 en 2 (1000, 1002, 1004, etc.).

Comando: **RN** (ReNombrar)

Sintaxis: **RN "nombre1.ext","nombre2[.ext]"**

Renombra el fichero **nombre1.ext** que pasa a llamarse **nombre2.ext**. Cuando el fichero solicitado no se halla en el disco, se provoca el **error 16**.

Ejemplo 1: **RN "EJEMPLO1.BIN", "EJEMPLO2.RSC"**

Renombra el fichero llamado EJEMPLO1.BIN, que pasa a llamarse EJEMPLO2.RSC.

Ejemplo 2: **RN "EJEMPLO1.ASM", "EJEMPLO2"**

Renombra el fichero llamado EJEMPLO1.ASM, que pasa a llamarse EJEMPLO2 (sin extensión).

NOTA: Cuando el programa que se quiere renombrar, tiene extensión (en los ejemplos anteriores: EJEMPLO1.BIN y EJEMPLO1.ASM), ésta habrá de ser puesta **OBLIGATORIAMENTE** dentro del nombre actual del programa, es decir, el que sigue al comando **RN**.

Comando: **SI** (SIMular)

Sintaxis: **SI[#] x1[,I]** <RETURN>

Simula un programa en lenguaje máquina, empezando por la dirección **x1**.

Ejemplo: **SI &H8800** <RETURN>

Empieza la simulación del programa, situado en el slot que está activo en ese momento, por la dirección **&H8800**.

Al utilizar este comando, se mostrará la siguiente pantalla:

Slot:		11111111				I R		P C		
		10101010				004A		8000		
A		B C	D E	H L	I X	I Y	SZ H	PNC		
00 .		0000	0000	0000	0000	0000	00000000			
00 .		0000	0000	0000			00000000			
							SP: FAE0			
		00 .	00 .	00 .	00 .	00 .	0000			
		00 .	00 .	00 .	00 .	00 .	0000			
		00 .	00 .	00 .	00 .	00 .	0000			
		00 .	00 .	00 .	00 .	00 .	0000			
		00 .	00 .	00 .	00 .	00 .	0000			
		00 .	00 .	00 .	00 .	00 .	0000			
		00 .	00 .	00 .	00 .	00 .	0000			
		00 .	00 .	00 .	00 .	00 .	0000			
		41 A	41 A	41 A	41 A	41 A	0000			
		55 U	55 U	55 U	55 U	55 U	FAE0			
32768	C9					RET				

En la primera línea aparece la configuración actual de los slots. Inmediatamente a continuación, se muestra la configuración actual de subslots (si su ordenador carece de ellos prescinda de esta infor-

mación) y el valor de los registros **I**, **R** y **PC**. Seguidamente aparece el resto de registros del Z-80, primero el juego que se halla conectado actualmente (**A**, **BC**, **DE**, **HL** y **F**), luego el segundo juego de registros (**A'**, **B'C'**, **D'E'** y **F'**) y el valor del **SP**. Finalmente, se muestra el contenido de la posición de memoria indicada por cada registro (y el de las nueve posiciones siguientes), los diez últimos datos depositados en la pila (**SP**), y la dirección de memoria en la que se encuentra el **PC**, con su código objeto y código fuente correspondiente.

A partir de ese momento se dispone de nueve comandos, representados por una letra, que permiten realizar diversas funciones, son los siguientes:

E = Ejecuta, REALMENTE, el paso actual (el que hay en pantalla) y pasa al siguiente. Mucho cuidado en las instrucciones del tipo **OUT (&HA8),A**.

S = Pasa (Salta) al siguiente paso sin ejecutar el actual. Este comando no produce ningún efecto y se limita a desensamblar por pasos.

C = Cambia el contenido de una, o varias, posiciones de memoria. Al pulsar este comando, aparecerán dos puntos (:) que indican que el Simulador se encuentra en disposición de recibir datos. Puede usted cambiar el contenido de una dirección de memoria, bien sea introduciendo directamente el código objeto, o por medio de código fuente. Por ejemplo: Vamos a introducir a partir de la dirección **&H8000** el comando **CALL 50000** (cuyo código máquina corresponde a **CD, 50, C3**). Pulsamos la letra "**C**", aparecen los dos puntos y escribimos: **&H8000 CALL 50000** <RETURN>, o bien: **&H8000 &HCD, &H50, &HC3** <RETURN>.

Veamos otro ejemplo: vamos a cambiar esa misma dirección por el comando **RET** (código máquina **&HC9**). Al aparecer los dos puntos, tecleamos: **&H8000 RET** <RETURN>, o bien: **&H8000 &HC9** <RETURN> Cualquiera que sea el modo que se utilice, NO COLOQUE mas espacios en blanco que los que aparecen en los ejem-

plos anteriores o provocará error y se interrumpirá la simulación.

