

Vircon32

CONSOLA VIRTUAL DE 32 BITS



Especificación del sistema

Parte 7: Otros componentes de la consola

Documento con fecha 2024.01.09

Escrito por Carra

¿Qué es esto?

Este documento es la parte número 7 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

Este documento se proporciona bajo la Licencia de Atribución Creative Commons 4.0 (CC BY 4.0). Puede leerse el texto completo de la licencia en el sitio web de Creative Commons: <https://creativecommons.org/licenses/by/4.0/>

Índice

La parte 7 de la especificación define el resto de componentes internos de la consola que no se habían cubierto hasta ahora. Para cada uno de ellos este documento describirá su comportamiento, comunicaciones, variables internas y el proceso general de su funcionamiento.

Al igual que en la parte 6, para agrupar el resto de chips en este único documento, cada sección para un chip en particular se hará en su lugar como una subsección aquí.

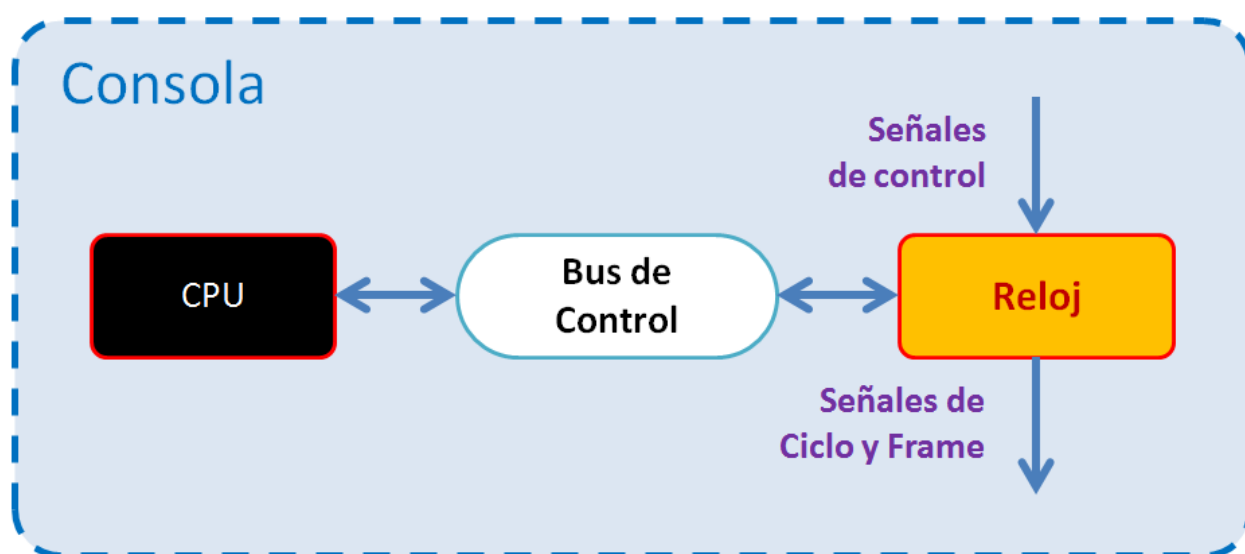
1 Reloj	3
2 Generador de Números Aleatorios (GNA)	9
3 Dispositivo nulo	12
4 Chip de RAM	13
5 Chip de la BIOS	15

1 Reloj

El reloj es el chip encargado de controlar la temporización global de la consola. Lo hace produciendo las señales de nuevo ciclo y nuevo frame que luego se envían a todos los componentes de la consola. Además de esto, el reloj también puede informar a la CPU información de la actual fecha y hora, a través de peticiones a puertos de control.

1.1 Conexiones externas

Como el reloj es sólo uno de los chips que forman la consola, no puede funcionar aislado. Esta figura muestra todas sus comunicaciones con otros componentes.



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

1.1.1 Señales de nuevo frame y nuevo ciclo

El reloj genera internamente las señales de nuevo ciclo y nuevo frame. A continuación emitirá estas señales para que otros elementos de la consola puedan recibirlas. Esto se describe en la sección 1.6 de este documento.

