SD81 BOOSTER

Manual de usuario



Edición revisada · Español
23 / 10 / 2026
SD · WiFi · Firmware · FPGA · ROM

Índice

Sobre esta edición ... 3
Guía de inicio rápido ... 3
1. Introducción ... 4
2. Descripción del hardware ... 4
3. Contenido de la caja ... 7
4. Instalación ... 7
5. Preparación de la tarjeta microSD ... 8
6. Primeros pasos ... 9
7. Carga y guardado desde la SD ... 9
8. Gestión de archivos y directorios ... 12
9. Funciones adicionales ... 13
10. Sonido ... 21
11. Gestión de memoria ... 25
12. Extensiones BASIC avanzadas ... 28
13. Ejemplos de programas ... 31
14. Códigos de error ... 34
15. Para programadores ... 35
16. Solución de problemas ... 44
17. Actualización por microSD: firmware, FPGA, ROM y WiFi ... 46
18. Glosario ... 49
19. Historial de versiones del firmware ... 50
20. Referencias ... 50
Apéndice A — Referencia completa del comando PLAY ... 51
Apéndice B — Referencia del generador de efectos PEG ... 53
Apéndice C — Diccionario del sintetizador de voz ... 54
Apéndice D — Sistema de paginación de memoria ... 55
Apéndice E — Puerto del interface Chroma81 (7FEFh) ... 56
Apéndice F — Modos Superfast y Spectrum ... 57
Apéndice G — Referencia técnica de audio: chip AY, VGM y alófonos ... 64
Apéndice H — Sprites por hardware ... 67
Apéndice I — CP/M: un sistema operativo alternativo ... 69
Apéndice J — El explorador de archivos ... 75
Apéndice K — El emulador EightyOne-CrossPlatform ... 79
Apéndice L — Depurador por hardware ... 87
Apéndice M — Red para programadores: MQTT y juegos por turnos ... 91
Fuentes de la revisión ... 94

Sobre esta edición

Revisión documental del repositorio de Alejandro Valero (wilco2009). Conserva los capítulos y apéndices de los manuales fuente, con revisión de las instrucciones de actualización, configuración NTP, referencias e índice. No equivale a una prueba de actualización en hardware real.

Revisión consultada: `39b367a7e980157512b4c1928d10e97aedf1cfad` (20/10/2026).

Fecha de esta edición: 23/10/2026.

Diseño de hardware y firmware MCU: Alejandro Valero (wilco2009). Código Z80 / ROM modificada: Pedro Gimeno (pgimeno). Textos e ilustraciones base: proyecto SD81 Booster.

Repositorio y fuentes · Releases

La sección 17 es la guía de actualización ampliada. Antes de utilizar binarios de una versión posterior, lee sus notas: los nombres o requisitos pueden cambiar.

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

INFO	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:

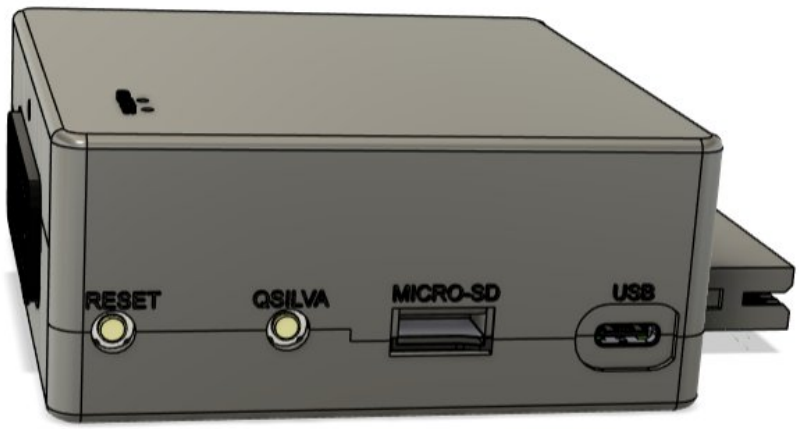
- Carga y guardado desde tarjeta microSD en formato .P, el mismo que utilizan los emuladores más populares.
- Hasta 512 KB de RAM mediante un mapeador de memoria en bloques de 8 KB.
- Emulación del chip de sonido AY-3-8910/12 con soporte para el comando PLAY.
- Reproductor de archivos VGM en segundo 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.

INFO	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



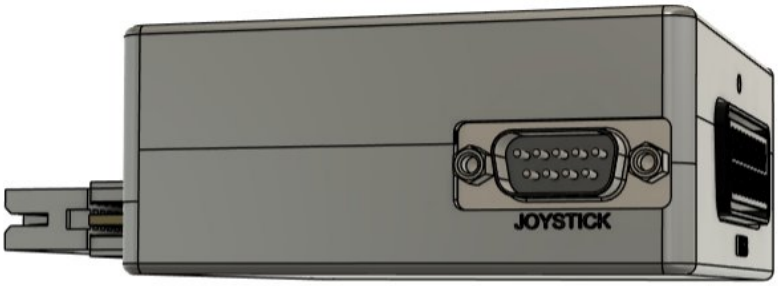
Vista lateral derecha: RESET, QSILVA, MICRO-SD y USB

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. Pulsado entre 1 y 3 segundos, pausa el programa y muestra el depurador; más de 3 segundos, guarda un snapshot (ver Apéndice L).
- Ranura MICRO-SD: inserta aquí la tarjeta microSD con los programas, ROMs y datos del sistema.
- Puerto USB-C: tiene dos usos. Como puerto de actualización de firmware en caso de recuperación via USB (ver sección 17). Como consola de depuración: al conectarlo al ordenador aparece un puerto serie asociado al chip CH340 (puede requerir drivers). Con un programa de terminal serie es posible monitorizar mensajes de estado y error del interface en tiempo real.

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, organizadas en tres fases:

Color / Patrón	Significado
Actualización de firmware	
Parpadeo Azul/Rojo	Error inicializando tarjeta SD
Amarillo fijo	Actualizando firmware
Parpadeo Blanco/Rojo	Error de actualización
Arranque	
Rojo parpadeante	Inicializando puerto serie
Naranja parpadeante	Esperando a la FPGA
Parpadeo Azul/Rojo	Error inicializando tarjeta SD
Parpadeo Naranja/Rojo	Error escribiendo la ROM en RAM
Parpadeo Amarillo	Inicializando RTC
Verde fijo	Sistema inicializado correctamente

Color / Patrón	Significado
Funcionamiento	
Verde fijo	Interface listo y modo Quick Silva apagado
Cyan fijo	Interface listo y modo Quick Silva encendido

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

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

4. Instalación

4.1 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.

4.2 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.

AVISO	Conectar o desconectar el interface con el ZX81 encendido puede dañar tanto el interface como el ordenador.

4.3 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
```

INFO	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.

5. Preparación de la tarjeta microSD

5.1 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:

1. 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.
2. 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.

AVISO	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.

5.2 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.

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

5.3 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)
```

5.4 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.

6. Primeros pasos

6.1 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
```

INFO	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
```

INFO	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).

6.2 Cargando tu primer programa desde la SD

1. Asegúrate de que tienes al menos un archivo .P en la tarjeta SD.
2. Enciende el ZX81 con el interface y la SD insertados.
3. En el cursor K, escribe el siguiente comando, sustituyendo NOMBRE por el nombre del archivo sin extensión:

```
LOAD FAST “NOMBRE”
```

1. Pulsa ENTER. El programa se cargará en unos instantes y arrancará automáticamente si tiene autoarranque.

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

7. Carga y guardado desde la SD

7.1 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
```

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

7.2 Guardar un programa

Para guardar el programa actual en la SD:

```
SAVE FAST “NOMBRE”
```

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

AVISO	Si ya existe un archivo con ese nombre, será sobreescrito sin aviso previo.

7.3 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
```

INFO	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.

AVISO	El token FAST es obligatorio para que CODE funcione correctamente. Si se usa LOAD “NOMBRE” CODE 30000 sin FAST (incluso con el modo SD activo), el archivo se cargará en la dirección por defecto 16393 (4009h), ignorando la dirección especificada. Esto es una limitación de compatibilidad del firmware.

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.

7.4 Cargar siempre desde cinta (independientemente del modo)

Si el modo SD está activo pero quieres cargar un archivo concreto desde cinta:

```
LOAD SLOW “NOMBRE”
```

7.5 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.
- **LOAD *ROMLOCK** antes de cargar: algunos programas antiguos escriben por su cuenta en direcciones de la zona de ROM (por ejemplo para detectar si hay RAM), sin saber que el interface usa algunas como puertos de configuración (2041-2062, 2090-2098 y los sprites, 2100-2128). Con LOAD *ROMLOCK esas escrituras se ignoran, como en una ROM real; LOAD *ROMLOCK STOP devuelve el comportamiento normal. Está desactivado por defecto y solo se puede cambiar con el comando, nunca con una escritura en memoria.

7.6 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.

AVISO	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.

7.7 Carga de snapshots de EightyOne — Comando Z81

El comando LOAD *Z81 restaura un snapshot en formato .Z81 guardado con el emulador EightyOne (ver Apéndice K): la memoria y el estado completo del procesador, de forma que el programa continúa exactamente en el punto en el que se guardó.

Cargar un snapshot:

> LOAD *Z81 “JUEGO.Z81”

El nombre debe incluir la extensión .Z81 y admite rutas, igual que LOAD FAST. Además de la memoria y los registros, se restauran el estado de las NMI, el modo WRX, el generador de caracteres (64, 128 o 256) y el color Chroma81 (RAM de color y modo), si el snapshot los incluye; lo que no aparezca en el fichero se deja como estaba. Si el fichero no existe o no es un snapshot válido, el comando termina con error. Los snapshots se cargan con el depurador por hardware (Apéndice L): hace falta el monitor /SYS/DEBUG.BIN en la tarjeta SD (viene en SD Content), y el comando no se puede usar con el programa parado en el depurador ni mientras se graba o se carga otro snapshot. En esos casos termina con el error P y no se carga nada.

Sirve también para programas que no arrancan bien con el interface conectado: se guarda en el emulador un snapshot con el programa ya en marcha y se carga directamente, sin pasar por su código de arranque. El explorador de archivos (Apéndice J) carga los .Z81 con este comando.

No uses LOAD FAST con un fichero .Z81: al no ser una extensión reconocida (ver 7.6), se cargaría en memoria tal cual, sin interpretarlo.

AVISO	Los snapshots de EightyOne se restauran tal como los ve el Z80 (de 8K a 64K). Los que graba el propio SD81 Booster (ver Apéndice L) guardan además las páginas de la RAM extendida y su asignación, los POKEs de control, el sonido y el estado del interface, y LOAD *Z81 los restaura enteros cuando el depurador está cargado (/SYS/DEBUG.BIN). Sin el depurador no se puede cargar ningún snapshot (error P).

8. Gestión de archivos y directorios

8.1 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 al estilo Unix:

- * representa cualquier número de caracteres (incluido ninguno).
- ? representa exactamente un carácter.

INFO	* casa con todos los archivos independientemente de si tienen extensión o no. En cambio, *.* solo casa con archivos que contienen un punto en el nombre. Para listar todos los archivos, usa *, no *.*.

Por ejemplo, para listar solo los archivos .P:

> LOAD *DIR “*.P”

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

8.2 Cambiar de directorio

Cambiar al directorio JUEGOS:

> LOAD *CD “JUEGOS”

Volver al directorio raíz:

> LOAD *CD “/”

Ver el directorio actual:

> LOAD *PWD

8.3 Crear y eliminar carpetas

Crear una carpeta nueva:

> LOAD *MD “NUEVACARPETA”

Eliminar una carpeta (debe estar vacía):

> LOAD *RD “CARPETAVACIA”

8.4 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”

INFO	TO es un token del BASIC (SHIFT + 4), no se escribe letra a letra.
INFO	La fecha y hora del archivo de destino no se preserva; el archivo copiado tendrá la fecha y hora del momento de la copia.

8.5 Espacio libre en la SD

> LOAD *FREE

INFO	Este cálculo puede tardar varios segundos en completarse.

8.6 Directorios T81 (función en fase alfa)

AVISO	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 “/”

INFO	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.

9. Funciones adicionales

9.1 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.

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

9.2 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$
```

9.3 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$
```

INFO	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.

9.4 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
```

INFO	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.

9.5 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”
```

Redirigir la salida de otros comandos a la ZX Printer:

El prefijo LOAD LPRINT puede usarse también con los comandos que normalmente muestran texto en pantalla, para redirigir su salida directamente a la ZX Printer:

```
LOAD LPRINT DIR

LOAD LPRINT DIR “*.P”

LOAD LPRINT FREE

LOAD LPRINT PWD

LOAD LPRINT VER
```

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 “arr/aba/izq/der/fue”

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

Ejemplo:

> LOAD *JOY “QAOP”

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

Teclas especiales:

Las teclas SHIFT y ENTER no tienen un carácter propio que pueda escribirse dentro de la cadena de configuración, por lo que se representan mediante dos caracteres sustitutos: * para SHIFT y < para ENTER.

> LOAD *JOY “QAOP<”

En este ejemplo el botón de fuego queda asignado a la tecla ENTER.

INFO	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.

9.7 Módulo WiFi (opcional)

El módulo WiFi es un accesorio opcional basado en un microcontrolador ESP32-C3 que se conecta al SD81 Booster y añade un servidor de archivos accesible por WiFi: permite listar, subir, descargar, borrar y organizar el contenido de la tarjeta microSD desde el navegador de un móvil, tablet u ordenador, sin sacar la tarjeta del interface.

INFO	El módulo WiFi no se incluye de fábrica en el SD81 Booster — es un accesorio opcional que se instala aparte. Si tu interface no lo lleva instalado, esta sección no aplica.

Configuración de la red WiFi:

Para que el módulo se conecte a tu red, crea un archivo de texto llamado WIFI.CFG dentro de la carpeta SYS de la tarjeta microSD, con el nombre de la red en una línea y la contraseña en la siguiente:

> MiRedWiFi
>
> MiContraseña

Puedes añadir más de una red repitiendo el mismo patrón (nombre, luego contraseña), una tras otra en el mismo archivo:

> CasaWifi
>
> passwordcasa
>
> TrabajoWifi
>
> passwordtrabajo

El módulo prueba las redes en el orden en que aparecen, hasta conectar con la primera disponible — útil para no tener que editar el archivo cada vez que te mueves de sitio (hasta 4 redes).

INFO	Guarda el archivo como texto plano, sin formato. Las redes deben ser de 2.4GHz — el módulo WiFi no es compatible con redes de 5GHz.

Acceso al servidor de archivos:

Con el módulo conectado a tu red, necesitas su dirección para acceder desde un navegador. Tienes varias formas de consultarla, de más a menos cómoda:

Desde el propio ZX81 (más sencillo, no necesita ningún otro dispositivo):

> LOAD THEN PRINT “*IP”

Muestra en pantalla la dirección IP actual del módulo.

Por nombre, sin teclear la IP:

Si tu dispositivo soporta mDNS (de serie en macOS, iOS, Android y Linux), accede directamente a http://sd81booster.local

INFO	En Windows, si no tienes instalado nada de Apple, es posible que el navegador no reconozca direcciones “.local”. No es necesario para nada más — puedes usar LOAD THEN PRINT “*IP” o la IP directa sin problema.

Alternativa: panel del router:

También puedes consultar la dirección IP asignada en la lista de dispositivos conectados del panel de administración de tu router (búscalo como “esp32” o similar).

Funciones disponibles:

- Listar y navegar el contenido de la SD por carpetas.
- Subir archivos, incluidos varios a la vez o una carpeta completa (en navegadores de escritorio).
- Descargar cualquier archivo al dispositivo desde el que navegas.
- Borrar archivos o carpetas.
- Crear carpetas nuevas.

Sincronización de hora vía NTP:

El módulo WiFi puede mantener en hora el reloj del interface sincronizándolo automáticamente con un servidor de hora (NTP) nada más conectarse a la red — útil si la pila del RTC se agota o simplemente para no tener que ajustar la fecha a mano.

Para activarla, crea un archivo de texto llamado NTP.CFG en la carpeta SYS de la tarjeta microSD:

```
SERVER=pool.ntp.org
MODE=SERVER
UTCOFFSET=1
DST=1
```

SERVER es la dirección del servidor NTP. MODE=SERVER activa la sincronización (usa MODE=LOCAL, o no crees el archivo, para dejarla desactivada). UTCOFFSET es el desfase horario respecto a UTC, en horas (puede ser negativo). DST suma una hora extra en horario de verano — es un interruptor manual que cambias tú mismo dos veces al año, igual que con cualquier otro reloj.

También puedes editar esta configuración, y forzar una sincronización inmediata, desde la propia página web del módulo (enlace “NTP time settings” en la portada del servidor de archivos).

Desde BASIC, sin necesidad de usar la web:

- LOAD *NTP — fuerza una sincronización inmediata.
- LOAD *NTP=“servidor” — cambia el servidor NTP.
- LOAD *NTP +n / LOAD *NTP -n — ajusta el desfase horario UTC, en horas.
- LOAD *SUMMER / LOAD *SUMMER STOP — activa o desactiva el horario de verano.

INFO	El firmware revisado intenta sincronizar al arrancar y, si falla, reintenta cada 60 segundos mientras hay conexión, hasta lograrlo una vez. Después cesan los reintentos automáticos. Si no puede sincronizar, conserva la hora del RTC; también puedes solicitar una sincronización manual.

Actualización del módulo: sigue la sección 17, que distingue firmware STM32, FPGA, ROM y WiFi, indica las ubicaciones exactas y explica el segundo arranque en módulos antiguos. La versión propia del WiFi en el paquete 1.7 es 1.4. La programación inicial y la recuperación por USB son procedimientos separados.

9.8 Terminal telnet — TELNET

TELNET es una terminal para conectarse a BBS y servidores telnet desde el ZX81, a través del módulo WiFi. Muestra 80 columnas por 24 filas en modo Superfast, con la fuente CP437 de 256 caracteres y colores ANSI, y entiende las secuencias de control que usan las BBS (ANSI y ADM-3A). Está en Software.zip, en la carpeta SD81/TOOLS/TELNET: copia la carpeta a la SD y carga el programa:

```
LOAD “TELNET.P”
```

La terminal funciona como un módem Hayes: los comandos AT se teclean en la propia pantalla.

Comando	Efecto
ATDT host:puerto	Conecta con el servidor (por ejemplo ATDT bbs.ejemplo.org:23). Responde CONNECT o NO CARRIER.
ATH	Cuelga.
+++	Vuelve a modo comando sin colgar (con un segundo de silencio antes y después).
ATO	Vuelve a la conexión.

