Vircon32

CONSOLA VIRTUAL DE 32 BITS

Especificación del sistema

Parte 6: Chips controladores

Documento con fecha 2024.01.07 Escrito por Carra

¿Qué es esto?

Este documento es la parte número 6 de la especificación del sistema Vircon32. Esta serie de documentos define el sistema Vircon32, y provee una especificación completa que describe en detalle sus características y comportamiento.

El principal objetivo de esta especificación es definir un estándar de lo que es un sistema Vircon32, y cómo debe implementarse un sistema de juego para que se considere conforme a él. Además, al ser Vircon32 es un sistema virtual, un importante objetivo adicional de estos documentos es proporcionar a cualquiera el conocimiento para crear sus propias implementaciones de Vircon32.

Sobre Vircon32

El proyecto Vircon32 fue creado de forma independiente por Carra. El sistema Vircon32 y su material asociado (incluyendo documentos, software, código fuente, arte y cualquier otro elemento relacionado) son propiedad del autor original.

Vircon32 es un proyecto libre y de código abierto en un esfuerzo por promover que cualquiera pueda jugar a la consola y crear software para ella. Para obtener información más detallada al respecto, se recomienda consultar los textos de licencia incluidos en cada uno de los programas disponibles.

Sobre este documento

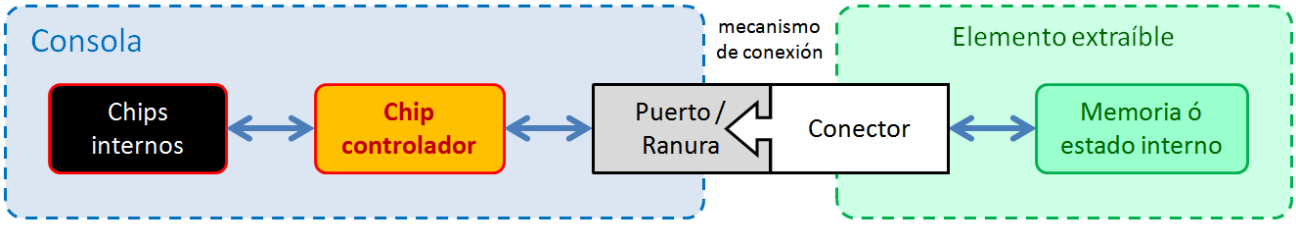
Índice

La parte 6 de la especificación define los 3 chips controladores de la consola. Para cada uno este documento describirá su comportamiento, sus puertos de control, sus variables internas y el proceso general de su funcionamiento.

1 Sobre los chips controladores

Una consola Vircon32 necesitará interactuar con algunos elementos externos (mandos, cartuchos y tarjetas de memoria) que son extraíbles, y pueden estar presentes o ausentes en un momento dado. Por eso no puede haber conexiones directas desde los componentes internos de la consola a estos elementos extraíbles. En su lugar, es necesario disponer de chips "controladores" que actúen como proxy en sus comunicaciones.

Los elementos extraíbles se conectan a la consola mediante algún tipo de mecanismo lógico o físico que permite a la consola determinar cuándo están presentes.



La función básica de los chips controladores es proporcionar una forma segura para la CPU de comunicarse con esos posibles elementos externos, permitiéndole:

1. Determinar cuándo está presente un determinado elemento externo.
2. Recibir una respuesta de error al intentar acceder a un elemento ausente.
3. Acceder a la memoria o estado del elemento cuando está presente.

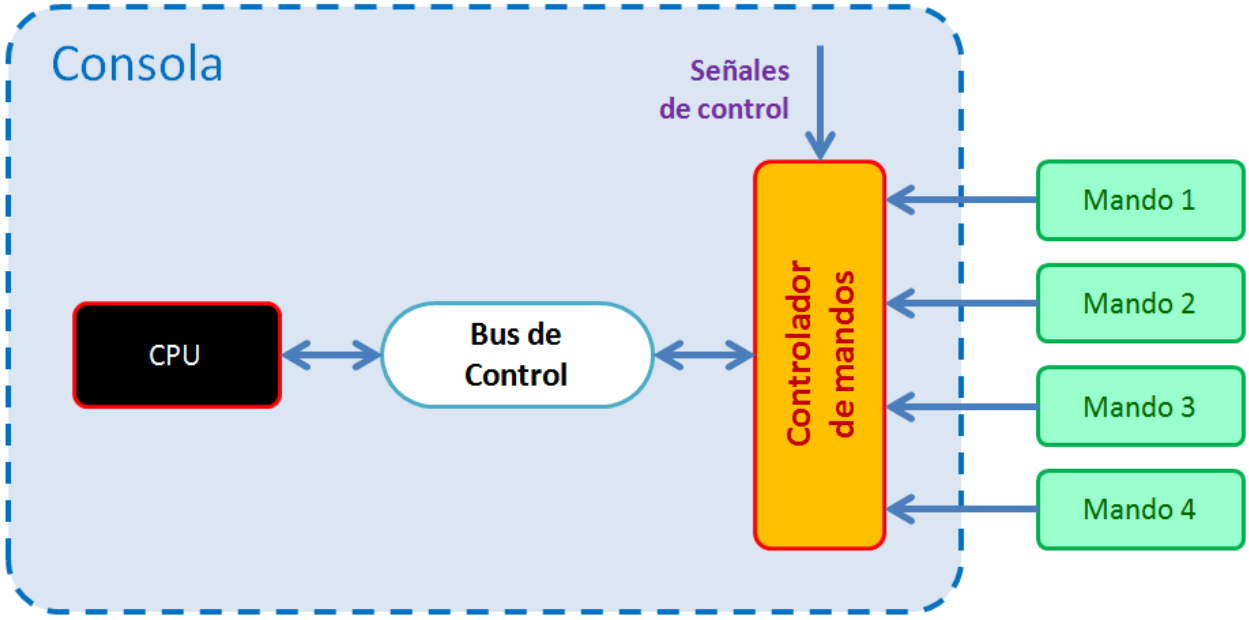
Existen 3 chips controladores: Controlador de mandos, Controlador de cartucho y Controlador de tarjeta. Las siguientes secciones describirán cada uno de estos chips en detalle. Para agrupar los 3 chips en este único documento, a diferencia de los documentos anteriores, cada sección de un chip en particular se presentará aquí como una subsección.