1.1.2 Señales de control

Como todos los componentes de consola, el reloj recibe la señal de reset. Las señales de nuevo frame y nuevo ciclo también pueden considerarse “recibidas” aquí como lo serían en cualquier otro componente, aunque las envíe el propio reloj. Las respuestas a las señales de control se detallan en el apartado 1.7 de este documento.

1.1.3 Bus de Control

El reloj está conectado como dispositivo esclavo al bus de control, con ID de dispositivo = 0. Esto permite al maestro del bus (la CPU) pedir operaciones de lectura o escritura en los

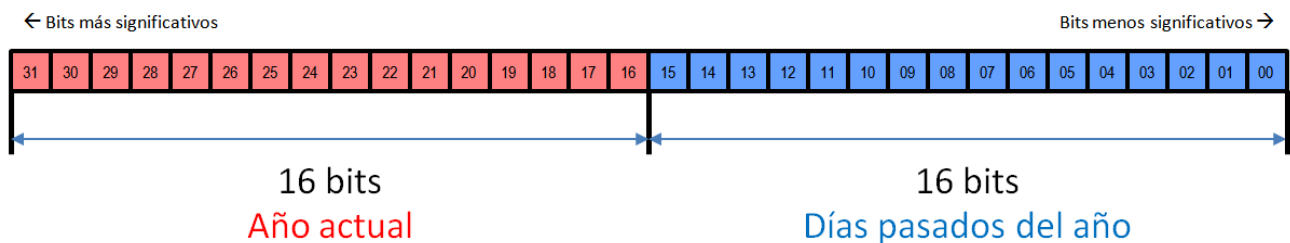
puertos de control que expone el reloj. La lista de sus puertos y las propiedades de éstos se detallan en secciones posteriores.

1.2 Fecha y hora actual

El reloj tiene un contador interno que se mantiene actualizado con la fecha y hora actual. Depende de la implementación definir cómo se gestiona. Una implementación hardware podría incluir un reloj con batería interna que sigue funcionando incluso cuando la consola está apagada. En cambio una implementación software puede actualizar la fecha y hora desde fuentes externas cada vez que se encienda la consola.

1.2.1 Formato interno de la fecha

El reloj representa internamente la fecha actual con una palabra de 32 bits. Este formato agrupa 2 campos. Cada uno se codifica como un entero de 16 bits sin signo, así:



Los 16 bits superiores son simplemente el número del año en curso (por ejemplo: 2022). Los 16 bits inferiores representan el número de días ya pasados de ese año. Así, por ejemplo, si han pasado 3 días ahora será 4 de Enero.

Los rangos válidos para estos 2 campos enteros en el formato de fecha son los siguientes:

- **AñoActual:** De 0 a 65535
- **DíasPasadosAño:** De 0 = 1 de Enero a 364 = 31 de Diciembre (*)

(*): Si el año actual se considera bisiesto, el límite superior de DíasPasadosAño se amplía a 365 para incluir el 29 de Febrero como nueva fecha del año.

1.2.2 Formato interno de la hora

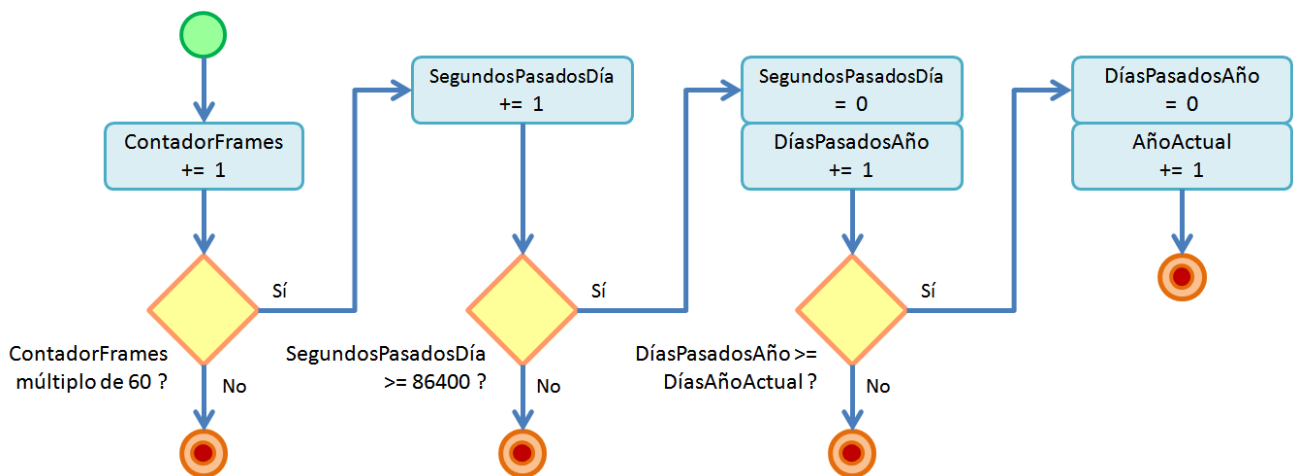
Dentro de una fecha determinada, el reloj también proporciona la hora actual dentro del día. Esta hora se representa internamente como un valor entero normal de 32 bits, que almacena el número de segundos transcurridos en el día actual.