Teclas especiales:

Tecla	Acción
ENTER + 0	Salir: cuelga si hay conexión y vuelve al BASIC.
ENTER + 9	Eco local sí/no, para servidores que no devuelven lo que se teclea.
ENTER + 8	Colores: blanco sobre negro (al arrancar), verde sobre negro o amarillo sobre azul. Los colores que pone la BBS se respetan.
SHIFT + 1	ESC.
SHIFT + ENTER y una tecla	CTRL + tecla.
ENTER + tecla	Símbolos: los serigrafiados y además @ \ | ~ ` { } [ ] _ ! # % &.
SHIFT + 0 / SHIFT + 9	DEL / retroceso.
SHIFT + 5/6/7/8	Flechas izquierda, abajo, arriba y derecha: envían las secuencias ANSI ESC [D/B/A/C, que es lo que entienden las BBS.

El teclado es el mismo que el de CP/M (ver la tabla de teclado de CP/M en el Apéndice I), salvo las flechas. Con el eco local activado, los códigos de control se muestran como ^X (^[es ESC y ^C es CTRL + C). La terminal contesta a la negociación telnet del servidor aceptando solo ECHO y SUPPRESS-GO-AHEAD, así que las BBS no muestran caracteres extraños al conectar. Usa la memoria de los bloques 4 y 5 ($8000-$97FF) para la pantalla, los colores y la fuente.

9.9 Juegos por turnos en red

Con el módulo WiFi (9.7), dos SD81 Booster pueden jugar por turnos por internet, cada uno en su casa y sin ningún servidor propio: los dos interfaces se conectan como clientes a un buzón público (un servidor MQTT) y se mandan en él las jugadas. Hay dos juegos, el 4 en raya (CONNECT4) y el ajedrez (CHESS), y un chat de prueba (NETCHAT). Están en Software.zip, en las carpetas SD81/TOOLS/CONNECT4, CHESS y NETCHAT. Copia la carpeta del juego a la SD y carga su programa .P, que es un cargador pequeño: lee del mismo directorio el .BIN con el juego, así que los dos archivos tienen que estar juntos.

Hace falta el módulo WiFi con la red configurada (/SYS/WIFI.CFG, ver 9.7), la FPGA con los modos Superfast. No hace falta el monitor del depurador.

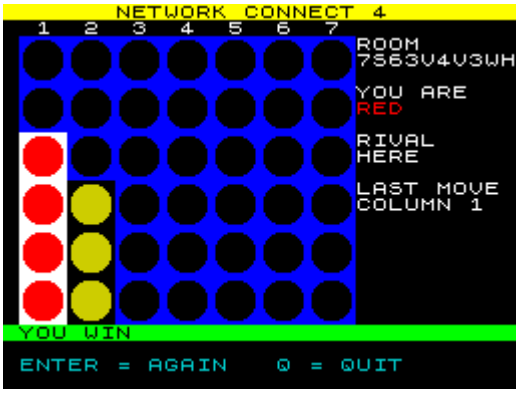
Empezar una partida:

- Partida nueva (1): el programa crea una sala y enseña su código, de 10 caracteres. Díselo al rival por cualquier medio. Quien crea la sala es el jugador 1: en el 4 en raya empieza, y en el ajedrez juega con las blancas.
- Unirse (2): teclea el código que te han dado (SHIFT+0 borra) y ENTER. Eres el jugador 2.
- Una sala admite dos jugadores. Un tercero que teclee el código ve THE ROOM IS FULL y no puede entrar; un código que nadie ha creado da THERE IS NO SUCH ROOM.
- Si sales del juego (0 o Q) o se corta la red, puedes volver con el mismo código desde «unirse»: recuperas tu sitio y la partida sigue donde estaba. Si se cayeron los dos, la sala queda vacía y el programa te deja elegir con qué jugador vuelves. Si alguien se cae sin despedirse, su sitio parece ocupado durante uno o dos minutos, hasta que el servidor se da cuenta.
- No hace falta que los dos estén conectados a la vez: si el rival no está, su última jugada espera en el buzón y la recibes al volver.

- Al acabar la partida, ENTER propone otra al rival.
- Guardar y seguir otro día: cada vez que cambia la partida, el programa la guarda en un fichero de la carpeta del juego, en la SD (CONNECT4.SAV o CHESS.SAV). Si apagas el ZX81, al volver el menú ofrece «3 = CONTINUE SAVED GAME»: el programa vuelve a entrar en la misma sala y con el mismo color, y la partida sigue donde estaba (en el ajedrez, con los relojes como estaban: el tiempo que pasas fuera no cuenta). No hace falta que los dos jugadores estén conectados a la vez: cada uno solo necesita su partida guardada y que el servidor conserve la última jugada del otro hasta que la recoja. Si el servidor la hubiera olvidado, quien vuelva primero vuelve a mandar su última jugada. Empezar otra partida, o unirse a otra, sustituye la guardada.

4 en raya:

El tablero se dibuja en alta resolución: 7 columnas de 6 filas en azul, con las fichas del jugador 1 en rojo y las del jugador 2 en amarillo. Los números 1 a 7 sobre las columnas dicen qué tecla suelta la ficha en cada una. La última ficha que cae queda resaltada con el fondo blanco y, al ganar, los cuatro en línea. En el panel de la derecha se ve la sala, tu color, si el rival está y la última jugada. R reenvía la última jugada, por si falló la red. Q sale.



4 en raya: ganan los cuatro rojos de la primera columna (captura de la pantalla real, a su tamaño).

Ajedrez:

La partida se dibuja en alta resolución (Superfast HiRes Spectrum): tablero de 8×8 casillas con sus coordenadas, la última jugada resaltada en magenta, el rey en jaque resaltado, y un panel con la sala, tu color y el estado. Cada uno ve el tablero desde su lado: las negras lo ven girado. Una jugada se teclea con la casilla de origen y la de destino, por ejemplo E2 E4, y ENTER; una tecla que no cuadra en esa posición no entra, y SHIFT+0 borra la última. Para coronar, una quinta letra elige la pieza (Q dama, R torre, B alfil, N caballo); sin ella, dama. 9 reenvía la última jugada y 0 sale. Se aplican todas las reglas salvo la triple repetición: enroque, captura al paso, coronación, jaque, mate, ahogado, material insuficiente y la regla de las 50 jugadas.



Ajedrez: la posición inicial (captura de la pantalla real, a su tamaño).

Relojes del ajedrez:

Al crear la sala eliges el tiempo de cada jugador: sin reloj, 5, 10, 15 o 30 minutos, sin incremento. El otro jugador lo hereda al teclear el código (va en su primera letra: A sin reloj, B 5 minutos, C 10, D 15, E 30). Los dos relojes salen en el panel; el del jugador que mueve, en amarillo y, con 30 segundos o menos, en rojo. Quien se queda sin tiempo pierde (TIME OUT).

Chat de prueba (NETCHAT):

Un chat mínimo para probar la red entre dos interfaces: uno elige sala nueva, el otro se une con el código, S envía una línea y Q sale.

Nota: las jugadas viajan sin cifrar por un servidor MQTT público, firmadas con una clave que sale del código de la sala. Eso impide que un extraño meta jugadas, pero no las oculta: no mandes nada sensible. Para programar tu propio juego, ver el Apéndice M.

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 “T120O45C5E5G9C”
```

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

INFO	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 “T1805C5E5G”
```

Octava

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

```
LOAD *PLAY “O35CO45CO55C”
```

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 “(5C5E5G)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).

INFO	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.

INFO	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
```

INFO	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.

INFO	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>

Cargar un programa PEG desde la SD (LOAD *PEB):

Además de la carga inline con cadena hexadecimal, es posible cargar un programa PEG compilado directamente desde un archivo en la SD:

> LOAD *PEB <dirección>,“<nombre>”

Donde <dirección> es la posición de inicio en la memoria PEG (0–255) y <nombre> es el nombre del archivo en la SD. El archivo debe tener extensión .PEB (PEG binary); si no se especifica extensión, el interface la añade automáticamente.

Ejemplo:

> LOAD *PEB 0,“EFECT”
>
> LOAD *PEG THEN RUN 0,0

Este ejemplo carga el archivo EFECT.PEB en la memoria PEG a partir de la posición 0 y arranca el hilo 0 desde esa misma dirección.

INFO	El archivo .PEB es el resultado de ensamblar un fuente .PEG con el ensamblador peg.py, un script Python incluido en la carpeta EXAMPLES/PEG/ del repositorio del proyecto. Para compilar: python peg.py efect.peg

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”

INFO	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.

INFO	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.

11. 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.

11.1 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.

AVISO	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.

11.2 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

INFO	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.

11.3 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

AVISO	El modo MC45 se implementa forzando a cero el pin /HALT 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.

Extensión de MC45 a los bloques 6 y 7

Por defecto, MC45 solo cubre los bloques 4 y 5 (direcciones 32768-49151): en el vídeo nativo del ZX81 la pantalla se genera saltando a D_FILE+$8000 y ejecutando NOPs hasta encontrar un HALT real (el propio D_FILE, en su espejo alto), y esa dirección nunca cae en los bloques 4/5 en uso normal, por lo que ejecutar código de verdad ahí es seguro.

En los modos Superfast (texto, HiRes nativo o HiRes Spectrum) el vídeo se genera íntegramente por hardware, sin que la CPU tenga que “ejecutar” D_FILE en ningún momento — así que, con código propio, también podría ser seguro ejecutar en los bloques 6 y 7 (49152-65535). El SD81 Booster permite activar esta extensión con un registro independiente:

```
POKE 2062, 170 : REM extiende MC45 tambien a los bloques 6 y 7

POKE 2062, 85 : REM vuelve a la restriccion habitual (solo bloques 4 y 5)
```

Para que tenga efecto hace falta también LOAD *MC45 activo; este registro solo añade los bloques 6 y 7 a la zona donde MC45 ya actúa, no lo sustituye.

AVISO	En el mapeo de bloques por defecto (sin RAM48 activada), los bloques 6 y 7 son un espejo físico de los bloques 2 y 3 — es decir, de la memoria donde viven el sistema de variables, el programa BASIC, la pila y el D_FILE real. Activar esta extensión mientras los bloques 6/7 siguen siendo ese espejo cuelga el sistema en cuanto se activa un modo Superfast, incluso con FAST. Actívala solo después de remapear los bloques 6 y/o 7 (con MAP) a páginas de SRAM libres, dedicadas exclusivamente a tu código propio.

11.4 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:

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

INFO	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.

11.5 Caracteres definibles por el usuario (128C / 64C / 256C)

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
```

Activar el modo de 256 caracteres:

```
LOAD *256C
```

Este modo solo funciona en Superfast texto. Amplía el conjunto a 256 caracteres, todos redefinibles, en la zona de memoria entre las direcciones 14336 y 16383 (3800h–3FFFh) — el doble de espacio que el modo de 128, alineado a 2K en vez de a 1K. Como con 128C, el bloque completo se precarga con el juego de caracteres de la ROM al arrancar, así que activar el modo sin haber redefinido nada no cambia lo que se ve en pantalla. Activar “128C” o “64C” desactiva “256C”.

Con el modo de 256 caracteres activo, la tabla de color de Chroma modo 0 (carácter por carácter) también pasa de 1 KB a 2 KB (14336–14463, $C000-$C7FF), para que cada uno de los 256 códigos tenga su propia entrada de color. El modo 1 (fichero de atributos) no se ve afectado, indexa por posición de pantalla y ya funciona igual con 128 o con 256 caracteres.

INFO	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
```

INFO	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
```

INFO	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.
- 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 50: espera indefinidamente a que se pulse cualquier tecla.

12. 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.

12.1 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$
```

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

12.2 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.

INFO	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.

12.3 Ejecución de código máquina

Ejecutar una rutina en código máquina (LOAD USR):

```
LOAD USR <dirección>
```

Equivale a RAND USR pero con tres ventajas importantes:

- 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 las pilas limpias.

- El resto de la línea no se analiza sintácticamente, lo que permite que la rutina realice su propio análisis de parámetros mediante RST 18h y RST 20h.

Al entrar en la rutina, el registro BC contiene la dirección llamada.

INFO	Limitación con literales numéricos: si la rutina usa la rutina SCANNING de la ROM para analizar parámetros, las expresiones con literales numéricos directos (ej. USR 40000) no funcionarán correctamente. Usa VAL “número” o CODE “carácter” como alternativa: por ejemplo, USR VAL “40000” en lugar de USR 40000.

12.4 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>

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

12.5 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 (RAMTOP queda en esa dirección más uno: LOAD THEN CLEAR 32767 deja el valor estándar de un ZX81 con 16K, RAMTOP = 32768). 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.

12.6 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.

INFO	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
```

INFO	[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.

13.6 Modos de pantalla ancha — Comandos COL80 / COL70 / COL32

El SD81 Booster permite ampliar el ancho de la pantalla de texto en modo Superfast, de las 32 columnas estándar del ZX81 a 70 u 80 columnas. A diferencia del POKE 2045 directo, estos comandos también adaptan las rutinas de la propia ROM (PRINT, PRINT AT, CLS, SCROLL, el editor de listados) al nuevo ancho, así que el BASIC estándar funciona igual que siempre, solo que con más columnas por línea.

Activar 80 columnas (caracteres de 7 píxeles):

```
LOAD *COL80
```

Activar 70 columnas (caracteres de 8 píxeles):

```
LOAD *COL70
```

Volver a 32 columnas, vídeo nativo:

```
LOAD *COL32
```

Los modos de 70 y 80 columnas ocupan exactamente el mismo ancho físico de pantalla (80 caracteres de 7 píxeles equivalen a 70 de 8), así que cambiar entre ellos no mueve ni redimensiona la imagen: solo cambia cuántas columnas de texto caben. Los tres comandos activan el modo FAST automáticamente (en modo Superfast el vídeo lo genera el interface, no la CPU, así que SLOW no aporta nada); COL32 hace lo contrario y vuelve a SLOW.

Al cambiar de modo, la pantalla se limpia (equivalente a CLS): el contenido de la pantalla del modo anterior no se conserva, porque el ancho de cada fila del área de pantalla cambia.

TAB y PLOT:

TAB y PLOT funcionan en los tres modos, igual que el resto de comandos de impresión (PRINT, PRINT AT, CLS, SCROLL). En 70 y 80 columnas, un TAB más allá del ancho de la línea da el error B, en vez de dar la vuelta como en 32 columnas. PLOT y UNPLOT admiten X de 0 a 139 en 70 columnas y de 0 a 159 en 80.

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.
P	No se pudo cargar: un archivo .B81 con errores (el detalle, con LOAD THEN PRINT “*B81ERR”), o un snapshot .Z81 sin el depurador (falta /SYS/DEBUG.BIN), con el depurador parado o con un snapshot en curso.

INFO	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
2013h	OutWaitEq — envía A al puerto de datos y espera reloj igual
2016h	WaitDiffBrk — como WaitClkDiff pero permite BREAK (no retorna si se pulsa)

Dirección	Contenido / Rutina
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).
AFh	Puerto de control MCU (escritura=reset MCU; bit 7 en lectura=bit de reloj, los bits 6..1 son el contador de VSYNC desde la ultima lectura y el bit 0 en lectura indica el estado instantáneo de VSYNC)

Sincronización con VSYNC

El bit 0 del puerto AFh 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.

Los bits 1 a 6 son un contador del numero de pulsos de VSYNC que se han producido desde la ultima lectura del puerto.

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

AVISO	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)

En modo de paginación simple (hasta 256 KB), los 8 bits escritos en el puerto E7h se interpretan así:

Bits	Función
D2, D1, D0	Número de bloque (0–7)
D7, D6, D5, D4, D3	Número de página (0–31)

Para acceder a las 64 páginas del modo completo (512 KB), se usa la instrucción OUT (C),r con el número de página en B y el número de bloque en otro registro. Si A contiene el número de bloque (0–7) y B el de página (0–63):

```
ld c,0E7h ; puerto del mapeador

out (c),a ; selecciona página B en bloque A
```

El cambio entre modo simple y completo se realiza mediante un comando al MCU (comandos FULLPAGING y HALFPAGING, ver sección 15.5).

15.4 Consola de depuración (puerto USB-C)

El puerto USB-C del interface también funciona como puerto serie de depuración. Al conectarlo al ordenador, el sistema operativo detecta un puerto serie virtual asociado al chip CH340G. En algunos sistemas puede ser necesario instalar los drivers del CH340.

Parámetros de conexión:

Parámetro	Valor
Velocidad	115200 baudios
Bits de datos	8
Paridad	Ninguna
Bits de parada	1
Control de flujo	Ninguno

Con cualquier programa de terminal serie (PuTTY en Windows, minicom en Linux, CoolTerm en macOS, o el propio Monitor Serie del IDE de Arduino) es posible monitorizar en tiempo real los mensajes del MCU, incluyendo: progreso del arranque, errores de acceso a la SD, progreso de actualizaciones de firmware, y mensajes de depuración del sistema de archivos, VGM, PEG y síntesis de voz.

La consola también acepta las órdenes del depurador por hardware (ver Apéndice L); h muestra su ayuda.

INFO	El firmware de producción emite mensajes básicos de estado por el puerto serie. Recompilando el firmware con la macro DEBUG activa se obtiene una salida mucho más detallada, útil para diagnóstico avanzado y desarrollo.

15.5 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

Código	Significado
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.
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.

Cód.	Nombre	Parámetros	Respuesta	Descripción
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 >.
53	F_OPEN	Handle(0..3)+nombre en ASCII	1B: status	Abre un fichero grande (tamaño 32 bits) en un handle de archivo especificado (0..3). El nombre es una cadena Pascal en ASCII.
58	F_OPEN_ZX81	Handle(0..3)+nombre en ZX81	1B: status	Abre un fichero grande (tamaño 32 bits) en un handle de archivo especificado (0..3). El nombre es una cadena Pascal en ZX81.