2 Controlador de mandos

El controlador de mandos es el chip encargado de controlar los 4 puertos de mandos de la consola. Todos los mandos de Vircon32 son idénticos, pero los puertos de mando están numerados. Por ejemplo: aunque sólo haya un mando conectado, se tratará como el mando 2 si se conecta al segundo puerto. Esto da como resultado un total de 16 combinaciones (2^4).

2.1 Conexiones externas

Como el controlador de mandos es sólo uno de los chips que forman la consola, no puede funcionar aislado. La figura muestra todas sus comunicaciones con otros componentes. Como se vio, las 4 conexiones a los mandos están numeradas y no son intercambiables.



Cada una de estas conexiones se explicará por separado en las secciones siguientes.

2.1.1 Señales de control

Como todos los componentes de la consola, el controlador de mandos recibe las señales de reset, nuevo frame y nuevo ciclo. Las respuestas a esas señales se detallan en el apartado 2.6 de este documento.

2.1.2 Bus de Control

El controlador de mandos está conectado como dispositivo esclavo al Bus de Control, con ID de dispositivo = 4. Esto permite al maestro del bus (la CPU) solicitar operaciones de lectura o escritura en los puertos de control expuestos por el controlador de mandos. La lista de sus puertos y sus propiedades se detallarán en secciones posteriores.

2.1.3 Mandos

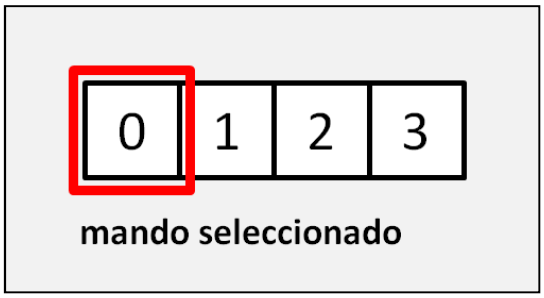
El controlador de mandos gestiona 4 puertos para mando numerados. La CPU puede consultar cada puerto para comprobar si actualmente tiene un mando conectado. Cuando hay un mando conectado, el controlador de mandos puede leer su estado actual.

2.2 Conceptos funcionales

Antes de explicar las funciones del controlador de mandos o las variables internas que les afectan, debemos presentar los conceptos básicos en los que se basa este chip.

2.2.1 Mando seleccionado

Todos los puertos de mando disponibles forman un rango usable de IDs de mando de 0 a 3. Para decidir qué mando usar al leer estados de controles o aplicar variables internas, el controlador de mandos siempre considera uno de esos IDs como "seleccionado". El mando seleccionado por defecto es el primero, correspondiente al ID de mando = 0.



2.2.2 Controles de los mandos

Los mandos de Vircon32 son todos idénticos. Cada mando cuenta con 11 controles: 4 direcciones y 7 botones. Estas listas los enumeran todos:

Direcciones:	**Botones:**
Izquierda, Derecha, Arriba, Abajo	A, B, X, Y, L, R, Start

Todos estos controles son digitales y los mandos representan su estado como un único bit: 1 (Verdadero) cuando está pulsado, 0 (Falso) cuando no lo está.

2.2.3 Formato del estado de un mando

Aunque cada control del mando se puede representar con 1 bit, el controlador de mandos guarda esos estados en formato entero. La razón de esto es que el controlador de mandos no se limita a leer y guardar el estado de los controles, sino que estos valores se van procesando a lo largo del tiempo para proporcionar más información y no sólo el estado actual de cada control.

El controlador de mandos realiza un seguimiento de los estados anteriores para informar también del tiempo (en frames) que el control ha estado en su estado actual. Esto se codifica en el signo y el valor del entero, siendo el signo positivo pulsado y el negativo no pulsado. Por ejemplo, si el estado actual del botón A es 20, la interpretación es esta:

Los juegos suelen necesitar que, en el instante que se pulsa un botón, se haga alguna acción una sola vez. Al proveer esta información extra, el controlador de mandos permite determinar al instante si se acaba de pulsar un botón, sin tener que guardar estados anteriores para saberlo.

Para que siempre se pueda determinar un signo, se garantiza que los estados de los controles nunca tendrán valor cero. Para asegurarlo se implementan estas 2 reglas:

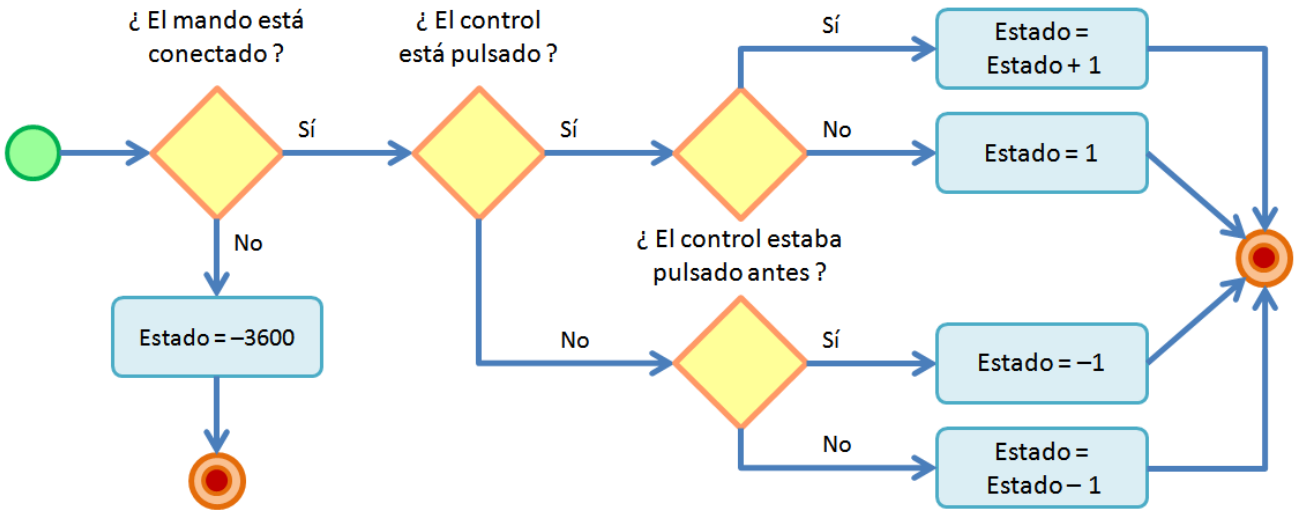
1. Los controles que acaban de cambiar de estado en el frame actual se pondrán a valor +/- 1 y comenzarán su cuenta de frames desde ahí si su estado se mantiene.

2. Si un puerto de mando dado no tiene conectado ningún mando, el controlador de mandos pondrá el estado de todos sus controles en su límite inferior de -3600. La interpretación de este valor podría ser que los controles llevan mucho tiempo sin pulsarse.

2.3 Proceso para leer los mandos

Con el proceso de lectura de mandos, el controlador de mandos lee el estado actual de los mandos y actualiza sus variables internas en consecuencia. La CPU podrá entonces hacer peticiones de lectura para determinar las pulsaciones actuales de los mandos.

Como en la mayoría de procesos de consola, el temporizado de lectura de los mandos está basado en frames. El proceso de lectura de mandos se activa al principio de cada frame y, para cada ejecución, hará un bucle sobre cada uno de los 4 puertos de mando. En cada puerto realizará el siguiente procesado para cada uno de los 11 controles de cada mando, para determinar sus respectivos estados actualizados (almacenados en sus respectivas variables internas).



Por su rango válido, los estados de un control no se pueden incrementar a más de 3600 ni decrementar a menos de -3600. En esos límites se detendrán. Además de actualizar los estados de controles con este algoritmo, el controlador de mandos también actualizará las variables booleanas que indican si cada mando está actualmente conectado.

2.3.1 Coherencia de la dirección

Vircon32 requiere que sus mandos impidan físicamente pulsar a la vez direcciones opuestas en un mismo eje. Esto significa que pulsar arriba + abajo, o izquierda + derecha no debería ser posible. Aún así, en algunas implementaciones podría ser inviable garantizar esto para todos los posibles mandos.

Cuando no se pueda garantizar esta característica del mando, el controlador de mandos deberá filtrar los eventos o estados de las direcciones, para que los programas nunca lean una dirección incoherente en la cruceta. Una forma habitual de hacerlo es considerar que cuando se pulsa un control de dirección, la dirección opuesta se suelta automáticamente. En otras palabras, se da prioridad a la última dirección pulsada en cada eje.

2.3.2 Eventos de conexión de mandos

Cuando se conecta o desconecta un mando, el controlador de mandos no da ninguna señal para informar del evento. La consola sólo lo sabrá cuando cambie la variable de conexión de un puerto, tras leer los estados del mando para el siguiente frame. Las implementaciones pueden funcionar usando internamente estos eventos si se necesita.

2.4 Variables internas

El controlador de mandos tiene un conjunto de variables que almacenan diferentes aspectos de su estado interno. Todas se guardan como valor de 32 bits, y se interpretan con los mismos formatos (entero, booleano, ...) descritos en la parte 2 de la especificación. Aquí enumeramos y detallamos todas las variables internas, organizadas en secciones.

2.4.1 Variables globales

Mando seleccionado	**Valor inicial:** 0
Formato: Entero	**Rango válido:** De 0 a 3

Este valor es el ID numérico del mando seleccionado actualmente. El mando seleccionado es el que se usará en todas las operaciones relacionadas con mandos.

Téngase en cuenta que los 4 IDs de mando siempre son seleccionables, incluso si sus mandos correspondientes no están conectados.

2.4.2 Estado de cada mando

Las variables listadas aquí son especiales. El controlador de mandos guarda una copia de cada una de estas variables por cada puerto de mando existente. Es decir: hay 4 copias. Las 4 existen siempre, sin importar qué mandos hay conectados.

Juntos, cada conjunto de estas variables describe las pulsaciones actuales para un solo mando. Como en cada momento sólo un conjunto de estas variables es accesible, se accede a cada variable de esta sección por puertos E/S a través "puntero". Cuando el mando seleccionado cambia, los puertos se redirigen a la copia de variables para ese mando.

Mando conectado	**Valor inicial:** (*)
Formato: Booleano	**Rango válido:** Verdadero/Falso

Este valor indica si actualmente el mando asociado está conectado o no.

(*) Su valor inicial viene dado por la presencia o ausencia de un mando en el puerto asociado al arrancar la consola.

