

Especificación funcionamiento del superupgrade

Características del interface

Las características del interface son las siguientes:

- Ampliación a 128/512Kb de RAM seleccionable por jumpers
- 2 conexiones de joystick multinorma (Kempston, Sinclair, Cursor)
- Botón de Reset
- Botón de NMI gestionable por hardware
- Memoria flash de 512Kb conteniendo hasta 32 ROMs de diferentes sistemas o juegos.
- Chip de sonido AY.
- Entrada de audio para mezclar el sonido del speaker con el del AY.
- Salida de video compuesto.
- Expansor de bus.
- Interface simple de 8 bits (en desarrollo)
- Compatible con divide
- Selección de la ROM activa por software o por microswitches.
- Conexión para interface de floppy multisistema incorporada.
- Grabación de la memoria flash desde el propio Spectrum.

La finalidad del interface es convertir un gomas en otras máquinas más avanzadas como un +3 o un 128K, permitiendo ejecutar sus ROMs y funcionar tal cual fueran estos, pudiendo arrancar en una máquina u otra desde un menú de opciones, o bien fijar la máquina desde la que quieres arrancar mediante microswitches.

Para emularlo habría que partir del modo de funcionamiento "Spectrum 16K", es decir, la máquina debe tener el mismo modelo de contienda que éste y el bus flotante se debe comportar de la misma manera que en un gomas.

Modelo de paginado

- 512Kb de RAM. Con paginación compatible con los modos del +3 y del pentagon 512, con las siguientes observaciones:
 - No funcionan los modos de allram.
 - No funciona la shadowram, siendo la página 7 una página más.
 - Funcionan correctamente las duplicidades tanto de la página 2 como de la página 5.
- 512Kb de ROM. Con paginación compatible con la del +3, y con un modo de paginación adicional basada en el puerto \$43B.

Puerto \$7FFD

	Descripción
D0	Bits 0..2 de la página de RAM seleccionada
D1	
D2	
D3	No usado
D4	Bit bajo de la selección de la página de ROM
D5	Deshabilitar paginado
D6	Bits 3 y 4 de la página de RAM seleccionada si el switch de habilitación de los modos Pentagon 512 está activo
D7	

La decodificación del puerto se hace de la misma forma que en un +3, es decir 01xx xxxx xxxx xx0x

Puerto \$1FFD

La decodificación del puerto se hace de la misma forma que en un +3, es decir 0001 xxxx xxxx xx0x

El puerto extra de paginado es el \$43B. Este puerto no se puede deshabilitar con el bit 5 del puerto \$7FFD y nos permite seleccionar cualquiera de las 32 ROMs existentes en la memoria flash.

	Descripción
D0	No usado
D1	No usado
D2	Bit alto de la selección de la página de ROM
D3	Motor del disco
D4	No usado
D5	No usado
D6	No usado
D7	No usado

La decodificación del puerto se hace de la siguiente forma: 0xxx xxxx 0x1x xx11

Todos los bits de este puerto se asignan a 0 durante el reset a excepción del D7 que se asignará a 1.

Existe un switch que permite elegir un modo manual para seleccionar la ROM activa.

- Interface simple de 8 bits.

Es un interface IDE que permite utilizar dispositivos IDE pero aprovechando solo 8 de los 16 bits disponibles en el bus IDE, por lo que solo está aprovechable la mitad de la capacidad del disco/CF.

Hay información sobre este interface aquí: <http://piters.tripod.com/simpif.htm>

- Compatibilidad con interface de disquetera.

Permite conectar un interface de disquetera compatible con con el del +3. Usando los puertos \$2FFD y \$3FFD al igual que este.

Cuando no está conectado dicho interface la consulta de los dos puertos devuelve \$FF.

- Gestión hardware de NMI's.

Si se pulsa el botón de NMI del interface automáticamente cambia a la ROM 0 y desactiva cualquier NMI entrante desde el bus de expansión. El hardware retorna a la página desde donde fue llamado cuando accedemos a la posición \$3FFF, en donde deberá haber un retn.

Esto último está todavía en desarrollo.