Cód.	Nombre	Parámetros	Respuesta	Descripción
54	F_SEEK	Handle(0..3)+Offset (4 bytes Little endian)	1B: status	Desplaza el puntero de lectura/escritura a la posición indicada en offset
55	F_READ	Handle(0..3)+Count(2B Little Endian)	count bytes + 1B:status	Lee count bytes. Siempre envia count bytes, si se termina el archivo rellena con ceros
56	F_WRITE	Handle(0..3)+Count(2B Little Endian)+info to write (count bytes)	1B:status	escribe count bytes.
57	F_CLOSE	Handle(0..3)	1B:status	Cierra el fichero
59	F_STAT	Handle(0..3)	4B:tamaño+2B:fecha+2B:hora+1B:status	Devuelve el tamaño (32 bits) y la fecha/hora de creación (formato FAT) de un fichero ya abierto con F_OPEN.
75	LOAD_Z81	String: nombre	Status (1B) + IM (1B)	LOAD *Z81 (ver 7.7), por el monitor del depurador (Apéndice L). El MCU lee el fichero entero antes de contestar: Status 0 = la ROM lanza la trampa y el MCU lo carga todo al parar; 1 no existe el fichero, 2 sin [MEMORY], 3 fichero mal hecho, 10 no se puede ahora (sin /SYS/DEBUG.BIN, depurador parado o snapshot en curso). IM es el modo de interrupción del snapshot. Hasta la 1.6 el comando 70 era un cargador propio, sin monitor; se quitó.

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.

Cód.	Nombre	Parámetros	Respuesta	Descripción
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.
65	SEL_256CHARS	—	—	Activa el modo de 256 caracteres definibles (solo Superfast texto). Equivale a LOAD *256C.
68	ROMLOCK_ON	—	—	Bloquea los puertos de configuración del bloque 0 (2041-2062, 2090-2098, sprites): las escrituras se ignoran. Equivale a LOAD *ROMLOCK.
69	ROMLOCK_OFF	—	—	Vuelve a activar los puertos de configuración del bloque 0. Equivale a LOAD *ROMLOCK 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.

Cód.	Nombre	Parámetros	Respuesta	Descripción
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).

Comandos de red (NET)

Cód.	Nombre	Parámetros	Respuesta	Descripción
66	**NET_READ**	**1B: max (0-255)**	**count (1B) + data[count] + avail (1B) + status (1B)**	**Lee hasta max bytes recibidos por el socket. count puede ser 0 aunque haya conexion — los datos llegan a rafagas. avail = bytes pendientes tras esta lectura, saturado a 255. Con max=0 se convierte en un ¿hay algo?» barato que solo refresca status, sin transferir datos.**
67	**NET_WRITE**	**1B: count + data[count]**	**accepted (1B) + status (1B)**	**Envia count bytes al socket. accepted puede ser menor que count si el buffer de salida esta lleno — el Z80 tiene que reenviar el resto.**

El campo status es el mismo en los dos comandos:

Valor	Significado
0	sin conexión

Valor	Significado
1	conectando
2	conectado
3	error / la conexión se cayó

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.
LED STAT parpadea en Azul/Rojo durante el arranque	Error inicializando la tarjeta SD. Comprueba que está insertada correctamente y formateada en FAT32.
LED STAT parpadea en Naranja/Rojo durante el arranque	Error escribiendo la ROM en RAM. Comprueba que la carpeta SYS contiene los archivos de ROM necesarios.
LED STAT parpadea en Naranja durante el arranque	El MCU está esperando respuesta de la FPGA. Si el parpadeo no termina, puede indicar un problema de hardware.
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. Si el error persiste, reformatea la tarjeta en FAT32. Si sigue fallando, puede ser una tarjeta SD defectuosa.
La SD no se reconoce	Verifica que está formateada en FAT32 (no exFAT ni NTFS).
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 “QAOP” (arriba/abajo/izquierda/derecha/fuego) 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 televisor o monitor está conectado al conector SCART del interface y que el canal de audio del SCART no está silenciado.
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.

16.6 Errores de actualización de firmware

Síntoma	Solución
LED STAT parpadea en Azul/Rojo al arrancar con firmware.bin en la SD	Error inicializando la tarjeta SD. Extrae la SD, comprueba el formato FAT32 y vuelve a intentarlo.
LED STAT parpadea en Blanco/Rojo tras intentar actualizar	Error durante la actualización. El firmware.bin permanece en la SD. Comprueba que el archivo no está corrupto y vuelve a encender el ZX81 para reintentar.
El LED STAT se queda en Amarillo fijo indefinidamente	El significado depende de la fase: puede ser escritura del MCU o un fallo WiFi. Consulta la sección 17.6 y no interrumpas una escritura activa.
Tras la actualización el interface no responde	Comprueba con LOAD *VER que la versión es correcta. Si el interface no arranca, sigue el procedimiento de recuperación de emergencia via USB descrito en la sección 17.

16.7 Uso de la consola de depuración como herramienta de diagnóstico

Cuando el LED STAT muestra un error pero no está claro cuál es la causa, la consola de depuración por USB-C puede proporcionar información adicional muy valiosa.

Cómo conectarse:

1. Conecta un cable USB-C entre el interface y el ordenador.

2. Abre un programa de terminal serie (PuTTY, Tera Term, minicom, el Monitor Serie del IDE de Arduino...) y conéctate al puerto COM/serie del CH340 con los parámetros: 115200 baudios, 8N1, sin control de flujo.

3. Enciende el ZX81 con el interface conectado.

4. Observa los mensajes que aparecen en la terminal durante el arranque y la operación normal.

El firmware de producción emite mensajes básicos de estado que permiten identificar en qué fase falla el arranque, si la tarjeta SD se reconoce correctamente, el resultado de las actualizaciones de firmware y otros eventos relevantes.

INFO	Para desarrolladores: recompilando el firmware con la macro DEBUG activa se obtiene una salida mucho más detallada, incluyendo el progreso de las operaciones de flash, el estado de los registros del AY y los detalles de cada comando recibido del Z80.

16.8 Programa de test del hardware — SD81TEST

SD81TEST comprueba el interface de forma sistemática: memoria y mapper, ejecución en MC45, puertos mapeados en memoria, MCU, RTC, SD, vídeo (Superfast, Chroma81, sprites, 128/256 caracteres, 70/80 columnas, doble buffer, WRX), sonido (chips AY, beeper, VGM, PEG y voz), teclado y joystick, y red a través del módulo WiFi. Sirve para comprobar un interface recién montado o para diagnosticar un fallo. Está en Software.zip, en la carpeta SD81/TOOLS/SD81TEST: copia la carpeta completa a la SD (los cinco ficheros .BIN, SD81TEST.VGM y SD81TEST.P) y carga el programa:

> LOAD “SD81TEST.P”

El menú agrupa las pruebas por categorías. Las automáticas terminan con RESULT: OK o RESULT: FAIL; las interactivas dibujan o hacen sonar algo y preguntan si se ve o se oye bien (Y/N).

Pasada automática e informe:

En el menú 9 (INFO AND SUMMARY), la opción 2 ejecuta seguidas las 20 pruebas automáticas y muestra un resumen con el resultado de cada una. El resumen se guarda también en el fichero SD81TEST.TXT, en la misma carpeta, junto con la fecha y las versiones del MCU, la ROM y la FPGA, para comparar máquinas o adjuntarlo a una consulta.

La prueba de memoria (menú 1, opción 1) es destructiva: borra el contenido de todas las páginas de RAM que no son de sistema (discos RAM de CP/M, pantallas guardadas...). Por eso no forma parte de la pasada automática. Las pruebas interactivas necesitan el modo SLOW (el de arranque), y las de red, el módulo WiFi configurado.

17. Actualización por microSD: firmware, FPGA, ROM y WiFi

17.1 Qué se actualiza y dónde va cada archivo

La actualización normal se hace mediante la microSD FAT32, sin abrir la carcasa ni conectar programadores. Hay cuatro componentes distintos: MCU STM32, FPGA, ROM Z80 y módulo WiFi ESP32-C3 (si está instalado). Utiliza los archivos de una misma versión del paquete; los números de versión de los componentes no tienen por qué coincidir.

Componente	Archivo y ubicación en la SD	Versión indicada en las notas del paquete 1.7
MCU STM32	`/firmware.bin`	2.7

Componente	Archivo y ubicación en la SD	Versión indicada en las notas del paquete 1.7
FPGA	`/sd81.mcs`	1.6
ROM Z80	`/SYS/SDBOOST.ROM`	1.8
WiFi ESP32-C3, opcional	`/ESP32FW.BIN`	1.4
Monitor de depuración	`/SYS/DEBUG.BIN`	Usar el del mismo paquete

Descargas: versiones publicadas del proyecto. Las notas y el script de empaquetado identifican `Firmware.zip`, `SD Content.zip` (el nombre puede aparecer con un punto en lugar del espacio), `Software.zip` y `Tools.zip`. Descarga los recursos adjuntos de la versión elegida; el ZIP automático «Source code» contiene fuentes y no sustituye a los binarios preparados. Esta edición documenta el código consultado, no certifica que cada cambio de la rama principal se haya publicado como binario.

17.2 Preparación y copia de seguridad

1. Apaga el ZX81 antes de sacar o insertar la microSD.
2. Haz una copia completa de la tarjeta en el ordenador, incluidos programas, partidas y configuraciones.
3. Conserva especialmente `/SYS/WIFI.CFG`, `/SYS/NTP.CFG` y, si lo utilizas, `/SYS/NET.CFG`. El paquete excluye WIFI.CFG, pero puede incluir NTP.CFG: comprueba que no pierdes tus ajustes al copiar.
4. Descomprime los paquetes. Copia su contenido a las ubicaciones de la tabla; no dejes los ZIP ni una carpeta contenedora adicional en lugar de los archivos.
5. Mantén la carpeta SYS en la raíz. Actualiza SDBOOST.ROM y DEBUG.BIN con los del mismo paquete que el MCU. Integra también los demás archivos de sistema suministrados, conservando tu configuración.
6. Expulsa la tarjeta de forma segura del ordenador e insértala con el ZX81 apagado.

No copies bootloader.bin como firmware.bin. El bootloader es para recuperación avanzada; no forma parte de la actualización normal por SD. No uses un binario de otro dispositivo ni un archivo ESP32 combinado de programación inicial como sustituto de ESP32FW.BIN.

17.3 Instalar la actualización completa

1. Enciende el ZX81 con los archivos preparados en la tarjeta.
2. El bootloader detecta firmware.bin, programa el MCU y verifica la escritura. Solo tras completarla correctamente elimina ese archivo y arranca la aplicación.
3. El firmware detecta sd81.mcs, programa y verifica la memoria flash de configuración de la FPGA. El archivo se elimina tras el éxito.
4. La ROM se carga desde `/SYS/SDBOOST.ROM` al arrancar. Este archivo **debe permanecer en la SD**: no es un instalador que se borre después.
5. Si hay módulo WiFi programado, este busca ESP32FW.BIN, aplica su actualización y se reinicia. En caso de éxito elimina ese archivo.
6. Espera a que termine el arranque y cese la actividad de actualización. No cortes la alimentación ni extraigas la SD mientras se está programando.

El WiFi 1.4 espera hasta 120 segundos a que responda el STM32. Versiones anteriores solo esperan unos 15 segundos: si actualizas MCU y FPGA en el mismo arranque, el WiFi puede quedar pendiente. **Una vez terminadas las otras actualizaciones**, vuelve a apagar y encender para que procese ESP32FW.BIN. No confundas el paquete general 1.7 con la versión propia del módulo WiFi 1.4.

17.4 Actualizar únicamente el WiFi

Con el resto del sistema compatible y actualizado, copia **ESP32FW.BIN en la raíz de la microSD** y reinicia el conjunto. Puedes copiarlo desde el ordenador o subirlo por el servidor web del módulo; si lo subes por WiFi, espera a que termine la transferencia antes de apagar. Actualizar desde la SD no requiere acceso a Internet: el ESP32 recibe el archivo a través del STM32.

El nombre antiguo era ESP32_FW.BIN. El firmware STM32 revisado traduce la petición antigua al nombre nuevo cuando corresponde, por lo que un módulo antiguo puede encontrar ESP32FW.BIN. Utiliza el nombre que suministra el paquete y actualiza primero el STM32 si vienes de una versión antigua.

Durante la lectura de actualización del WiFi, STAT parpadea en rosa. Un error puede dejarlo amarillo. **El color por sí solo no confirma que todo el paquete esté instalado:** comprueba las versiones y los archivos de diagnóstico.

Si la imagen es inválida o no cabe, el código revisado elimina el archivo rechazado y escribe `/ESP32FWFAILED.TXT` con el motivo. Si falla la comunicación durante la transferencia, conserva el archivo para reintentarlo al siguiente arranque. Copia otra vez una imagen válida del paquete correcto; la desaparición del BIN sin comprobar la versión no basta como prueba de éxito.

17.5 Comprobar el resultado

- Ejecuta `LOAD *VER` para consultar la versión del firmware y `LOAD *FPGA` para consultar la FPGA.
- Consulta la versión WiFi al pie de su servidor web o en la información de `/MAN/IP.TXT`. La dirección se consulta con `LOAD THEN PRINT "*IP"`.
- Comprueba que SDBOOST.ROM y DEBUG.BIN siguen en SYS. En el paquete 1.7, `LOAD *Z81` depende del monitor DEBUG.BIN; sin él, el comando devuelve el informe P.
- Prueba un programa, vídeo y sonido. Para un diagnóstico más amplio, usa SD81TEST (sección 16.8); su informe registra versiones. Su prueba de memoria borra RAM y no debe ejecutarse con trabajo sin guardar.
- Si el WiFi no cambia de versión después de actualizar el resto, realiza el segundo arranque indicado arriba y revisa ESP32FWFAILED.TXT si aparece.

17.6 Errores y recuperación

Síntoma	Comprobación
MCU: alternancia azul/rojo	Fallo al inicializar la SD en el bootloader. Revisar tarjeta FAT32 y contacto.
MCU: alternancia blanco/rojo	Fallo al grabar o verificar. El archivo permanece para reintentar con una copia válida.
Amarillo durante programación	Puede corresponder a escritura del MCU o a error WiFi; distinguir la fase y consultar diagnóstico, sin cortar una escritura activa.
WiFi: BIN eliminado pero versión sin cambiar	Revisar ESP32FWFAILED.TXT; puede haberse rechazado la imagen.
Faltan comandos o no arranca la expansión	Revisar SYS, ROM y coherencia del paquete antes de intentar recuperación por USB.

La programación inicial de un MCU o ESP32 sin firmware y la recuperación de un dispositivo que no puede leer la SD son tareas distintas. El repositorio describe recuperación STM32 con JP7 y STM32CubeProgrammer, y programación ESP32-C3 con Arduino. El script de empaquetado sitúa las instrucciones y herramientas de recuperación en **Tools.zip**. Consulta las instrucciones de la versión concreta o el servicio técnico antes de abrir la unidad; esta guía de actualización normal no requiere hacerlo.

RTC: una actualización normal no requiere inicializar el reloj. Reserva ese procedimiento para programación inicial o corrupción de su configuración; no pulses QuickSilva al encender como parte de una actualización habitual.

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.
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).
FPGA	Circuito lógico programable (Xilinx Spartan-6 XC6SLX9) 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.
1.1	07/2026	Comprobación e instalación de actualizaciones desde la página web del módulo WiFi; registro de diagnóstico del módulo en la SD.
1.2	09/2026	Sprites por hardware, scroll fino en Superfast, 256 caracteres definibles; correcciones del color Chroma81 y del joystick.
1.3	09/2026	Puente de red tipo módem (BBS/telnet, comandos AT), modos de 70 y 80 columnas, MC45 en los bloques 6 y 7, pantalla y atributos alternativos, y CP/M 3.
1.4	09/2026	Snapshots de EightyOne (LOAD *Z81), LOAD *ROMLOCK y corrección de la basura en el borde con juegos WRX.
1.5	09/2026	Terminal TELNET y programa de test SD81TEST. FPGA: captura síncrona de escrituras y lectura de FRAMES en Superfast. MCU: DEL con rutas absolutas y lectura de los registros 14 y 15 del AY.
1.6	10/2026	Depurador por hardware (Apéndice L): pantalla en el ZX81, símbolos, traza e historial, consola y paneles en el navegador; snapshots completos con LOAD *Z81. Interrupciones simuladas en la FPGA. ROM: PLOT y TAB en los modos de 70 y 80 columnas (gracias a Kelly Abrantes Murta por avisar del fallo de PLOT). MCU: el sonido calla mientras se carga un snapshot; el servidor web del módulo WiFi va más fluido. Nuevo Apéndice K con la guía del emulador EightyOne.
1.7	10/2026	Juegos por turnos en red (sección 9.9 y Apéndice M): 4 en raya y ajedrez con relojes, dibujados en alta resolución, que guardan la partida en la SD y se siguen otro día. Módulo WiFi: comandos AT de MQTT, SHA-256, HMAC y aleatorios. 64 sprites, hasta 12 por línea. Depurador: captura de pantalla y teclado del ZX81 en la web, sym !, y CP/M. LOAD *Z81 solo por el monitor del depurador (informe P; desaparece el comando 70 del MCU). El archivo de actualización del módulo WiFi pasa a ser ESP32FW.BIN y se puede actualizar todo de una vez. Los WAV de 32 bits suenan bien. Versiones: STM32 2.7, ROM 1.8, FPGA 1.6, WiFi 1.4.

INFO	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://github.com/wilco2009/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 recuperación via USB en caso de emergencia:

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/

Créditos del proyecto

- Diseño del hardware y firmware del MCU: Alejandro Valero (wilco2009)
- Código Z80 / ROM modificada: Pedro Gimeno (pgimeno)

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

Valor	Nombre	Duración a 60 bpm
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)
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)

Parámetro	Descripción
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
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

Instrucción	Codificación	Descripción
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

INFO	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).

Bit	Función
5	0 = modos de color disponibles. Siempre 0, lo que indica que el modo color está siempre activo.
4–1	No usado (reservado).

INFO	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.

Comando equivalente:

LOAD *COLOR : REM activar color, borde blanco (7) por defecto

LOAD *COLOR <color> : REM activar color, borde = <color> (0-15)

LOAD *COLOR <color>,<modo> : REM color = <color> (0-15), modo Chroma = <modo> (0-1)

LOAD *COLOR STOP : REM desactivar color

(equivalente al OUT 7FEFh anterior; activa el bit 5 y fija el modo de código de carácter automáticamente. **<color>** son los 4 bits bajos del puerto, brillo + GRB combinados en 0-15. **<modo>** es opcional (0 = tabla de color por código de carácter, 1 = fichero de atributos; 0 si se omite). El modo 1 solo está implementado por hardware en los submodos Superfast: en modo nativo (sin Superfast) no tiene efecto real.