{*control*} mando	**Valor inicial:** -3600
Formato: Entero	**Rango válido:** De -3600 a 3600 (excepto 0)

Son 11 variables, correspondientes a cada uno de los 11 controles del mando que se enumeró en la sección 2.2.2. Cada variable representa el estado actual para ese control en el mando asociado. Éste es el estado ya procesado, no sólo el valor booleano. Por lo tanto se interpreta como {signo, valor}, tal y como se describe en la sección 2.2.3.

2.5 Puertos de control

Esta sección detalla el conjunto de puertos de control expuestos por el controlador de mandos, a través de su conexión al bus de control de la CPU como dispositivo esclavo. La siguiente tabla enumera todos los puertos expuestos, junto con sus propiedades básicas.

Lista de puertos de control expuestos			
Dirección externa	Dirección interna	Nombre del puerto	Acceso L/E
400h	00h	Mando Seleccionado	Lectura y Escritura
401h	01h	Mando Conectado	Sólo Lectura
402h	02h	Izquierda Mando	Sólo Lectura
403h	03h	Derecha Mando	Sólo Lectura
404h	04h	Arriba Mando	Sólo Lectura
405h	05h	Abajo Mando	Sólo Lectura
406h	06h	Botón Start Mando	Sólo Lectura
407h	07h	Botón A Mando	Sólo Lectura
408h	08h	Botón B Mando	Sólo Lectura
409h	09h	Botón X Mando	Sólo Lectura
40Ah	0Ah	Botón Y Mando	Sólo Lectura
40Bh	0Bh	Botón L Mando	Sólo Lectura
40Ch	0Ch	Botón R Mando	Sólo Lectura

2.5.1 Acciones en peticiones de lectura/escritura de puertos

Los puertos de control del controlador de mandos no son simples registros hardware. Los efectos causados por una petición de lectura/escritura a un puerto concreto pueden ser distintos de lectura o escritura de valores. Esta sección detalla cómo se comporta cada puerto del controlador de mandos.

Obsérvese que, además de las acciones realizadas, será necesario dar una respuesta de éxito/fallo a la petición, como parte de la comunicación del bus de control. Si no se indica lo contrario siempre se asumirá que la respuesta es de éxito. Cuando la respuesta sea de fallo, el controlador de mandos no realizará más acciones y la CPU activará un error HW.

Puerto “Mando Seleccionado”

En peticiones de lectura:
El controlador de mandos proveerá el valor actual de la variable interna “Mando seleccionado”.

En peticiones de escritura:
El controlador de mandos comprobará si el valor recibido corresponde a un ID de mando válido. Si no lo es la petición será ignorada. Para valores válidos, el controlador de mandos sobrescribirá la variable interna “Mando seleccionado” con el valor recibido. Después redirigirá todos los puertos de estado del mando para apuntar a las variables del nuevo mando seleccionado.

Puerto “Mando Conectado”

En peticiones de lectura:
El controlador de mandos proveerá el valor actual de la variable interna “Mando conectado” asociada al ID de mando seleccionado actualmente.

En peticiones de escritura:
Como este puerto es de sólo lectura, se dará una respuesta de fallo al bus de control.

Puerto “{ *control* } Mando”

Estos mismos comportamientos aplican a los 11 puertos (con direcciones de 02h a 0Ch), que corresponden a los 11 controles del mando listados en la sección 2.2.2.

En peticiones de lectura:
El controlador de mandos proveerá el valor actual de la variable interna “{ *control* } mando” asociada al ID de mando seleccionado actualmente.

En peticiones de escritura:
Como estos puertos son de sólo lectura, se dará una respuesta de fallo al bus de control.

2.6 Respuestas a señales de control

Como todos los componentes de la consola, cada vez que se produzca una señal de control, el controlador de mandos la recibirá y reaccionará para procesar ese evento. Para cada una de las señales de control, el controlador de mandos responderá realizando las siguientes acciones:

Señal de Reset:

- Todas las variables internas del controlador de mandos se restablecen a sus valores iniciales. Esto incluye las variables de estado de todos los mandos.
- Para cada puerto de mando, el controlador de mandos comprobará si hay un mando conectado y actualizará su correspondiente variable “Mando conectado”.
- Cualquier efecto adicional asociado a esos cambios en las variables internas se aplica inmediatamente, como se indica en las peticiones de escritura en puertos de control.

Señal de Frame:

- El controlador de mandos activará el proceso de lectura de mandos, como se describe en la sección 2.3.

Señal de Ciclo:

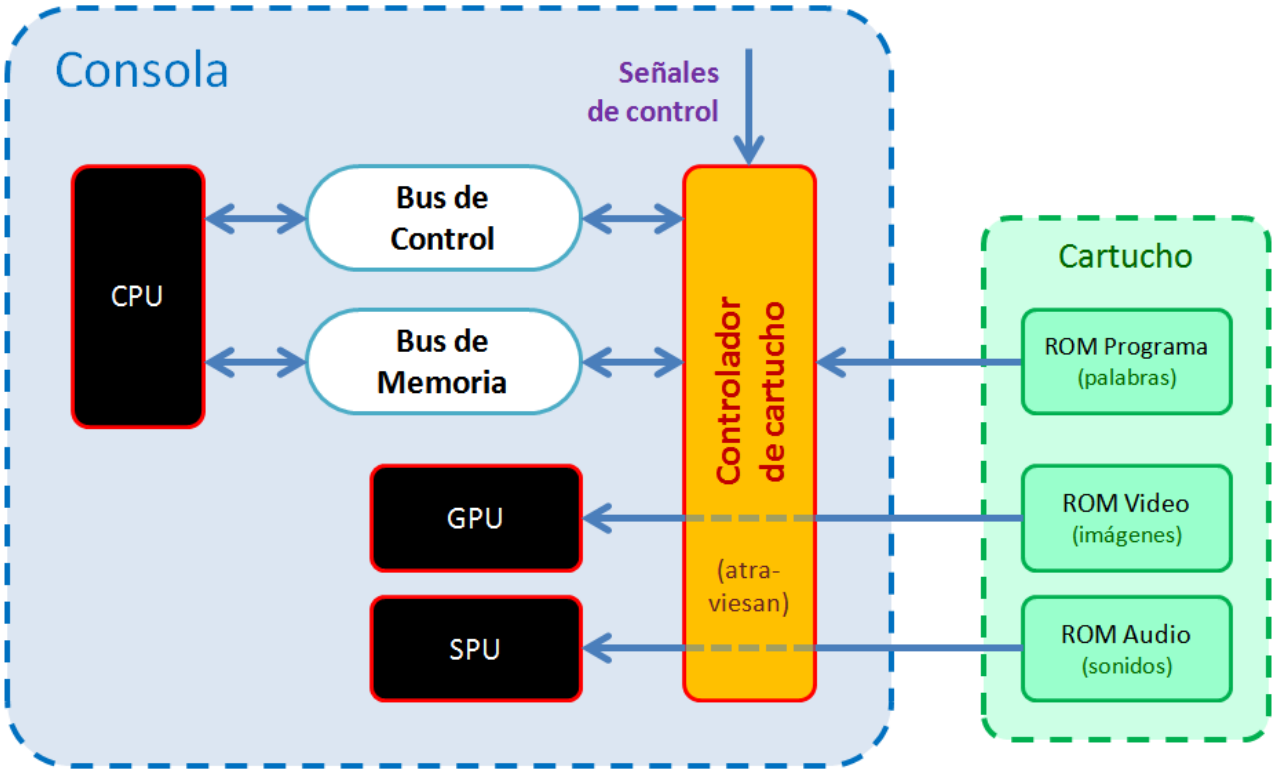
- El controlador de mandos no necesita reaccionar a esta señal, a menos que se requiera por detalles concretos de la implementación.

3 Controlador de cartucho

El controlador de cartucho es el chip encargado de controlar la ranura para cartucho de la consola. A diferencia de los mandos, sólo puede haber 1 cartucho conectado. Sin embargo cada cartucho tendrá un contenido diferente: de las 3 ROM del cartucho (programa, vídeo y audio) sólo la primera existirá siempre, y todas ellas pueden contener un número distinto de elementos en cada cartucho.

3.1 Conexiones externas

Como el controlador de cartucho es sólo uno de los chips que forman la consola, no puede funcionar aislado. Esta figura muestra todas sus comunicaciones con otros componentes. Como se ha dicho, el cartucho conectado (si lo hay) puede ser uno distinto cada vez.



Cada una de estas conexiones se explicará por separado en las secciones siguientes.

3.1.1 Señales de control

Como todos los componentes de la consola, el controlador de cartucho recibe las señales de reinicio, nuevo frame y nuevo ciclo. Las respuestas a estas señales se detallan en la sección 3.5 de este documento.

3.1.2 Bus de Control

El controlador de cartucho está conectado como esclavo al Bus de Control, con ID de dispositivo = 5. Esto permite al maestro del bus (la CPU) pedir operaciones de lectura o

escritura en los puertos de control expuestos por el controlador de cartucho. La lista de sus puertos y las propiedades de éstos se detallan en secciones posteriores.

3.1.3 Bus de Memoria

El controlador de cartucho está conectado como esclavo al Bus de Memoria, con ID de dispositivo = 2. Esto permite al maestro del bus (la CPU) pedir operaciones de lectura o escritura en las direcciones de memoria expuestas por el cartucho. Las propiedades y el rango de direcciones de memoria del cartucho se cubren en secciones posteriores.

3.1.4 Cartucho

El controlador de cartucho gestiona la ranura de cartucho. La CPU puede consultar si esta ranura tiene conectado actualmente un cartucho. Cuando haya un cartucho el controlador de cartucho podrá leer su contenido.

La ranura de cartucho cuenta con un mecanismo de seguridad que la bloquea mientras la consola esté encendida. Por esta razón, es seguro asumir que la conexión y el contenido del cartucho no cambiarán durante el funcionamiento de la consola.

3.1.5 GPU y SPU

La GPU y la SPU necesitan acceder a la ROM de vídeo y audio del cartucho respectivamente. Como estas ROMs pueden no estar presentes, esas 2 conexiones tienen que hacerse a través del controlador del cartucho. Esto es necesario para que cada chip determine si su ROM de destino está presente, y el número de elementos que contiene.

Una vez conocida esa información inicial, las conexiones desde la GPU y la SPU se conciben como pasantes: cada chip puede leer esos contenidos libremente. Sin embargo, el mecanismo real para configurar y gestionar este acceso lo definirá la implementación.

3.2 Memoria del cartucho

La consola en sí misma no contiene ningún programa para la CPU aparte de las rutinas de la BIOS, así que necesita leer las palabras de memoria almacenadas en los cartuchos conectándose a su ROM de programa. Una ROM de programa es una región de memoria de sólo lectura que contiene una secuencia de palabras de 32 bits.

El controlador de cartuchos está conectado al Bus de Memoria para permitir que un cartucho conectado exponga el contenido de su ROM de programa (que siempre existe) a la CPU. Si una ROM de programa de cartucho contiene N palabras, como el controlador de cartucho usa la ID = 2 en el Bus de Memoria, su rango de direcciones será:

Direcciones internas → De 0 a N – 1
Direcciones externas → De 20000000h a (20000000h + N – 1).

El tamaño de la ROM de programa variará en cada cartucho. Puede contener cualquier número de palabras desde 1 hasta su límite de tamaño de 128 x 1024 x 1024 palabras.

