Manual de Usuario

Interfaz de expansión para Sinclair ZX81

SD · AY Sound · Speech · 512KB RAM · RTC

Versión 1.0

Hardware y software de código abierto

Índice de contenidos

Al abrir el documento, haz clic derecho sobre el índice y selecciona «Actualizar campo» para ver los números de página.

Guía de inicio rápido

Si acabas de abrir la caja y quieres empezar cuanto antes, sigue estos cinco pasos:

1. Prepara la tarjeta microSD

- Formatea una tarjeta microSD en FAT32.
- Copia la carpeta SYS y todo su contenido en la raíz de la tarjeta. Sin esta carpeta el interface no arrancará.
- Copia tus archivos .P en la tarjeta, organizados en carpetas si lo deseas.

2. Conecta el interface

- Apaga el ZX81.
- Conecta el SD81 Booster al puerto de expansión trasero del ZX81.
- Inserta la tarjeta microSD en la ranura del interface.

3. Enciende y comprueba

- Enciende el ZX81. Debe arrancar con normalidad mostrando el cursor K.
- El LED STAT del panel superior debe iluminarse en verde fijo.

4. Carga tu primer programa

```
LOAD FAST "NOMBRE"
```

Sustituye NOMBRE por el nombre del archivo sin la extensión .P.

5. ¿Quieres explorar qué hay en la SD?

```
LOAD *DIR
```

> ℹ️ *Para más detalles sobre cualquiera de estos pasos, consulta las secciones correspondientes del manual.*

1. Introducción

El SD81 Booster es una interfaz de expansión de hardware abierto para el Sinclair ZX81 que amplía considerablemente las capacidades originales del ordenador. Entre sus principales características:

- Carga y guardado desde tarjeta microSD en formato .P, el mismo que utilizan los emuladores más populares.
- Hasta 512 KB de RAM mediante un mapeador de memoria en bloques de 8 KB.
- Emulación del chip de sonido AY-3-8910/12 con soporte para el comando PLAY.
- Reproductor de archivos VGM en segundo plano.
- Síntesis de voz con muestras de audio almacenadas en la memoria interna del microcontrolador.
- Hasta 128 caracteres definibles por el usuario, con compatibilidad con el modo de definición de caracteres del interface QuickSilva.

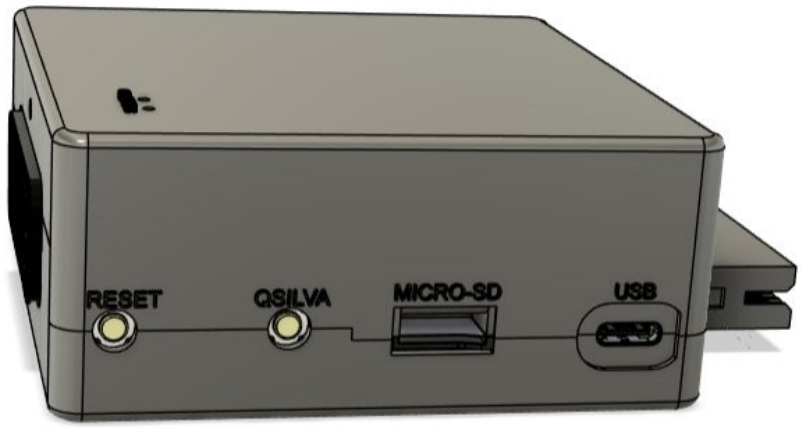
- Compatibilidad con el interface Chroma81, que proporciona salida de vídeo RGB color para el ZX81.
- Compatibilidad total con las rutinas originales de cinta y la ZX Printer.

2. Descripción del hardware

El SD81 Booster es una caja compacta que se conecta al puerto de expansión trasero del ZX81. En su exterior encontrarás todos los conectores y controles necesarios para aprovechar sus funciones.

> ℹ️ *Las imágenes de esta sección son renders provisionales del diseño 3D y serán sustituidas por fotografías del producto final antes del lanzamiento.*

2.1 Panel lateral derecho

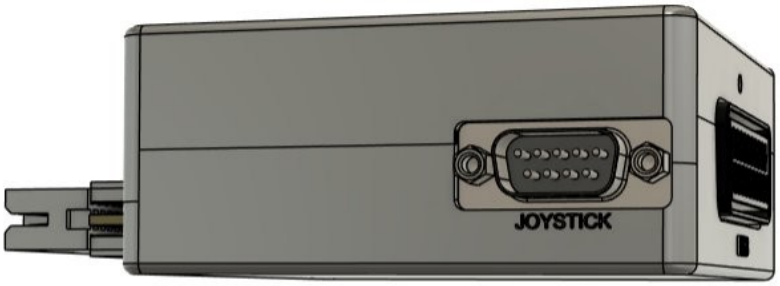


Vista lateral derecha: RESET, QSILVA, MICRO-SD y USB

El panel lateral derecho presenta, de izquierda a derecha:

- Botón RESET: reinicia el sistema completo (ZX81 + interface).
- Botón QSILVA: alterna entre el juego de caracteres del interface QuickSilva y el juego de caracteres estándar de la ROM del ZX81. Cada pulsación conmuta entre uno y otro.
- Ranura MICRO-SD: inserta aquí la tarjeta microSD con los programas, ROMs y datos del sistema.
- Puerto USB-C: puerto de programación del microcontrolador y puerto serie de depuración. No es necesario para el uso normal del interface.

2.2 Panel lateral izquierdo y panel trasero



Vista lateral izquierda: conector JOYSTICK DB9

- Conector JOYSTICK (DB9): puerto de joystick compatible con joysticks estándar de 9 pines (tipo Atari/Commodore). El mapeo de botones a teclas del ZX81 es totalmente programable mediante el comando LOAD *JOY (ver sección 9.6).

Vista trasera: conector RGB SCART

- Conector RGB (SCART): salida de vídeo RGB color compatible con el interface Chroma81. Permite conectar el ZX81 a monitores y televisores con entrada SCART para obtener imagen en color de alta calidad.

2.3 Panel superior — LEDs de estado

Vista superior: LEDs STAT y SD

El panel superior incorpora dos indicadores luminosos:

- LED STAT (estado): indica el estado general del interface mediante diferentes colores y parpadeos (ver tabla en sección 2.4).
- LED SD (acceso a tarjeta): parpadea durante las operaciones de lectura o escritura en la tarjeta microSD.

2.4 Tabla de estados del LED STAT

El LED STAT indica el estado del interface mediante combinaciones de color y parpadeo:

Color / Patrón	Significado
Verde fijo	Interface listo. Sistema funcionando con normalidad.
Amarillo fijo	Cargando archivo desde la tarjeta SD.
Azul fijo	Guardando archivo en la tarjeta SD.
Rojo fijo	Error de acceso a la tarjeta SD.
Púrpura parpadeante	Reproduciendo audio (WAV o síntesis de voz).
Rojo parpadeante rápido	Error crítico del sistema.

3. Contenido de la caja

Al abrir la caja del SD81 Booster encontrarás:

- 1× Interface SD81 Booster

- 1× Tarjeta microSD (preformateada)
- 1× Pila botón CR2032 (preinstalada en el interface)
- Este manual de usuario

> ⚠ *Si alguno de estos elementos falta o está dañado, contacta con el vendedor antes de conectar el interface.*

3. Instalación

3.1 Antes de conectar el interface

- Asegúrate de que el ZX81 está apagado antes de conectar o desconectar el interface.
- El SD81 Booster se conecta al puerto de expansión trasero del ZX81.
- El interface no es compatible con dispositivos que reemplacen la ROM interna del ZX81.

3.2 Conexión

1. Apaga el ZX81.
2. Alinea el conector del SD81 Booster con el puerto de expansión trasero del ZX81. Asegúrate de que los pines están correctamente alineados.
3. Empuja suavemente hasta que el conector quede completamente insertado. No fuerces la conexión.
4. Inserta la tarjeta microSD en la ranura del interface (ver sección 4).
5. Enciende el ZX81.

> ⚠ *Conectar o desconectar el interface con el ZX81 encendido puede dañar tanto el interface como el ordenador.*

3.3 Comprobación de la instalación

Al encender el ZX81 con el SD81 Booster correctamente instalado, el ordenador debe arrancar con normalidad mostrando el cursor K habitual. El interface no modifica el arranque del sistema.

Para verificar que el interface está funcionando, escribe el siguiente comando en BASIC y pulsa ENTER:

```
LOAD *VER
```

> ℹ *El asterisco * se obtiene pulsando SHIFT + B. El interface mostrará la versión del firmware instalado.*

Si el ordenador se queda bloqueado o no arranca correctamente, desconecta el interface y asegúrate de que el conector está bien alineado.

4. Preparación de la tarjeta microSD

4.1 Formato de la tarjeta

El SD81 Booster requiere una tarjeta microSD formateada en FAT32. La tarjeta incluida en la caja ya viene formateada y preparada correctamente.

Si utilizas una tarjeta propia, sigue estos pasos:

6. Formatea la tarjeta en FAT32: en Windows, botón derecho sobre la tarjeta → Formatear → FAT32. En macOS o Linux, utiliza una herramienta de formato de disco y selecciona FAT32.
7. Copia la carpeta SYS y todo su contenido (incluida en el paquete de software del interface) en la raíz de la tarjeta SD. Esta carpeta contiene archivos imprescindibles para el funcionamiento del interface.

> ⚠ *El interface no soporta los formatos exFAT ni NTFS. La carpeta SYS es imprescindible: contiene los archivos de ROM necesarios para el arranque del interface. Sin ella, el interface no funcionará.*

4.2 Caracteres permitidos en nombres de archivo

Debido a las limitaciones del teclado del ZX81, solo se pueden usar los siguientes caracteres en los nombres de archivo y carpeta:

- Letras: A a Z (siempre en mayúsculas al guardar)
- Números: 0 a 9
- Símbolos: . , ; $ ( ) = + -
- El carácter / se utiliza como separador de directorios y no puede usarse en nombres de archivo.

> 💡 *Evita terminar un nombre de archivo con un espacio o un punto, ya que algunos sistemas operativos podrían tener problemas para leer ese archivo.*

4.3 Estructura de carpetas recomendada

El interface buscará en la raíz de la tarjeta SD por defecto. Puedes organizar tus programas en subcarpetas. Se recomienda la siguiente estructura:

```
/
├── AUTOEXEC.P          ← Programa que se carga al arrancar
├── JUEGOS/
│   ├── MANIC.P
│   └── PACMAN.P
├── DEMOS/
└── SYS/                ← Carpeta del sistema (obligatoria, no
modificar)
```

4.4 El programa AUTOEXEC

Si existe un archivo llamado AUTOEXEC.P en la raíz de la SD, este se cargará y ejecutará automáticamente al escribir el comando RUN en un ZX81 sin ningún programa cargado. Es útil para crear menús de inicio personalizados.