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

Con los comandos equivalentes:

LOAD *SFAST : REM Superfast texto

LOAD *SFHR <direccion> : REM Superfast HiRes nativo, HFILE = <direccion>

LOAD *SFSP <direccion> : REM Superfast HiRes Spectrum, HFILE = <direccion>

LOAD *SFAST STOP : REM desactivar (equivalente a *SFHR STOP / *SFSP STOP)

(equivalentes a los POKE anteriores, por si se prefiere desde código máquina). **LOAD *SFHR** y **LOAD *SFSP** escriben HFILE (2043/2044) y activan el modo en un solo paso; no hace falta indicar HFILE para **LOAD *SFAST** porque el modo texto no usa RAM extendida.

Scroll horizontal fino (Superfast)

Los tres submodos Superfast admiten un desplazamiento horizontal fino de 0 a 7 píxeles, que adelanta la búsqueda de carácter/atributo/píxel esa cantidad de píxeles en cada fila:

POKE 2090, <offset> : REM offset de scroll fino, 0-7 pixeles

Comando equivalente:

LOAD *SCROLL <offset> : REM equivalente a POKE 2090

El desplazamiento se aplica por igual a los tres submodos Superfast (texto, HiRes nativo y HiRes Spectrum); no tiene efecto en modo nativo. En Superfast texto, el hueco de píxeles que queda a la derecha de cada fila lo rellena el byte de NEWLINE del DFILE (posición 32 de cada fila, sin uso en este modo): el software es responsable de escribir ahí el contenido que deba aparecer al hacer scroll. Esto ocurre con o sin color activado (la propia búsqueda de carácter no depende del Chroma); lo único que puede variar según el modo Chroma es el color con el que se muestra esa columna, y solo en modo 1 (fichero de atributos), que no calcula bien el color de esta columna extra. En los demás submodos (HiRes nativo y HiRes Spectrum) ese hueco no tiene un contenido definido todavía.

El desplazamiento puede activarse o desactivarse por fila, para dejar fijos marcadores o puntuaciones mientras el resto de la pantalla se desplaza. Cada fila de texto (0-23) tiene un bit en uno de tres registros de 8 bits:

POKE 2091, <mapa> : REM filas 0-7 (bit0=fila 0)

POKE 2092, <mapa> : REM filas 8-15

POKE 2093, <mapa> : REM filas 16-23

Bit a 1 = esa fila se desplaza con el offset de **LOAD *SCROLL**; bit a 0 = esa fila se queda fija. Por defecto (tras un reset) los tres registros valen 255 (todas las filas se desplazan), igual que antes de que existiera este registro. Comando equivalente:

LOAD *SCROWS <lo>,<mid>,<hi> : REM equivalente a POKE 2091/2092/2093

(equivalente a los tres POKE anteriores en un solo paso; cada argumento es un byte 0-255).

Pantalla alternativa (Superfast)

Los submodos de texto Superfast (32, 70 y 80 columnas) toman normalmente la dirección de pantalla de la variable de sistema D_FILE (16396/16397), igual que en modo nativo. El SD81 Booster permite sustituir esa dirección por otra, sin tocar D_FILE, mediante un registro alternativo:

> POKE 2096, <dir_baja> : REM byte bajo de la direccion alternativa
>
> POKE 2097, <dir_alta> : REM byte alto
>
> POKE 2098, 170 : REM activa el registro alternativo

> POKE 2098, 85 : REM lo desactiva, vuelve a D_FILE

Mientras el registro alternativo está desactivado (el valor por defecto, tanto tras un reset como después de POKE 2045,85), el comportamiento es exactamente el de siempre: nada cambia para el software existente. Al activarlo con POKE 2098,170, la pantalla Superfast se lee de la dirección indicada en 2096/2097 en lugar de D_FILE, sin que el BASIC ni las rutinas de la ROM (que siguen usando D_FILE para lo suyo) se enteren del cambio.

Esto permite, por ejemplo, mantener dos pantallas en memoria e ir alternando cuál se muestra sin necesidad de mover D_FILE de un lado a otro en cada fotograma.

Importante:

Volver a vídeo nativo (POKE 2045,85) desactiva automáticamente el registro alternativo, para que “modo estándar” signifique siempre el comportamiento de siempre. Cambiar entre los distintos submodos Superfast (POKE 2045,170/171/172/173/174) no lo desactiva: si estaba activo, sigue estándolo.

Tabla de atributos alternativa (Chroma modo 1, Superfast)

Normalmente, el modo 1 de Chroma (fichero de atributos) coloca la tabla de color en la misma dirección que la pantalla pero con el bit 15 forzado a 1, es decir, a $8000 de distancia de D_FILE (o de la dirección alternativa de la sección anterior, si está activa). El SD81 Booster permite independizar también esta tabla, con el mismo mecanismo:

> POKE 2059, <base_baja> : REM byte bajo de la tabla de atributos
>
> POKE 2060, <base_alta> : REM byte alto
>
> POKE 2061, 170 : REM activa la tabla alternativa
>
> POKE 2061, 85 : REM la desactiva, vuelve al comportamiento de siempre

Desactivada (el valor por defecto), el modo 1 sigue funcionando exactamente igual que hasta ahora. Activada con POKE 2061,170, la tabla de color deja de estar atada a la posición de la pantalla: puede vivir en cualquier dirección de 16 bits, sin el límite de módulo 32K que imponía forzar el bit 15. La tabla usa la misma aritmética de fila y columna que la pantalla (incluido el byte de relleno inicial), así que el software puede reutilizar el mismo cálculo de posición para las dos.

Este registro es independiente del de la sección anterior: se puede activar la dirección de pantalla alternativa sin activar ésta, o al revés, aunque lo habitual será usar ambos juntos para tener control total sobre dónde vive cada cosa.

Importante:

Igual que con la dirección de pantalla alternativa, volver a vídeo nativo (POKE 2045,85) desactiva también este registro automáticamente.

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.

AVISO	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

Comando equivalente (tinta del patrón + borde en modo Spectrum):

LOAD *BORDER <color> : REM tinta del patron (0-15) y borde en modo Spectrum (bits 2-0)

(equivalente a **POKE 2046,<color>** más el puerto ULA FBh anterior en un solo paso; no activa el patrón por sí solo, hace falta **POKE 2047,170**. El color de fondo del borde en modo nativo se controla con **LOAD *COLOR**, no con este comando.)

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 AFh, 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.

Doble buffer (present-blit)

Cómo genera la imagen el interface (sin doble buffer)

Toda la RAM del sistema (hasta 512 KB) vive en un único chip de memoria externo (la **SRAM**), organizado en páginas de 8 KB que se asignan a los 8 bloques del mapa de memoria del Z80 mediante el mapper (E7h). Ahora bien, el circuito de vídeo **no lee esa SRAM directamente** — si lo hiciera, competiría con la CPU por el mismo chip de memoria en cada ciclo. Para evitarlo, el interface mantiene dentro de la propia FPGA una **memoria interna de vídeo** (64 KB) que actúa como un **espejo** de los 8 bloques: cada vez que la CPU escribe un byte en la SRAM, ese mismo byte se copia automáticamente al espejo. El circuito de vídeo genera la imagen leyendo siempre este espejo interno, nunca la SRAM.

El espejo es fiel en todo momento — demasiado fiel, de hecho: si en un instante dado la CPU acaba de borrar un sprite, el espejo ya refleja el hueco vacío, y si el haz de vídeo pasa por esa zona justo entonces, dibuja ese hueco. Como CPU y vídeo comparten el mismo dato “en vivo”, el haz puede pillar estados intermedios del dibujado — de ahí el parpadeo y el tearing que el doble buffer viene a resolver.

Qué cambia el doble buffer

POKE 2057, 168+B : REM activar doble buffer; front buffer = bloque B (0-7)

POKE 2057, 85 : REM desactivar doble buffer

Al activarlo, el interface deja de mantener el espejo en tiempo real **solo para el bloque B** de la memoria interna de vídeo (el resto de bloques del espejo siguen funcionando como siempre). A partir de ese momento:

- El vídeo deja de mirar el espejo del bloque HFILE (la pantalla habitual) y pasa a mirar el espejo del bloque B.

- El espejo del bloque B ya **no se actualiza escritura a escritura**. En cada parpadeo vertical (VSYNC), el hardware copia de golpe, byte a byte, todo el contenido actual del espejo del bloque HFILE dentro del espejo del bloque B. Entre un VSYNC y el siguiente, el espejo del bloque B queda **congelado**: es una fotografía fija, no un reflejo en vivo.

> **El punto que más confunde — léelo dos veces:** el bloque lógico B sigue existiendo también como memoria SRAM normal, tan accesible y escribible por la CPU como cualquier otro bloque. Pero mientras el doble buffer está activo, **esa SRAM deja de tener ninguna relación con lo que ves en pantalla**: es el espejo interno de ese bloque el que ya no la refleja. Lo que escribas en la SRAM del bloque B no aparecerá en la imagen, y lo que ves en pantalla no es esa SRAM sino la fotografía que la FPGA renovó en el último VSYNC, copiada del bloque HFILE. Por eso el bloque B conviene tratarlo como “reservado para el hardware” mientras el doble buffer está activo, aunque técnicamente sigas pudiendo usarlo como RAM para otras cosas (variables, código) que nada tengan que ver con la pantalla.

Dicho de otro modo, bajo el mismo número de bloque conviven tres cosas distintas que no hay que confundir:

Capa	Qué contiene	Quién la actualiza
SRAM física del bloque B	La memoria real del sistema (8 KB)	La CPU, con cada lectura/escritura normal
Espejo interno del bloque B — doble buffer OFF	Copia en vivo de esa SRAM	Automáticamente, byte a byte, en cada escritura de la CPU
Espejo interno del bloque B — doble buffer ON	Fotografía fija del bloque HFILE	Solo la FPGA, de golpe, una vez por VSYNC

El resultado práctico:

- La pantalla muestra siempre una **instantánea completa** tomada en el último VSYNC: nunca se ven borrados ni dibujados a medias.

- El programa dibuja siempre sobre **una única superficie** (la página HFILE de siempre): no hay que alternar entre dos páginas ni redibujar “el frame de hace dos”.

- Lo que se relee de la pantalla (la página HFILE, no el bloque B) es siempre lo último escrito — operaciones de leer-modificar-escribir coherentes.

Uso correcto: espera el flanco de subida del VSYNC (bit 0 del puerto AFh) y realiza todo el borrado/dibujado a continuación. Desde ese momento dispones de unos **16 ms** antes de que el hardware tome la siguiente instantánea (la copia comienza justo al terminar el área visible). Si el dibujado excede ese tiempo, la instantánea podría capturar un estado intermedio — el mismo comportamiento que un doble buffer clásico.

Elección del bloque front (B): como el espejo de ese bloque deja de reflejar su SRAM, **no debe apuntarse a él ningún elemento de vídeo** (HFILE, DFILE, atributos Chroma81) mientras el doble buffer esté activo — su espejo ya no serviría para mostrarlos. Bloques recomendados: **4 o 5** ($8000-$9FFF / $A000-$BFFF, si no coinciden con tu HFILE). Evita: 0 (glyphs ROM del modo texto), 1 (chr RAM), 2-3 (DFILE), 6-7 (atributos Chroma81). Si el bloque front coincide con el de HFILE, la pantalla se queda congelada para siempre: el blit se limita a copiar el bloque sobre sí mismo cada VSYNC (sin efecto) y la máscara de escritura bloquea permanentemente que tus dibujos nuevos lleguen al mirror --- la SRAM sí se actualiza, pero nada de eso llega a verse.

> **Nota:** en modo HiRes nativo solo se doble-bufferiza el bitmap; los atributos (zona $C000) se leen en directo. En modo Spectrum se doble-bufferiza el bloque completo (bitmap + atributos). El modo texto no usa el doble buffer.

Ejemplo típico (HFILE en $8000 = bloque 4, front en bloque 5):

POKE 2043,0 : REM HFILE bajo

POKE 2044,128 : REM HFILE alto ($8000)

POKE 2045,172 : REM Superfast HiRes Spectrum

POKE 2057,173 : REM doble buffer ON, front = bloque 5 (168+5)

Con el comando equivalente:

LOAD *DBUF 5 : REM doble buffer ON, front = bloque 5

LOAD *DBUF STOP : REM doble buffer OFF

Control por puerto de E/S (pseudo-bloque 8): el `POKE 2057` deja de funcionar si el programa ha desactivado la ventana de POKEs de control escribiendo en 2056 (modo “RAM plana”, p. ej. CP/M). Para esos casos el doble buffer también se controla a través del **mapper port (E7h)** usando el bloque ficticio 8:

asm

ld a,08h ; pseudo-bloque 8

ld b,32+5 ; valor: bit5=activar, bits2:0=bloque front (aquí 5)

ld c,0e7h

out (c),a ; doble buffer ON, front = bloque 5

ld b,0 ; valor 0 = desactivar

out (c),a

Esta vía está disponible cuando el modo **full paging** está activo o cuando se ha escrito en 2056. Restricción: en half paging tras 2056, no asignes una página impar al bloque 0 con el mapper port (ese patrón de datos, x8h, coincide con el pseudo-bloque 8).

Modo MANUAL: doble buffer sin copia automática

Además del modo AUTO descrito arriba (con blit automático en cada VSYNC), existe un **modo MANUAL** en el que la FPGA no copia nada: el propio Z80 dibuja el frame completo directamente en el bloque que en cada momento no es el front, y solo conmuta cuál de los dos bloques se muestra. Se ahorra el coste de la copia automática (~630 µs por frame), a cambio de que el redibujado completo corre por cuenta del Z80 (viable en modos Superfast, donde la CPU está libre de refresco de vídeo). A diferencia del modo AUTO, aquí no hay un único HFILE que la FPGA copie: hacen falta dos bloques reales (por ejemplo 4 y 5).

POKE 2057, 200+B : REM doble buffer ON modo MANUAL; front = bloque B (0-7)

Los valores 168+B (modo AUTO) y 200+B (modo MANUAL) comparten el mismo POKE 2057; usa 85 para desactivar en ambos casos.

Uso típico:

1. Elige dos bloques reales para los dos buffers (p. ej. 4 y 5). HFILE deja de ser relevante para este mecanismo mientras el doble buffer esté activo.

2. Actívalo con el bloque inicial como front (POKE 2057, 200+4) y dibuja el primer frame en el otro bloque (5).

3. En cada frame: dibuja el frame completo en el bloque que no es el front actual; espera VSYNC (bit 0 del puerto AFh); conmuta el front con POKE 2057, 200+B (B = el bloque que acabas de pintar).

4. En el frame siguiente, pinta el bloque que ha quedado libre (el antiguo front).

La máscara de escritura de la FPGA sigue protegiendo automáticamente al bloque front frente a escrituras de la CPU, en ambos modos.

Por puerto de E/S (pseudo-bloque 8): igual que en modo AUTO, pero con un bit adicional (bit4 del valor B) para seleccionar MANUAL:

ld a,08h ; pseudo-bloque 8

ld b,32+16+5 ; bit5=activar, bit4=modo MANUAL, bits2:0=bloque front (aqui 5)

ld c,0e7h

out (c),a ; doble buffer ON modo MANUAL, front = bloque 5

Consulta el ejemplo completo en código máquina en `EXAMPLES/DBUF/` (pelota rebotando con conmutación del doble buffer en tiempo real).

WRX con la RAM de 8-16K

Algunos programas de alta resolución WRX colocan sus gráficos en la zona de **8-16K** ($2000-$3FFF) y apuntan ahí el registro I (por ejemplo, **Hi-res Chess** de Psion). En un ZX81 real esto requiere una ampliación de RAM en esa zona; en el SD81 Booster esa región existe, pero por defecto el interface la trata como **generador de caracteres en RAM** (la función de LOAD *128C y los juegos con caracteres definibles), que necesita justo el comportamiento contrario durante el refresco de vídeo. Son dos usos legítimos de la misma zona y no es posible distinguirlos automáticamente, así que se selecciona con el comando:

LOAD *WRX : REM activar modo WRX en 8-16K

LOAD *WRX STOP : REM desactivar (modo generador de caracteres, por defecto)

(equivalente a POKE 2058,170 / POKE 2058,85, por si se prefiere desde código máquina). Activa el modo WRX **antes de cargar** el programa (por ejemplo LOAD *WRX y después LOAD “HRCHESS”). Tras un reset vuelve al modo por defecto. No afecta a los programas WRX que sitúan sus gráficos en $4000 o superior, que funcionan siempre sin necesidad de este comando.

Resumen de POKEs de control

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

Dirección	Valor	Función
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)
2056	-	Desactiva los pokes de control y activa la escritura en el bloque 0
2057	168+B	Activar doble buffer (front buffer = bloque B, 0-7)
2057	200+B	Activar doble buffer modo MANUAL (front buffer = bloque B, 0-7)
2057	85	Desactivar doble buffer
2058	170	Activar WRX en la RAM de 8-16K
2058	85	Desactivar WRX (modo generador de caracteres)
2090	0-7	Scroll horizontal fino Superfast (0-7 pixeles)
2091	<mapa>	Filas 0-7 con scroll fino activo (bit0=fila 0)
2092	<mapa>	Filas 8-15 con scroll fino activo
2093	<mapa>	Filas 16-23 con scroll fino activo
2096	<dir_baja>	Byte bajo de la direccion de pantalla alternativa (Superfast)
2097	<dir_alta>	Byte alto de la direccion de pantalla alternativa
2098	170 / 85	Activar / desactivar la direccion de pantalla alternativa
2059	<base_baja>	Byte bajo de la tabla de atributos alternativa (Chroma modo 1)
2060	<base_alta>	Byte alto de la tabla de atributos alternativa
2061	170 / 85	Activar / desactivar la tabla de atributos alternativa
2062	170 / 85	Extender MC45 también a los bloques 6 y 7 / volver solo a 4 y 5

Con LOAD *ROMLOCK activo, las escrituras en todas las direcciones de esta tabla y en las de los sprites (2100-2128, Apéndice H) no tienen ningún efecto, como en una ROM real. LOAD *ROMLOCK STOP las vuelve a activar (ver 7.5).

Apéndice G — Referencia técnica de audio: chip AY, VGM y alófonos

Registros del chip AY-3-8910/12

El emulador AY del SD81 Booster es compatible a nivel de registro con el chip original. Soporta tres voces independientes, envolvente y ruido.