3.2.1 Conexión a la ROM de programa

Cuando se conecta un cartucho que con N palabras, las primeras N direcciones de memoria interna para el controlador del cartucho desde 0 se asignan a cada palabra, en orden. El resto de direcciones hasta (128x1024x1024 - 1) no se usan y no son accesibles.

Este proceso de conexión ocurre cada vez que se inserta un nuevo cartucho. Aún así, en los sistemas Vircon32 los cartuchos sólo pueden insertarse con la consola apagada. Por lo tanto, es seguro que las implementaciones retrasen los procedimientos necesarios para que el controlador de cartuchos establezca esta conexión hasta el encendido de la consola.

Depende de la implementación decidir cómo establecer una conexión con las ROMs de programa y localizar y leer sus palabras. Por ejemplo, se puede leer de antemano toda la ROM de programa al establecer la conexión. Otra opción sería mantener un puntero a la memoria del cartucho para que el controlador del cartucho lea palabras sobre la marcha.

3.2.2 Acciones en peticiones de lectura/escritura de memoria

Cualquier dirección de memoria dentro del rango usado actualmente se puede leer libremente. Por ello las peticiones de lectura para ese rango siempre proveerán el valor de la palabra guardada y se responderán con éxito. Los intentos de leer fuera del rango usado serán respondidos con fallo y provocarán un error de hardware.

Las ROMs de programa de los cartuchos son siempre memorias de sólo lectura, por lo que las peticiones de escritura se responderán con fallo y provocarán un error hardware.

3.3 Variables internas

El controlador de cartuchos dispone de un conjunto de variables que guardan diferentes aspectos de su estado interno. Cada una de ellas se almacena como un valor de 32 bits, y todas se interpretan con los mismos formatos (entero, booleano, etc.) descritos en la parte 2 de la especificación. Aquí listamos y detallamos todas las variables internas.

Cartucho conectado	**Valor inicial:** (*)
Formato: Booleano	**Rango válido:** Verdadero / Falso

Este valor indica si actualmente hay un cartucho conectado a la consola.

(*)Su valor inicial viene dado por la presencia o ausencia de un cartucho en la ranura de cartucho al arrancar la consola. Este valor no puede cambiar con la consola funcionando.

Tamaño ROM de Programa	**Valor inicial:** (*)
Formato: Entero	**Rango válido:** De 1 a 134217728 (= 128*1024*1024)

Representa el tamaño de la ROM de programa en palabras de 32bits para el cartucho conectado actualmente. Este valor es 0 cuando el cartucho no está presente.

(*) Su valor inicial viene dado por el cartucho presente (si lo hay) al arrancar la consola y su ROM de programa. Este valor no puede cambiar con la consola funcionando.

Número de texturas	**Valor inicial:** (*)
Formato: Entero	**Rango válido:** De 0 a 256

Representa el número de texturas contenidas en la ROM de vídeo del cartucho conectado actualmente. Este valor es 0 si el cartucho está ausente o no contiene una ROM de vídeo.

(*) Su valor inicial viene dado por el cartucho presente (si lo hay) al arrancar la consola y su ROM de video. Este valor no puede cambiar con la consola funcionando.

Número de sonidos	**Valor inicial:** (*)
Formato: Entero	**Rango válido:** De 0 a 1024

Representa el número de sonidos contenidos en la ROM de audio del cartucho conectado actualmente. Este valor es 0 si el cartucho está ausente o no contiene una ROM de audio.

(*) Su valor inicial viene dado por el cartucho presente (si lo hay) al arrancar la consola y su ROM de audio. Este valor no puede cambiar con la consola funcionando.

3.4 Puertos de control

Esta sección detalla el conjunto de puertos de control expuestos por el controlador de cartucho a través de su conexión al bus de control de la CPU como dispositivo esclavo. Todos los puertos expuestos, junto con sus propiedades básicas, se enumeran en la siguiente tabla:

Lista de puertos de control expuestos			
Dirección externa	Dirección interna	Nombre del puerto	Acceso L/E
500h	00h	Cartucho Conectado	Sólo Lectura
501h	01h	Tamaño ROM de Programa	Sólo Lectura
502h	02h	Número de Texturas	Sólo Lectura
503h	03h	Número de Sonidos	Sólo Lectura

3.4.1 Acciones en peticiones de lectura/escritura de puertos

A diferencia de otros chips, los puertos del controlador de cartucho sí pueden modelarse como registros de sólo lectura. El efecto al leer estos puertos es sólo obtener su valor asociado, como se detalla en esta sección.

Obsérvese que, además de las acciones realizadas, será necesario dar una respuesta de éxito/fallo a la petición, como parte de la comunicación del bus de control. Esta respuesta siempre será de éxito en peticiones de lectura y de fallo en las de escritura. En este último caso el controlador de cartucho no realizará más acciones y la CPU activará un error HW.

Puerto “Cartucho Conectado”

En peticiones de lectura:
El controlador de cartucho proveerá el valor actual de la variable interna “Cartucho conectado”.

Puerto “Tamaño ROM de Programa”

En peticiones de lectura:
El controlador de cartucho proveerá el valor actual de la variable interna “Tamaño ROM de Programa”.

Puerto “Número de Texturas”

En peticiones de lectura:
El controlador de cartucho proveerá el valor actual de la variable interna “Número de texturas”.

Puerto “Número de Sonidos”

En peticiones de lectura:
El controlador de cartucho proveerá el valor actual de la variable interna “Número de sonidos”.

3.5 Respuestas a señales de control

Como todos los componentes de la consola, cada vez que se produzca una señal de control, el controlador de cartucho la recibirá y reaccionará para procesar ese evento. Para cada una de las señales de control, el controlador de cartucho responderá realizando las siguientes acciones:

Señal de Reset:

- Si no hay cartucho presente y no se ha extraído ningún cartucho anteriormente, el controlador de cartucho realizará el mismo procesado que cuando se extrae el cartucho. En caso contrario no realizará ninguna acción.