Un día tiene $24 \times 60 \times 60 = 86400$ segundos. Por tanto, este formato para representar los segundos transcurridos dentro del día tendrá un rango válido de 0 = 00:00:00 a 86399 = 23:59:59.

1.3 Proceso para actualizar la hora

Según pasa el tiempo, el reloj necesita mantener actualizada su fecha y hora interna. El formato interno de hora tiene una resolución de 1 segundo, por lo que sólo será necesario actualizarlo cada vez que haya pasado un segundo completo.

La referencia más cercana que tiene el reloj para saber esto es la señal de nuevo frame. Los frames de la consola ocurren exactamente a 60 Hz, por lo que para actualizar la fecha y hora el reloj activará este proceso cada vez que produzca una señal de nuevo frame. La lógica usada para añadir el frame a la fecha y hora actuales es propagar cualquier desbordamiento de rango en los contadores internos, como se muestra en este algoritmo:



Debe tenerse en cuenta que esta lógica necesita determinar si el año actual es bisiesto. Con esto se fijará si los días del año actual deben ser 365 ó 366.

1.4 Variables internas

El reloj tiene un conjunto de variables que guardan diferentes aspectos de su estado interno. Cada una se guarda como un valor de 32 bits, y todas se interpretan con los mismos formatos (entero, booleano, etc) vistos en la parte 2 de la especificación, excepto el formato de fecha actual ya descrito. Aquí se lista y detalla todas las variables internas.

Fecha actual	Valor inicial: (*)
Formato: Fecha	Rango válido: De 1 Enero de 0 a 31 Diciembre de 65535

Este valor indica la fecha actual, en el formato de fecha descrito en la sección 1.2.1.

(*) Su valor inicial viene dado por la fecha actual cuando se arranca la consola.

Hora actual	Valor inicial: (*)
Formato: Entero	Rango válido: De 0 a 86399

Este valor indica la hora actual, expresada como se describe en la sección 1.2.2.

(*) Su valor inicial viene dado por la hora actual cuando se arranca la consola.

Contador de frames	Valor inicial: 0
Formato: Entero	Rango válido: Valores no negativos del entero de 32 bits

Indica los frames transcurridos desde el último encendido o reinicio de la consola.

Contador de ciclos	Valor inicial: 0
Formato: Entero	Rango válido: De 0 a 249999

Este valor indica el número de ciclos de CPU transcurridos dentro del frame actual.

1.5 Puertos de control

Esta sección detalla el conjunto de puertos de control expuestos por el reloj a través de su conexión al bus de control de la CPU como dispositivo esclavo. La siguiente tabla enumera todos los puertos expuestos y sus propiedades básicas:

Lista de puertos de control expuestos			
Dirección externa	Dirección interna	Nombre del puerto	Acceso L/E
000h	00h	Fecha Actual	Sólo Lectura
001h	01h	Hora Actual	Sólo Lectura
002h	02h	Contador de Frames	Sólo Lectura
003h	03h	Contador de Ciclos	Sólo Lectura

1.5.1 Acciones en peticiones de lectura/escritura de puertos

A diferencia de otros chips, los puertos del reloj sí pueden modelarse como registros de sólo lectura. El efecto de leerlos es sólo obtener su valor relacionado, como se explica.