Reg	Descripción	B7	B6	B5	B4	B3	B2	B1	B0
R0	Canal A — período tono (bajo)	B7	B6	B5	B4	B3	B2	B1	B0

Reg	Descripción	B7	B6	B5	B4	B3	B2	B1	B0
R1	Canal A — período tono (alto)	—	—	—	—	B3	B2	B1	B0
R2	Canal B — período tono (bajo)	B7	B6	B5	B4	B3	B2	B1	B0
R3	Canal B — período tono (alto)	—	—	—	—	B3	B2	B1	B0
R4	Canal C — período tono (bajo)	B7	B6	B5	B4	B3	B2	B1	B0
R5	Canal C — período tono (alto)	—	—	—	—	B3	B2	B1	B0
R6	Período de ruido	—	—	—	B4	B3	B2	B1	B0
R7	Habilitación de canales	—	—	Ruido C	Ruido B	Ruido A	Tono C	Tono B	Tono A
R8	Amplitud canal A	—	—	—	Env.	L3	L2	L1	L0
R9	Amplitud canal B	—	—	—	Env.	L3	L2	L1	L0
R10	Amplitud canal C	—	—	—	Env.	L3	L2	L1	L0
R11	Período envolvente (bajo)	B7	B6	B5	B4	B3	B2	B1	B0
R12	Período envolvente (alto)	B7	B6	B5	B4	B3	B2	B1	B0
R13	Forma de envolvente	—	—	—	—	—	B2	B1	B0

INFO	En R7, un bit a 0 activa el canal; a 1 lo desactiva. En R8–R10, si el bit Env. está activo, la amplitud la controla la envolvente (R11–R13) en lugar de L3–L0.

Puertos de E/S — dos chips AY compatibles ZonX-81

El SD81 Booster implementa dos chips AY físicos, compatibles con el interface estándar ZonX-81, decodificados de forma parcial: solo se comprueban los bits A1, A2, A3, A5 y A7 de la dirección; los bits A0, A4 y A6 son indiferentes.

Chip	Selección de registro (latch)	Escritura de dato
Chip A (ZonX-81 estándar, A3=1)	$CFh / $DFh	$0Fh / $1Fh
Chip B (extensión SD81 Booster, A3=0)	$C6h	$06h

El bit A7 hace de línea BC1 del AY: durante una escritura, A7=1 selecciona el registro (latch de dirección) y A7=0 escribe el dato en el registro ya seleccionado. La lectura de estado del PSG (BC1=1, BDIR=0) está implementada en la FPGA pero no expuesta actualmente por el firmware/BASIC.

Opcodes del reproductor VGM

El reproductor VGM del interface solo interpreta los opcodes referentes al chip AY. El resto se ignoran sin producir error.

Opcode	Parámetros	Descripción
61h	nn nn	Esperar n muestras (little-endian, 0–65535; ≈ 1,49 s máximo). Pausas largas se representan con varios comandos consecutivos.
62h	—	Esperar 735 muestras (1/60 de segundo). Equivale a 61h DFh 02h.
63h	—	Esperar 882 muestras (1/50 de segundo). Equivale a 61h 72h 03h.

Opcode	Parámetros	Descripción
A0h	aa dd	Escribir el valor dd en el registro AY número aa.

Tabla de alófonos SP0256-AL2

Los alófonos se usan con el comando MCU BINARY SAY (16h) para síntesis fonética directa de precisión. El acceso de alto nivel mediante texto (LOAD *SAY “texto”) no requiere conocer estos códigos.

Código	Alófono	Ejemplo	Código	Alófono	Ejemplo
$00	PA1	pausa 10 ms	**$20**	AW	out
$01	PA2	pausa 30 ms	**$21**	DD2	do
$02	PA3	pausa 50 ms	**$22**	GG3	wig
$03	PA4	pausa 100 ms	**$23**	VV	vest
$04	PA5	pausa 200 ms	**$24**	GG1	got
$05	OY	boy	**$25**	SH	ship
$06	AY	sky	**$26**	ZH	azure
$07	EH	end	**$27**	RR2	brain
$08	KK3	comb	**$28**	FF	food
$09	PP	pow	**$29**	KK2	sky
$0A	JH	dodge	**$2A**	KK1	can’t
$0B	NN1	thin	**$2B**	ZZ	zoo
$0C	IH	sit	**$2C**	NG	anchor
$0D	TT2	to	**$2D**	LL	lake
$0E	RR1	rural	**$2E**	WW	wool
$0F	AX	succeed	**$2F**	XR	repair
$10	MM	milk	**$30**	WH	whig
$11	TT1	part	**$31**	YY1	yes (corto)
$12	DH1	they	**$32**	CH	church
$13	IY	see	**$33**	ER1	fir (corto)
$14	EY	beige	**$34**	ER2	fir (largo)
$15	DD1	could	**$35**	OW	beau
$16	UW1	too	**$36**	DH2	they
$17	AO	aught	**$37**	SS	vest
$18	AA	hot	**$38**	NN2	no
$19	YY2	yes (largo)	**$39**	HH2	hoe
$1A	AE	hat	**$3A**	OR	store
$1B	HH1	he	**$3B**	AR	alarm

Código	Alófono	Ejemplo	Código	Alófono	Ejemplo
$1C	BB1	business (corto)	**$3C**	YR	clear
$1D	TH	thin	**$3D**	GG2	guest
$1E	UH	book	**$3E**	EL	saddle
$1F	UW2	food	**$3F**	BB2	business (largo)

Apéndice H — Sprites por hardware

El SD81 Booster incluye un sistema de sprites gestionado enteramente por hardware: 64 sprites de 8×8 píxeles, cada uno con su propia máscara de transparencia y su propio color por fila (varios colores dentro del mismo sprite), compuestos automáticamente sobre la imagen de fondo sin coste de CPU. Se controlan con POKEs a un bloque de direcciones fijo, o de forma más cómoda con cuatro comandos LOAD * dedicados.

Sistema de coordenadas

La posición de cada sprite se expresa en un sistema de coordenadas desplazado 32 píxeles respecto a la pantalla, para que un sprite pueda entrar y salir por cualquier borde con un recorte limpio en vez de reaparecer bruscamente por el lado opuesto:

coordenada de sprite 32 = pixel 0 de pantalla

X: 0-318 (32 fuera por la izquierda, 256 visibles, 32 por la derecha)

Y: 0-255 (32 fuera por arriba, 192 visibles, 32 por abajo)

Cualquier píxel del sprite que caiga fuera del área visible real de la pantalla (0,0)-(255,191) sencillamente no se dibuja. Por eso X necesita 9 bits (dos POKEs: parte baja y parte alta) mientras que Y cabe en un solo byte.

Mapa de direcciones (POKE 2100-2128)

Igual que HFILE o el patrón de borde, estas direcciones caen dentro de los primeros 8 KB del mapa de memoria del Z80 (zona de ROM): el interface intercepta esas escrituras concretas en vez de dejarlas caer al vacío, así que no hace falta memoria real ahí. Antes de escribir el color, los datos de píxel o la máscara de un sprite, hay que seleccionarlo primero con POKE 2100,n.

Dirección	Valor	Función
2100	0-63	Selecciona el sprite activo. Las escrituras siguientes afectan a este sprite.
2101	0/1	Activa (1) o desactiva (0) el sprite seleccionado.
2102	0-255	Coordenada X, 8 bits bajos.
2103	0/1	Coordenada X, bit 8 (rango completo 0-318).
2104	0-255	Coordenada Y (32 = primera línea visible).
2105-2112	byte	Color de cada una de las 8 filas: tinta×16+papel (0-15 cada uno).
2113-2120	byte	Patrón de píxel de cada una de las 8 filas.
2121-2128	byte	Máscara de transparencia de cada una de las 8 filas.

Cómo se pinta cada píxel

Para cada píxel del sprite se combinan tres bits, uno de cada matriz en la misma fila y columna:

1. Máscara — si el bit es 0, ese punto es transparente y se ve lo que hay debajo (fondo, texto, otro sprite de menor prioridad). Si es 1, el sprite pinta algo ahí.

2. Dato de píxel — solo importa si la máscara es 1: bit=1 usa la tinta de esa fila, bit=0 usa el papel de esa fila.

Como el color se define fila a fila (no un único color por sprite), un mismo sprite puede combinar varios colores — por ejemplo, una bola con bandas de color distintas en cada línea.

Formato de color

El byte de color de cada fila es tinta×16+papel, con tinta y papel cada uno en la escala 0-15 (la misma que usan LOAD *COLOR y LOAD *BORDER): bit 3 = brillo, bit 2 = verde, bit 1 = rojo, bit 0 = azul. Este formato es el mismo tanto en modo CHROMA como en modo SPECTRUM — el color del sprite se aplica después de que el hardware ya ha resuelto cuál de los dos modos está activo, así que no hace falta preocuparse por el modo al definir el color de un sprite.

Prioridad entre sprites solapados

Si en un mismo píxel hay más de un sprite activo, gana el de índice más alto (el sprite 63 tapa al 0). No hay ningún otro control de orden de dibujado aparte del número de sprite.

Comandos BASIC

LOAD *SPRITE <n>,<x>,<y> — selecciona el sprite <n> (0-63), fija su posición y lo activa, en un solo paso.

LOAD *SPRITE 0,150,88

LOAD *SPRITE <n> STOP — oculta el sprite <n> sin tocar su posición ni sus datos gráficos.

LOAD *SPRITE 0 STOP

LOAD *SPRCOL / LOAD *SPRPIX / LOAD *SPRMASK <n>,“<16 caracteres hex>” — cargan, respectivamente, el color, los datos de píxel o la máscara del sprite <n>: 16 caracteres hexadecimales = 8 bytes = una fila por byte. Cada comando selecciona automáticamente el sprite indicado.

LOAD *SPRCOL 0,“0202020606060606”

LOAD *SPRPIX 0,“070810182626181C”

LOAD *SPRMASK 0,“070F1F1F3F3F1F1F”

Se usan 16 caracteres hex en vez de un único bloque de 48 (color+píxel+máscara juntos) precisamente para reducir el riesgo de error al escribir una cadena tan larga a mano.

Capacidad

El firmware actual soporta 64 sprites, con posicionamiento libre por píxel y color por fila, pero como máximo 12 de ellos pueden aparecer en una misma línea de pantalla. Si más de 12 sprites activos cortan la misma línea, en esa línea se dibujan los 12 de índice más alto y los demás no aparecen; en las líneas donde no coinciden se ven con normalidad. Es un límite parecido al de la NES (que tenía 8 por línea), y los programas que mueven muchos sprites juntos suelen rotar cuáles quedan fuera en cada imagen para que parpadeen en vez de desaparecer. Cuenta cualquier sprite activo que cruce la línea, esté donde esté en horizontal, y un sprite activo fuera de la pantalla también ocupa hueco: conviene desactivarlo (LOAD

*SPRITE n STOP). Un cambio en los datos de un sprite se ve a partir de la línea siguiente. Es una cifra fija para el usuario final ---depende del bitstream concreto de la FPGA que lleve instalado el interface---, no algo configurable desde BASIC.

Apéndice I — CP/M: un sistema operativo alternativo

Además del BASIC de serie del ZX81, el SD81 Booster puede arrancar CP/M, el sistema operativo estándar de los microordenadores profesionales de 8 bits de finales de los 70 y los 80. Existen dos puertos para este hardware:

CP/M 2.2, la versión clásica, sin bancos de memoria: hasta unos 41 KB de TPA (área de programa transitorio) en modo MC45.

CP/M 3 (CP/M Plus) bancado, que usa el paginador de memoria del SD81 Booster para repartir la RAM en bancos de 8 KB independientes de sistema y de usuario. Es el que documenta este apéndice.

CP/M 3 aporta, sobre el 2.2: más TPA (el sistema y el usuario ya no comparten el mismo banco), un disco RAM, el reloj en tiempo real integrado en el sellado de fecha de los ficheros, una consola con 256 caracteres y color, y un driver de red para BBS y sistemas de comunicaciones por WiFi.

El sistema sigue el estándar de Digital Research (DRI) hasta donde el hardware lo permite: mismo formato de CCP, BDOS y BIOS, mismas convenciones de programa .COM, misma tabla pública de llamadas al BIOS. Lo específico de este puerto es la capa de paginación de memoria (el XIOS) y los drivers de dispositivo — disco, teclado, consola, reloj y red — escritos para el hardware del SD81 Booster.

CP/M 2.2: los dos modos de compilación

CP/M 2.2 se distribuye en dos variantes, seleccionadas al compilar, según se use o no el modo MC45 (ver el capítulo 11, «Gestión de memoria», y en particular «Extensión de MC45 a los bloques 6 y 7»):

Modo	MC45	TPA	RAM de vídeo	Recomendado para
32 KB	No	~24 KB	$8000	Uso general de CP/M
48 KB	Sí	~41 KB	$C000	Turbo Pascal y programas grandes

El modo de 32 KB funciona en cualquier SD81 Booster sin nada especial que activar. El de 48 KB necesita MC45 encendido, y da casi el doble de TPA — la diferencia entre poder compilar con Turbo Pascal o no.

CP/M 3 necesita toda la memoria por encima de los 32 KB para su paginación en bancos, no solo los bloques 4 y 5 — de ahí que use la extensión de MC45 a los bloques 6 y 7 descrita en el capítulo 11 (POKE 2062,170). El propio arranque de CP/M 3 la activa por su cuenta, sin que el usuario tenga que hacer nada.

Instalación y arranque

El sistema se distribuye como un único fichero, SYSTEM.BIN, que contiene el BIOS, la BDOS y el CCP ya enlazados. Se carga desde BASIC igual que cualquier bloque de código máquina y se arranca con USR:

```
LOAD FAST “SYSTEM.BIN” CODE 24576

RAND USR 24576
```

Además de SYSTEM.BIN, la tarjeta necesita las imágenes de disco: A.IMG a D.IMG (256 KB de directorio, 2 MB de datos cada una) en la raíz de la SD. Las imágenes de CP/M 3 tienen un formato propio y NO son compatibles con las de CP/M 2.2 ni con las de BASIC — no se pueden mezclar. Una quinta unidad, E:, es un disco RAM que no necesita imagen: se crea vacío en cada arranque y su contenido se pierde al apagar o resetear.

Memoria bancada

El SD81 Booster pagina la memoria en 8 bloques de 8 KB. CP/M 3 usa esa capacidad para mantener dos contextos completos: uno de sistema (donde vive el BIOS y la BDOS) y uno de usuario (la TPA, donde corren los programas). El bloque más alto ($E000-$FFFF) es común a los dos contextos — ahí vive todo lo que necesita ser alcanzable sin importar cuál esté activo, incluida la tabla pública de llamadas al BIOS (ver más abajo).

Un programa de usuario no necesita saber nada de esto: el cambio de contexto lo gestiona el propio sistema en cada llamada a la BDOS o al BIOS. Solo importa si se escribe código que accede a memoria fuera de la TPA, o que necesita ir más rápido que una llamada normal a la BDOS (ver «Llamar al BIOS directamente»).

Discos: A: a D: (tarjeta SD) y E: (RAM)

A: a D: son las cuatro unidades respaldadas por la tarjeta SD, con formato CP/M 3 estándar (2 KB por bloque de asignación, 256 entradas de directorio). E: es un disco RAM de unos 390 KB, rápido pero volátil — útil para compilaciones, ficheros temporales o cualquier cosa que no necesite sobrevivir a un reset.

MOUNT3.COM permite cambiar en caliente la imagen montada en una unidad sin reiniciar el sistema — necesario para intercambiar disquetes virtuales durante una sesión larga.

Teclado

El ZX81 no tiene teclado ASCII completo: le faltan símbolos, y solo tiene una tecla de flecha por combinación con SHIFT. La tabla siguiente resume cómo se llega a cada carácter. Ver «Tabla de teclado de CP/M» para el listado completo.

Directa y SHIFT+tecla: letras, dígitos y las flechas (estilo WordStar: SHIFT+5/6/7/8 = izquierda/abajo/arriba/derecha).

ENTER+tecla: los símbolos que el ZX81 no tiene serigrafiados directamente — incluidos los que hacen falta para programar en C o usar una BBS (@ \ | ~ ` _ # % & !, entre otros).

SHIFT+ENTER, y luego una tecla: modo control. Da ^A-^Z con las letras, y NUL con la barra espaciadora.

SHIFT+9 y SHIFT+0: retroceso (BS, $08) y borrado (DEL, $7F) — las dos, porque distinto software remoto espera un byte distinto para la misma función.

SHIFT+.: tabulador (TAB, $09).

Reloj en tiempo real

CP/M 3 sella la fecha y hora de creación/modificación de cada fichero usando el RTC de la placa. La hora se ajusta con el propio RTC del SD81 Booster (por ejemplo desde BASIC, o sincronizándolo por NTP si hay módulo WiFi) — CP/M solo lee lo que el RTC ya tiene.

Consola: terminal con 256 caracteres y color

La consola local entiende un subconjunto de secuencias ANSI/VT100, suficiente para BBS, editores de pantalla completa y cualquier programa que pinte con color:

Movimiento de cursor: ESC[fila;colH (o f), ESC[nA/B/C/D (arriba/abajo/derecha/izquierda), ESC[s/ESC[u (guardar/restaurar posición).

Borrado: ESC[2J (pantalla), ESC[K (fin de línea).

Color: ESC[...m (SGR) — ver la utilidad SGR más abajo para la lista de códigos soportados.

ESC[6n (petición de posición del cursor): se contesta de verdad, con ESC[fila;colR — necesario porque algunos programas (BBS incluidas) miden el tamaño del terminal así antes de arrancar.

256 caracteres (juego CP437 completo, no solo el ASCII imprimible).

Llamar al BIOS directamente

Todas las funciones del BIOS son alcanzables desde un programa de usuario sin pasar por la BDOS, a través de una tabla pública de saltos de 3 bytes cada uno, empezando en la dirección $FF00:

Función	Índice	Dirección
CONST	2	$FF06
CONIN	3	$FF09
CONOUT	4	$FF0C
AUXOUT	6	$FF12
AUXIN	7	$FF15
AUXIST	18	$FF36
AUXOST	19	$FF39

(dirección = $FF00 + índice × 3; la lista completa de 33 entradas sigue el orden estándar de CP/M 3).

Es la misma técnica que usan WordStar y Turbo Pascal para no pagar el coste de la BDOS en operaciones de consola frecuentes, y está pensada para programas que necesiten el máximo rendimiento posible — por ejemplo, un terminal de comunicaciones. El programa TERM incluido con el sistema (ver más abajo) es un ejemplo real: llama a CONOUT así, y eso multiplica por varias veces la velocidad de la consola frente a pasar por la BDOS.