Grabación de la memoria flash

El interface superupgrade incorpora un chip de memoria flash 29C040 que puede ser programado desde el propio Spectrum.

El protocolo de protección/desprotección, grabación y borrado del chip puede ser consultado en el datasheet de este: http://www.datasheetcatalog.net/es/datasheets_pdf/2/9/C/0/29C040.shtml

Se trata de una memoria flash de 512Kb basado en sectores de 256 bytes.

Las escrituras se realizan en dos fases:

Fase 1: Almacenamiento en el registro interno.

Cuando escribimos en cualquier posición de la ROM, el chip empezará a almacenar todos los datos que escribamos en el registro interno del sector de 256 bytes que le corresponda a la dirección donde hemos escrito, poniendo inicialmente todas las posiciones de ese registro intermedio a \$FF. Todas las escrituras durante esta fase deberán hacerse al mismo sector, ignorándose las escrituras que se hagan a cualquier otro sector (esto último no lo acabo de tener claro, ya que en el datasheet lo único que dice es que deben hacerse todas las escrituras al mismo sector, pero no dice lo que pasa si escribes en otro)

Después de esto podemos ir escribiendo aleatoriamente en cualquier posición de dicho sector y el chip irá almacenando los datos en el registro intermedio.

Entre cada escritura de cada byte individual puede pasar un máximo de 200 useg, lo que corresponde a 532 T-states en memoria contenida y 710 T-states en memoria no contenida.

Si dejamos de escribir en ese sector durante más de esos T-states el chip pasará automáticamente a la segunda fase de escritura interna.

Fase 2:

Se escribirán físicamente los datos en el chip, determinándose el final de esta fase en el momento que el contenido del último byte escrito sea igual al byte que intentábamos escribir. Esta fase tiene una duración de 10mseg, lo que corresponde a 26600 T-states en memoria contenida y 35500 T-states en memoria no contenida. Los datos que no hayan sido escritos dentro de ese bloque de 256 bytes se escribirán como \$FF. Se podrá considerar terminada la escritura cuando se pueda leer el valor que intentábamos escribir en la última posición del sector.

El chip tiene dos formas de trabajo en lo que a escritura se refiere, modo protegido o modo desprotegido.

La gestión de protección/desprotección o escritura con el chip protegido se realiza mediante comandos, que consisten en secuencias de escrituras en determinadas posiciones de la memoria flash. Por lo

anterior, una secuencia de escrituras identificada como un comando no pasará a la fase de escritura física.

En el modo protegido hay que enviar una secuencia concreta antes de enviar una secuencia de escritura, en caso contrario se ignorará.

En el modo desprotegido funcionaremos escribiendo directamente sin interponer ningún comando.

La secuencia para activar el modo protegido es la siguiente:

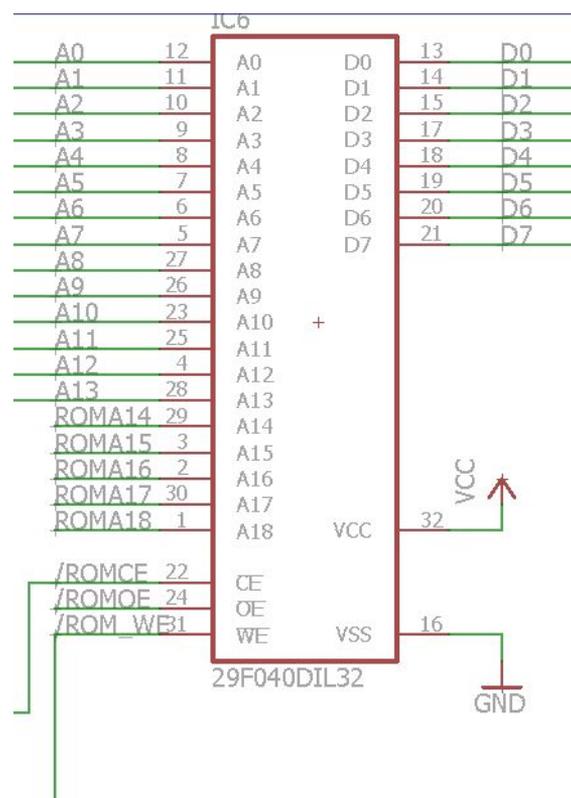
Escribir AA en la dirección \$5555
 Escribir 55 en la dirección \$2AAA
 Escribir A0 en la dirección \$5555
 escribir un sector de 256 bytes de datos.