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 “Fecha Actual”

En peticiones de **lectura**:

El reloj proveerá el valor actual de la variable interna “Fecha Actual”.

Puerto “Hora Actual”

En peticiones de **lectura**:

El reloj proveerá el valor actual de la variable interna “Hora Actual”.

Puerto “Contador de Frames”

En peticiones de **lectura**:

El reloj proveerá el valor actual de la variable interna “Contador de frames”.

Puerto “Contador de Ciclos”

En peticiones de **lectura**:

El reloj proveerá el valor actual de la variable interna “Contador de ciclos”.

1.6 Envío de las señales de tiempo

Para controlar la temporización global de la consola, el reloj debe ser capaz de medir el tiempo con precisión. El mecanismo concreto para ello debe definirlo la implementación: una implementación hardware podría usar un oscilador, mientras que una versión software puede usar cronómetros del sistema con resolución suficiente.

Mientras la consola esté encendida, el reloj producirá nuevas señales de frame a intervalos regulares de 60 Hz. Esta frecuencia debe ser lo más precisa posible para evitar problemas de sincronización cuando la consola esté encendida durante mucho tiempo.

Entre cada 2 señales de frame consecutivas, el reloj debe producir también 250.000 señales de ciclo. A diferencia de las señales de frame, no es necesario que las señales de ciclo se produzcan a intervalos regulares. Pueden producirse a cualquier ritmo dentro del frame que las contiene, incluso de forma irregular. Las implementaciones son libres de decidir cómo distribuir las señales de ciclo dentro de cada frame.

1.7 Respuestas a señales de control

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

Señal de Reset:

- Las variables internas “Contador de frames” y “Contador de ciclos” se ponen a sus valores iniciales.
- Las variables internas “Fecha actual” y “Hora actual” se ajustan para representar la fecha y hora actuales respectivamente. Si es necesario, el reloj accederá a la información de fecha y hora de la implementación y la convertirá a los formatos internos descritos en la sección 1.2.
- Por último, el reloj genera una señal de nuevo frame para comenzar el frame inicial.

Señal de Frame:

- El reloj activa el proceso para actualizar la hora, como describe la sección 1.3.
- La variable interna “Contador de ciclos” se pone a 0.

Señal de Ciclo:

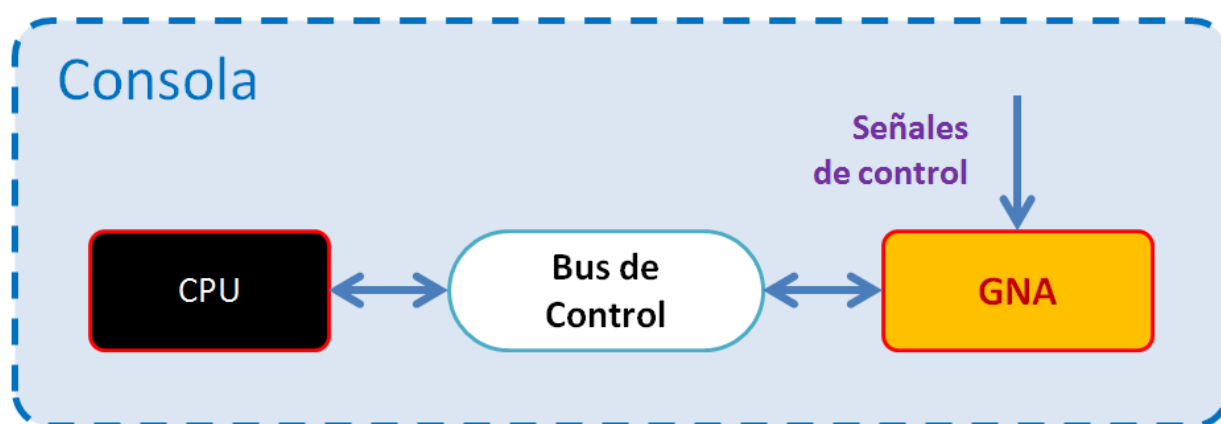
- La variable interna “Contador de ciclos” se incrementa en 1.