Comunicaciones

El sistema incluye un driver de red (NET) que expone al Z80 un flujo de bytes crudo, como si tuviera un módem conectado — el propio software decide qué hacer con ese flujo (por ejemplo, mandar comandos AT para marcar una conexión). El puente WiFi real lo gestiona el módulo ESP32 de la placa; el Z80 no sabe nada de sockets ni de direcciones.

TERM.COM es el terminal incluido para usar NET. A diferencia de un programa CP/M típico, NO pasa por el dispositivo lógico AUX: ni por la BDOS: habla con el hardware de red directamente, con el mismo truco de llamada directa al BIOS descrito arriba para la salida por pantalla. La razón es de rendimiento — un terminal genérico que respetara DEVICE AUX: sería varias veces más lento — y significa que TERM está pensado específicamente para NET, no como sustituto de un terminal de comunicaciones de propósito general.

Teclas de TERM (todas con ENTER+tecla):

Tecla	Función
ENTER+0	Salir
ENTER+9	Conmutar el eco local (verlo escrito mientras se teclea)
ENTER+8	Conmutar el volcado a fichero (con TERM fichero.BIN)

Antes de conectar, lo que el Z80 escribe por NET se interpreta como comandos de módem Hayes (AT). Una vez conectado, todo lo que se escribe va directo al socket — comandos AT incluidos, salvo que se vuelva a modo comando con “+++”. Comandos soportados:

Comando	Efecto
AT	Comprobación. Responde OK.
ATDT host:puerto	Abre el socket (espacio tras «DT» opcional). Responde CONNECT o NO CARRIER.
ATDL	Remarca el último host:puerto marcado, con éxito o sin él. ERROR si nunca se marcó nada.
ATH / ATH0	Cuelga. Responde OK.
ATO / ATO0	Vuelve a modo datos sin colgar. Responde CONNECT.
ATE0	Apaga el eco de los comandos en modo comando.
ATE1	Enciende el eco de los comandos (por defecto).
ATZ	Reinicia al estado inicial: cuelga y restaura el eco. Responde OK.
+++	Escapa a modo comando sin colgar, con un segundo de silencio antes y después. Responde OK.

Las respuestas siempre terminan en CR/LF: OK, CONNECT, NO CARRIER, ERROR. Si el otro extremo cierra la conexión, TERM recibe NO CARRIER y vuelve a modo comando por su cuenta — no hay reconexión automática de la sesión: para volver a marcar hay que usar ATDT o ATDL de nuevo, igual que en un módem real.

También existe una página web de configuración (http://<ip-del-módulo-wifi>/telnet) que permite fijar el destino y ver el estado de la conexión sin usar comandos AT — pensada sobre todo para pruebas. El camino que usa el software real es ATDT/ATDL.

Utilidades incluidas

Programa	Función
MOUNT3.COM	Cambia en caliente la imagen montada en una unidad
TERM.COM	Terminal de comunicaciones (ver arriba)
CLS.COM	Borra la pantalla y sitúa el cursor en el origen
SGR.COM	Cambia el color de tinta/papel de la consola
BORDER.COM	Cambia el color del borde

SGR parámetros manda la secuencia ANSI ESC[parámetros m a la consola, separados por ;:

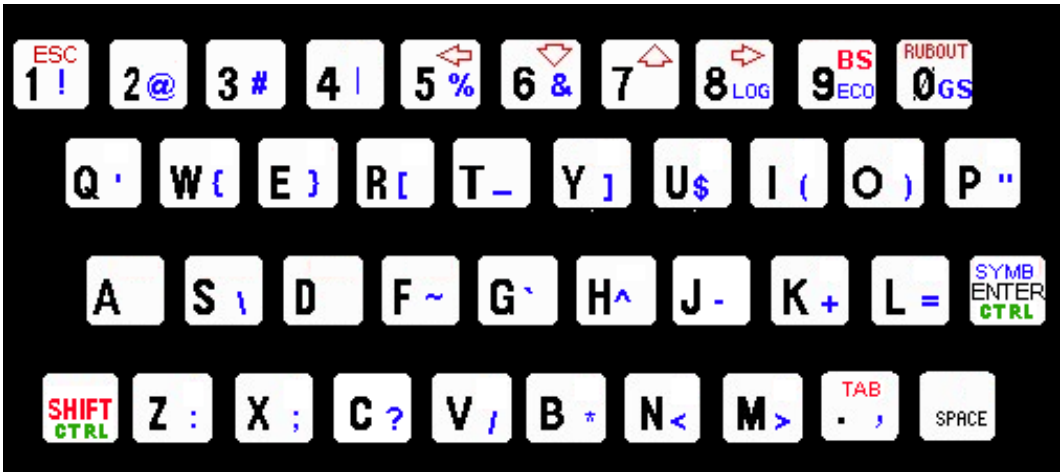
Código	Efecto
0	Restaurar los colores por defecto
1	Tinta brillante
30-37	Color de tinta: negro, rojo, verde, amarillo, azul, magenta, cian, blanco
40-47	Color de papel, misma tabla de colores

> SGR 0 reset
>
> SGR 33 tinta amarilla
>
> SGR 1;33;44 tinta amarilla brillante, papel azul

BORDER n, con n de 0 a 7 (color normal) o de 8 a 15 (mismo color brillante, sumando 8) — ver el Apéndice E, «Puerto del interface Chroma81 (7FEFh)», sección «Escritura», para el formato completo del puerto.

> BORDER 4 borde azul
>
> BORDER 12 borde azul brillante

Tabla de teclado de CP/M



Las teclas ENTER+8 y ENTER+9 se rotulan por su función (LOG, ECO) en vez de por el código de control que mandan ($1E, $1C): son las que usa TERM para su propio menú.

Tecla	Directa	SHIFT	ENTER
1	1	ESC	!
2	2	—	@
3	3	—	#
4	4	—	|
5	5	←	%
6	6	↓	&
7	7	↑	—
8	8	→	—

Tecla	Directa	SHIFT	ENTER
9	9	BS	^\
0	0	DEL	^]
Q	q	Q	’
W	w	W	{
E	e	E	}
R	r	R	[
T	t	T	_
Y	y	Y	]
U	u	U	$
I	i	I	(
O	o	O	)
P	p	P	”
A	a	A	—
S	s	S	\
D	d	D	—
F	f	F	~
G	g	G	`
H	h	H	^
J	j	J	-
K	k	K	+
L	l	L	=
Z	z	Z	:
X	x	X	;
C	c	C	?
V	v	V	/
B	b	B	*
N	n	N	<
M	m	M	>
.	.	TAB	,
ESPACIO	espacio	espacio	espacio
ENTER	CR	CR	CR

Además, SHIFT+ENTER y luego una letra da ^A-^Z (letra AND $1F), y SHIFT+ENTER+espacio da NUL.

Mapa de memoria de CP/M 3 (resumen)

Rango	Contenido
$0000-$DFFF	Bancado: TPA (contexto de usuario) o BIOS/BDOS/CCP (contexto de sistema), según cuál esté activo
$E000-$FFFF	Común a los dos contextos: el SCB ($FE00), la tabla pública de llamadas al BIOS ($FF00), y el resto del estado que tiene que verse igual desde ambos lados

El tamaño exacto de la TPA depende de la configuración final del sistema; se puede consultar arrancando y mirando el mensaje inicial de CP/M 3 (Banked memory, NN.NK TPA).

CP/M y el depurador

El depurador por hardware (Apéndice L) funciona con CP/M. Pulsa el botón QuickSilva entre 1 y 3 segundos y aparece su pantalla; S, O, U y C ejecutan paso a paso o siguen, y volver a pulsar el botón también sigue. Con CP/M el bloque 0 ($0000-$1FFF) es RAM de CP/M, así que los sprites y los POKEs de control, que viven en ese bloque, no existen mientras CP/M está en marcha. Mientras el depurador está en pantalla, el bloque 0 vuelve a comportarse como la ROM de siempre y no se toca la RAM de CP/M: al seguir, todo está como estaba.

Los snapshots también valen: el botón QuickSilva más de 3 segundos (o snap en la consola) guarda el estado de CP/M y LOAD *Z81 lo restaura. El historial (th) sigue apuntando las instrucciones aunque el bloque 0 sea RAM.

Apéndice J — El explorador de archivos

El explorador (EXPLORER.BIN) es un navegador de la tarjeta SD en modo Superfast HiRes Spectrum (42 columnas, con una fuente comprimida de 6 píxeles), pensado para no tener que recordar nombres de fichero ni comandos LOAD para cargar un programa, ver una imagen o gestionar la tarjeta.

Arranque

```
2 LET ORG=24576
3 LOAD FAST
10 FAST
12 LOAD *FULLPAG
15 LOAD THEN CLEAR ORG-1
20 LOAD FAST "/EXPLORER/EXPLORER.BIN" CODE ORG
30 LET N=USR ORG
40 IF N=0 THEN GOTO 5000
50 LET F$=""
55 LET EXT = 255
60 FOR I=0 TO N-1
65 LET A$=CHR$ (PEEK (ORG+3+I))
```

```
67 IF A$=“.” THEN LET EXT = I+1

70 LET F$=F$+A$

80 NEXT I

85 SLOW

90 LET E$=F$(EXT TO )

95 LOAD THEN CLEAR 32767

100 IF (E$ = “.P”) OR (E$=“.81”) OR (E$=“.WAV”) OR (E$=“.ROM”) THEN LOAD FAST F$ THEN
GOTO 1

130 IF (E$ = “.Z81”) THEN LOAD *Z81 F$

300 GOTO 10

5000 LOAD THEN CLEAR 32767

5010 SLOW
```

Al salir del explorador con un archivo seleccionado (tecla ENTER u 8 sobre un fichero que no sea .VGM, .PEB, .SCR ni .TXT — esos cuatro los gestiona el propio explorador, ver más abajo), este stub recoge el nombre en F$ y decide qué hacer según la extensión. Tal como está, carga con LOAD FAST los .P, .81, .WAV y .ROM, y restaura con LOAD *Z81 los snapshots .Z81 (ver 7.7); para cualquier otra extensión que quieras cargar de otra forma, añade tu propia condición antes de la línea 300.

Las líneas 95 y 5000 devuelven RAMTOP a 32768, el valor estándar de un ZX81 con 16K, que la línea 15 había bajado a ORG para proteger el explorador. Fíjate en que es 32767 y no 32768: LOAD THEN CLEAR deja RAMTOP en la dirección indicada más uno, y hay juegos que se cuelgan si RAMTOP no tiene el valor estándar.

La tarjeta SD que se descarga del repositorio ya viene configurada para arrancar así: incluye este arranque como /AUTOEXEC.P, así que con el interface conectado basta con encender el ZX81 y pulsar RUN y ENTER para que el explorador se cargue y arranque solo, sin teclear nada (ver 5.4, «El programa AUTOEXEC»).

Navegación

Tecla	Acción
6	Bajar en la lista
7	Subir en la lista
1	Página anterior
2	Página siguiente
5	Subir a la carpeta padre
8 / ENTER	Entrar en la carpeta seleccionada, o abrir/cargar el archivo
ESPACIO	Salir al BASIC (sin ningún archivo seleccionado)
.	Filtrar el listado por comodines (* ?)
SHIFT+1	Quitar el filtro activo

El filtro persiste mientras navegas entre carpetas; solo se borra con SHIFT+1.

Qué pasa al seleccionar un archivo

Según la extensión, ENTER/8 hace una de estas cuatro cosas sin salir del explorador, o devuelve el nombre al BASIC:

Extensión	Comportamiento
.VGM	Se carga y empieza a sonar en el acto
.PEB	Igual que .VGM, para efectos generados con el editor PEG
.SCR	Se muestra a pantalla completa (captura Spectrum, 6912 bytes); cualquier tecla vuelve al listado
.TXT	Se abre en el visor de texto (ver abajo)
cualquier otra	El explorador sale al BASIC con el nombre del archivo, para que el stub decida qué hacer

Operaciones de fichero

Tecla	Acción
N	Nueva carpeta (pide el nombre)
D	Borrar el fichero o carpeta seleccionado (pide confirmación; las carpetas solo se borran si están vacías)
R	Renombrar (pide el nombre nuevo, precargado con el actual)
C	Marcar el seleccionado para copiar
X	Marcar el seleccionado para mover
V	Pegar (copia o mueve, según lo marcado con C/X, en la carpeta actual)

Al marcar con C o X, el icono correspondiente de la barra inferior se tiñe de azul como recordatorio de que hay algo pendiente de pegar.

Edición de texto (nueva carpeta / renombrar / filtro)

Editor con cursor real (inserta y borra en la posición del cursor, no solo al final):

Tecla	Acción
letras, dígitos, .	Carácter literal
SHIFT+Z/X/C/V	: ; ? /
SHIFT+J/K/L	- + =
SHIFT+N/M/.	< > ,
SHIFT+B	*
SHIFT+ESPACIO	` (comilla inversa)
SHIFT+5 / SHIFT+8	Mover el cursor izquierda / derecha
SHIFT+0	Borrar (el carácter a la izquierda del cursor)

Tecla	Acción
SHIFT+1	Restaurar el nombre original (solo al renombrar)
ENTER	Confirmar
ESPACIO	Cancelar

Panel de configuración (S)

La tecla S abre y cierra un panel lateral con el estado de la FPGA y del reproductor de música, y permite cambiarlo sin salir al BASIC:

Tecla	Acción
W	Conmutar WRX (generador de caracteres en la RAM de 8-16K)
F	Conmutar paginación completa (FULLPAG)
M	Conmutar MC45
A	Conmutar el juego de caracteres entre 64 y 128
J	Editar las cinco teclas del joystick (arriba, abajo, izquierda, derecha, fuego)
T	Parar la música (VGM/PEB) en curso
Y	Pausarla
U	Continuarla
I	Ver /MAN/IP.TXT (la IP del módulo WiFi, si hay conexión)

El panel también muestra, en su última línea, la versión del MCU, la ROM y la FPGA (Mx.y,Rx.y,Fx.y). W/F/M/A/J/T/Y/U solo actúan con el panel visible.

Visor de texto (.TXT)

Tecla	Acción
6 / 7	Línea siguiente / anterior
1 / 2	Página anterior / siguiente
ESPACIO	Salir al listado

Visor hexadecimal

Se abre desde el panel de configuración con la tecla H, sobre el archivo seleccionado en el listado (no disponible para carpetas):

Tecla	Acción
6 / 7	Fila siguiente / anterior
1 / 2	Página anterior / siguiente
G	Ir a una dirección (hexadecimal)

Tecla	Acción
Z	Alternar la columna derecha entre ASCII y códigos de carácter ZX81
ESPACIO	Salir al listado

Cómo se devuelve un archivo al BASIC

El explorador vive en RAM como código máquina suelto (cargado con LOAD ... CODE), así que no puede tocar él mismo la zona de programa y variables BASIC — se limita a devolver el nombre del archivo seleccionado (para los tipos que no gestiona internamente) y dejar que el stub decida. Al salir:

1. Copia el nombre (ya en ASCII) a una dirección fija, ORG+3 (24579 con el ORG de arriba), que sobrevive a la restauración de vídeo.

2. Restaura el modo de vídeo estándar y el mapeo de memoria normal (identidad: cada bloque a su página del mismo número).

3. Devuelve en BC la longitud del nombre (0 si se salió con ESPACIO) — en ZX81, USR dirección como función devuelve el contenido de BC.

Ensamblar

Con pasmo (http://pasmo.speccy.org/):

```
pasmo explorer.asm EXPLORER.BIN
```

Apéndice K — El emulador EightyOne-CrossPlatform

El emulador, EightyOne-CrossPlatform, se descarga de https://codeberg.org/wilco2009/EightyOne-CrossPlatform

Este apéndice explica **cómo usar el SD81 Booster dentro del emulador EightyOne**: activarlo, apuntarlo a una tarjeta SD virtual y trabajar con las ventanas, menús, snapshots, consola serie y depurador hardware del emulador.

No describe los comandos propios del SD81 Booster (LOAD *..., PLAY, MAP, ...) ni el hardware. Para eso, consulta el resto de este manual, que se aplica igual en el emulador.

Qué se emula

EightyOne emula el SD81 Booster completo, no solo sus comandos:

Parte	En el emulador
MCU (STM32)	Su protocolo en los puertos $A7 / $AF, los comandos del sistema de ficheros, PLAY, VGM, PEG, voz, RTC, joystick
FPGA	Paginación de 64 × 8 KB, MC45, modos de vídeo Superfast y Spectrum, modos de texto con pantallas anchas, sprites hardware, doble búfer, Chroma81, AY ZonX81, interrupciones simuladas
Depurador del firmware	El DEBUGGER.cpp del propio firmware, ejecutándose sin modificar, más la parte FPGA del depurador hardware

Parte	En el emulador
Tarjeta SD	Una carpeta de tu ordenador
LED y botón QuickSilva	Se muestran y se manejan desde la ventana de la consola serie

Como el emulador ejecuta el código real del firmware para el depurador y la ROM real del SD81 (SDBOOST.ROM) para todo lo demás, los programas y herramientas escritos para la placa real se comportan igual.

Activar el SD81 Booster

1. Abre **Options → Hardware…** (F6).

2. Selecciona la máquina **ZX81**, **TS1000** o **TS1500**. El SD81 Booster no está disponible en ZX80, Lambda, R470, TK85, Jupiter Ace ni ZX97LE; allí la casilla aparece atenuada.

3. Marca **SD81 Booster**.

4. Pulsa **SD Card…** y elige la carpeta que hará de tarjeta virtual (ver «La tarjeta SD virtual»).

5. Pulsa **Apply / OK**. La máquina se reinicia con el SD81 Booster instalado.

Con el SD81 Booster activo, el emulador configura el resto de la máquina como la placa real, y estos ajustes quedan forzados:

- Sonido: chip AY **ZonX81**.
- Joystick: **Programmable** (teclas por defecto: 5 izquierda, 8 derecha, 7 arriba, 6 abajo, Espacio disparo).
- Alta resolución: **WRX**.
- Color: **Chroma81**, siempre activo (no hay interruptor).
- Generador de caracteres: **QuickSilva**, que la ROM del SD81 activa cuando hace falta.
- La zona 8–16K es RAM.

El ajuste del SD81 Booster se guarda en el INI y se recuerda entre sesiones. La opción Save Configuration (**Options → Configuration**) lo guarda junto con el resto de ajustes de hardware.