Señales de Frame y Ciclo:

- El controlador de cartucho no necesita reaccionar a estas señales, a menos que se requiera por detalles concretos de la implementación.

Además de reaccionar a las señales de control, el controlador de cartucho también tendrá que realizar el siguiente procesado en respuesta a eventos a nivel de la consola:

Cuando se extrae el cartucho:

- El controlador del cartucho pondrá su variable “Cartucho conectado” a Falso.
- Las variables internas “Tamaño ROM de programa”, “Número de texturas” y “Número de sonidos” se pondrán todas a 0.
- Todas las direcciones de memoria del controlador de cartucho serán desasignadas.

Cuando se conecta un nuevo cartucho:

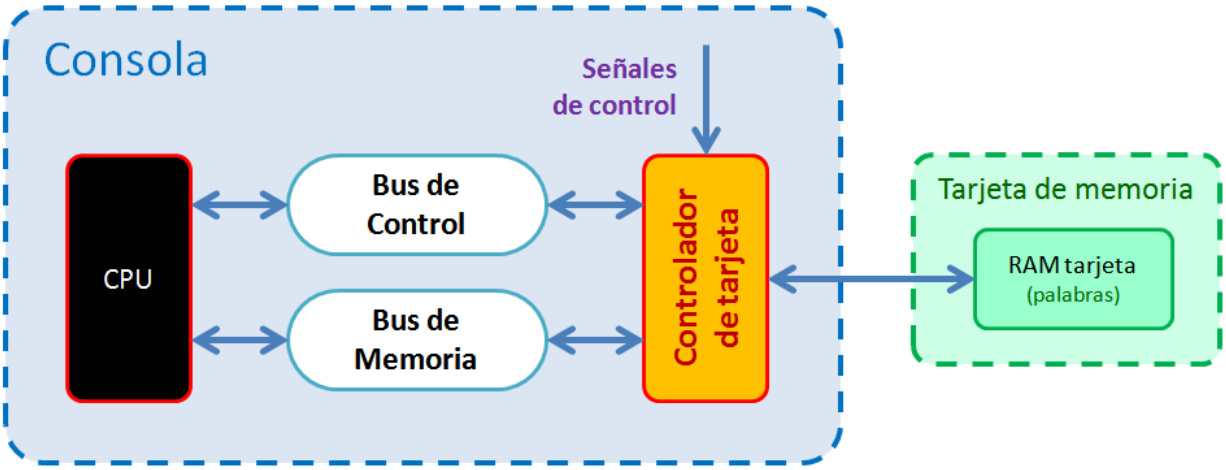
- El controlador del cartucho pondrá su variable “Cartucho conectado” a Verdadero.
- El controlador del cartucho realizará el proceso de conexión descrito en la sección 3.2.1 para asignar direcciones de memoria a su ROM de programa y acceder a las palabras que contiene.
- La variable interna “Tamaño de ROM de programa” se ajustará al número de palabras contenidas en la ROM de programa del cartucho conectado.
- Si el cartucho conectado tiene una ROM de video, el controlador de cartucho ajustará su variable interna “Número de texturas” al número de texturas que contiene.
- Si el cartucho conectado tiene una ROM de audio, el controlador de cartucho ajustará su variable interna “Número de sonidos” al número de sonidos que contiene.

4 Controlador de tarjeta

El controlador de tarjeta es el chip encargado de controlar la ranura para tarjeta de memoria de la consola. Al igual que los cartuchos, sólo puede haber 1 tarjeta presente. La memoria de todas las tarjetas es del mismo tamaño y sólo variará su contenido almacenado. Sin embargo, a diferencia de los cartuchos, las tarjetas de memoria se pueden conectar y desconectar en cualquier momento.

4.1 Conexiones externas

Como el controlador de tarjeta es sólo uno de los chips que forman la consola, no puede funcionar aislado. Esta figura muestra todas sus comunicaciones con otros componentes. Como se ha dicho, la tarjeta conectada (si la hay) puede ser una distinta cada vez.



Cada una de estas conexiones se explicará por separado en las secciones siguientes.

4.1.1 Señales de control

Como todos los componentes de la consola, el controlador de tarjeta recibe las señales de reinicio, nuevo frame y nuevo ciclo. Las respuestas a estas señales se detallan en la sección 4.5 de este documento.

4.1.2 Bus de Control

El controlador de tarjeta está conectado como esclavo al Bus de Control, con ID de dispositivo = 6. Esto permite al maestro del bus (la CPU) pedir operaciones de lectura o escritura en los puertos de control expuestos por el controlador de tarjeta. La lista de sus puertos y las propiedades de éstos se detallan en secciones posteriores.

4.1.3 Bus de Memoria

El controlador de tarjeta está conectado como esclavo al Bus de Memoria, con ID de dispositivo = 3. Esto permite al maestro del bus (la CPU) pedir operaciones de lectura o escritura en las direcciones de memoria expuestas por la tarjeta. Las propiedades y el rango de direcciones de memoria de la tarjeta se cubren en secciones posteriores.

4.1.4 Tarjeta de memoria

El controlador de tarjeta gestiona la ranura para tarjeta de memoria. La CPU puede consultar si esta ranura tiene conectada actualmente un tarjeta. Cuando haya una tarjeta el controlador de tarjeta podrá leer su contenido.

La ranura para tarjeta de memoria no dispone de un mecanismo de bloqueo como la ranura de cartucho. Los programas siempre deben comprobar si hay una tarjeta antes de leer o escribir en ella. Cualquier intento de acceder a la tarjeta cuando no está presente provocará un error hardware y detendrá el programa.