L = Llama una rutina y la ejecuta de una sola vez. Este comando es útil cuando una rutina es muy larga y no se quiere ejecutarla por pasos. NO utilice el comando **L** para instrucciones de salto (**JR**, **DJNZ** ó **JP**) si no quiere bloquear el ordenador.

B = Permite cambiar los Bancos (slots) de memoria. Pulse **B** y aparecerán dos puntos (:). Introduzca la nueva configuración (preferiblemente en binario) y pulse <RETURN> (p.e.: **&B11111100**). Si quiere modificar también el subslot, introdúzcalo a continuación del slot, separándolo de éste por un guión, y pulse <RETURN> (p.e.: **&B11111100 - &B10101000**). Procure no cambiar NUNCA la **página 1** (bits **2** y **3**), ya que es donde se aloja RSC II, ni la **página 3** (bits **6** y **7**), que debe estar en contacto permanente con la ROM, porque bloqueará el ordenador (los bits se numeran de **0 a 7** empezando por la derecha).

I = Coloca una interrupción en cualquier parte del programa. De este modo, es posible ejecutar de una sola vez, un cierto número de instrucciones. Pulse **B** y aparecerán dos puntos (:). Introduzca, sin dejar espacios, la dirección de memoria donde deberá producirse la interrupción y pulse <RETURN>. Tenga mucho cuidado donde coloca la interrupción, ya que si ésta no se pone al principio de una instrucción los resultados son imprevisibles.

R = Este comando permite llamar a una rutina situada en otro slot, sin tener que cambiar la configuración del mismo, con el comando **B**. El programa solicitará el slot y/o subslot, en el que se halla la rutina en cuestión (con el mensaje **Slot:**). Responda siguiendo las instrucciones dadas en el apartado 4.5 (opción [#]). NO utilice el comando **R** para instrucciones que no sean del tipo: **CALL** o **RST**.

M = Permite modificar el contenido de los dos jue-

gos de registros del Z-80 mas **IX** e **IY**. Al pulsar **M** el cursor se situará en la primera línea de registros (los actuales). Procure no equivocarse ni salirse de los límites que marcan los propios registros, o provocará error y se interrumpirá la simulación. Tanto si hace alguna modificación, como si no, finalice con <RETURN> y pasará a la siguiente línea (**registros de reserva**). Actúe de igual modo.

O = Permite modificar el resto de registros, o sea **PC** y **SP** (los valores de **I** y **R** son puramente informativos y no pueden ser modificados). Pulse "**O**" y aparecerán dos puntos (:). En función del registro que quiera cambiar, introduzca (sin espacios) una de las siguientes expresiones:

PC = x1	<RETURN>
SP = x1	<RETURN>

En cualquier caso, **x1** ha de ser el nuevo valor que desea darle al registro en cuestión. Sea cuidadoso al modificar el **SP**.

Si una vez pulsados los comandos: **C**, **B**, **I**, **R**, **M** ú **O** no desea utilizarlos, puede anular su actuación pulsando <CTRL+STOP>.

NOTA: Cuando se utiliza la opción [**I**], el listado se efectúa por **impresora**.

ATENCIÓN: Al utilizar el comando **I** (Interrupción) del Simulador, cuando el número de instrucciones a simular (antes de la interrupción) sea muy grande, se dará el caso de que el programa tardará bastante tiempo en ejecutarlas. Ello puede dar la impresión de que el ordenador se ha quedado bloqueado. En estos casos, hay que tener un poco de paciencia y esperar a que termine.

Comando: **TR** (TRasladar)

Sintaxis: **TR[#] x1,x2,x3** <RETURN>

Traslada el bloque de bytes, comprendido entre las direcciones **x1** y **x2** (ambas inclusive) y lo deposita a partir de la dirección **x3**.

Ejemplo 1: **TR &H8000, &H9000, &HA000** <RETURN>

Traslada el contenido del bloque comprendido entre las direcciones **&H8000** y **&H9000**, y lo deposita en la dirección **&HA000** y siguientes.

Cuando utilice la opción **[#]**, siga las instrucciones dadas para la misma en el apartado 4.5

Ejemplo 2: **TR# &H4000, &H7FFF, &H8000** <RETURN>

Traslada el bloque de bytes, comprendido entre las direcciones **&H4000** y **&H7FFF** (del slot que se haya indicado), y lo deposita a partir de la dirección **&H8000** del slot que está activo, es decir, el slot que contiene memoria RAM. Cuando se utiliza la opción **[#]** en este comando, el traslado tarda varios segundos en efectuarse (dependiendo del tamaño del bloque a trasladar). Esto podría, en un principio, dar la impresión de que el ordenador se ha quedado bloqueado.