La tarjeta SD virtual

La “tarjeta SD” es una carpeta normal. En ella está todo lo que tendría la tarjeta real:

> <carpeta SD>/
>
> AUTOEXEC.P se ejecuta al arrancar (si existe)
>
> SYS/
>
> SDBOOST.ROM la imagen de la ROM del SD81 (obligatoria)
>
> DEBUG.BIN el monitor del depurador (opcional, ver «El depurador hardware»)
>
> <tecla>.ROM ROMs alternativas (ver «Teclas de arranque»)
>
> ... tus programas, carpetas, snapshots .Z81, ficheros VGM, etc.

Elegir la carpeta

- **Options → Hardware… → SD Card…** abre un selector de carpeta. En el primer arranque apunta por defecto a la carpeta SD81 junto al emulador.
- Puedes usar cualquier carpeta. Aplica las mismas reglas de nombres que en la tarjeta real (ver el resto de este manual).
- Cambiar la carpeta requiere aplicar el diálogo de hardware, lo que reinicia la máquina.

La ROM

El emulador lee **`SYS/SDBOOST.ROM`** de la SD virtual, exactamente como hace la MCU en la placa real, así que puedes probar una ROM nueva simplemente sustituyendo ese fichero y pulsando **Hard Reset**.

Si el fichero no existe, el emulador recurre a la ROM cargada por el cargador de ROM propio de EightyOne. Algunas funciones (por ejemplo, cargar snapshots a través del monitor del depurador) necesitan la SDBOOST.ROM real.

Editar la tarjeta con el emulador en marcha

La carpeta se lee bajo demanda, así que puedes añadir o cambiar ficheros con tu gestor de archivos mientras el emulador funciona. Los listados de directorio (LOAD *DIR) muestran siempre el contenido actual. Un fichero que el SD81 tiene abierto (por ejemplo un VGM en reproducción) lo mantiene abierto el emulador, como en la tarjeta real, y no se puede borrar desde Windows hasta que se cierre.

Arranque: reset, teclas de arranque y AUTOEXEC

Menú	Qué hace
Control → Hard Reset (Power Cycle) (Shift+F2)	Reinicializa la máquina y el SD81 Booster, como si se apagara y encendiera. La ROM se recarga desde la SD.
Control → Soft Reset (RESET Line) (F2)	Pulsa la línea de reset. El SD81 Booster conserva su estado y no recarga nada.
Control → NMI (Shift+F3)	Genera una NMI.
Control → Pause (F3)	Pausa la emulación.

Teclas de arranque

La ROM del SD81 mira las teclas numéricas 0–9 durante el arranque para elegir una ROM alternativa (/SYS/<tecla>.ROM). En el emulador, **mantén pulsada la tecla numérica mientras pulsas Hard Reset** (o mientras arranca el emulador). El emulador conserva el estado del teclado durante el reset para que la ROM vea la tecla, como en la máquina real.

AUTOEXEC

Si la raíz de la SD contiene AUTOEXEC.P, la ROM del SD81 lo ejecuta al arrancar, como en la placa real.

Pantalla, sonido y joystick

Pantalla

- **Options → Display Settings…** (F5) y **View → Zoom / Border** funcionan como en cualquier máquina. Las pantallas extendidas del SD81 (Spectrum 256 × 192, texto Superfast, modos de texto anchos) se muestran en la ventana normal.

- **Options → TV Emulation…** (F10) ajusta brillo, contraste, color, ruido, fantasma (ghosting) y líneas de barrido (scanlines). Los valores por defecto son los adecuados para la combinación ZX81 + SD81 Booster.
- Un reset devuelve la pantalla al modo ZX81 por defecto y apaga el color Chroma, como la placa real.

Sonido

El sonido de los chips AY (ZonX81, el segundo AY de la FPGA y el AY de la MCU que usan PLAY, VGM y PEG) va a la salida de audio del emulador. **Options → Sound…** (F8) y **Tools → Sound Output…** se aplican como siempre.

Joystick

- **Control → Connect Joystick 1/2** conecta un mando físico al joystick programable.
- El mapeo de teclas se cambia con el comando JOY (ver el resto de este manual) o desde el diálogo de hardware.
- Las opciones de **autodisparo** están en el menú Control.

Snapshots

Los snapshots son ficheros .Z81. Con el SD81 Booster activo llevan una sección extra [SD81BOOSTER] con el estado de la placa (el formato está en DOC/z81_snapshot_format.md). Un snapshot hecho con el SD81 restaura todo el estado: las 64 páginas de memoria, el mapeo de páginas, el modo de vídeo, el borde, Chroma, las pantallas anchas, los sprites, los chips AY, un VGM que estuviera sonando, los programas PEG, los ficheros que estuvieran abiertos y el directorio actual.

Desde el menú del emulador

- **File → Save Snapshot…** (F12) guarda un snapshot.
- **File → Load Snapshot…** (F11) carga uno. Arrastrar y soltar en la ventana solo admite ficheros .P, no snapshots.

Desde el ZX81

- LOAD *Z81 “nombre” carga un snapshot desde la tarjeta SD. Cuando el monitor del depurador del SD81 está instalado (ver «El depurador hardware»), el emulador lo carga exactamente igual que la placa real: el sonido se silencia mientras carga y el VGM, el PEG y los ficheros abiertos se reanudan al final.
- Los snapshots hechos desde el depurador hardware (el comando snap, o manteniendo el botón QS 3 segundos) son ficheros .Z81 normales y también se cargan desde el menú del emulador.

Cargar con el SD81 desactivado

Un snapshot con sección [SD81BOOSTER] cargado con el SD81 desactivado carga solo la memoria y la CPU; activa primero el SD81 para obtener el estado completo.

CP/M Disk Manager

Tools → CP/M Disk Manager… permite explorar imágenes de disco CP/M y copiar ficheros entre ellas y tu ordenador, sin herramientas externas.

- Formatos admitidos: SD81 **CP/M+ (2 MB)** y SD81 **CP/M 2.2 (250 KB)**.

- Puedes crear una imagen nueva, copiar ficheros desde y hacia la imagen y borrar ficheros.
- CP/M se ejecuta desde la tarjeta SD (LOAD *CPM...); consulta el resto de este manual. **Help → CP/M Keyboard Map** muestra el mapa de teclas de CP/M.

Ventana Debug: panel SD81 Booster

Tools → Debug Window... (Ctrl+F6) muestra un panel extra **SD81 Booster** en su esquina inferior derecha cuando la placa está activa. Los valores aparecen atenuados mientras el emulador funciona y se actualizan cuando está detenido.

Campo	Significado
Mode	Modo de vídeo actual (TEXT, Spectrum, ancho, ...)
Chroma	Registro de modo de Chroma81
HFILE	Dirección del fichero de pantalla
Border	Modo y color del borde
NMI gen	Estado del generador de NMI (y fotogramas)
CDFLAG	Estado del indicador FAST/SLOW
WRX	Alta resolución WRX activada o no
MC45	Modo MC45: off (código hasta $8000), on (todo el rango) u on ($C000)
Block / Page	Los 8 bloques de memoria y la página de 8 KB mapeada actualmente en cada uno

La **Live Memory View** (Ctrl+F5) muestra la memoria tal como la ve la CPU, con el mapeo actual.

Si usas los símbolos de la ventana Debug, los ficheros de símbolos (.SYM) se resuelven por página de memoria, de modo que las etiquetas del código paginado se resuelven en la página correcta.

La consola serie

La placa real tiene un puerto serie USB-C que se usa para diagnóstico y para el depurador hardware. El emulador ofrece lo mismo en una ventana.

Ábrela con **Tools → SD81 Booster Serial Console...**.

La ventana

> [línea de entrada] [QS (hold)] [LED] [Clear]
>
> +----------+
>
> | salida de la MCU |
>
> | ... |
>
> +----------+

- **Línea de entrada** (arriba): escribe un comando y pulsa **Enter**. La línea se repite en la salida con el prefijo >. Las teclas ↑ y ↓ recuperan las líneas anteriores.
- **Salida**: todo lo que imprime la MCU, en fuente monoespaciada. La ventana conserva la parte más reciente de una salida larga.
- **Clear**: vacía la salida.

- **QS (hold)** y **LED**: ver más abajo.

El botón QS y el LED

La placa tiene un **botón QuickSilva** y un **LED de estado** tricolor. En el emulador:

- Mantén pulsado con el ratón el botón **QS (hold)** y suéltalo cuando el LED muestre el color que quieres. Al soltar se ejecuta la acción correspondiente al tiempo que lo has mantenido.
- El **cuadrado LED** junto al botón muestra en tiempo real el color del LED de la placa, incluido el parpadeo.

Pulsado durante	LED	Al soltar
menos de 1 s	verde (o cian si QuickSilva está apagado)	conmuta QuickSilva
1 – 3 s	magenta	pausa / continúa el programa (depurador hardware)
más de 3 s	amarillo	hace un snapshot

Otros colores reflejan actividad: amarillo parpadeando mientras se escribe un snapshot, cian parpadeando mientras se carga uno, magenta cuando el depurador tiene el programa parado.

INFO	El LED y la medición del tiempo de pulsación se actualizan una vez por fotograma emulado, así que el color puede ir retrasado respecto al ratón hasta un fotograma (20 ms). El botón no se evalúa mientras el propio emulador está en pausa.

Comandos de la consola

Comando	Acción
DBG_PAUSE	Detiene el programa en el siguiente límite de instrucción (depurador hardware)
DBG_RELOAD	Recarga /SYS/DEBUG.BIN y reinicia el Z80 y la FPGA
DBG_TRACE / DBG_TRACE on / DBG_TRACE off	Muestra o cambia el trazado de comandos de la MCU en la consola
DBG_FPGA	Vuelca el estado de la FPGA y del mapper emulados (solo emulador, para diagnóstico)
Comandos en minúsculas (r, s, m, ...)	Los comandos del depurador del firmware (ver «El depurador hardware»)

Atajos de menú: **Control → SD81 hardware debugger: pause** y **reload monitor and reset** envían los comandos DBG_ correspondientes. Pulsar **Arriba y Abajo** a la vez en el joystick también pausa el programa.

Mensajes

La consola también muestra los mensajes de estado de la MCU. Con el trazado activo (DBG_TRACE on) verás además una línea por cada comando recibido del Z81 (COMMAND=… y AFTER COMMAND=…), útil al informar de un problema.

El depurador hardware

El SD81 Booster tiene un depurador hardware: la FPGA detiene el programa en un límite de instrucción y llama a un pequeño programa monitor (/SYS/DEBUG.BIN) que reside en la página de memoria 63, y la MCU ofrece la línea de comandos por el puerto serie. El emulador lo implementa todo, con el código real del depurador del firmware, así que se comporta como la placa real. Los comandos, la pantalla del depurador y el botón QS se describen en el Apéndice L; aquí se explica cómo se manejan en el emulador.

Preparación

1. Coloca el fichero del monitor en **`SYS/DEBUG.BIN`** dentro de la carpeta SD (8 KB como máximo). Se carga en cada hard reset y la consola imprime Debug monitor loaded (/SYS/DEBUG.BIN, … bytes). Si el fichero no existe, el depurador queda desactivado.

2. Asegúrate de que la ROM de la SD es una versión que soporte el depurador.

3. Abre **Tools → SD81 Booster Serial Console…**.

Uso

1. Pausa el programa con p en la consola, con la opción de menú **Control → SD81 hardware debugger: pause**, con el botón QS (1–3 s) o con Arriba + Abajo del joystick.

2. Escribe comandos del depurador en la línea de entrada. La ayuda del propio firmware los lista todos; escribe ? para verla. En resumen:

Grupo	Comandos
Ejecución	p pausa, c continuar, s [n] paso, o paso por encima, u paso hasta salir, g dir ejecutar hasta una dirección
Breakpoints y watchpoints	b dir, bc, bl; w r|w|io dir
Inspección	r registros, d desensamblar, m volcar memoria, io puerto
Modificación	e dir bytes, x reg=val
Traza	t, th (historial)
Símbolos	sym fichero (pasmo .sym); las direcciones admiten símbolo o símbolo+n
Pantalla	ui muestra la pantalla propia del depurador en el ZX81 (texto 80 × 24); v muestra el vídeo del programa mientras está parado
Snapshots	snap [-a] [nombre] escribe un fichero .Z81

Los números son hexadecimales. Se admiten programas en modo **SLOW** y **FAST**.

Sesión típica

> > p detiene el programa en ejecución
>
> > r muestra los registros y la instrucción actual
>
> > b 8000 pone un breakpoint
>
> > c continúa; el programa se detiene en 8000
>
> > s 5 ejecuta cinco instrucciones
>
> > snap guarda un snapshot del programa detenido
>
> > c sigue adelante

Snapshots desde el depurador

snap escribe un fichero .Z81 en la tarjeta SD (el nombre por defecto es el del último programa cargado). Cárgalo con LOAD *Z81, con File → Load Snapshot... o en la placa real. Por defecto se guardan las páginas asignadas a algún bloque y las que el programa ha escrito desde el último reset; snap -a guarda las 64.

Línea de comandos y opciones del INI

Línea de comandos

Por línea de comandos y al arrastrar y soltar sobre la ventana, el emulador solo acepta ficheros .P.

Opciones del INI para arranques automáticos

En [MAIN] del fichero INI:

Clave	Significado
AutoType	Una secuencia de **teclas** que se teclea sola una vez arrancada la máquina, como si se escribiera en el teclado. \n es ENTER. Ejemplo: AutoType=J*CD”MIPROYECTO”\nR\n teclea LOAD *CD”MIPROYECTO” y luego RUN, lo que ejecuta AUTOEXEC.P de esa carpeta.

Son teclas, no texto: en un ZX81 LOAD es la tecla única J; los símbolos que necesitan SHIFT los gestiona el emulador. Está pensado para arranques rápidos y pruebas, no para el uso normal.

La carpeta SD se guarda en el INI con la clave SD81VirtualSD, y el estado del SD81 con la clave SD81.

Diferencias con el hardware real

El emulador reproduce la placa, pero algunas cosas son distintas por naturaleza:

- **La tarjeta SD es una carpeta.** No hay límites de formato FAT, la tarjeta nunca se “extrae” y los nombres de fichero siguen el sistema de ficheros del anfitrión (Windows no distingue mayúsculas).
- **Tiempos.** La MCU y la SD son instantáneas comparadas con lo real; los programas que dependen de la velocidad de la SD van más rápido. La temporización de VGM, PLAY y PEG sigue el reloj del anfitrión.
- **Los comandos de actualización de FPGA y firmware** (LOAD *FPGA, actualización del firmware) no tienen efecto en el emulador.
- **Hardware no presente:** el Wi-Fi y el servidor de ficheros, la batería del reloj de tiempo real (BAT) y los LEDs de la placa distintos del LED de estado.
- **La consola serie** sustituye al terminal serie USB-C; no hay velocidad ni puerto que configurar.
- El LED de estado y el botón QS se manejan con el ratón en la ventana de la consola; la pulsación corta o larga de la placa real se reproduce por el tiempo que mantienes pulsado el botón del ratón.

Resolución de problemas

Problema	Qué comprobar
La casilla SD81 Booster aparece atenuada	La máquina elegida no lo soporta. Elige ZX81, TS1000 o TS1500.
Pantalla negra / no arranca nada al activarlo	Falta SYS/SDBOOST.ROM en la carpeta SD o no es válida. Revisa la ruta en **Hardware → SD Card...**.

Problema	Qué comprobar
LOAD *Z81 termina con pantalla negra	Usa una SDBOOST.ROM reciente; con una ROM antigua el snapshot no se puede cargar a través del monitor del depurador. Cargar desde **File → Load Snapshot…** funciona siempre.
La consola dice “No debug monitor loaded”	Falta SYS/DEBUG.BIN o pesa más de 8 KB. Añádelo y haz **Hard Reset** (o DBG_RELOAD).
La consola no muestra nada	Pulsa **Clear** y envía un comando; comprueba que **Tools → SD81 Booster Serial Console…** es la ventana en la que escribes.
No se puede borrar un fichero desde Windows	El emulador lo tiene abierto (por ejemplo un VGM en reproducción). Detén la reproducción desde el programa o reinicia la máquina.
La pantalla sale mal tras cambiar de modo	Haz un **Hard Reset**. Si puedes reproducirlo, guarda un snapshot e infórmalo.
Falta el sonido de los chips AY	Revisa **Options → Sound…**, que la salida de sonido esté activada y que el SD81 esté activo (fuerza el AY ZonX81).

Para informar de un problema, activa el trazado con DBG_TRACE on, reprodúcelo y copia la salida de la consola.

Apéndice L — Depurador por hardware

El SD81 Booster incluye un depurador por hardware: puede parar cualquier programa en cualquier momento, enseñar sus registros y su memoria, ejecutarlo paso a paso, poner puntos de ruptura y de vigilancia, guardar el estado completo de la máquina en un snapshot y volver a cargarlo. El programa no tiene que estar preparado para ello.

Qué hace falta

El depurador es un pequeño monitor en código máquina, /SYS/DEBUG.BIN, que el interface carga al encender en una página de la RAM extendida reservada para él (la 63). Si el fichero no está en la SD, el depurador no está disponible y todo lo demás funciona igual. El firmware del MCU, la ROM y la FPGA tienen que ser de la misma versión del paquete.

Se maneja de cuatro formas: con el botón QuickSilva y el teclado del ZX81, sin nada más; desde la consola USB (ver 15.4); desde el navegador, con el módulo WiFi; y desde el propio programa, con la trampa OUT.

El botón QuickSilva

Pulsación	Qué hace
Corta (menos de 1 s)	Alterna QuickSilva, como siempre.
De 1 a 3 s (el LED se pone magenta)	Pausa: el programa se para y aparece la pantalla del depurador. Con el programa parado, otra pulsación de 1 a 3 s lo hace seguir.
Más de 3 s (el LED se pone amarillo)	Snapshot: guarda el estado en un fichero .Z81 y el programa sigue.

Mientras el programa está parado, el LED STAT está en magenta. Un programa en modo SLOW se para al llegar a su siguiente interrupción de vídeo.

La pantalla del depurador

Al pausar con el botón aparece, en el propio ZX81, una pantalla de 80 columnas con los registros, los flags, los puntos de ruptura, el desensamblado alrededor del PC (resaltado), la pila y un volcado de memoria. Se maneja con el teclado del ZX81 y cada tecla actúa al soltarla. Las direcciones se teclean en hexadecimal: con 4 cifras se aceptan solas, con menos hay que pulsar ENTER, y cualquier otra tecla cancela.