5. Primeros pasos

5.1 Modos de carga: SD o cinta

Por defecto, los comandos LOAD y SAVE del BASIC funcionan con la cinta, exactamente igual que en un ZX81 sin interface. Para utilizar la tarjeta SD tienes dos opciones:

Opción A — Activar el modo SD de forma permanente: Escribe en BASIC:

```
LOAD FAST
```

> ℹ️ *La palabra FAST se obtiene con SHIFT + F, no escribiendo las letras una a una. A partir de ese momento todos los LOAD y SAVE usarán la SD por defecto. Si necesitas cargar desde cinta estando en modo SD, usa LOAD SLOW "nombre" para esa operación concreta.*

Para volver al modo cinta por defecto:

```
LOAD SLOW
```

> ℹ️ *SLOW se obtiene con SHIFT + D. A partir de ese momento todos los LOAD y SAVE vuelven a usar la cinta, y para cargar desde SD habrá que añadir FAST explícitamente.*

Opción B — Cargar desde SD sin cambiar el modo: Añade FAST directamente al comando de carga (ver sección 6).

5.2 Cargando tu primer programa desde la SD

8. Asegúrate de que tienes al menos un archivo .P en la tarjeta SD.
9. Enciende el ZX81 con el interface y la SD insertados.
10. En el cursor K, escribe el siguiente comando, sustituyendo NOMBRE por el nombre del archivo sin extensión:
    ```
    LOAD FAST "NOMBRE"
    ```
11. Pulsa ENTER. El programa se cargará en unos instantes y arrancará automáticamente si tiene autoarranque.

> 💡 *Si no recuerdas el nombre exacto del archivo, puedes ver el contenido de la SD con el comando LOAD *DIR (ver sección 7).*

6. Carga y guardado desde la SD

6.1 Cargar un programa

Para cargar un archivo desde la SD:

```
LOAD FAST "NOMBRE"
```

El interface buscará primero el archivo con el nombre exacto. Si no lo encuentra, intentará añadir la extensión .P automáticamente.

Cargar desde una subcarpeta:

```
LOAD FAST "JUEGOS/PACMAN"
```

Cargar y ejecutar desde una línea específica:

```
LOAD FAST "NOMBRE" THEN GOTO 100
```

Cargar sin ejecutarlo automáticamente:

```
LOAD FAST "NOMBRE" THEN STOP
```

> ℹ️ *THEN, GOTO y STOP son tokens del BASIC del ZX81, no se escriben letra a letra.*

6.2 Guardar un programa

Para guardar el programa actual en la SD:

```
SAVE FAST "NOMBRE"
```

El interface añadirá automáticamente la extensión .P al archivo guardado.

> ⚠️ *Si ya existe un archivo con ese nombre, será sobreescrito sin aviso previo.*

6.3 Cargar y guardar bloques de memoria (código máquina)

Para cargar un bloque de datos en una dirección de memoria específica:

```
LOAD FAST "NOMBRE" CODE 30000
```

> ℹ️ *A diferencia del ZX Spectrum, la dirección después de CODE es obligatoria. Los archivos en SD no almacenan la dirección de carga en una cabecera.*

Para guardar un bloque de memoria en la SD:

```
SAVE FAST "NOMBRE" CODE 30000,2048
```

Donde 30000 es la dirección de inicio y 2048 es la longitud en bytes.

6.4 Notas sobre compatibilidad de juegos

Algunos juegos necesitan una inicialización especial antes de ser cargados. Los casos más comunes son:

- **LOAD *128C** antes de cargar: el juego utiliza el modo de 128 caracteres definibles por el usuario.
- **LOAD FAST** antes de cargar: algunos juegos tienen múltiples archivos. Ejecutar LOAD FAST sin nombre activa el modo SD para todos los LOAD y SAVE a partir de ese momento, permitiendo que el juego cargue sus archivos adicionales automáticamente.

> ℹ️ *Consulta el archivo de compatibilidad incluido con el interface para ver las instrucciones específicas de cada juego.*

6.6 Formatos de archivo reconocidos

El comando LOAD FAST detecta automáticamente el tipo de archivo por su extensión y actúa de forma diferente según el caso:

Extensión	Comportamiento
`.P`	Programa BASIC estándar del ZX81. El interface calcula el tamaño real del programa desde las variables del sistema y descarta los bytes sobrantes al final del archivo.
`.81`	Igual que .P.
`.P81`	Formato multi-programa. El interface salta el nombre del fichero embebido antes de leer los datos.
`.ROM`	Archivo de ROM. El interface carga el contenido en la dirección 0 del espacio de direccionamiento y resetea el sistema. El control no vuelve al BASIC.
`.WAV`	Archivo de audio sin comprimir (PCM). El interface lo reproduce directamente en lugar de cargarlo en memoria.
`Otras`	El archivo se carga íntegramente en memoria tal cual, sin ningún procesado.

⚠ *Cargar un archivo con extensión .ROM mediante LOAD FAST provoca un reset inmediato del sistema. Asegúrate de que el archivo contiene una ROM válida antes de cargarlo, ya que un archivo corrupto podría dejar el sistema en un estado irrecuperable hasta que se reinicie con otra ROM.*

7. Gestión de archivos y directorios

7.1 Ver el contenido de la SD

Para listar los archivos del directorio actual:

```
LOAD *DIR
```

Para listar los archivos de una carpeta concreta:

```
LOAD *DIR "JUEGOS"
```

Se pueden usar comodines: * representa cualquier número de caracteres, ? representa exactamente un carácter. Por ejemplo, para listar solo archivos .P:

```
LOAD *DIR "*.P"
```

ℹ *Si el listado no cabe en pantalla, aparecerá ... en la línea inferior. Pulsa cualquier tecla para continuar, o SPACE para cancelar.*

7.2 Navegar entre directorios

Cambiar al directorio JUEGOS:

```
LOAD *CD "JUEGOS"
```

Volver al directorio raíz:

```
LOAD *CD "/"
```

Ver el directorio actual:

```
LOAD *PWD
```

7.3 Crear y eliminar carpetas

Crear una carpeta nueva:

```
LOAD *MD "NUEVACARPETA"
```

Eliminar una carpeta (debe estar vacía):

```
LOAD *RD "CARPETAVACIA"
```

7.4 Borrar, renombrar y copiar archivos

Borrar un archivo:

```
LOAD *DEL "ARCHIVO.P"
```

Renombrar o mover un archivo:

```
LOAD *MV "VIEJO.P" TO "NUEVO.P"
```

Copiar un archivo:

```
LOAD *CP "ORIGEN.P" TO "DESTINO.P"
```

ℹ️ *TO es un token del BASIC (SHIFT + 4), no se escribe letra a letra.*

7.5 Espacio libre en la SD

```
LOAD *FREE
```

ℹ️ *Este cálculo puede tardar varios segundos en completarse.*

7.6 Directorios T81 (función en fase alfa)

⚠️ *Esta funcionalidad está actualmente en fase alfa. Puede contener errores y su comportamiento o interfaz podría cambiar en versiones futuras. No se recomienda su uso en entornos de producción.*

El SD81 Booster soporta un formato de archivo especial con extensión .T81 que actúa como un contenedor de múltiples programas ZX81, de forma similar a como un archivo ZIP contiene varios archivos. Esto permite distribuir colecciones de programas en un único archivo.

Para acceder al contenido de un archivo T81, se usa el comando LOAD *CD como si fuera un directorio normal:

```
LOAD *CD "COLECCION.T81"
```

A partir de ese momento, los comandos LOAD, LOAD *DIR y LOAD *OPENDIR operan sobre el contenido del archivo T81 en lugar del sistema de archivos FAT, permitiendo navegar y cargar los programas que contiene de la misma manera que se haría con archivos normales.

Para salir del directorio T81 y volver al sistema de archivos normal:

```
LOAD *CD "/"
```

> ℹ️ *Limitaciones en fase alfa: Las operaciones de escritura (LOAD *DEL, LOAD *MD, LOAD *RD, LOAD *MV, LOAD *CP, SAVE) no están soportadas dentro de un directorio T81 y devolverán un error.*

8. Funciones adicionales

8.1 Reproducción de archivos WAV

El interface puede reproducir archivos de audio en formato WAV sin comprimir directamente desde la tarjeta SD, simplemente cargándolos con el comando habitual:

```
LOAD FAST "SONIDO.WAV"
```

Si el archivo tiene extensión .WAV, el interface lo detecta automáticamente y lo reproduce en lugar de intentar cargarlo como programa. No es necesario ningún comando especial.

> ℹ️ *Solo se admiten archivos WAV sin comprimir (PCM). Otros formatos de audio no son compatibles.*

8.2 Reloj en tiempo real — Comando RTC

El SD81 Booster incorpora un reloj en tiempo real (RTC) con batería de reserva. El comando LOAD *RTC permite consultar y ajustar la hora y la fecha desde BASIC.

Mostrar la hora y fecha actuales en pantalla:

```
LOAD *RTC
```

Guardar la hora y fecha en una variable de cadena:

```
LOAD *RTC TO R$
```

Poner el reloj en hora:

```
LOAD *RTC="cadena"
```

El comando acepta varios formatos de cadena. Puedes ajustar fecha y hora a la vez, o solo uno de los dos:

Formato	Ejemplo	Descripción
`AAAA-MM-DD HH:MM:SS.CC`	`2025-04-30 18:30:00.00`	Fecha y hora completas con centésimas
`AAAA-MM-DD HH:MM:SS`	`2025-04-30 18:30:00`	Fecha y hora completas
`AAAA-MM-DD`	`2025-04-30`	Solo fecha
`HH:MM:SS.CC`	`18:30:00.00`	Solo hora con centésimas
`HH:MM:SS`	`18:30:00`	Solo hora
`HH:MM`	`18:30`	Hora y minutos (segundos a cero)

Ejemplos:

```
LOAD *RTC="2025-04-30 18:30:00"
LOAD *RTC="2025-04-30"
LOAD *RTC="18:30:00"
```

Ejemplo en un programa BASIC:

```
10 LOAD *RTC TO R$
20 PRINT "Fecha y hora: ";R$
```

8.3 Estado de la batería del RTC — Comando BAT

El SD81 Booster incorpora una pila botón CR2032 que mantiene el reloj en hora cuando el ZX81 está apagado. Esta pila viene preinstalada de fábrica y tiene una vida útil estimada de varios años en condiciones normales de uso. Cuando se agote, puede sustituirse por cualquier pila CR2032 estándar disponible en comercios de electrónica.

El estado de carga de la pila puede consultarse desde BASIC con el comando LOAD *BAT.

Mostrar el estado de la batería en pantalla:

```
LOAD *BAT
```

Guardar el estado en una variable de cadena:

```
LOAD *BAT TO B$
```

💡	*Si el reloj pierde la hora frecuentemente al apagar el ordenador, consulta el estado de la batería con este comando para saber si necesita ser reemplazada.*

8.4 Modo RAM extendida — Comando RAM48

El comando LOAD *RAM48 activa el modo de RAM extendida de 48 KB, que amplía la memoria disponible para programas BASIC y datos más allá de los límites habituales.

Activar el modo RAM extendida:

```
LOAD *RAM48
```

Desactivarlo para restaurar la compatibilidad estándar:

```
LOAD *RAM48 STOP
```

ℹ️	*STOP es el token del BASIC, no se escribe letra a letra. Si algún programa presenta problemas de compatibilidad con el modo RAM48 activo, desactívalo con LOAD *RAM48 STOP antes de cargarlo.*

8.5 Visualización de archivos de texto — Comando THEN PRINT

El comando LOAD THEN PRINT es el equivalente al comando TYPE de MS-DOS o cat de Linux: muestra el contenido de un archivo de texto directamente en la pantalla del ZX81.