Comando: **VD** (Ver Directorio)

Sintaxis: **VD[,I]** <RETURN>

Muestra todos los ficheros que se encuentran en el disco, con su tamaño en bytes y la fecha y hora de grabación.

Ejemplo 1: **VD** <RETURN>

Lista por pantalla, el directorio del disco que se encuentra dentro de la unidad.

Ejemplo 2: **VD, I** <RETURN>

Lista por **impresora** el directorio del disco que se encuentra dentro de la unidad.

NOTA: *Este comando sólomente es válido para RSC II en versión disco.*

Comando: **VE** (VERificar)

Sintaxis: **VE "nombre" [,x1][-x2]** <RETURN>

Este comando va unido al comando **GT**, y verifica si el bloque de texto comprendido entre las líneas **x1** y **x2** (ambas inclusive) es el mismo que se acaba de grabar con **GT**, en el casete. Si **x1** y **x2** son omitidos la verificación se hace de todo el fichero.

Ejemplo 1: **VE "PRUEBA"** <RETURN>

Verifica si el fichero que hay en memoria es idéntico al que se acaba de grabar en el casete, con el comando: **GT "PRUEBA"**.

Ejemplo 2: **VE "PRUEBA", 100 - 500** <RETURN>

Verifica si el bloque de texto, que comprenden las líneas 100 y 500 (ambas inclusive) del fichero que hay en memoria, es igual al que se acaba de grabar en el casete, con: **GT "PRUEBA",100-500**.

NOTA: Este comando *SOLAMENTE* es válido para RSC II en versión *CASETE*, puesto que en la versión *disco*, los comandos de grabación activan, por sí mismos, el flag de verificación.

Comando: **VM** (Volcar Memoria)

Sintaxis: **VM[#] x1[-x2][,I]** <RETURN>

Efectúa un volcado de memoria hexadecimal (con sus correspondientes valores ASCII) de las direcciones comprendidas entre **x1** y **x2** (ambas inclusive).

Ejemplo 1: **VM &HC000** <RETURN>

Efectúa sólo el volcado de la dirección **&HC000**.

Ejemplo 2: **VM &H8000 - &HC000** <RETURN>

Efectúa el volcado desde la dirección **&H8000** hasta la **&HC000** (ambas inclusive).

Al utilizar la opción **[#]**, siga las instrucciones dadas para la misma en el apartado 4.5

NOTA: Cuando se utiliza la opción **[,I]**, el listado se efectúa por **impresora**.

3. INTRODUCCIÓN DE LÍNEAS DE TEXTO

La introducción de código fuente se efectúa a base de líneas numeradas (igual que en BASIC). Una línea puede contener hasta cuatro campos distintos:

NÚMERO	ETIQUETA	INSTRUCCIÓN	COMENTARIO
10	SUMA:	LD A,B	;Carga A con B
20		LD C,20	;Carga C con 20
30		ADD A,C	;Suma A y C
40		RET NZ	;Retorna si A<>0
50		JR SUMA	

NÚMERO = Es imprescindible colocarlo puesto que es lo que indica que se trata de una línea.

ETIQUETA = Es opcional y puede ponerse siempre que sea preciso. Cuando se define una etiqueta, es decir cuando se halla al principio de la línea (p.e: SUMA:), ésta debe finalizar, OBLIGATORIAMENTE, con dos puntos (:). Podrá contener un máximo de 8 caracteres que, a excepción del primero que debe ser SIEMPRE una letra, pueden ser números, letras o la combinación de ambos. Al llamar a una etiqueta, es decir, cuando forma parte de la instrucción (p.e.: JR SUMA), no debe llevar NUNCA los dos puntos finales. Es importante destacar, que no hay palabras reservadas para las etiquetas, por lo que es posible que estén formadas por nombres de comandos del lenguaje ensamblador del Z-80 ó pseudocomandos.

INSTRUCCION = Se divide en dos partes: el COMANDO (p.e.: RET ó JR) y el OPERANDO (p.e.: NZ ó SUMA). El comando es imprescindible (a menos que la línea esté formada únicamente por un comentario) y puede ser cualquiera de los autorizados para Z-80, ó uno de los **11 pseudocomandos** que admite RSC II (ver apartado 4.1). El operando, cuando lo haya (algunas instrucciones no lo llevan), puede formarse de registros, condiciones, direcciones, etc.. Cuando se trate de algún valor numérico, éste puede ser dado mediante una etiqueta, o en uno de los tres siste-

mas numéricos que se admiten (decimal, hexadecimal ó binario). Cuando se trate de valores hexadecimales o binarios, éstos deberán ir precedidos de: **&H** y **&B**, respectivamente.