Tecla	Acción
S	Un paso: ejecuta una instrucción.
O	Paso por encima: CALL, RST, DJNZ, LDIR y similares se ejecutan enteros.
U	Salir de la rutina actual (hasta la dirección de vuelta de la pila).
C / ESPACIO	Seguir.
G dirección	Ejecutar hasta esa dirección.
B dirección	Poner o quitar un punto de ruptura (B ENTER: en el PC).
W R/W/I dirección	Punto de vigilancia de lectura, escritura o E/S: para justo después del acceso (W ENTER lo quita).
R registro valor	Cambiar un registro: P (PC), S (SP), A (AF), B (BC), D (DE), H (HL), X (IX), Y (IY), I.
E dirección bytes ENTER	Escribir bytes en memoria (2 cifras cada uno).
D dirección	Desensamblar desde esa dirección (D ENTER: volver al PC).
M dirección	Volcado de memoria desde esa dirección. 5, 6, 7 y 8 lo mueven.
T número	Traza: ejecuta ese número de instrucciones de una en una (T ENTER: hasta un punto de ruptura). ESPACIO la para.
H	Cambia el desensamblado por las últimas instrucciones ejecutadas (ver Traza e historial).
V	Ver la pantalla del programa (V vuelve al depurador).
Z	Snapshot, sin seguir.
L	Cargar el último snapshot grabado en esta sesión.
Q	Quitar la pantalla del depurador (queda la del programa).

En la pantalla del programa (después de V o Q) el teclado sigue activo: S guarda un snapshot y sigue, Z guarda un snapshot, L carga el último, C o ESPACIO siguen y V vuelve al depurador. Una vez abierta, la pantalla del depurador aparece en cada parada (punto de ruptura, paso...) hasta quitarla desde la consola con ui.

Snapshots

Un snapshot guarda en la SD el estado completo de la máquina, en el formato .Z81 de EightyOne ampliado: la memoria, los registros, las páginas de la RAM extendida y su asignación, los POKEs de control, los colores Chroma81, los sprites, los chips de sonido y el estado del interface (directorio actual, ficheros abiertos, música VGM...).

Se graba en la carpeta actual con el nombre del último programa cargado y un número (MAZOGS001.Z81, MAZOGS002.Z81...), o NONAME001.Z81 si no se ha cargado ninguno. Se graba con el botón QS (más de 3 s), con Z en la pantalla del depurador (o S en la del programa, que además sigue) o desde la consola:

> snap [-a] [NOMBRE]

Por defecto se guardan las páginas de la RAM extendida que están asignadas a algún bloque y las que el programa ha escrito desde el último reset (la 63, la del depurador, nunca). Un LOAD no borra la cuenta, así que un programa que carga sus datos desde la SD en varias tandas conserva todo. Con -a se guardan las 64 páginas. Para volver a cargarlo, LOAD *Z81 “NOMBRE.Z81” (ver 7.7), el explorador de archivos o L en la pantalla del depurador.

La consola USB

Conectando el USB-C a un PC y abriendo un terminal serie (ver 15.4), todo lo anterior está disponible como órdenes de texto; h muestra la lista completa. Los números van en hexadecimal y una dirección también puede ser un símbolo. Las principales:

Orden	Qué hace
p / c	Pausa / seguir.
s [n]	n pasos (1).
o / u	Paso por encima / salir de la rutina.
g dir	Ejecutar hasta dir.
b dir / bc [dir] / bl	Poner / quitar uno o todos / listar los puntos de ruptura.
w r|w|io dir / w	Poner / quitar el punto de vigilancia.
r / x reg=valor	Ver los registros / cambiar uno.
d [dir] [n] / m dir [n]	Desensamblar / volcado de memoria.
e dir bytes / io puerto [valor]	Escribir en memoria / leer o escribir un puerto.
snap [-a] [nombre]	Snapshot.
t [n] / th [n]	Traza lenta / ver las últimas instrucciones.
tron / troff	Historial encendido / apagado.
sym [fichero | ! | -]	Cargar / releer / quitar los símbolos.
ui	Pantalla del depurador en el ZX81, sí / no.
v / v dir	Con el programa parado, ver su pantalla en Superfast texto / su mapa de bits WRX en dir.

Desde el navegador

Con el módulo WiFi (ver 9.7), la página http://dirección-IP/debug, enlazada desde el servidor de ficheros como Debugger, muestra el depurador en paneles: desensamblado (clic en el margen para poner o quitar un punto de ruptura, doble clic en una línea para ejecutar hasta ella), registros y flags (clic para cambiar un registro), puntos de ruptura y de vigilancia, pila, volcado de memoria (clic en un byte para cambiarlo) y una consola con las mismas órdenes que la USB. Hace falta actualizar a la vez el firmware del MCU y el del módulo WiFi. Debajo de los paneles, Captura de pantalla (el botón Capture) dibuja en el navegador la imagen que el ZX81 está mostrando: para el programa un instante (del orden de décimas de segundo, según el modo), lee de la memoria de vídeo lo necesario y lo deja seguir. Es una foto, no vídeo: la velocidad del enlace no da para más. Dibuja el ZX81 nativo, el texto Superfast de 32, 70 y 80 columnas,

HiRes y Spectrum, con el color de Chroma81, el scroll fino y los sprites (con su límite de 12 por línea, salvo en 70 y 80 columnas, donde los sprites no están implementados). Con el programa ya parado, la captura no lo hace seguir. Los modos que no reconoce, como WRX con la fuente en I, no los dibuja. El borde es el del Chroma81; con el modo Spectrum no se conoce y se dibuja igual. La orden cap de la consola hace la captura sin web (solo para probar). Si el programa usa un D_FILE alternativo (POKE 2096-2098), la captura trae también el del sistema y la casilla D_FILE alternativo elige cuál se dibuja con una FPGA anterior: el interface no puede saber en ese caso si la FPGA tiene activo el alternativo (un POKE 2045,85 lo apaga y volver a 170 no lo enciende), así que si la imagen sale mezclada hay que probar la otra opción. La casilla sprites los muestra u oculta; sin marcar antes de capturar, no se leen y la captura es más rápida. La línea de información de la captura dice si el estado del vídeo (modo, D_FILE y atributos alternativos, doble búfer) viene de la FPGA, que lo informa desde la versión de este paquete, o se ha deducido de los POKEs; si viene de la FPGA, la casilla queda desactivada porque no hay nada que elegir. Debajo de la captura hay un teclado del ZX81: haz clic en sus teclas (SHIFT se queda pulsado hasta la siguiente tecla) o marca «use my keyboard» para que las teclas de tu ordenador vayan al ZX81. Letras, cifras, ENTER, el punto y la barra espaciadora van directas; los símbolos se mandan con el SHIFT que les toque en el ZX81 (por ejemplo, las comillas son SHIFT+P), Retroceso es RUBOUT (SHIFT+0), las flechas son SHIFT+5 a SHIFT+8 y Escape es BREAK (SHIFT+SPACE). Las teclas llegan al ZX81 como si estuvieran pulsadas de verdad, también para la pantalla del depurador. Si la página deja de dar señales (se cierra, se pierde la conexión), el interface suelta todas las teclas al cabo de 1,5 segundos. Hace falta actualizar a la vez la FPGA, el firmware del MCU y el del módulo WiFi.

Abre la página con la IP del módulo (LOAD THEN PRINT “*IP”) mejor que con el nombre sd81booster.local: en algunos ordenadores el navegador tarda de 10 a 30 segundos en resolver el nombre en cada operación y las órdenes parecen lentas.

Las órdenes que necesitan el programa parado (puntos de ruptura, registros, memoria, pasos) avisan con una línea roja arriba si el programa está en marcha: pulsa Pause antes. En la consola de la página, History es th 20, Trace 100 ejecuta 100 instrucciones en traza lenta y Symbols relee el fichero de símbolos (ver Símbolos).

Símbolos

Si junto al programa hay un fichero de símbolos con su mismo nombre y la extensión .SYM (MIJUEGO.P y MIJUEGO.SYM), se lee al cargar el programa: el desensamblado muestra las etiquetas y las direcciones se pueden escribir por su nombre (b BUCLE, g INICIO+10). Es el fichero que genera el ensamblador pasmo con un tercer argumento:

```
pasmo mijuego.asm mijuego.bin MIJUEGO.SYM
```

También se puede cargar a mano con sym FICHERO.SYM. Los símbolos se leen al cargar el programa y no se vuelven a leer solos: si vuelves a ensamblar y cambias el .SYM sin volver a cargar el programa, el depurador sigue con el anterior y las etiquetas salen en direcciones que ya no son las del programa. sym ! lo relee, y sym solo dice cuántos símbolos hay y de qué fichero. En la página web, el botón Symbols hace lo mismo y, con un nombre escrito en la caja de al lado, carga ese fichero.

Traza e historial

El interface apunta continuamente la dirección de cada instrucción que ejecuta el programa, sin frenarlo: al parar por lo que sea, th 20 en la consola (o H en la pantalla del depurador) enseña las últimas instrucciones que llevaron hasta ahí. Se apaga y se enciende con troff y tron.

La traza lenta (t n en la consola, o T en la pantalla) ejecuta n instrucciones de una en una y apunta además los registros de antes de cada una, que th enseña. Va a unas 200 instrucciones por segundo.

Para programadores: la trampa OUT

Un programa puede pararse a sí mismo en un punto concreto escribiendo 16 en el puerto $3FEF; si el depurador no está cargado, no pasa nada:

```
LD BC,$3FEF
LD A,16
OUT (C),A
```

Limitaciones

- La página 63 de la RAM extendida y el puerto $3FEF están reservados para el depurador.
- Un punto de ruptura en código que se carga después (o que el programa sobrescribe) desaparece. Para esos casos, G o un punto de vigilancia de escritura.
- Si el programa se para en mitad de una operación con la SD (un LOAD, por ejemplo), conviene dejarlo seguir antes de usar otras funciones del depurador.
- Parado, un programa en SLOW o en FAST no tiene imagen propia: V la muestra en Superfast texto. Los programas que generan su propia imagen (WRX, pseudo alta resolución) pueden no verse bien.
- Con CP/M en marcha (bloque 0 en RAM) los sprites y los POKEs de control están apagados; el resto del depurador funciona igual (ver el Apéndice I).

Apéndice M — Red para programadores: MQTT y juegos por turnos

El módulo WiFi (9.7) incluye un cliente MQTT, resumen criptográfico y números aleatorios, todo con comandos AT. Se escriben en el terminal TELNET (9.8) o los manda un programa propio por el puente de red del interface (comandos NET_READ y NET_WRITE del MCU, ver la sección 15). Encima, el Z80 tiene una librería de salas y mensajes firmados (net.inc) y un controlador de partidas por turnos (game.inc), que son los que usan los juegos de 9.9. Los fuentes están en EXAMPLES/NETCHAT y EXAMPLES/NETGAME del repositorio.

Comandos MQTT

Comando	Qué hace
AT+MQTT=host,puerto[,tls[,usuario,clave]]	Conecta a ese servidor. tls vale 0 o 1. Bloquea hasta unos 8 segundos si el servidor no contesta.
AT+MQTT	Sin argumentos: prueba por orden los servidores de /SYS/NET.CFG, recuerda el último que funcionó y pasa al siguiente si uno falla tres veces seguidas.
AT+MQTTID=cadena	Fija el identificador del cliente (por defecto, sd81- y 8 caracteres hexadecimales al azar).
AT+MQTTWILL=topic,retain,texto	Mensaje de última voluntad: se publica si la conexión se corta sin despedirse. Se manda antes de conectar.
AT+MQTTSUB=topic[,qos]	Se suscribe a un topic (admite los comodines + y #). Espera la confirmación hasta 3 segundos.
AT+MQTTUNSUB=topic	Cancela la suscripción.
AT+MQTTPUB=topic,qos,retain,len	Publica un mensaje: el módem contesta > y después se envían len bytes tal cual (cualquier valor, hasta 120). Con len 0 publica un mensaje vacío, que borra el retenido. Con qos 1 espera la confirmación 3 segundos y reintenta una vez.

Comando	Qué hace
AT+MQTTRX	Saca un mensaje de la cola: +MQTTRX:topic,len, un salto de línea y los len bytes; o solo OK si no hay ninguno.
AT+MQTT?	Estado: +MQTT:estado,servidor,en_cola,perdidos,error.
AT+MQTTCLOSE	Cierra la sesión MQTT. ATZ cierra la sesión MQTT y el socket TCP y olvida todo.

- Los datos van por longitud, en binario: el módem no interpreta los bytes que siguen al >.
- La recepción es por sondeo (AT+MQTTRX), nunca con mensajes que aparezcan solos. El módulo guarda hasta 8 mensajes de 128 bytes; si se llena descarta el más antiguo y lo cuenta en perdidos. Un mensaje de más de 120 bytes o con un topic de más de 64 se descarta.
- Hay una sola sesión MQTT, independiente del socket TCP de ATDT. Se reconecta sola, con esperas de 2, 4, 8, 16 y 30 segundos, y vuelve a suscribirse a lo que estuviera suscrita.

Resumen criptográfico y números aleatorios

Comando	Qué hace
AT+SHA256=len	El módem contesta > y se envían len bytes. Devuelve +SHA256: y 64 caracteres hexadecimales.
AT+HMAC=lenclave,lendatos	Después del >, la clave y los datos seguidos. Devuelve +HMAC: y 64 caracteres hexadecimales (HMAC-SHA256).
AT+RAND=n	Devuelve +RAND: y n bytes al azar en hexadecimal (n hasta 32), del generador del módulo.

Los servidores: /SYS/NET.CFG

Un archivo de texto en la carpeta SYS de la SD, con un servidor por línea (hasta 6): broker, el nombre, el puerto y, si hace falta TLS, la palabra tls. Las líneas vacías, las que empiezan por # y las mal formadas se ignoran. Sin el archivo (o sin ninguna línea válida) se usan estos tres, sin TLS:

> broker test.mosquitto.org 1883
>
> broker broker.emqx.io 1883
>
> broker broker.hivemq.com 1883

Son servidores públicos de prueba, sin garantía de servicio. Los comandos AT+MQTT con argumentos ignoran el archivo.

Estados y errores

Valor	Significado
estado 0 / 1 / 2 / 3	apagado / conectando / conectado / reconectando
error 0	ninguno (es el último error: se borra al conectar con éxito)
error 1	sin WiFi
error 2	no abre el socket TCP
error 3	el servidor rechaza la conexión

Valor	Significado
error 4	tiempo agotado
error 5	argumentos malos
error 6	cola o búfer lleno
error 7	sin sesión MQTT
error 8	sin confirmación de un mensaje con qos 1

Salas y mensajes de los juegos

Los topics de un servidor público los puede leer cualquiera, así que el código de la sala no va en el topic. El código son 10 caracteres de un alfabeto de 32 (50 bits), sacados de AT+RAND. El topic es sd81/ más los 12 primeros hexadecimales de SHA256("topic"+código), y la clave es SHA256("key"+código) en hexadecimal, usada tal cual como clave del HMAC. Quien ve el topic no obtiene el código.

Cada jugador publica, retenido y con qos 1, en <topic>/0 (quien creó la sala) o <topic>/1 (quien se unió). El mensaje es: versión (1 byte), secuencia (2, empieza en un valor al azar y sube por emisor), turno (2), datos (0 a 100) y los 8 primeros bytes del HMAC de todo lo anterior. El receptor descarta lo que no verifica, lo de su propia ranura y la secuencia repetida.

La presencia va en <topic>/p0 y <topic>/p1, retenida: 1 al entrar y 0 al salir; el mensaje de última voluntad publica el 0 si la conexión se corta sin despedirse (el servidor tarda unos 90 segundos en notarlo). Antes de entrar, la librería lee las dos presencias: así sabe si la sala está llena, si no existe o si hay que volver a ocupar un sitio.

Esto protege de travesuras, no de un atacante decidido: 50 bits se pueden romper sin conexión, y los datos van en claro.

La librería net.inc y el controlador game.inc

net.inc (EXAMPLES/NETCHAT) se incluye en el programa y necesita una RAM de trabajo de unos 1,2 KB, cuya dirección se define con NB_BASE. Rutinas: net_init, room_new, room_join, room_send, room_poll, room_leave y, para un bucle de juego, room_poll_send y room_poll_recv (el sondeo en dos pasos, que deja pintar entre medias). Devuelven un código en A: 0 bien, 1 el módem dijo ERROR, 2 plazo agotado, 3 respuesta inesperada, 4 código inválido, 5 mensaje demasiado largo, 6 sala llena, 7 la sala no existe, 8 la sala existió pero está vacía (hay que elegir ranura con room_join_as).

game.inc (EXAMPLES/NETGAME) es el controlador de una partida por turnos: la jugada, el turno, el hash del estado, el estado completo, la otra partida y la reconexión. No sabe nada de las reglas: el juego define seis rutinas, G_STATE_LEN (el tamaño del estado) y G_MOVE_LEN (1 o 2 bytes por jugada). Mueve el jugador 1 con un número de jugada par y el 2 con uno impar.

Rutina	Qué hace
g_reset	Deja la posición inicial.
g_valid	HL = la jugada, B = el jugador: devuelve CF = 1 si no es legal.
g_apply	HL = la jugada, B = el jugador: la hace.
g_winner	A = 0 sigue, 1 o 2 gana ese jugador, 3 tablas.
g_save / g_load	HL = destino / origen: los G_STATE_LEN bytes del estado.

Los mensajes del controlador son: 1 jugada (con el hash del estado resultante), 2 pido el estado, 3 estado completo y 4 otra partida; cualquier desfase se arregla pidiendo el estado completo. El fuente de connect4.asm es el ejemplo más corto, y chess.asm el más completo (HiRes, relojes).

Probarlo a mano

Con el terminal TELNET (9.8), escribe estas líneas en dos interfaces, o en uno solo y publica con un cliente MQTT de escritorio (por ejemplo mosquitto_pub):

```
AT+MQTT=test.mosquitto.org,1883

AT+MQTTSUB=sd81test/hola,1

AT+MQTTPUB=sd81test/hola,1,0,5

>hello

AT+MQTTRX

AT+MQTT?
```

Manual de Usuario SD81 Booster v1.0 — Hardware y software de código abierto

Fuentes de la revisión

- MANUAL/ES/SD81_Manual_ES.md
- release_notes_v1.7.0.md
- Arduino/bootloader/bootloader.ino
- Arduino/SD81BoosterV2_039_STM32/SD81BoosterV2_039_STM32.ino
- Arduino/SD81BoosterV2_039_STM32/WIFI_HANDLER.cpp
- Arduino/Wifi_module_01/Wifi_module_01.ino
- TOOLS/build_release.py