Mostrar un archivo de texto en pantalla:

```
LOAD THEN PRINT "FICHERO"
```

Enviar el contenido a la impresora ZX Printer:

```
LOAD THEN LPRINT "FICHERO"
```

Sistema de ayuda integrado

Si se antepone un asterisco * al nombre del fichero, el interface busca automáticamente un archivo con ese nombre en la carpeta /MAN/ de la SD y le añade la extensión .TXT. Esto permite implementar un sistema de ayuda al estilo del comando man de Linux:

```
LOAD THEN PRINT "*PLAY"
```

Este comando buscaría el archivo /MAN/PLAY.TXT en la SD y mostraría su contenido en pantalla. Puedes crear tus propios archivos de ayuda en esa carpeta para documentar tus programas o comandos.

Ejemplo — mostrar instrucciones de un juego:

```
10 LOAD THEN PRINT "*INSTRUCCIONES"
```

Esto mostraría el contenido de /MAN/INSTRUCCIONES.TXT, ideal para mostrar las instrucciones de un juego o programa desde dentro del propio programa BASIC.

9.6 Joystick programable — Comando JOY

El SD81 Booster incorpora un puerto de joystick DB9 cuyo mapeo de botones es totalmente configurable. El comando LOAD *JOY permite asignar una tecla del ZX81 a cada dirección y al botón de fuego del joystick.

Sintaxis:

```
LOAD *JOY "izq/der/arr/aba/fue"
```

La cadena de configuración contiene exactamente cinco caracteres, uno por cada función del joystick en este orden: izquierda / derecha / arriba / abajo / fuego.

Ejemplo:

```
LOAD *JOY "OPQA "
```

- Izquierda → tecla O
- Derecha → tecla P
- Arriba → tecla Q
- Abajo → tecla A
- Fuego → tecla espacio

💡	*Consulta los controles de cada juego antes de configurar el joystick. Muchos juegos del*

	ZX81 usan combinaciones de teclas diferentes, y con este comando puedes adaptarlas a cualquier joystick estándar de 9 pines sin modificar el software.

10. Sonido

El SD81 Booster incorpora un emulador del chip de sonido AY-3-8910/12, el mismo que usaban ordenadores como el ZX Spectrum 128K o el Amstrad CPC. Esto permite reproducir música de hasta tres voces simultáneas, efectos de sonido programables y música en segundo plano, todo ello desde BASIC o desde código máquina.

10.1 Comando PLAY — Música con el chip AY

El comando PLAY permite reproducir música directamente desde BASIC mediante una cadena de texto que describe las notas, la duración, el tempo y otros parámetros. Admite hasta tres voces simultáneas (canales A, B y C del AY).

Sintaxis básica:

```
LOAD *PLAY "cadena1"
LOAD *PLAY "cadena1","cadena2","cadena3"
```

Cada cadena corresponde a una voz. Pueden usarse de una a tres cadenas simultáneamente.

Ejemplo sencillo — melodía en una sola voz:

```
LOAD *PLAY "T120 O4 5C 5E 5G 9C"
```

Esto toca las notas Do, Mi, Sol y Do (acorde de Do mayor) a 120 pulsaciones por minuto en la octava 4.

Notas

Las notas se escriben con las letras C D E F G A B (Do Re Mi Fa Sol La Si). Se pueden modificar con:

- = antes de la nota: sube un semitono (sostenido). Ejemplo: =F es Fa sostenido.
- £ antes de la nota: baja un semitono (bemol). Ejemplo: £E es Mi bemol.
- Letra en vídeo normal: toca la nota en la octava actual.
- Letra en vídeo inverso (SHIFT + 9 sobre la letra): toca la misma nota en la octava siguiente, sin cambiar la octava activa. Es equivalente al uso de mayúsculas en el ZX Spectrum.
- £ sin nota a continuación: pausa o silencio.

Duración

Un número del 1 al 12 antes de una nota establece su duración desde ese punto en adelante:

Valor	Nombre	Duración a 60 bpm
1	Semicorchea	0.25 s
3	Corchea	0.5 s
5	Negra	1 s
7	Blanca	2 s
9	Redonda	4 s

> 💡 *Puedes ligar duraciones con el signo -. Por ejemplo, 4-3A combina corchea con puntillo (7/8 de negra).*

Tempo

T<número> establece el tempo en pulsaciones por minuto (bpm), entre 60 y 240. El valor por defecto es 120. Solo tiene efecto en la primera voz.

```
LOAD *PLAY "T180 5C 5E 5G"
```

Octava

O<número> selecciona la octava, de 0 (muy grave) a 8 (muy agudo). La octava por defecto es 4.

```
LOAD *PLAY "O3 5C O4 5C O5 5C"
```

Repeticiones

- (...) repite una sección una vez más.
-) sin (correspondiente repite la cadena entera indefinidamente.
- H detiene el comando PLAY aunque haya voces en bucle infinito.

```
LOAD *PLAY "(5C 5E 5G) H",")"
```

Efectos de volumen (envolvente)

W<número> selecciona uno de los 8 efectos de volumen del chip AY (W0 a W7), como ataque, decaimiento o tremolo. U activa el efecto en el canal. X<número> ajusta la velocidad del efecto (0–65535; a mayor valor, más lento).

> ℹ️ *Para una descripción completa de todos los parámetros, incluyendo los efectos de envolvente y el modo de ruido, consulta el Apéndice A de este manual.*

10.2 Reproductor VGM — Música en segundo plano

El SD81 Booster puede reproducir archivos en formato VGM (Video Game Music) en segundo plano mientras el ZX81 ejecuta cualquier otro programa. Esto permite añadir música a tus propios programas BASIC sin consumir tiempo de CPU.

> ℹ️ *Solo se admiten archivos VGM que contengan datos del chip AY-3-8910 o AY-3-8912. Los VGMs con otros chips de sonido no son compatibles.*

Preparar un archivo VGM:

```
LOAD *VGM "MUSICA"
```

Carga el archivo MUSICA.VGM de la SD y lo deja listo para reproducir, sin iniciarlo todavía.

Iniciar la reproducción:

```
LOAD *VGM THEN RUN
```

Pausar y reanudar:

```
LOAD *VGM THEN PAUSE
LOAD *VGM THEN CONT
```

Detener la reproducción:

```
LOAD *VGM THEN STOP
```

Activar el modo bucle (la música se reinicia al terminar):

```
LOAD *VGMLOOP
```

Desactivar el modo bucle:

```
LOAD *VGMLOOP STOP
```

> ℹ️ *THEN, RUN, CONT, PAUSE y STOP son tokens del BASIC del ZX81, no se escriben letra a letra.*

Ejemplo de uso típico en un programa BASIC:

```
10 LOAD *VGM "MUSICA"
20 LOAD *VGMLOOP
30 LOAD *VGM THEN RUN
40 REM --- el programa continúa mientras suena la música ---
```

10.3 Generador de efectos PEG — Efectos de sonido programables

El PEG (Programmable Effects Generator) es una pequeña máquina virtual integrada en el interface que ejecuta programas de efectos de sonido de forma completamente independiente al Z80, sin consumir tiempo de CPU del ZX81.

El PEG accede directamente a los registros del chip AY y dispone de hasta tres hilos de ejecución paralelos, lo que permite reproducir varios efectos simultáneamente.

> ℹ️ *El PEG está orientado principalmente a desarrolladores. Para crear programas PEG se recomienda usar el ensamblador PEG incluido en el repositorio del proyecto. Para una descripción completa del juego de instrucciones, consulta el Apéndice B de este manual.*

Cargar un programa PEG en memoria:

```
LOAD *PEG <dirección>,"<hexadecimal>"
```

Donde <dirección> es la posición en la memoria PEG (0–255) y <hexadecimal> es la secuencia de instrucciones en formato hexadecimal. Cada instrucción ocupa 4 caracteres hexadecimales (2 bytes).

Iniciar un hilo PEG:

```
LOAD *PEG THEN RUN <hilo>,<dirección>
```

Donde <hilo> es el número de hilo (0, 1 o 2) y <dirección> es la dirección de inicio del programa PEG.

Detener, pausar y reanudar un hilo:

```
LOAD *PEG THEN STOP <hilo>
LOAD *PEG THEN PAUSE <hilo>
LOAD *PEG THEN CONT <hilo>
```

10.4 Síntesis de voz — Comando SAY

El SD81 Booster incorpora un sintetizador de voz que permite reproducir frases en inglés directamente desde BASIC. El sintetizador se basa en los fonemas del chip SP0256, un sintetizador de voz ampliamente utilizado en la época que formaba parte de interfaces clásicos como el Currah MicroSpeech para el ZX Spectrum o The Voice para el Videopac G7000/Odyssey 2. Las muestras de audio de los fonemas están almacenadas en la memoria interna del microcontrolador, por lo que no se necesita ningún archivo adicional en la SD.

Sintaxis:

```
LOAD *SAY "frase"
```

El sintetizador analiza la cadena de texto e intenta construir la pronunciación combinando fonemas del inglés. El texto debe escribirse en inglés, en mayúsculas.

Ejemplos:

```
LOAD *SAY "HELLO WORLD"
LOAD *SAY "ZX81 COMPUTER"
LOAD *SAY "ERROR 5"
```

Los números se leen en inglés automáticamente, desde cero hasta los billones.

Reproducción en background

Por defecto, la CPU espera a que la voz termine antes de continuar ejecutando el programa. Si se antepone un * al inicio de la cadena, la reproducción continúa en segundo plano y el ZX81 sigue ejecutando el programa inmediatamente:

```
LOAD *SAY "*HELLO WORLD"
```

No es posible añadir sonidos o fonemas personalizados. El conjunto de fonemas disponibles está fijo en la memoria interna del microcontrolador.

Cómo funciona el sintetizador

El sintetizador descompone el texto de izquierda a derecha, buscando siempre la coincidencia más larga posible en su diccionario interno. Las palabras reconocidas como tales suenan mejor que las que el sintetizador debe construir sílaba a sílaba.

Los espacios y signos de puntuación producen pausas de duración creciente: el espacio produce una pausa corta, la coma una pausa media, y el punto y coma o los dos puntos una pausa larga.

> 💡 *Para mejorar la pronunciación de palabras no reconocidas, escríbelas fonéticamente en inglés. Por ejemplo, SINCLAIR puede sonar mejor como SINCLER. Consulta el Apéndice C para ver el diccionario completo de palabras y fonemas reconocidos.*

10. Gestión de memoria

El SD81 Booster incorpora hasta 512 KB de RAM, muy por encima de los 1 KB originales del ZX81 o de las expansiones de memoria convencionales. Esta memoria se gestiona mediante un mapeador de memoria que divide tanto la RAM como el espacio de direcciones del Z80 en bloques de 8 KB, permitiendo asignar cualquier página de memoria a cualquier bloque del mapa de direcciones.

Para la mayoría de los programas BASIC no es necesario gestionar la memoria manualmente: el interface la configura automáticamente al arrancar y el BASIC del ZX81 puede usar la RAM disponible de forma transparente.