COMENTARIO = Es opcional y puede ponerse cuando se desee. Deber ir precedido de un punto y coma (;), seguido (o no) del texto correspondiente.

A la hora de introducir una línea, no es necesario dejar ningún espacio en blanco (aunque pueden ponerse los que se quieran) para separar los distintos campos que la componen, EXCEPTO el espacio que separa el comando del operando (si lo hay), dentro del campo de la instrucción, que deberá ser respetado SIEMPRE.

Cuando quiera que una línea se componga simplemente de una etiqueta, acompáñela de un punto y coma. Por ejemplo: **ETIQUETA: ;**

4. OTROS COMANDOS Y FUNCIONES

4.1 PseudoCOMANDOS

RSC II admite hasta **11 pseudocomandos** que efectúan diversas funciones y son de gran utilidad a la hora de programar en lenguaje ensamblador. Los pseudocomandos solamente funcionan dentro de una línea (modo programa) y no deben ser introducidos en modo directo. Los parámetros que acompañan a algunos de ellos (**x1**, **x2**, etc.), se pueden dar mediante un valor numérico, una etiqueta ó una expresión matemática (ver apartado 4.4).

Comando: **ORG**

Sintaxis: **ORG x1**

Marca la dirección donde debe depositarse el código objeto que es generado al ensamblar un fichero. El Ensamblador dispone de un **contador** que marca la dirección donde ha de depositarse el código objeto ó lenguaje máquina. Este contador toma por defecto el valor **&H8000** (32768) al empezar a ensamblar. Si en una línea se encuentra un **ORG**, el contador toma el valor que éste le indica (**x1**) y empieza a depositar el código objeto en esa dirección.

Comando: **DEFB**

Sintaxis: **DEFB x1,x2,x3,...xn**

Define desde **1** hasta **n** valores (tantos como quepan en una línea) de **un byte**, que serán depositados a partir de la dirección que marca el contador en ese momento.

Comando: **DEFM**

Sintaxis: **DEFM "<cadena>"**

Define una <cadena> con tantos caracteres alfanuméricos (encerrados entre comillas) como quepan en una línea. Los respectivos valores ASCII son depositados a partir de la dirección que marca el contador en ese momento.

NOTA: Véase la diferencia entre **DEFB 9** y **DEFM "9"**.

Comando: **DEFW**

Sintaxis: **DEFW x1,x2,x3,...xn**

Define desde **1** hasta **n** valores (tantos como quepan en una línea) de **dos bytes**, que son depositados a partir de la dirección que marca el contador en ese momento.

Comando: DEFS

Sintaxis: **DEFS x1**

Reserva una zona de memoria. Cuando el ensamblador encuentra el comando **DEFS**, rellena con un **0** (cero) tantos bytes como le indique **x1** (es decir, el contador se incrementa **x1** bytes) y sigue ensamblando a continuación.

Comando: EQU

Sintaxis: **EJEMPLO: EQU x1**

Permite definir una etiqueta con un valor concreto donde **EJEMPLO** es un nombre de etiqueta (máx. 8 caracteres) y **x1** el valor que se le asigna.

Puesto que las etiquetas toman sus valores, dependiendo de la posición que ocupan dentro del código fuente, y la dirección en la que éste se ensambla, el comando **EQU** es muy útil para asignar a una etiqueta un valor fijo, independientemente de la dirección que indique **ORG**.

Comando: END

Sintaxis: **END**

El Ensamblador RSC II efectúa dos pasadas. Durante la primera de ellas el comando **END** es ignorado. En el segundo paso, **END** indica el final del ensamblado y todas las líneas que haya tras él serán ignoradas (no ensambladas). Por contra, al linkar (comando **LK**), END es ignorado en ambas pasadas.

Comando: MAC

Sintaxis: **EJEMPLO: MAC**

Define una MACro, donde **EJEMPLO** será el nombre que se le asigna (**máx. 8 caracteres**). Todas las macros deben empezar con una etiqueta y el comando **MAC**.

Comando: **ENDM**

Sintaxis: **ENDM**

Marca el final de una macro. Las macros han de finalizar SIEMPRE con **ENDM**.

NOTA: Pueden anidarse entre sí hasta **16 macros** (al llamarlas, NO al definir las). Es decir, dentro de una macro se puede llamar a otra, ó a otras.

Comando: **IF**

Sintaxis: **IF x1**

Condiciona el ensamblado al resultado de la expresión **x1**. Cuando **x1** es igual a **cero**, las líneas que le siguen no son ensambladas. Por el contrario, si **x1** es diferente de cero, se ignora la condición IF y el ensamblado prosigue normalmente.