4.2 Memoria de la tarjeta

La consola en sí misma no tiene ningún almacenamiento permanente que puedan usar los programas. Para guardar y recuperar datos entre sesiones necesita leer y escribir las palabras de la tarjeta conectándose a su memoria RAM. Una memoria RAM es una región de memoria de lectura y escritura que contiene una secuencia de palabras de 32 bits.

El controlador de tarjeta está conectado al Bus de Memoria para permitir que una tarjeta conectada exponga el contenido de su RAM a la CPU. El controlador de tarjeta usa el ID de dispositivo = 3 dentro del Bus de Memoria. La RAM de una tarjeta de memoria contiene (256 x 1024) palabras, así que el rango de direcciones para la memoria de la tarjeta será:

Direcciones internas → De 0 a 3FFFh (= 256 x 1024 – 1)
Direcciones externas → De 30000000h a 30003FFFh

4.2.1 Conexión a la RAM de la tarjeta

Cuando se conecta una tarjeta, el controlador de tarjeta asigna todas sus direcciones de memoria interna (256 x 1024) a las palabras RAM de la tarjeta en orden, empezando en 0.

Este proceso de conexión se produce cada vez que se inserta una nueva tarjeta. Esto puede ocurrir en cualquier momento, por lo debe realizarse la conexión sin demora. Pero si se conecta una tarjeta con la consola apagada, la implementación puede optar por retrasar este proceso de conexión hasta el siguiente encendido de la consola.

La implementación debe decidir cómo establecer una conexión con la RAM de la tarjeta y acceder a sus palabras. Potencialmente los accesos a la tarjeta podrían ocurrir cada ciclo, por lo que una opción posible si la implementación no es lo bastante rápida es trabajar con una copia en la RAM del sistema y sólo escribir los cambios en la tarjeta real en determinados momentos (por ejemplo, usando la señal de nuevo frame).

4.2.2 Acciones en peticiones de lectura/escritura de memoria

Cualquier dirección de memoria dentro del rango establecido se puede leer y escribir libremente, por lo que cualquier petición en ese rango siempre accederá al valor de la palabra guardada y se responderá con éxito. Los intentos de lectura o escritura fuera del rango se responderán con fallo y provocarán un error hardware.

4.2.3 Eventos de conexión de tarjetas

Cuando se inserta o extrae una tarjeta, el controlador de tarjeta no da ninguna señal para informar del evento. Aún así la consola puede saber esto observando cuándo cambia el valor de la variable de "Tarjeta conectada". Las implementaciones pueden funcionar usando internamente estos eventos si se necesita.

4.3 Variables internas

El controlador de tarjeta dispone de una única variable que almacena su estado interno. Esta variable se almacena como un valor de 32 bits, y se interpreta con los mismos formatos (entero, booleano, etc.) descritos en la parte 2 de la especificación. A continuación detallamos esta variable interna.

Tarjeta conectada	**Valor inicial:** (*)
Formato: Booleano	**Rango válido:** Verdadero / Falso

Este valor indica si actualmente hay una tarjeta de memoria conectada a la consola.

(*) Su valor inicial viene dado por la presencia o ausencia de una tarjeta en la ranura para tarjeta de memoria al arrancar la consola.

4.4 Puertos de control

Esta sección detalla el único puerto de control expuesto por el controlador de tarjeta a través de su conexión al bus de control de la CPU como dispositivo esclavo. El puerto expuesto y sus propiedades básicas son las de la siguiente tabla:

Lista de puertos de control expuestos			
Dirección externa	Dirección interna	Nombre del puerto	Acceso L/E
600h	00h	Tarjeta Conectada	Sólo Lectura

4.4.1 Acciones en peticiones de lectura/escritura de puertos

A diferencia de otros chips, el puerto del controlador de tarjeta sí puede modelarse como un registro de sólo lectura. El efecto al leer este puerto es sólo obtener su valor asociado, como se detalla en esta sección.

Puerto “Tarjeta Conectada”

En peticiones de lectura:
El controlador de tarjeta proveerá el valor actual de la variable interna “Tarjeta conectada”. Se dará una respuesta de éxito al bus de control.

En peticiones de escritura:
Como este puerto es de sólo lectura, se dará una respuesta de fallo al bus de control.

4.5 Respuestas a señales de control

Como todos los componentes de la consola, cada vez que se produzca una señal de control, el controlador de tarjeta la recibirá y reaccionará para procesar ese evento. Para cada una de las señales de control, el controlador de tarjeta responderá realizando las siguientes acciones:

Señal de Reset:

- Si no hay tarjeta presente y no se ha extraído ninguna tarjeta anteriormente, el controlador de tarjeta realizará el mismo procesado que cuando se extrae la tarjeta. En caso contrario no realizará ninguna acción.

Señales de Frame y Ciclo:

- El controlador de tarjeta no necesita reaccionar a estas señales, a menos que se requiera por detalles concretos de la implementación.

Además de reaccionar a las señales de control, el controlador de tarjeta también tendrá que realizar el siguiente procesado en respuesta a eventos a nivel de la consola:

Cuando se conecta una tarjeta de memoria:

- El controlador de tarjeta pondrá su variable “Tarjeta conectada” a Verdadero.
- El controlador de tarjeta realizará el proceso de conexión descrito en la sección 4.2.1 para asignar direcciones de memoria a la RAM de la tarjeta y obtener acceso a las palabras que contiene.

Cuando se extrae la tarjeta de memoria:

- El controlador de tarjeta pondrá su variable “Tarjeta conectada” a Falso.
- Todas las direcciones de memoria del controlador de tarjeta serán desasignadas.

(Fin de la parte 6)