10.1 Conceptos básicos: bloques y páginas

El espacio de direcciones del Z80 (64 KB) se divide en 8 bloques de 8 KB cada uno:

Bloque	Rango de direcciones	Uso habitual
0	0000–1FFF	ROM del ZX81 (solo lectura)
1	2000–3FFF	ROM de expansión del interface
2	4000–5FFF	RAM principal / pantalla
3	6000–7FFF	RAM principal
4	8000–9FFF	RAM ampliada
5	A000–BFFF	RAM ampliada
6	C000–DFFF	Espejo de bloque 2 (necesario para el vídeo)
7	E000–FFFF	Espejo de bloque 3 (necesario para el vídeo)

Los 512 KB de RAM se dividen en 64 páginas de 8 KB. Cada bloque puede apuntar a cualquiera de estas 64 páginas, lo que permite acceder a toda la memoria simplemente cambiando qué página está asignada a qué bloque.

> ⚠ *Los bloques 6 y 7 deben mantenerse como espejo de los bloques 2 y 3 respectivamente para que el sistema de vídeo del ZX81 funcione correctamente. Modificarlos sin tener esto en cuenta puede causar que la pantalla deje de funcionar.*

10.2 Comando MAP — Asignar páginas a bloques

Para asignar una página de memoria a un bloque determinado:

```
LOAD *MAP <bloque>,<página>
```

Donde <bloque> es un número del 0 al 7 y <página> es un número del 0 al 63.

Para leer qué página está asignada actualmente a un bloque:

```
LOAD *MAP <bloque> TO <variable>
```

Ejemplo — cambiar el bloque 4 a la página 10:

```
LOAD *MAP 4,10
```

A partir de ese momento, cualquier lectura o escritura en las direcciones 32768–40959 accederá a la página 10 de la RAM.

Ejemplo — leer la página asignada a un bloque:

```
10 LOAD *MAP 4 TO A
20 PRINT "Bloque 4 apunta a la pagina ";A
```

> ℹ *Para una referencia completa del sistema de paginación, incluyendo el modo de paginación completa de 512 KB y el uso desde código máquina, consulta el Apéndice D de este manual.*

10.3 Modo MC45 — Código máquina en bloques 4 y 5

Por diseño del hardware del ZX81, las instrucciones de código máquina situadas en los bloques 4 y 5 (direcciones 32768–49151) son ejecutadas de forma incorrecta: los opcodes en los rangos 00h–3Fh y 80h–BFh son interpretados como NOP, lo que hace imposible ejecutar código normal en esa zona.

El modo MC45 (Machine Code 4 and 5) desactiva esta limitación, permitiendo ejecutar cualquier instrucción Z80 en esa área de memoria.

Activar el modo MC45:

```
LOAD *MC45
```

Desactivarlo:

```
LOAD *MC45 STOP
```

> ⚠ *El modo MC45 se implementa forzando a cero el pin M1 del Z80 de forma intermitente y durante intervalos de tiempo muy breves, lo que mantiene la carga sobre dicho pin en niveles bajos y hace que la posibilidad de daño sea muy reducida. Además, el interface ya incorpora internamente la resistencia de protección necesaria, por lo que no es necesario realizar ninguna modificación en el ZX81. Pese a todo, el uso de esta característica se*

	realiza bajo la responsabilidad exclusiva del usuario. Cuando MC45 está activo no es posible cargar ni escribir programas BASIC de más de 16 KB.

10.4 Arranque con ROM alternativa

El SD81 Booster permite arrancar con una ROM diferente a la estándar del ZX81 sin necesidad de usar ningún comando. Basta con tener en la carpeta /SYS/ de la tarjeta SD uno o más archivos de ROM con los nombres 0.ROM, 1.ROM, 2.ROM... hasta 9.ROM.

Para arrancar con una ROM alternativa:

12. Mantén pulsado el número correspondiente al archivo de ROM deseado mientras enciendes el ZX81.
13. Suelta la tecla cuando el ordenador haya arrancado.

💡	*Esta función es muy útil para probar ROMs alternativas o modificadas sin necesidad de reprogramar ningún chip. La ROM estándar del ZX81 se carga siempre si no se pulsa ninguna tecla durante el arranque.*

10.5 Caracteres definibles por el usuario (128C / 64C)

Por defecto el ZX81 dispone de 64 caracteres definidos por la ROM. El SD81 Booster permite ampliar este conjunto a 128 caracteres, todos ellos completamente redefinibles, escribiendo en la zona de memoria entre las direcciones 15360 y 16383 (3C00h–3FFFh).

Activar el modo de 128 caracteres:

```
LOAD *128C
```

Volver al modo estándar de 64 caracteres:

```
LOAD *64C
```

ℹ️	*En el modo de 128 caracteres, los 64 caracteres superiores se muestran automáticamente en vídeo inverso por el hardware. Para mostrarlos en vídeo normal debes almacenar el gráfico invertido en esa posición de memoria.*

Si necesitas restaurar el juego de caracteres original después de haberlo modificado:

```
LOAD *LDIR 7680,15360,512
LOAD *LDIR 7680,15872,512
```

ℹ️	*La activación del modo de 128 caracteres es incompatible con el generador interno de caracteres HRG (ver sección 10.6). Ambos modos no pueden estar activos simultáneamente.*

Ejemplo: pantalla de inicio estilo Spectrum

El siguiente programa ilustra el uso del modo de 128 caracteres para cargar un juego de caracteres alternativo desde la SD y mostrar una pantalla al estilo del ZX Spectrum:

```
 5 LOAD *128C
20 LOAD FAST "SPEC-81-128.BIN" CODE 15360
30 CLS
35 POKE 16418,0
40 PRINT AT 23,1;CHR$ 8;" 1982 S[INCLAIR] R[ESEARCH] L[TD]."
45 POKE 16418,2
50 IF INKEY$="" THEN GOTO 50
```

> ℹ️ *Los textos entre corchetes indican vídeo inverso: S[INCLAIR] significa que la S es normal e INCLAIR va en vídeo inverso; R[ESEARCH] que la R es normal y ESEARCH en vídeo inverso; L[TD] que la L es normal y TD en vídeo inverso. Para introducir caracteres en vídeo inverso pulsa SHIFT + 9 antes de cada carácter. CHR$ 8 corresponde al carácter gráfico de cuadrícula del ZX81 (código 8, símbolo © en el juego de caracteres del Spectrum).*

Explicación línea a línea:

- Línea 5: activa el modo de 128 caracteres definibles.
- Línea 20: carga el archivo SPEC-81-128.BIN desde la SD en la dirección 15360 (3C00h), zona de caracteres definibles. Este archivo contiene el juego de caracteres del ZX Spectrum.
- Línea 30: limpia la pantalla.
- Línea 35: activa el modo Superfast (POKE 16418,0) para liberar la CPU del control del vídeo.
- Línea 40: imprime en la línea 23 el mensaje de copyright del Spectrum, usando CHR$ 8 como símbolo © y las palabras de la firma en vídeo inverso.
- Línea 45: restaura el modo de vídeo estándar (POKE 16418,2).
- Línea 50: espera indefinidamente a que se pulse cualquier tecla.

11. Extensiones BASIC avanzadas

Esta sección recoge comandos adicionales del BASIC extendido del SD81 Booster orientados principalmente a la programación: manipulación de cadenas, acceso a memoria, comunicación con puertos de entrada/salida y acceso al directorio desde un programa.

11.1 Manipulación de cadenas

Invertir caracteres de una cadena (*INV):

Invierte el bit 7 de todos los caracteres de una variable de cadena, convirtiendo los caracteres normales en inversos y viceversa:

```
LOAD *INV A$
```

También admite porciones de cadena:

```
LOAD *INV A$(2 TO 7)
```

Forzar vídeo inverso en una cadena (*BOLD):

Fuerza el bit 7 de todos los caracteres de una variable de cadena a 1, poniendo todos los caracteres en vídeo inverso:

```
LOAD *BOLD A$
```

> 💡 *Para poner caracteres en vídeo normal a partir de una cadena en inverso, aplica primero *BOLD y luego *INV para invertir el resultado.*

11.2 Copia y relleno de bloques de memoria

Copiar un bloque de memoria en orden ascendente (*LDIR):

```
LOAD *LDIR <origen>,<destino>,<longitud>
```

Copia <longitud> bytes desde <origen> hasta <destino> en orden ascendente. Equivale a la instrucción Z80 LDIR.

Copiar en orden descendente (*LDDR):

```
LOAD *LDDR <origen>,<destino>,<longitud>
```

Igual que *LDIR pero en orden descendente. Equivale a la instrucción Z80 LDDR.

> ℹ️ *Cuando origen y destino se solapan, el orden de copia importa: usa *LDIR si el destino está antes del origen en memoria, y *LDDR si está después, para evitar que los datos se sobreescriban durante la copia.*

Cargar datos hexadecimales en memoria (*HEX):

```
LOAD *HEX <dirección>,"<hexadecimal>"
```

Por ejemplo, LOAD *HEX 30000,"0A014020" carga los valores 10 (0Ah), 1, 64 (40h) y 32 (20h) a partir de la dirección 30000.

11.3 Ejecución de código máquina

Ejecutar una rutina en código máquina (LOAD USR):

```
LOAD USR <dirección>
```

Equivale a RAND USR pero con tres ventajas: no modifica el generador de números aleatorios, la rutina se llama desde el nivel superior del BASIC dejando los registros alternativos BC', DE' y HL' disponibles, y el resto de la línea no se analiza sintácticamente permitiendo que la rutina realice su propio análisis de parámetros. Al entrar en la rutina, el registro BC contiene la dirección llamada.

11.4 Acceso a puertos de entrada/salida

Escribir en un puerto (*OUT):

```
LOAD *OUT <puerto>,<valor>
```

Leer de un puerto (*IN):

```
LOAD *IN <puerto> TO <variable>
```

> *TO es el token del BASIC (SHIFT + 4), no se escribe letra a letra.*

11.5 Acceso a memoria de 16 bits

Leer un valor de 16 bits de memoria (LOAD PEEK):

```
LOAD PEEK <dirección> TO <variable>
```

Lee dos bytes consecutivos de memoria a partir de <dirección> y los almacena como un valor de 16 bits en <variable>.

Escribir un valor de 16 bits en memoria (LOAD THEN POKE):

```
LOAD THEN POKE <dirección>,<valor16>
```

Establecer el límite superior de memoria del BASIC (LOAD THEN CLEAR):

```
LOAD THEN CLEAR <dirección>
```

Establece la última dirección de RAM disponible para el BASIC. A diferencia del comando CLEAR estándar, este no borra las variables; solo limpia la pila de GOSUB. Usa un CLEAR separado si también quieres borrar las variables.

11.6 Acceso al directorio desde un programa

Estos comandos permiten leer el contenido de un directorio de la SD desde dentro de un programa BASIC, útil para construir menús de selección de archivos.

Abrir un directorio para lectura (*OPENDIR):

```
LOAD *OPENDIR <cadena>
```

La cadena puede incluir una ruta completa y comodines. Si se usan comodines, el número máximo de entradas recuperables es 512. Este comando deja el directorio preparado para usar *ROW.

Leer una entrada del directorio (*ROW):

```
LOAD *ROW <número> TO <variable$>
```

Lee la entrada de directorio con el número indicado (empezando desde 1) y la almacena en la variable de cadena. Si el número está fuera de rango, devuelve una cadena vacía.