Comando: **ENDI**

Sintaxis: **ENDI**

Da por finalizada una condición IF, cualquiera que sea, y las líneas que siguen se ensamblan con toda normalidad. Las condiciones IF, siempre han de finalizar con **ENDI**.

NOTA: No es posible anidar entre sí, las condiciones IF/ENDI.

4.2 Comandos del Ensamblador

El Ensamblador admite hasta tres comandos que, sin ejercer ninguna influencia sobre el código objeto, proporcionan ciertas utilidades a la hora de hacer los listados. Estos comandos han de precederse del signo "^" y son los siguientes:

Comando: ^C

Sintaxis: ^C "<cadena>"

Permite definir una <cadena> alfanumérica (que debe ir encerrada entre comillas) de hasta **45 caracteres**, que se añade a la cabecera de presentación.

Comando: ^P

Sintaxis: ^P

Efectúa un salto de página inmediatamente a continuación de la línea en la que se encuentra.

Comando: ^S

Sintaxis: ^S

Detiene el listado, y el ensamblado, hasta que se pulsa una tecla.

4.3 Macros

Una macro, abreviación de macro-instrucción, es un conjunto de instrucciones que deben utilizarse repetidas veces a lo largo de un programa. Veamos un simple ejemplo:

```

10 TRASLADA:  MAC
20           LD HL,&H0100
30           LD DE,&H8000
40           LD BC,&H3F00
50           LDIR
60           ENDM

```

Cuando en algún programa deben utilizarse a menudo instrucciones como éstas, se ahorrará tiempo y memoria definiendo una macro. De este modo, cada vez que se necesite ensamblar el texto del que se compone, no será necesario repetirlo. Simplemente, se llama a la macro por su nombre. Por ejemplo:

4800 TRASLADA

Cuando el Ensamblador encuentre una línea donde se llama a una macro, ensamblará (en la dirección que marque el contador en ese momento) todas las instrucciones de las que está compuesta. Pero observe que los valores que contiene (&H0100, &H8000 etc.) son siempre los mismos. Para que las macros puedan ser aún más útiles, es posible sustituir cualquier valor numérico por una variable, representada por un **interrogante**. El ejemplo anterior quedaría así:

```

10 TRASLADA:  MAC
20           LD HL, ?
30           LD DE, ?
40           LD BC, ?
50           LDIR
60           ENDM

```

En este caso, cada vez que se llama a la macro, el nombre de la misma debe ir seguido de tantos valores numéricos, separados por comas, como variables (o interrogantes) haya dentro de ella. De este modo, es posible darle en cada caso el valor que mas convenga. Por ejemplo:

4800 TRASLADA &H6F00, &H9E40, &H2A80

El valor **&H6F00** tomaría el lugar del **primer interrogante**, **&H9E40** el del **segundo** y así sucesivamente. Estos valores, podrían darse mediante una etiqueta ó expresión matemática.

NOTA: Una macro no puede contener ninguna etiqueta más que la del nombre que la define.

4.4 Signos aritméticos

En ocasiones, es interesante poder efectuar operaciones matemáticas entre etiquetas y/o valores numéricos. El Ensamblador puede efectuar **ocho** de estas **operaciones** (dentro de una línea) que son:

+	SUMA
-	RESTA
*	MULTIPLICACIÓN
/	DIVISIÓN ENTERA
!	AND
\$	OR
%	XOR
#	MOD

Veamos un par de ejemplos:

```
10 LD A, 8 + 4 $ 2 * ETIQUETA - 10
```

```
20 SUB ETIQUETA / 4 * 2
```

NOTA: Ningún signo tiene preferencia sobre los demás, por lo que éstos serán calculados en el orden en que se van encontrando.

4.5 Slots/subslots

Todos los comandos que disponen de la opción **[#]**, actúan en las direcciones pertenecientes al slot o slots que se hallan activos (y que varían según el ordenador). Pero RSC II permite inspeccionar cual-

quier slot (ó subslots, si los hay) de la máquina. Por ello, al utilizar la opción [#], aparece:

Slot:

Introduzca el número de slot y pulse <RETURN>. Si se trata de un subslot, indíquelo después del slot (separado de éste por un guión, p.e.: 3-2) y pulse <RETURN>.

5. INFORMACION IMPORTANTE

5.1 Direcciones inferiores a &H8000

Cuando se ensambla (o se linka) en direcciones inferiores a &H8000, el código objeto o lenguaje máquina que se genera no puede ser depositado en estas direcciones, porque allí se encuentra RSC II y el fichero de texto (o código fuente) introducido, y, si así se hiciera, el programa desaparecería de la memoria. Por ello, en estos casos, el Ensamblador, y/o Linkador, actúan del siguiente modo:

Cuando la dirección indicada por **ORG**, sea inferior a &H4000 (de 0 a &H3FFF), el código objeto correspondiente se depositará en la dirección indicada + &H8000. Por ejemplo, si un programa empieza en la dirección &H100 y termina en &H2000, el código objeto generado se encontrará entre las direcciones: &H8100 (&H100+&H8000) y &HA000 (&H2000+&H8000).