Una vez protegido el chip, si queremos escribir tenemos que interponer de nuevo la misma secuencia de escritura anterior (sin leer los 256 bytes) y después escribir el sector que queremos

Para desactivar el modo protegido:

Escribir AA en la dirección \$5555
 Escribir 55 en la dirección \$2AAA
 Escribir 80 en la dirección \$5555
 Escribir AA en la dirección \$5555
 Escribir 55 en la dirección \$2AAA
 Escribir 20 en la dirección \$5555
 Escribir un sector de 256 bytes de datos.

Esquema de conexión de la memoria flash en el superupgrade



En la imagen se puede ver como está conectada la memoria flash al sistema.

Como la ROM del spectrum solo ocupa el espacio de direcciones que va desde \$0000 hasta \$4000, solo son necesarias las líneas de direcciones A0..A13 que van conectadas directamente a las líneas de direcciones del spectrum, de tal forma que uno solo de los 32 slots de 16KB que caben en el chip es accesible simultáneamente.

Es decir, el Superupgrade divide el chip en 32 slots de 16Kb.

Por un lado el Z80 percibe cualquier acceso a ROM el direccionamiento mediante un valor entre 0 y \$3FFF, debiendo utilizar los puertos de cambio de ROM para acceder a unas zonas u otras de la memoria flash.

Sin embargo, para el chip de flash no existen los slots, solo son un artificio creado por el superupgrade para poder aprovechar los 512Kb del chip. Desde el punto de vista del chip cualquier acceso a ROM consta de una dirección de 19 bits (A18..A0). Es el

superupgrade el que, al detectar un acceso a ROM (dirección 0..\$3FFF) la traduce ese acceso a una dirección de 19 bits que pueda entender el chip.

La selección del slot se realiza con las líneas ROMA18..ROMA14 que van conectadas a las líneas A18..A14 de la memoria flash por un lado y a la CPLD por el otro. Cuando la CPLD recibe una escritura a uno de los puertos de selección de página/slot (\$43B/\$7FFD/\$1FFD) activa o desactiva las línea ROMA18..ROMA14 según corresponda.

Ejemplo: Si tenemos seleccionado el slot 5, las líneas ROMA18..ROMA14 tendrían el estado 00101b=\$5, por lo tanto un acceso desde el Z80 en la dirección \$1000=1000000000000b (que está en ROM) acabaría accediendo a la posición \$B000=1011000000000000b de la memoria flash. Sin embargo un acceso a la dirección \$B000 el superupgrade la interpretará como un acceso a RAM, por lo que no activará el chip.

Los comandos de memoria la flash empiezan todos escribiendo en la dirección \$5555 del espacio de direcciones de la memoria flash (no confundir con el espacio de direcciones del Z80). Si desde el z80 escribimos directamente en la dirección \$5555 activaríamos las líneas del bus de direcciones correspondientes a ese valor, con lo que estaríamos accediendo a una posición de la memoria RAM y el chip de memoria flash no se enteraría.

Por esto último, para acceder a la dirección \$5555 deberíamos combinar una selección de slot (slot 1=00001b) con un acceso a la dirección de la ROM \$1555 lo que el superupgrade acabará traduciendo como un acceso a la dirección \$5555 de la memoria flash

Para acceder a la dirección \$2AAA del chip habría que cambiar al slot 0 y escribir en la dirección \$2AAA.