2 Generador de Números Aleatorios (GNA)

El chip GNA sólo tiene una función: generar secuencias de números pseudoaleatorios. Esto es útil en juegos que necesitan simular la suerte o comportamientos aleatorios. Este tipo de función es bastante fácil de implementar en software, pero tener un chip dedicado a ello sigue siendo valioso: proporciona a todos los sistemas Vircon32 un algoritmo de generación estándar en el que todas las implementaciones deben coincidir.

2.1 Conexiones externas

Como el GNA es sólo uno de los chips que forman la consola, no puede funcionar aislado. Esta figura muestra todas sus comunicaciones con otros componentes.



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 GNA recibe las señales de reset, nuevo frame y nuevo ciclo. Las respuestas a estas señales se detallan en la sección 2.5.

2.1.2 Bus de Control

El GNA se conecta como dispositivo esclavo al bus de control, con ID de dispositivo = 1. Esto permite al maestro del bus (la CPU) pedir operaciones de lectura o escritura sobre los puertos de control que expone el GNA. La lista de sus puertos así como sus propiedades se detallan en secciones posteriores.

2.2 Variables internas

El GNA dispone de una única variable que guarda 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 presentamos y detallamos esta variable interna.

Valor actual	Valor inicial: 1
Formato: Entero	Rango válido: De 1 a 7FFFFFFEh

Este valor representa la semilla actual utilizada por el algoritmo de generación pseudo-aleatoria. También es el siguiente valor que se dará como parte de la secuencia.

2.3 Puertos de control

Esta sección detalla el conjunto de puertos de control expuestos por el GNA a través de su conexión como dispositivo esclavo al bus de control de la CPU. La siguiente tabla lista el único puerto expuesto, junto con sus propiedades básicas:

Lista de puertos de control expuestos			
Dirección externa	Dirección interna	Nombre del puerto	Acceso L/E
100h	00h	Valor Actual	Lectura y Escritura

2.3.1 Acciones en peticiones de lectura/escritura de puertos

El puerto de control del GNA no es simplemente un registro hardware. Los efectos causados por una petición de lectura/escritura a este puerto son diferentes de la lectura o escritura de un valor. En esta sección se detallan estos comportamientos.

Además de las acciones realizadas, la respuesta dada al bus de control para cualquier petición de lectura y escritura a este puerto siempre será de éxito.

Puerto “Valor Actual”

En peticiones de **lectura**:

El GNA proveerá el valor actual de la variable interna “Valor actual”. Después, el GNA activará el proceso de generación descrito en el apartado 2.4 para actualizar dicha variable con el siguiente valor de la secuencia.

En peticiones de **escritura**:

El GNA comprobará si el valor recibido está fuera del rango de la variable interna “Valor actual”. Si lo está la solicitud será ignorada. Para valores válidos el GNA sobrescribirá la variable interna “Valor actual” con el valor recibido. Esto hará que la siguiente petición de lectura a este puerto proporcione el nuevo valor escrito.

2.4 Proceso para generar la secuencia

Los algoritmos para generar números pseudoaleatorios suelen basarse en un valor semilla inicial. La secuencia de valores obtenida se determina completamente a partir de esa semilla inicial concreta. La semilla que usa el GNA es su variable interna "Valor actual". En cada paso de la generación, esa variable se sobrescribe con el siguiente valor de la secuencia. Ese valor actualizado servirá a su vez como semilla para el siguiente paso de generación, y así sucesivamente.

Todas las implementaciones de Vircon32 debe usar un algoritmo de generación equivalente. Esto significa que cualquier sistema debe generar la misma secuencia para un mismo valor inicial de semilla. El algoritmo de referencia del GNA es el siguiente:



Nótese que este algoritmo hace uso de una variable temporal entera de 64 bits que llamaremos Valor64. La razón de esto es que, si usamos sólo valores de 32 bits, puede haber inconsistencias en la forma en que las diferentes plataformas manejan los bits de desbordamiento producidos al multiplicar. Usando una variable de 64 bits para hacer la multiplicación podemos asegurar un resultado consistente.

Los valores numéricos usados para este algoritmo se eligen para ser los mismos que los de la implementación de `minstd_rand` en C++11.