Cuando la dirección indicada por **ORG**, se encuentre comprendida entre &H4000 y &H7FFF, el código objeto correspondiente será depositado en la dirección indicada + &H4000. Por ejemplo, si un programa empieza en la dirección &H4000 y finaliza en &H7FFF, el código objeto generado será depositado entre la dirección &H8000 (&H4000+&H4000) y &HBFFF (&H7FFF+&H4000).

Sin embargo, el programa se habrá ensamblado perfectamente para funcionar en la dirección indicada por **ORG** y los listados y etiquetas (si las hay) se referirán a esa dirección.

Esta particularidad, lejos de ser un inconveniente es una ventaja, ya que igualmente el código objeto debería ser trasladado a partir de **&H8000** para ser grabado y, posteriormente, cargado desde BASIC.

No obstante, a la hora de grabar el código objeto, debe tenerse en cuenta en qué dirección se encuentra éste, a fin de dar correctamente las direcciones inicial, final, etc., en comandos de grabación como **GB** o **GC**. Si, por ejemplo, se graba un programa **COM**, que siempre empieza en la dirección **&H100**, el código objeto se encontrará a partir de **&H8100**, y ésta será la dirección inicial que debe indicarse al grabarlo.

5.2 Grabación y carga de programas

Cuando en los comandos **CB**, **CC**, **CT**, **GB**, **GC** y **GT**, se omite la extensión, ésta es colocada, automáticamente, por RSC II (**BIN** para **CB** y **GB**, **COM** para **CC** y **GC** y **ASM** para **CT** y **GT**). Para grabar o cargar algún programa sin extensión, debe ponerse únicamente el punto dentro del nombre del programa. Por ejemplo: **GT "EJEMPLO1."**, **CB "EJEMPLO2."**, etc..

5.3 Consejos y notas finales

- Cuando se linkan varios ficheros fuente es aconsejable, a fin de evitar errores, que solo el primero de ellos marque (con el pseudocomando **ORG**) la **dirección de inicio**. De esta manera, por mucho que se modifiquen los distintos ficheros, no habrá que preocuparse de cambiar la dirección en la que debe

empezar cada uno de ellos, puesto que serán empalmados uno tras otro.

- Todos los listados podrán ser detenidos parcialmente pulsando la tecla **<STOP>** (en este caso podrá reanudarse el listado pulsando cualquier tecla), o totalmente pulsando **<CTRL+STOP>**.

- Es posible hacer un **RESET** desde RSC II tecleando lo siguiente: **PR 0** **<RETURN>**.

- Por razones obvias de seguridad, NO se puede **DE**sensamblar, **BU**scar, **SI**mular, **TR**asladar, **VO**lcar **ME**moria, etc., de las direcciones comprendidas entre **&H4000** y **&H7FFF** del slot en el que se encuentra alojado RSC II.

- El pasar de **ENS**amblar a **DE**sensamblar, **VO**lcar **ME**moria, **SI**mular, etc., no afecta para nada al código fuente o código objeto que haya en memoria. EXCEPTO cuando se utiliza la opción **[,G]** del comando **DE** que borra el fichero de texto (si lo hay) de la memoria del Ensamblador.

- Las **etiquetas** pueden tener un **máximo de ocho caracteres** (aparte de los dos puntos). En el caso de que se exceda de esta cantidad, el programa tomará los ocho primeros e ignorará los restantes.

- Una **Macro**, al contrario que una etiqueta, NO podrá ser definida nunca, con el mismo nombre que un comando del Z-80, o el de un pseudocomando.

- Aún cuando las macros suelen definirse al inicio del texto, pueden ser definidas en cualquier parte del mismo. La única precaución a tener en cuenta, es que deberán definirse antes de ser llamadas.

- Los saltos relativos (**JR** y **DJNZ**) y los desplazamientos en los registros índice (**IX** e **IY**), se pueden dar mediante una etiqueta, o bien indicando el número de bytes directamente. En este último caso,

los saltos y desplazamientos positivos (hacia adelante) y negativos (hacia atrás), han de ir precedidos de los signos "+" y "-", respectivamente. En los saltos positivos, puede omitirse el signo "+".

- En todos los comandos que efectúan algún tipo de listado, aparece el mensaje:

Modo (H/D):

que solicita si el listado debe hacerse en **Hexadecimal** o **Decimal**. Responda **H** o **D**, según prefiera.