Ejemplo, para copiar 256 bytes desde \$C000 al slot 3 dirección \$400 (que corresponde con un sector completo), teniendo la memoria protegida deberemos hacer lo siguiente:

Para escribir AAH en la dirección \$5555 (101010101010101) del chip, deberemos cambiar al slot 1 y luego escribir AAH en la dirección 1555H.

```
ld a,1
ld bc,$43b
out ($43b),a
ld hl,$1555
ld (hl),$AA
```

Para escribir 55H en la dirección 2AAAH (010101010101010) del chip, deberemos cambiar al slot 0 y luego escribir 55H en la dirección \$2AAA.

```
ld a,0
ld bc,$43b
out ($43b),a
ld hl,$2AAA
ld (hl),$55
```

Para escribir A0H en la dirección \$5555 (101010101010101) del chip, deberemos cambiar al slot 1 y luego escribir A0H en la dirección 1555H.

```
ld a,1
ld bc,$43b
out ($43b),a
ld hl,$1555
```

```
ld (hl), $A0
```

Y por último deberemos cambiar al slot 3 y allí escribir los 256 bytes en la dirección \$400.

```
ld a, 3
ld bc, $43b
out ($43b), a
ld hl, $C000
ld de, $0400
ld bc, $0100
ldir
```

Y luego esperaremos a que el último byte (\$499) sea igual a \$C100, o bien esperaremos 10mS, lo que corresponde a 26600 T-states en memoria contenida y 35500 T-states en memoria no contenida.

Algoritmo de escritura para emuladores.

Seguidamente expongo el algoritmo de funcionamiento de la gestión de la escritura en memoria flash.

Time1 = 200 useg. (532 T-states en memoria contenida y 710 T-states en memoria no contenida)
Time2 = 10mS (26600 T-states en memoria contenida y 35500 T-states en memoria no contenida)

```
function FlushInternalRegister()
begin
    for i = 0 to 255
        Memory[ActiveRegister*256+i] = InternalRegister[i]
    for i = 0 to 255
        InternalRegister[i] = $FF
    ActiveRegister = -1
end

function Escribe (Addr, Data)
begin
    if Not Protegido or (Estado = P3) then
        begin
            if ActiveRegister = -1 then
                begin
                    ActiveRegister = Addr DIV 256
                end
            InternalRegister[Addr MOD 256] = Data
        end
end

function flushCola()
begin
    for i = 0 to lenCola-1
        Escribe(Cola[i].Addr, Cola[i].Data)
    LenCola = 0
end

function DropCola()
begin
    LenCola = 0
end

function AddCola(Addr, Data)
begin
    Cola[LenCola] = (Addr, Data)
```

```

LenCola = LenCola + 1
end
Main ()
DropCola()
Estado = E0
ActiveRegister = -1
Do
    if Time > Timeout2 then
    begin
        FlushInternalRegister()
    end
    if Time > Timeout1 then
    begin
        FlushCola()
        Estado = E0
    end
    If Acceso_de_Escritura then
    begin
        AddCola(Addr, Data)
        case Estado of
        E0:
            If (slot = $0001) and (Addr = $1555) and (Data=$AA) Then
            begin
                Estado = P1
            end else begin
                FlushCola()
                Estado = E0
            end
        P1:
            If (slot = $0000) and (Addr = $2AAA) and (Data=$55) Then
            begin
                Estado = P2
            end else begin
                FlushCola()
                Estado = E0
            end
        P2:
            If (slot = $0001) and (Addr = $1555) and (Data=$A0) Then
            begin
                Estado = P3
                Protegido = 1
                DropCola()
            end else If (slot = $0001) and (Addr = $1555) and (Data=$80) Then
            begin
                Estado = U3
            end else begin
                FlushCola()
                Estado = E0
            end
        P3: //Escritura de datos protegida
            begin
                Escribe(Addr, Data)
            end else begin
                Estado = E0
            end
        U3:
            If (slot = $0001) and (Addr = $1555) and (Data=$AA) Then
            begin
                Estado = U4
            end else begin
                Flush(Cola)
                Estado = E0
            end
        U4:
            If (slot = $0000) and (Addr = $2AAA) and (Data=$55) Then
            begin
                Estado = U5
            end else
            begin
                Flush(Cola)

```

```
                Estado = E0
            end
U5:            If (slot = $0001) and (Addr = $1555) and (Data=$20) Then
                begin
                    Protegido = 0
                    Estado = E0
                end else
                begin
                    Flush(Cola)
                    Estado = E0
                end
            end case
        end
    end
end
```