2.5 Respuestas a señales de control

Como todos los componentes de la consola, cada vez que se produce una señal de control, el GNA la recibe y produce una respuesta para procesar ese evento. Para cada una de las señales de control, el GNA responderá realizando las siguientes acciones:

Señal de Reset:

- La variable interna "Valor actual" se pone a su valor inicial.
- Los efectos adicionales asociados a ese cambio en la variable interna se aplican inmediatamente, como describe el comportamiento de escritura del puerto de control.

Señales de Frame y Ciclo:

- El GNA no necesita reaccionar a estas señales, salvo que lo requieran detalles específicos de la implementación.

3 Dispositivo nulo

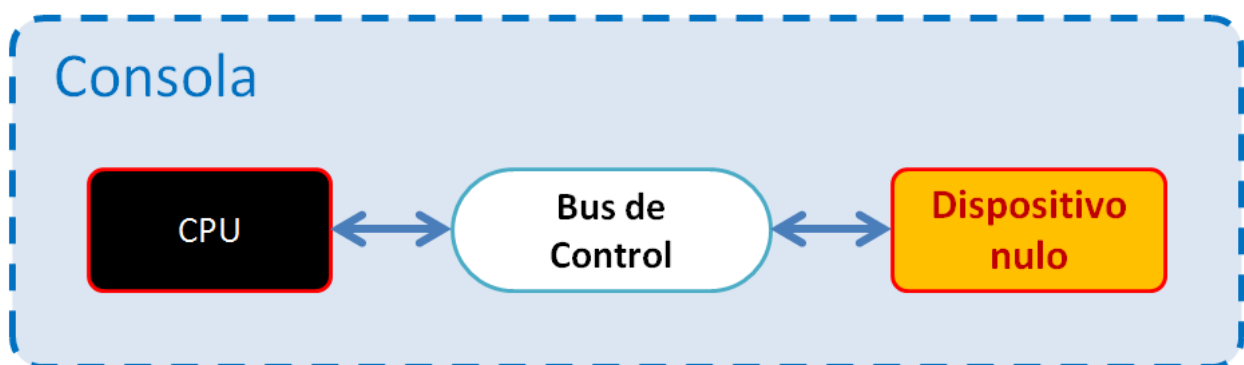
Este dispositivo no se ha mencionado en ningún documento hasta ahora, porque no es un chip real de la consola: es sólo un chip ficticio opcional. Puede ser útil en algunas implementaciones para ayudar a asegurar que cualquier petición de acceso a puertos de control inexistentes se responda siempre con fallo.

Como se vio en la parte 2 de la especificación, las direcciones globales del bus de control se componen de 3 bits (ID del dispositivo) + 8 bits (dirección local del puerto en ese dispositivo). Esto hace que el espacio de direcciones del bus de control esté efectivamente dividido en 8 IDs de dispositivos separadas. Pero como el bus de control sólo tiene 7 dispositivos conectados (IDs 0 a 6), el ID = 7 queda sin usar y no corresponde a ningún dispositivo esclavo.

Conectar el dispositivo nulo en el último ID de dispositivo, permite a las implementaciones tratar todas las peticiones de lectura/escritura en puertos de la misma manera: simplemente envían cada petición al dispositivo correspondiente para que la procese.

3.1 Conexiones externas

La única función del dispositivo nulo es estar conectado al bus de control como dispositivo esclavo, con ID = 7, y responder con fallo a cualquier petición recibida. Como dispositivo ficticio no se comunica con ninguno de los chips reales de la consola. Tampoco necesita recibir ninguna señal de control.