Ejemplo — menú de selección de archivos en BASIC:

```
10 LOAD *OPENDIR "JUEGOS/*.P"
20 FOR I=1 TO 10
30 LOAD *ROW I TO A$
40 IF A$="" THEN GOTO 70
50 PRINT I;". ";A$
60 NEXT I
70 INPUT "Selecciona: ";N
80 LOAD *ROW N TO A$
90 LOAD FAST A$
```

13. Ejemplos de programas

Esta sección recoge programas de ejemplo que ilustran el uso de las funciones principales del SD81 Booster. Todos están escritos en BASIC estándar del ZX81 con las extensiones del interface.

> ℹ️ *Los textos entre corchetes (por ejemplo [SINCLAIR]) se introducen en vídeo inverso en el ZX81, pulsando SHIFT + 9 antes de cada carácter. CHR$ 8 corresponde al carácter gráfico de cuadrícula del ZX81 (código 8).*

13.1 Reloj en tiempo real (RTC)

Demuestra los tres formatos de ajuste del reloj: fecha y hora completas, solo fecha y solo hora.

```
10 LET A$="2025-11-10 13:48:00.00"
15 LOAD *RTC
16 PRINT
20 LOAD *RTC=A$
25 LOAD *RTC
26 PRINT
30 LOAD *RTC="2026-11-10"
35 LOAD *RTC
36 PRINT
40 LOAD *RTC="12:20"
45 LOAD *RTC
46 PRINT
50 LOAD *RTC="13:20:35"
55 LOAD *RTC
56 PRINT
```

Muestra la hora actual (línea 15), luego la ajusta mediante una variable de cadena (línea 20), después cambia solo la fecha (línea 30), luego solo la hora con formato corto (línea 40) y finalmente con hora, minutos y segundos (línea 50). Tras cada ajuste imprime el resultado para verificarlo.

13.2 Estado de la batería del RTC (BAT)

```
10 LOAD *BAT TO A$
20 PRINT A$
```

Lee el estado de la batería del reloj en tiempo real y lo muestra en pantalla. Si la batería está baja, el valor mostrado lo indicará.

13.3 Comprobación del modo MC45

Comprueba si el modo de código máquina en bloques 4 y 5 está activo cargando una pequeña rutina en ensamblador y ejecutándola:

```
 10 LOAD *HEX 40000,"010203C9"
 20 IF USR 40000=770 THEN GOTO 100
 30 PRINT "MC45 INACTIVE"
 40 STOP
100 PRINT "MC45 ACTIVE"
```

Carga en la dirección 40000 (bloques 4-5) una rutina Z80 que devuelve el valor de BC al salir (C9=RET). Si MC45 no está activo, las instrucciones en esa zona no se ejecutarán correctamente y el valor devuelto no será 770. Si MC45 está activo, la rutina se ejecuta correctamente y salta a la línea 100.

13.4 Modo Superfast — demostración de velocidad

Compara visualmente la velocidad de refresco en modo estándar ZX81 frente al modo Superfast del SD81 Booster:

```
   4 SLOW
   5 PRINT "ZX81 SLOW MODE"
   6 PAUSE 250
   7 POKE 2045,85
   8 POKE 16418,0
   9 CLS
  10 GOSUB 1000
  15 CLS
  20 POKE 2045,170
  30 PRINT "SD81-BOOSTER SUPER FAST MODE"
  31 PAUSE 250
  35 CLS
  36 FAST
  37 GOSUB 1000
  40 GOTO 40
  75 POKE 1024,2
  76 POKE 1024,3
  77 POKE 1024,4
1000 PRINT "[CHR$ 0]123456789ABCDEFGHIJKLMNOPQRSTUV";
1010 FOR N=1 TO 22
1020 PRINT "0123456789ABCDEFGHIJKLMNOPQRSTUV";
1030 NEXT N
1040 PRINT "0123456789ABCDEFGHIJKLMNOQRSTUV[CHR$ 0]";
1050 RETURN
2000 PRINT "[CHR$ 0]123456789ABCDEFGHIJKLMNOPQRSTUV";
2010 FOR N=1 TO 22
2020 PRINT "0123456789ABCDEFGHIJKLMNOPQRSTUV";
2030 NEXT N
2040 PRINT "0123456789ABCDEFGHIJKLMNOQRSTUV[CHR$ 0]";
```

```
2050 RETURN
```

> ℹ️ *[CHR$ 0] en las líneas 1000, 1040, 2000 y 2040 representa el carácter 0 en vídeo inverso visible en el listado original.*

El programa muestra primero una pantalla llena de texto en modo SLOW estándar del ZX81, donde la CPU dedica tiempo al refresco de vídeo. Luego activa el modo Superfast (POKE 2045,170) y repite el mismo relleno de pantalla con FAST, mostrando la diferencia de velocidad.

13.5 Carga de imagen en modo Spectrum

Carga una imagen en formato Spectrum (.SCR) desde la carpeta /SCR/ de la SD y la muestra usando el modo HiRes Spectrum del interface:

```
  5 LOAD *CD "/SCR"
 15 LET HFILE=32768
 20 POKE 2044,HFILE/256
 25 POKE 2045,172
 30 LOAD *OUT 32751,39
 40 LOAD *OUT 251,6
 50 LOAD FAST "Z.SCR" CODE HFILE
190 IF INKEY$="" THEN GOTO 15
200 POKE 2045,85
210 LOAD *OUT 32751,0
```

Explicación línea a línea:

- Línea 5: cambia al directorio /SCR de la SD.
- Línea 15: define HFILE=32768 (8000h), inicio del bloque 4.
- Línea 20: escribe la parte alta de la dirección del fichero de pantalla en el registro del interface.
- Línea 25: activa el modo Superfast HiRes Spectrum (POKE 2045,172).
- Líneas 30-40: configura los registros de color del borde mediante el puerto de salida del interface.
- Línea 50: carga el archivo Z.SCR en la dirección HFILE (32768).
- Línea 190: espera a que se pulse cualquier tecla; al pulsarla vuelve a la línea 15 para recargar.
- Línea 200: desactiva el modo Superfast.
- Línea 210: restaura el registro de salida a 0.

14. Códigos de error

Cuando se produce un error, el ZX81 muestra un código en la parte inferior de la pantalla seguido del número de línea donde ocurrió. Los códigos relacionados con el SD81 Booster son:

Código	Significado
A	Argumento inválido (nombre de archivo incorrecto, parámetro fuera de rango, o cadena hexadecimal con longitud incorrecta en *PEG o *HEX).
D	El usuario pulsó BREAK para interrumpir una operación (por ejemplo, durante un listado de directorio).
G	Archivo no encontrado en la SD.
H	Error al acceder a la tarjeta SD. Comprueba que está insertada correctamente y formateada en FAT32.
I	Error de E/S en la tarjeta SD durante una operación de lectura o escritura.
J	Disco lleno: no hay espacio suficiente en la SD para guardar el archivo.
K	Archivo o directorio ya existe con ese nombre.
L	Nombre de archivo demasiado largo o con caracteres no permitidos.
M	El directorio no está vacío (al intentar eliminar con *RD).
N	Permiso denegado o archivo protegido contra escritura.

Si obtienes el error H de forma repetida, extrae la tarjeta SD, comprueba el formato FAT32 y vuelve a insertarla. Si el error persiste, prueba con otra tarjeta.

15. Para programadores

Esta sección está dirigida a desarrolladores que deseen aprovechar las capacidades avanzadas del SD81 Booster desde código máquina Z80, incluyendo las rutinas de la ROM de expansión y el sistema de comunicación con el microcontrolador.

15.1 Mapa de la ROM de expansión

La ROM de expansión ocupa el bloque 1, a partir de la dirección 8192 (2000h):

Dirección	Contenido / Rutina
2000h	Cadena 'SD81' en codificación ZX81
2004h	Byte de versión ROM (nibble alto = mayor, nibble bajo = menor). PEEK 8196.
2005h	Rutina: devuelve en HL la dirección de retorno del llamador
2006h	Rutina: ejecuta JP (HL) — emula CALL (HL)
2007h	GetMCUVersion — versión MCU en BC (B=0, C=versión). USR 8199.
200Ah	WaitClkDiff — espera bit de reloj diferente al bit 7 de C
200Dh	WaitClkEq — espera bit de reloj igual al bit 7 de C
2010h	OutWaitDiff — envía A al puerto de datos y espera reloj diferente

Dirección	Contenido / Rutina
2013h	OutWaitEq — envía A al puerto de datos y espera reloj igual
2016h	WaitDiffBrk — como WaitClkDiff pero permite BREAK (no retorna si se pulsa)
2019h	WaitEqBrk — como WaitDiffBrk pero espera igualdad
201Ch	SendString — envía cadena al MCU con longitud prefijada. B=longitud, DE=dirección
201Fh	SendStrLoop — envía B bytes al MCU desde DE sin esperar cambios de reloj
2022h	ReportStatus — lee byte del MCU; si ≠0 genera error BASIC (1=G, 2=H, ...)
2025h	PrintBPaged — imprime carácter con paginación. B=carácter (0–63 o 128–191)
2028h	Cmd64C — activa modo 64 caracteres (= LOAD *64C)
202Bh	Cmd128C — activa modo 128 caracteres (= LOAD *128C)
202Eh	GetPhase — estado del reloj en bit 7 de C
2031h	GetData — lee puerto de datos del MCU. Resultado en A. No modifica flags.
2034h	SD81_RESET — punto de entrada RESET. Requiere NMI y DI deshabilitados; HL=dirección de continuación.
2037h	SD81LOADCMD — punto de entrada para el comando LOAD extendido
203Ah	SD81SAVECMD — punto de entrada para el comando SAVE extendido
203Dh	SD81RUNCMD — punto de entrada para el comando RUN extendido

Obtener la versión desde BASIC:

```
10 LET V=USR 8199
20 LET MAJ=INT(V/16)
30 LET MIN=V-16*MAJ
40 PRINT "VERSION MCU: ";CHR$(MAJ+28);".";CHR$(MIN+28)
```

15.2 Puertos de E/S y protocolo MCU

Puerto	Función
E7h	Mapeador de memoria
A7h	Puerto de datos MCU (lectura y escritura). El bit 0 en lectura indica el estado de la interrupción VSYNC.
AFh	Puerto de control MCU (escritura=reset MCU; bit 7 en lectura=bit de reloj)

Sincronización con VSYNC

El bit 0 del puerto A7h refleja el estado de la interrupción de sincronismo vertical (VSYNC). Esto permite a la CPU esperar al inicio del refresco de pantalla de forma precisa, sin necesidad de interrupciones ni del comando HALT.

En el ZX Spectrum, muchos juegos usaban HALT o una rutina de interrupción IM1/IM2 para sincronizarse con el barrido vertical. En el SD81 Booster este mecanismo sustituye esa funcionalidad y es especialmente útil al portar juegos de Spectrum.

Ejemplo de bucle de espera a VSYNC en ensamblador Z80:

```
WAIT_VSYNC:
        in      a,(0A7h)        ; leer puerto de datos MCU
        and     01h             ; aislar bit 0 (VSYNC)
        jr      nz,WAIT_VSYNC   ; esperar hasta que VSYNC = 0
WAIT_VSYNC2:
        in      a,(0A7h)
        and     01h
        jr      z,WAIT_VSYNC2   ; esperar flanco (VSYNC = 1)
        ; Sincronizados con el inicio del frame
```

⚠ *Escribir cualquier valor en el puerto AFh provoca un reset software del MCU. No debe hacerse mientras el MCU esté guardando o copiando un archivo.*

La comunicación se sincroniza mediante el bit de reloj (bit 7 del puerto AFh), que se invierte con cada lectura o escritura en A7h. El Z80 debe esperar a que el bit cambie antes de la siguiente operación.