- Durante la **SIMulación** (comando **SI**), el programa utiliza la zona comprendida entre **&HF959** y **&HFAF4**.

- Los números **Binarios** y **Hexadecimales**, deberán ir precedidos de **&B** y **&H** respectivamente. Todo número que no se preceda de ninguno de estos dos símbolos será tratado como decimal.

- Obsérvese que mediante el comando **PR**, puede ejecutarse cualquier rutina del **B.I.O.S.**

5.4 Errores

Error 01 = Error de sintaxis. Se ha cometido algún error a la hora de escribir un comando.

Error 02 = Función ilegal. Se ha utilizado un comando de manera no autorizada.

Error 03 = Se ha llamado a alguna etiqueta que no está definida.

Error 04 = Se ha definido la misma etiqueta en dos ocasiones (ya estaba definida anteriormente).

Error 05 = Valor excesivo. Se ha dado algún valor mayor de **65535** (**&HFFFF**), o mayor de **255** (**&HFF**), en

lugares donde sólo se permiten valores de un byte.

Error 06 = Memoria completa. Se ha llenado completamente la memoria (**16255 bytes**), a base de cargar y/o introducir código fuente. También aparece este error si, al **ENsamblar** o **LinKar**, no se dispone de espacio suficiente para depositar las etiquetas.

Error 07 = Impresora desconectada. Se ha utilizado la opción **[,I]** y no hay impresora, o bien ésta no se encuentra ON LINE.

Error 08 = Se ha definido un salto relativo, fuera de los límites permisibles (de **-128** a **+127 bytes**).

Error 09 = Error E/S en disco. Se ha producido algún error al acceder a la unidad de disco.

Error 10 = Interrumpido. Se ha interrumpido un comando que efectuaba cualquier tipo de listado.

Error 11 = Se ha intentado dividir algún valor numérico por cero.

Error 12 = Se ha encontrado un comando **ENDM** o **ENDI** sin que previamente se hubiera definido una macro, con **MAC** o una condición con **IF**, respectivamente.

Error 13 = Disco desconectado. Se ha accedido a la unidad de disco sin que ésta se encuentre conectada, o bien, no hay ningún disquete dentro de ella. En la versión **casete**: Verificar cinta. La grabación efectuada con **GT**, es defectuosa.

Error 14 = Se ha definido una **MACro** (o un **IF**) dentro de otra, o la **MACro** no se ha cerrado con **ENDM**.

Error 15 = El espacio reservado para macros con el comando **LK**, es insuficiente.

Error 16 = Fichero inexistente. Se ha intentado la carga de un programa que no se halla en el disco.

Error 17 = Disco protegido. Se ha intentado grabar en un disco protegido contra escritura.

Error 18 = Disco completo. Se ha intentado grabar en un disco que está totalmente lleno.

Error 19 = ORG incorrecto. El código objeto alcanza la zona de memoria utilizada por el **SP** y **RSC II** (véase el comando **EN**).

Error 20 = Nombre erróneo. El nombre dado a un fichero, contiene caracteres no autorizados.

5.5 Comandos autorizados para Z-80

ADC	ADD	AND	BIT	CALL	CCF	CP
CPD	CPDR	CPI	CPIR	CPL	DAA	DEC
DI	DJNZ	EI	EX	EXX	HALT	IM
IN	INC	IND	INDR	INI	INIR	JP
JR	LD	LDD	LDDR	LDI	LDIR	NEG
NOP	OR	OTDR	OTIR	OUT	OUTD	OUTI
POP	PUSH	RES	RET	RETI	RETN	RL
RLA	RLC	RLCA	RLD	RR	RRA	RRC
RRCA	RRD	RST	SBC	SCF	SET	SLA
SRA	SRL	SUB	XOR			

5.6 Un programa de ejemplo

Este programa de ejemplo le permitirá practicar la mayoría de comandos que se han visto. Para empezar teclee: **IN** <RETURN> e introduzca este texto:

```

10 ;=====
20 ;Programa de prueba
30 ;=====
40 ;
50          ORG &HC000
60 VALOR:   EQU 4

```

70 SIGNO:	DEFM "\$"	
80	LD HL,0	;HL=Inicio VRAM
90 INICIO:	LD A,(SIGNO)	;A=ASC("\$")
100	CALL &H4D	;Rutina VPOKE (BIOS)
110	INC HL	
120	LD A,VALOR	;A=4 (línea 60)
130	CP H	;¿H=4?
140	JR NZ, INICIO	;Si no lo es, repite
150	CALL &H9F	;Pulsar una tecla
160	RET	;Retorna a RSC II

Si tiene un MSX-2 y está trabajando en 80 columnas cambie el número **4** de la línea 60 por un **8**.