Ejemplo — comando GETBYTE (índice 16 de la memoria interna del MCU):

```
        in      a,(0AFh)        ; leer bit de reloj inicial
        ld      c,a
        ld      a,20h           ; código GETBYTE
        out     (0A7h),a        ; enviar comando
WAIT1:  in      a,(0AFh)
        xor     c
        jp      p,WAIT1         ; esperar reloj diferente
        ld      a,16            ; índice
        out     (0A7h),a        ; enviar parámetro
WAIT2:  in      a,(0AFh)
        xor     c
        jp      m,WAIT2         ; esperar reloj igual
        in      a,(0A7h)        ; leer respuesta — resultado en A
```

15.3 Mapeador de memoria (puerto E7h)

Modo simple (hasta 256 KB): bits D2-D0 = bloque (0–7), bits D7-D3 = página (0–31).

Modo completo (hasta 512 KB): usar instrucción OUT (C),r con número de página en B y número de bloque en el registro que se envía:

```
        ld      c,0E7h
        out     (c),a           ; A = bloque (0-7), B = página (0-63)
```

15.4 Tabla completa de comandos MCU

Los comandos se envían al MCU escribiendo su código en el puerto de datos A7h, siguiendo el protocolo de sincronización por bit de reloj descrito en la sección 15.2. Todos los parámetros de cadena van precedidos de un byte con la longitud.

Códigos de error devueltos por los comandos

Código	Significado
0	Éxito
1	Archivo o directorio no encontrado
2	No es un directorio
3	Error de operación (no se pudo crear/borrar/renombrar)
4	El archivo o directorio ya existe
5	Archivo demasiado grande
6	No se pudo crear el archivo destino
7	Error de escritura
8	Error de lectura parcial
12	No hay ningún archivo VGM abierto
13	Operación no permitida en directorio T81
14	Parámetro de joystick inválido

Comandos de sistema

Cód.	Nombre	Parámetros	Respuesta	Descripción
0	NOP	—	—	Sin operación. Solo sincroniza el reloj.
1	VERSION	—	1 byte: versión	Devuelve la versión del MCU. Mismo formato que el byte en 2004h.
32	GETBYTE	1 byte: índice (0–255)	1 byte: valor	Lee un byte de la memoria interna. Índices 0–127: variables volátiles. Índices 128–255: EEPROM (persistente).
33	SETBYTE	1 byte: índice + 1 byte: valor	—	Escribe un byte en la memoria interna.

Comandos de sistema de archivos

Cód.	Nombre	Parámetros	Respuesta	Descripción
2	PWD	—	String + EOT + status	Devuelve el directorio actual en codificación ZX81.
3	CD	String: ruta	Status	Cambia el directorio actual. Admite rutas absolutas (/) y relativas.
4	DEL	String: archivo	Status	Borra un archivo del directorio actual. Sin comodines.

Cód.	Nombre	Parámetros	Respuesta	Descripción
5	MKDIR	String: nombre	Status	Crea un subdirectorio.
6	RMDIR	String: nombre	Status	Elimina un directorio vacío.
7	MOVE	String: origen + String: destino	Status	Renombra o mueve un archivo.
8	COPY	String: origen + String: destino	Status	Copia un archivo. La fecha/hora no se preserva.
9	LOAD	String: nombre	2B longitud + N bytes + Status	Carga un archivo. .P/.81: calcula tamaño real. .ROM: carga en dirección 0 y resetea. .WAV: reproduce.
10	SAVE	String: nombre + 2B longitud + N bytes	Status	Guarda un bloque de datos como archivo en la SD.
11	TYPE	String: nombre	String char a char + EOT + Status	Envía el contenido de un archivo de texto. Con * busca en /MAN/ con extensión .TXT.
12	DIR	String: ruta/comodín	String char a char + EOT + Status	Lista el directorio, incluyendo tamaños de archivo.
14	FREE_TXT	—	String + EOT + Status	Devuelve espacio total y libre de la SD como texto.
15	FREE	—	4B total + 4B libre + Status	Espacio total y libre en KB como valores de 32 bits little-endian.
16	OPENDIR	String: ruta/comodín	Status	Abre un directorio y construye un array interno (máx. 512 entradas).
17	GETROWLEN	2B: índice	1B: longitud + Status	Longitud del nombre de la entrada índice del array abierto con OPENDIR.
18	GETROW	2B: índice	1B: longitud + N bytes + Status	Nombre de la entrada índice en codificación ZX81. Índice 0 = directorio actual. Directorios entre < y >.

Comandos de control del hardware

Cód.	Nombre	Parámetros	Respuesta	Descripción
19	**ENABLE_MC45**	**—**	**—**	**Activa el modo MC45 (código máquina en bloques 4 y 5).**
20	DISABLE_MC45	—	—	Desactiva el modo MC45.
21	JOY	String: 5 bytes de teclas ZX81	Status	Configura el mapeo del joystick: izquierda, derecha, arriba, abajo, fuego.
27	SEL_128CHARS	—	—	Activa el modo de 128 caracteres definibles. Equivale a LOAD *128C.
28	SEL_64CHARS	—	—	Activa el modo estándar de 64 caracteres. Equivale a LOAD *64C.
29	FULLPAGING	—	—	Activa el modo de paginación completa (512 KB, 64 páginas).
30	HALFPAGING	—	—	Activa el modo de paginación simple (256 KB, 32 páginas).
48	ENABLE_48K	—	—	Activa el modo RAM extendida de 48 KB. Equivale a LOAD *RAM48.
49	DISABLE_48K	—	—	Desactiva el modo RAM extendida. Equivale a LOAD *RAM48 STOP.

Comandos de síntesis de voz

Cód.	Nombre	Parámetros	Respuesta	Descripción
22	**BINARY_SAY**	**String: bytes de alófonos**	**Status**	**Reproduce alófonos en formato binario. Síncrono (bloquea hasta terminar).**
23	SAY	String: texto ASCII	Status	Convierte texto a fonemas y reproduce. Con * como primer carácter: background. Equivale a LOAD *SAY.

Comandos AY / sonido

Cód.	Nombre	Parámetros	Respuesta	Descripción
24	**AY_SET_REG**	**1B: registro (0–15) + 1B: valor**	**—**	**Escribe un valor en un registro del emulador AY.**
25	AY_GET_REG	1B: registro (0–15)	1B: valor	Lee el valor actual de un registro del emulador AY.
26	AY_PLAY	String: canal A + String: B + String: C	Status	Reproduce hasta tres cadenas PLAY simultáneas. Con * en canal A: background. Equivale a LOAD *PLAY.

Comandos VGM

Cód.	Nombre	Parámetros	Respuesta	Descripción
34	**PLAY_VGM**	**String: nombre de archivo**	**Status**	**Abre y comienza a reproducir un archivo VGM en background. Añade .vgm si no tiene extensión.**
35	STOP_VGM	—	—	Detiene la reproducción VGM y reinicia el emulador AY.
36	PAUSE_VGM	—	—	Pausa la reproducción VGM.
37	CONT_VGM	—	—	Reanuda la reproducción VGM pausada.
38	LOOP_VGM	1B: modo (0=no bucle, 1=bucle)	—	Establece el modo de bucle del reproductor VGM.

Comandos PEG

Cód.	Nombre	Parámetros	Respuesta	Descripción
40	**LOAD_PEG**	**1B: dirección + String: datos hex**	**—**	**Carga instrucciones PEG en la memoria del generador. 2 bytes por instrucción en little-endian.**
41	PLAY_PEG	1B: hilo (0–2) + 1B: dirección	—	Inicia la ejecución de un programa PEG en el hilo indicado.
42	STOP_PEG	1B: hilo (0–2)	—	Detiene y reinicia el hilo PEG indicado.
43	PAUSE_PEG	1B: hilo (0–2)	—	Pausa el hilo PEG indicado.
44	CONT_PEG	1B: hilo (0–2)	—	Reanuda el hilo PEG indicado.
45	SDLOAD_PEG	String: nombre + 1B: dirección	Status	Carga un archivo .PEB desde la SD en la memoria PEG. Tamaño máximo: 512 bytes.

Comandos RTC y batería

Cód.	Nombre	Parámetros	Respuesta	Descripción
50	RTC	String: fecha/hora (o vacío para leer)	Si lectura: String ZX81 + Status. Si escritura: Status	Sin parámetros: devuelve fecha/hora. Con parámetros: ajusta el reloj. Formatos: AAAA-MM-DD HH:MM:SS.CC / AAAA-MM-DD HH:MM:SS / AAAA-MM-DD / HH:MM:SS.CC / HH:MM:SS / HH:MM.
52	BAT	—	5 bytes ASCII + Status	Devuelve el nivel de batería del RTC como string de 5 caracteres en formato V.mmm (codificación ZX81).

16. Solución de problemas

16.1 El interface no arranca o el ZX81 se queda bloqueado

Síntoma	Solución
El ZX81 no muestra nada al encender	Comprueba que el interface está correctamente insertado en el puerto de expansión. Desconéctalo y vuelve a conectarlo con el ZX81 apagado.
El LED STAT no se ilumina	Comprueba que la tarjeta SD está insertada y que contiene la carpeta SYS con su contenido completo. Sin ella el interface no arranca.
Pantalla en blanco o con ruido	Asegúrate de que los pines del conector de expansión no están doblados o sucios.
El ZX81 arranca pero los comandos del interface no funcionan	Verifica la versión del firmware con LOAD *VER. Para actualizar, copia firmware.bin en la raíz de la SD y enciende el ZX81.

16.2 Problemas con la tarjeta microSD

Síntoma	Solución
Error H al intentar cargar	Comprueba que la SD está correctamente insertada. Extráela y vuélvela a insertar.
La SD no se reconoce	Verifica que está formateada en FAT32 (no exFAT ni NTFS).
El directorio aparece vacío	Comprueba que los archivos tienen extensión .P y nombres con caracteres permitidos.
Error G al cargar un programa	El archivo no existe con ese nombre. Usa LOAD *DIR para ver los nombres exactos.
Error J al guardar	La tarjeta SD está llena. Usa LOAD *FREE para comprobar el espacio disponible.

16.3 El reloj pierde la hora al apagar

El reloj en tiempo real se alimenta de la pila botón CR2032. Si el reloj pierde la hora sistemáticamente al apagar el ZX81, probablemente la pila necesita ser reemplazada.

Comprueba el nivel de carga con:

```
LOAD *BAT
```

Si el voltaje es inferior a 2.5V aproximadamente, sustituye la pila por una CR2032 nueva. La pila se encuentra en la placa del interface y puede extraerse con una herramienta plana fina.

16.4 El joystick no responde

Síntoma	Solución
El joystick no hace nada	Configura el mapeo con LOAD *JOY "OPQA " (u otro mapeo según el juego) antes de lanzar el programa.
Solo funciona alguna dirección	Comprueba que la cadena de configuración tiene exactamente 5 caracteres.
El joystick mueve pero no dispara	Verifica que el quinto carácter de la cadena JOY corresponde a la tecla de fuego del juego.

16.5 El sonido no funciona

Síntoma	Solución
No hay sonido con LOAD *PLAY	Comprueba que el cable de audio está conectado a la salida correspondiente del interface.
La voz no se entiende	Prueba con frases cortas en inglés. Escribe las palabras fonéticamente si el resultado no es satisfactorio.
El VGM no suena	Verifica que el archivo es un VGM con datos del chip AY únicamente. Los VGMs con otros chips no son compatibles.

17. Actualización del firmware

El SD81 Booster tiene dos componentes de firmware actualizables de forma independiente: el microcontrolador (MCU) y la FPGA/CPLD. Cada uno utiliza una herramienta y un proceso diferente.

> ⚠ *No interrumpas el proceso de actualización una vez iniciado. Una actualización incompleta puede dejar el interface en un estado no operativo que requeriría asistencia técnica para recuperarlo.*

17.1 Actualización del microcontrolador (MCU)

El SD81 Booster incorpora un bootloader en el MCU que permite actualizar el firmware de forma muy sencilla, sin necesidad de ninguna herramienta de programación externa ni cable especial: basta con copiar el archivo de firmware en la tarjeta microSD.

Requisitos:

- Tarjeta microSD del interface.
- Archivo de firmware firmware.bin, disponible en el repositorio del proyecto.

Proceso:

14. Descarga el archivo firmware.bin desde el repositorio del proyecto.
15. Copia firmware.bin en la raíz de la tarjeta microSD (no en ninguna subcarpeta).
16. Inserta la tarjeta en el interface con el ZX81 apagado.
17. Enciende el ZX81. El bootloader detectará automáticamente el archivo y comenzará la actualización. El LED STAT indicará el progreso.
18. Cuando la actualización termine correctamente, el bootloader borrará el archivo firmware.bin de la SD y arrancará el nuevo firmware automáticamente.

⚠ *No apagues el ZX81 ni extraigas la tarjeta SD durante la actualización. Si el proceso se interrumpe, el interface podría quedar en un estado no operativo.*

ℹ *Si el LED STAT muestra un error al arrancar tras la actualización, comprueba que el archivo firmware.bin era el correcto para tu versión del hardware y repite el proceso.*

Verifica la versión instalada con:

```
LOAD *VER
```

17.2 Actualización de la FPGA/CPLD

La FPGA/CPLD gestiona el vídeo, el mapeador de memoria, el puerto Chroma81 y los modos Superfast. Su actualización está orientada exclusivamente a personal técnico especializado y requiere un cable programador Xilinx Platform Cable USB, diferente al USB-C usado para el MCU.

Los binarios de la FPGA/CPLD están disponibles en el repositorio del proyecto. Para las instrucciones detalladas del proceso de programación con Xilinx ISE/Vivado, consulta la documentación técnica del repositorio.

⚠ *Una programación incorrecta de la FPGA/CPLD puede dejar el interface permanentemente inoperativo. No realices esta operación a menos que tengas experiencia con herramientas de programación Xilinx.*

18. Glosario

Término	Definición
Alófono	Unidad mínima de sonido del habla usada por el sintetizador de voz. El SD81 Booster usa los alófonos del chip SP0256.

Término	Definición
Bloque	División de 8 KB del espacio de direccionamiento del Z80. El SD81 Booster divide los 64 KB del Z80 en 8 bloques (0–7).
CPLD / FPGA	Circuito lógico programable que implementa por hardware la lógica de vídeo, el mapeador de memoria y otras funciones del interface.
FAT32	Sistema de archivos requerido por la tarjeta microSD del interface. Incompatible con exFAT y NTFS.
FAST	Token del BASIC del ZX81 (SHIFT+F). En el SD81 Booster, activa el modo de carga/guardado desde la SD.
Fichero de pantalla (HFILE)	Bloque de memoria que contiene los datos de la pantalla en los modos Superfast.
HRG	High Resolution Graphics. Modo de alta resolución del ZX81.
MCU	Microcontrolador. El chip que gestiona la SD, el sonido, el RTC y la comunicación con el Z80.
Página	División de 8 KB de la RAM del interface. Los 512 KB se dividen en 64 páginas (0–63) que pueden mapearse a cualquier bloque.
PEG	Programmable Effects Generator. Máquina virtual para reproducir efectos de sonido en background sin usar la CPU del ZX81.
RTC	Real Time Clock. Reloj en tiempo real incorporado en el SD81 Booster, alimentado por una pila CR2032.
SLOW	Token del BASIC del ZX81 (SHIFT+D). En el SD81 Booster, activa el modo de carga/guardado desde cinta (audio).
SP0256	Chip sintetizador de voz de General Instrument, base del sintetizador del SD81 Booster. Usado también en el Currah MicroSpeech y The Voice.
Superfast	Modo en el que el hardware del SD81 Booster gestiona el refresco de pantalla liberando la CPU del ZX81 para otras tareas.
T81	Formato de archivo contenedor que agrupa múltiples programas ZX81. El interface puede navegar su contenido como si fuera un directorio. (fase alfa)
Token	En el BASIC del ZX81, cada palabra reservada se almacena como un único byte. Se introducen con combinaciones de SHIFT.
VGM	Video Game Music. Formato de archivo de música que el SD81 Booster puede reproducir en background usando el emulador AY.
Vídeo inverso	Modo de visualización del ZX81 en el que el fondo y el carácter intercambian colores. Se activa con SHIFT+9 antes del carácter.

19. Historial de versiones del firmware

Versión	Fecha	Novedades principales
1.0	2025	Primera versión de lanzamiento público.

> *Este historial se actualizará con cada nueva versión del firmware. Consulta el repositorio del proyecto para ver el registro completo de cambios.*

20. Referencias

El proyecto SD81 Booster

Repositorio oficial — código fuente, firmware, esquemas y documentación técnica:

https://codeberg.org/Retrostuff/SD81-Booster

Va de Retro — foro de retroinformática en español donde se anunció una versión anterior del interface:

https://www.va-de-retro.com/foros/portal

Herramientas

STM32CubeProgrammer — herramienta de programación del MCU STM32, necesaria para actualización vía USB en caso de recuperación:

https://www.st.com/en/development-tools/stm32cubeprog.html

Documentación técnica de referencia

Chip sintetizador de voz SP0256-AL2 (General Instrument) — hoja de características del chip en el que se basa el sintetizador de voz del SD81 Booster:

https://rarewaves.net/wp-content/uploads/2018/09/SP0256-AL2.pdf

Interface Chroma81 — documentación del interface de color para ZX81 cuya funcionalidad implementa el SD81 Booster. El enlace original ya no está disponible; puede encontrarse en archivos web:

http://www.fruitcake.plus.com/Sinclair/ZX81/Chroma/ChromaInterface_Documentation.htm

Formato de archivo .P y .P81 — especificación técnica de los formatos de programa del ZX81:

https://k1.spdns.de/Develop/Projects/zasm/Info/O80%20and%20P81%20Format.txt

Especificación del formato VGM (Video Game Music) — formato de archivo de música utilizado por el reproductor VGM del interface:

https://vgmrips.net/wiki/VGM_Specification

ROM del ZX81

El SD81 Booster incluye una ROM modificada basada en la disassembly original del ZX81. Créditos:

Geoff Wearmouth — disassembly comentada de la ROM del ZX81 (preservada en archive.org):

https://web.archive.org/web/20150815035607/http://www.wearmouth.demon.co.uk/zx81.htm

Tomaž Šolc — preservación de la disassembly:

https://www.tablix.org/~avian/spectrum/rom/

Apéndice A — Referencia completa del comando PLAY

Tabla de duraciones

Valor	Nombre	Duración a 60 bpm
1	Semicorchea	0.25 s
2	Semicorchea con puntillo	0.375 s
3	Corchea	0.5 s
4	Corchea con puntillo	0.75 s
5	Negra	1 s
6	Negra con puntillo	1.5 s
7	Blanca	2 s
8	Blanca con puntillo	3 s
9	Redonda	4 s
10	Tresillo de semicorchea	0.1667 s
11	Tresillo de corchea	0.3333 s
12	Tresillo de negra	0.6667 s

Efectos de envolvente (W)

Código	Forma	Descripción
W0	|_____	Decaimiento, luego silencio
W1	/|_____	Ataque, luego silencio
W2	|‾‾‾‾‾	Decaimiento, luego sostenido
W3	/‾‾‾‾‾	Ataque, luego sostenido
W4	|||||	Decaimiento repetido (tremolo)

Código	Forma	Descripción
W5	/|/|/|/|	Ataque repetido
W6	/\/\/\/	Ataque y decaimiento repetidos
W7	\/\/\/\	Decaimiento y ataque repetidos

Tabla resumen de parámetros

Parámetro	Descripción
C..B	Nota en octava actual
Inv(C..B)	Nota en octava siguiente (vídeo inverso)
=	Sostenido (siguiente nota)
£	Bemol (siguiente nota) o silencio
1..12	Duración desde este punto
-	Ligadura de duración
N / espacio	Separador de números
O<n>	Octava (0–8, defecto 4)
T<n>	Tempo en bpm (60–240, defecto 120) — solo canal A
V<n>	Volumen (0–15)
W<n>	Efecto de envolvente (0–7)
U	Activa envolvente en el canal
X<n>	Velocidad de envolvente (0–65535; 6927 ≈ 1 s)
M<n>	Selección de canales activos y modo (0–63)
( )	Repetir sección una vez más
)	Repetir desde el inicio indefinidamente
H	Detener PLAY en todos los canales
*	(Solo canal A) Reproducción en background

Apéndice B — Referencia del generador de efectos PEG

El PEG es una máquina virtual de 16 bits con acceso a los 16 registros del chip AY (R0–R15) y 16 variables de propósito general (V0–V15). Soporta hasta 3 hilos de ejecución paralelos y hasta 256 palabras de programa.

Instrucción	Codificación	Descripción
LD R,XX	0R XX	Carga registro AY con valor 8 bits

Instrucción	Codificación	Descripción
`ADD R,XX`	`1R XX`	Suma valor 8 bits a registro AY
`LD V,XX`	`2R XX`	Carga variable con valor 8 bits (resto cero)
`ADD V,XX`	`3R XX`	Suma valor 8 bits a variable
`LD R,R`	`40 RR`	Carga registro con otro registro
`LD R,V`	`41 RV`	Carga registro con variable
`LD V,R`	`42 VR`	Carga variable con registro
`LD V,V`	`43 VV`	Carga variable con otra variable
`ADD V,V`	`44 VV`	Suma dos variables
`SUB V,V`	`45 VV`	Resta segunda de primera
`ADC V,V`	`46 VV`	Suma con acarreo
`SBC V,V`	`47 VV`	Resta con acarreo
`NOT V,V`	`48 VV`	Primera = complemento a 1 de segunda
`AND V,V`	`49 VV`	AND bit a bit
`OR V,V`	`4A VV`	OR bit a bit
`XOR V,V`	`4B VV`	XOR bit a bit
`MUL V,V`	`4C VV`	Multiplica (resultado 32 bits en V y V+1)
`DIV V,V`	`4D VV`	Divide (cociente en V, resto en V+1)
`SHR V,X`	`4E VX`	Desplazamiento a la derecha
`SHL V,X`	`4F VX`	Desplazamiento a la izquierda
`MUL V,XX`	`5V XX`	Multiplica variable por constante
`DIV V,XX`	`6V XX`	Divide variable por constante
`SUB V,XX`	`7V XX`	Resta constante de variable
`DJNZ V,XX`	`8V XX`	Decrementa y salta si no es cero
`WAIT XXX`	`9X XX`	Espera tiempo en ms
`WAIT V`	`A0 0V`	Espera tiempo indicado en variable (ms)
`HALT`	`A0 10`	Detiene el efecto
`JR XX`	`A1 XX`	Salto relativo

> ℹ️ *Los offsets de salto son relativos a la instrucción siguiente. El ensamblador PEG incluido en el repositorio gestiona esto automáticamente.*

Apéndice C — Diccionario del sintetizador de voz

El sintetizador se basa en los fonemas del chip SP0256 de General Instrument (Currah MicroSpeech, The Voice). El texto se analiza de izquierda a derecha buscando la coincidencia más larga posible.