Cuando lo tenga introducido, pruebe el comando **LT**. Si quiere grabar el código fuente, utilice el comando **GT** (por ej. **GT "PRUEBA" <RETURN>**). Ahora ensámblelo (**EN <RETURN>**), utilizando la opción **1**. Utilice el comando **LE** para listar las etiquetas. Si no hay errores ni etiquetas ausentes, puede grabar el código objeto con **GB** (para poderlo cargar desde BASIC con **BLOAD**). Para ello teclee:

```
GB "PRUEBA", &HC000, &HC013, &HC001 <RETURN>
```

Y ahora ya puede probarlo (**PR &HC001 <RETURN>**). El programa llena toda la pantalla con el signo "\$". Pulse una tecla, y el control volverá al editor de RSC II. Desensamble (comando **DE**) la parte que comprenden las direcciones **&HC002** y **&HC013** y aparecerá el código fuente que tecleó anteriormente (pero lógicamente, sin pseudocomandos ni etiquetas). Teclee: **LT <RETURN>** y verá como el código fuente (si no lo ha borrado) sigue intacto en memoria. Pruebe a borrarlo (**NU <RETURN>**). Si ahora lista el texto, con **LT**, no aparecerá nada. Vuelva a cargar el texto con: **CT "PRUEBA" <RETURN>**. Si dispone de unidad de disco, utilice el comando **VD** para ver las grabaciones que ha hecho. Trate de utilizar todos los comandos del editor de RSC II, para familiarizarse con ellos.

5.7 Ejemplo de macros anidadas

```

10 ;=====
20 ;Programa que anida 3 macros
30 ;=====
40 ;
50 MACRO1:  MAC
60          LD  A,?
70          ENDM
80 MACRO2:  MAC
90          MACRO1 20
100         LD  C,10
110        ENDM
120 MACRO3: MAC
130        MACRO2
140        ADD A,C
150        ENDM
160 ;
170        ORG &HC000
180        MACRO3
190        RET
200 FINAL:  END

```

Este pequeño programa (muy poco útil por otra parte) es un ejemplo de como anidar las macros. Véase que no puede haber una macro dentro de otra al definirse, es decir, antes de definir otra macro, es preciso cerrar la anterior con **ENDM**.

La línea **180**, llama a la **MACRO3**. Al ejecutarla, se encuentra con que ésta llama a la **MACRO2**, que a su vez llama a la **MACRO1**. Puesto que ésta última contiene una variable (**?**), debe acompañarse al nombre de un valor de un **byte** (ya que el registro **A** solamente puede contener hasta el número **255**). Una vez ejecutada la **MACRO1**, vuelve a la **MACRO2** que otorga un valor a **C** y regresa a **MACRO3** que **suma A y C**. Al finalizar ésta, se vuelve a la línea **190** que marca el **final del programa**.

6. EL PROGRAMA CONVERT

El nuevo Macro-Ensamblador/Linkador RSC II modifica, ligeramente, el formato con que el que se graban los ficheros de texto (código fuente). De este modo, se acelera la velocidad de ensamblado, y los ficheros pueden ser linkados. Por ello, se incluye el programa **CONVERT** que convierte cualquier fichero grabado con el Ensamblador RSC, al nuevo formato de RSC II (en menos de un segundo). Así, pueden aprovecharse todos los ficheros fuente anteriores, sin tener que reescribirlos.

CONVERT puede trabajar desde BASIC o desde RSC II. Para cargarlo desde BASIC, teclee: **BLOAD "CONVERT.BIN",R**. Si desea cargarlo desde RSC II. utilice el comando **CB "CONVERT"** y, cuando esté cargado, ejecútelo con: **PR &HC000**.

Una vez en marcha, el programa le pedirá el nombre del fichero que desea convertir. Introdúzcalo (ENTRE COMILLAS) y pulse <RETURN>. Si omite la extensión, ésta será **ASM**. Cuando termine la conversión, CONVERT pedirá el nombre que desea darle al fichero convertido. Introdúzcalo de igual manera que lo hizo antes (entre comillas) y recuerde que si omite la extensión, ésta será **ASM**. En el caso de querer poner una extensión diferente, colóquela separada del nombre por un punto (p.e.: **"PRUEBA.RSC"**).

Finalmente, el programa preguntará si quiere convertir algún otro fichero. En caso afirmativo, podrá repetir la operación tantas veces como sea necesario. De lo contrario, CONVERT finalizará y devolverá el control al BASIC o a RSC II.

Cuando se cometa algún error de disco, CONVERT finalizará devolviendo el control a donde fue llamado (BASIC o RSC II).