Palabras reconocidas (selección por longitud)

13 caracteres: INVESTIGATORS, IRRESPONSIBLE

12 caracteres: INVESTIGATOR

11 caracteres: INVESTIGATE

10 caracteres: CORRECTING

9 caracteres: COGNITIVE, CORRECTED, SEPTEMBER, SINCERELY, SINCERITY, INTERFACE

8 caracteres: CHECKERS, CHECKING, COMPUTER, CORRECTS, DAUGHTER, DECEMBER, EIGHTEEN, FEBRUARY, FREEZERS, FREEZING, NINETEEN, NOVEMBER, PLEDGING, SATURDAY

7 caracteres: BOOSTER, CHECKED, CHECKER, CORRECT, FREEZER, JANUARY, MINUTES, OCTOBER, PLASTIC, SIXTEEN, TUESDAY, COLLIDE

6 caracteres: AUGUST, COOKIE, EQUALS, EXTENT, FRIDAY, FROZEN, MONDAY, SUNDAY, TALKED, TALKER, TWENTY

5 caracteres: APRIL, CHECK, CROWN, EIGHT, EQUAL, ERROR, FIFTY, HELLO, MARCH, MONTH, SIXTY, TALKS, THREE, WORLD

4 caracteres: DATE, FIVE, FOUR, HAVE, JUNE, NINE, RAYS, TALK, THIS, TIME, WHAT, WHOA, WILL, ZX81

3 caracteres: ACK, ACT, ADD, AMP, ASH, ASK, BAD, BED, BIG, BOX, BUT, CAR, END, EST, GET, HAS, HIM, ICK, IMP, ING, INK, JOB, KEY, MAY, NOT, NOW, OLD, OUR, OUT, RAY, RED, SIX, SUN, TEN, THE, TOP, TWO, YES — y todas las sílabas con dígrafos DH, NG, SH, TH, WH.

Puntuación y pausas

Carácter	Efecto
`Espacio`	Pausa corta
`,`	Pausa media
`.` `;` `:`	Pausa larga

Alófonos directos (uso avanzado)

Pausas: PA1, PA2, PA3, PA4, PA5

Vocales: AA, AE, AH, AO, AW, AX, AY, EH, ER1, ER2, EY, IH, IY, OW, OY, UH, UW1, UW2, XR, YR

Consonantes: BB1, BB2, CH, DD1, DD2, DH1, DH2, EL, FF, GG1, GG2, GG3, HH1, HH2, JH, KK1, KK2, KK3, LL, MM, NG, NN1, NN2, OR, PP, RR1, RR2, SH, SS, TH, TT1, TT2, VV, WH, WW, YY1, YY2, ZH, ZZ

Apéndice D — Sistema de paginación de memoria

Asignación inicial de páginas

Bloque	Página inicial	Rango / Uso
0	0	0000–1FFF (ROM, solo lectura)
1	1	2000–3FFF (ROM de expansión)
2	2	4000–5FFF (RAM principal)
3	3	6000–7FFF (RAM principal)
4	4	8000–9FFF (RAM ampliada)
5	5	A000–BFFF (RAM ampliada)
6	2	C000–DFFF (espejo de bloque 2)
7	3	E000–FFFF (espejo de bloque 3)

Reglas de uso

- El bloque 0 es siempre de solo lectura. Los bloques 1–7 son siempre de lectura/escritura.
- Una misma página puede estar mapeada en más de un bloque simultáneamente.
- Los bloques 6 y 7 deben espejar los bloques 2 y 3 respectivamente para que el sistema de vídeo funcione. Excepción: si el fichero de pantalla está completamente en el bloque 2, el bloque 7 puede mapearse libremente, y viceversa.
- Los bloques 4 y 5 siempre pueden mapearse libremente.
- Por las peculiaridades del hardware del ZX81, los bloques 4–7 solo pueden usarse para datos, no para ejecutar código (salvo con el modo MC45 activo para los bloques 4 y 5).

Modificación de la ROM

Es posible modificar la ROM mapeando la página 0 a cualquier bloque con escritura habilitada y realizando los cambios allí. También puede sustituirse completamente la ROM cargando el contenido deseado en cualquier página y mapeándola al bloque 0.

Apéndice E — Puerto del interface Chroma81 (7FEFh)

El SD81 Booster implementa la interfaz de color del Chroma81 a través del puerto 7FEFh (01111111 11101111 en binario). Este puerto puede leerse y escribirse.

Escritura (OUT 7FEFh)

Permite configurar el modo de color y el color del borde:

Bits	Función
7–6	Reservado para uso futuro. Siempre a 0.
5	Activar modo color: 1 = color activado.
4	Modo de color: 0 = código de carácter, 1 = fichero de atributos.
3	Bit de brillo del color del borde.
2–0	Color del borde en formato GRB (Green-Red-Blue).

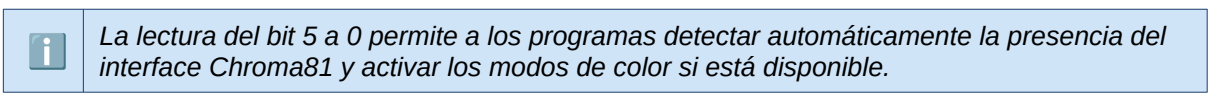
Formato del color de borde (bits 2–0, formato GRB)

Valor (GRB)	Color
000	Negro
001	Azul
010	Rojo
011	Magenta
100	Verde
101	Cian
110	Amarillo
111	Blanco

Lectura (IN 7FEFh)

Permite detectar si el modo color está disponible y leer el estado del VSync:

Bit	Función
7–6	No usado (reservado).
5	0 = modos de color disponibles. Siempre 0, lo que indica que el modo color está siempre activo.
4–1	No usado (reservado).
0	Estado de la interrupción VSync. 1 = el haz de pantalla está en el periodo de blanking vertical (pantalla pintada).



La lectura del bit 5 a 0 permite a los programas detectar automáticamente la presencia del interface Chroma81 y activar los modos de color si está disponible.

Sincronización con VSync:

El bit 0 permite a la CPU esperar a que la pantalla haya terminado de pintarse antes de actualizar su contenido, evitando parpadeos y artefactos visuales. Muchos juegos del ZX Spectrum usaban la instrucción HALT o una rutina de interrupción para sincronizarse con el VSync; en el SD81 Booster este mecanismo es el equivalente directo para esa funcionalidad:

```
        ; Esperar al inicio del VSync
WAIT:   in   a,($A7)
        rrca              ; bit 0 al carry
        jr   nc,WAIT      ; si carry=0, pantalla todavía pintándose
        ; aquí ya se ha completado el refresco, seguro actualizar
vídeo
```

Apéndice F — Modos Superfast y Spectrum

El problema del vídeo en el ZX81 original

El ZX81 en modo SLOW gestiona el vídeo por software: durante el refresco de la pantalla, la CPU debe ejecutar instrucciones NOP mientras el hardware genera la señal de vídeo, lo que consume una parte importante del tiempo de procesador disponible. El SD81 Booster resuelve esto con el modo Superfast.

Modo Superfast

En el modo Superfast, el hardware del interface toma el control del bus de datos durante el refresco de pantalla, colocando los datos de vídeo directamente sin intervención de la CPU. Esto libera al procesador para ejecutar código útil durante todo el ciclo, aumentando significativamente la velocidad efectiva del sistema.

Antes de activar el modo es necesario indicar la dirección del fichero de pantalla (HFILE):

```
POKE 2043, HFILE_low    : REM parte baja de la dirección del fichero
de pantalla
POKE 2044, HFILE_high   : REM parte alta de la dirección del fichero
de pantalla
```

Las tres variantes del modo Superfast se seleccionan mediante POKE en la dirección 2045:

POKE	Modo
`POKE 2045, 170`	Superfast texto (modo texto estándar acelerado)
`POKE 2045, 171`	Superfast HiRes nativo (alta resolución en formato ZX81)
`POKE 2045, 172`	Superfast HiRes Spectrum (alta resolución en formato Spectrum)
`POKE 2045, 85`	Desactivar modo Superfast

Modo Spectrum

El modo Spectrum (POKE 2045,172) reordena las líneas de pantalla para que coincidan con la organización de la pantalla del ZX Spectrum, facilitando la conversión de programas entre ambas plataformas. En el ZX81 estándar las líneas se organizan de forma diferente a como lo hace el Spectrum; este modo elimina esa diferencia por hardware.

> ⚠ *En el modo Spectrum se emula el puerto del beeper y borde del Spectrum (ULA write port = FBh), donde los bits 2–0 controlan el color del borde y los bits 4–3 controlan el beeper. En este modo el puerto FBh queda ocupado y se pierde la compatibilidad con la ZX Printer.*

Control del borde

Cambiar los atributos del borde:

```
POKE 2046, <attr>
```

Definir y activar un patrón de borde:

Es posible definir un patrón de 8 bytes que se repetirá a lo largo del borde de la pantalla, permitiendo rellenarlo completamente con un diseño personalizado:

```
POKE 2048, byte0  : REM primer byte del patrón
POKE 2049, byte1
...
POKE 2055, byte7  : REM último byte del patrón
POKE 2047, 170    : REM activar patrón de borde
POKE 2047, 85     : REM desactivar patrón de borde
```

Puerto ULA en modo Spectrum (FBh):

Bits	Función
4–3	Control del beeper
2–0	Color del borde

Sincronización con VSYNC

En el ZX Spectrum, muchos juegos utilizaban la instrucción HALT o una rutina de interrupción IM1/IM2 para sincronizarse con el barrido vertical de la pantalla y conseguir animaciones fluidas sin parpadeo.

En el SD81 Booster esta funcionalidad se sustituye mediante la lectura del bit 0 del puerto A7h, que refleja el estado de la interrupción de sincronismo vertical (VSYNC). La CPU puede esperar a este bit para sincronizarse con el inicio del refresco de pantalla sin necesidad de interrupciones ni de HALT. Consulta la sección 14.2 para el ejemplo de código completo.

Resumen de POKEs de control

Dirección	Valor	Función
2043	<bajo>	Parte baja de la dirección del fichero de pantalla

Dirección	Valor	Función
2044	<alto>	Parte alta de la dirección del fichero de pantalla
2045	170	Activar Superfast texto
2045	171	Activar Superfast HiRes nativo
2045	172	Activar Superfast HiRes Spectrum
2045	85	Desactivar Superfast
2046	<attr>	Cambiar atributos del borde
2047	170	Activar patrón de borde
2047	85	Desactivar patrón de borde
2048–2055	<datos>	Definir patrón de borde (8 bytes)