3.2 Puertos de control

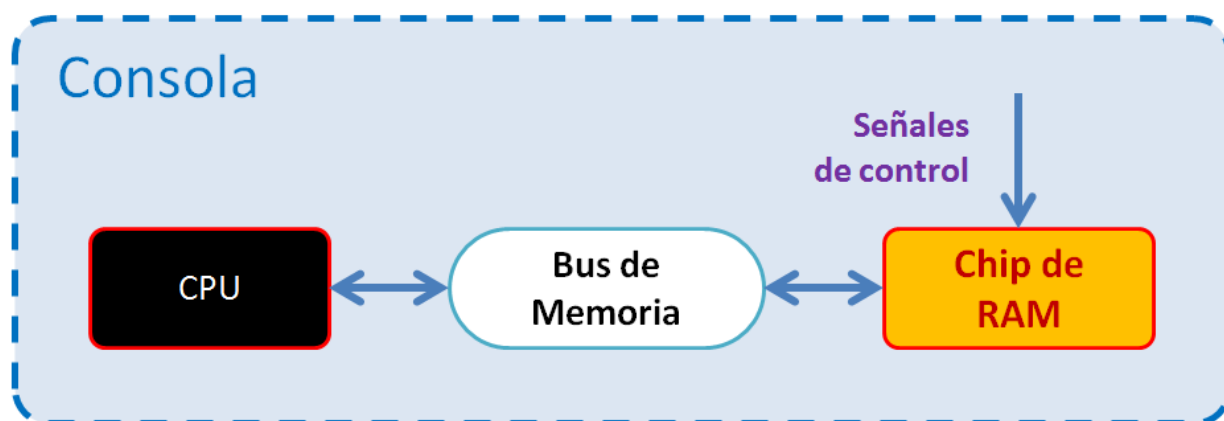
El dispositivo nulo en realidad no expone ningún puerto. En su lugar rechaza de forma automática cualquier petición dentro de su rango completo de puertos (direcciones externas de 700h a 7FFh) dando una respuesta de fallo al bus de control.

4 Chip de RAM

El chip de la memoria RAM es básicamente un componente pasivo: no realiza ninguna función salvo gestionar la memoria que contiene y procesar las peticiones de lectura ó escritura en ella. Por eso el chip de RAM no tiene variables internas ni puertos de control.

4.1 Conexiones externas

Como la RAM es sólo uno de los chips que forman la consola, no puede funcionar aislada. Esta figura muestra todas sus comunicaciones con otros componentes.



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

4.1.1 Señales de control

Como todos los componentes de la consola, el chip de RAM recibe las señales de reset, nuevo frame y nuevo ciclo. Las respuestas a esas señales se detallan en la sección 4.3.

4.1.2 Bus de Memoria

El chip de RAM está conectado como dispositivo esclavo al bus de memoria, con ID de dispositivo = 0. Esto permite al maestro del bus (la CPU) pedir operaciones de lectura ó escritura en las direcciones de memoria que expone la RAM. El rango y las propiedades de las direcciones de memoria RAM se verán en secciones posteriores.

4.2 Memoria RAM

La memoria RAM es una secuencia numerada de palabras de 32 bits. Tiene un tamaño de 4 MWords = $4 \times 1024 \times 1024$ palabras. RAM significa memoria de acceso aleatorio: cada una de estas palabras individuales se puede leer y modificar de forma independiente.

La memoria RAM no es persistente: sólo puede almacenar su contenido mientras la consola está encendida. Los datos que contiene se borrarán al apagar la consola.

4.2.1 Conexión a la memoria

El chip de RAM está conectado al bus de memoria para exponer el contenido de su memoria a la CPU. La RAM usa el ID de dispositivo = 0 dentro del bus de memoria, por lo que el rango de direcciones para la memoria RAM será:

Direcciones internas → De 0 a 3FFFFFFh (= 4 x 1024 x 1024 – 1)

Direcciones externas → De 00000000h a 003FFFFFFh.

4.2.2 Rendimiento de la memoria

No se especifican los niveles de rendimiento, pero se da por supuesto que la memoria RAM es lo bastante rápida para que, en cada ciclo, responda a todas las peticiones de la CPU a tiempo para que la instrucción actual termine dentro del mismo ciclo.

4.3 Respuestas a señales de control

Como todos los componentes de la consola, cada vez que se envía una señal de control, el chip de RAM la recibe y produce una respuesta para procesar ese evento. Para cada una de las señales de control, el chip de RAM responderá realizando las siguientes acciones:

Señal de Reset:

- Todas las direcciones de memoria RAM se pondrán al valor 0.

Señales de Frame y Ciclo:

- El chip de RAM no necesita reaccionar a estas señales, salvo que lo requieran detalles concretos de la implementación.

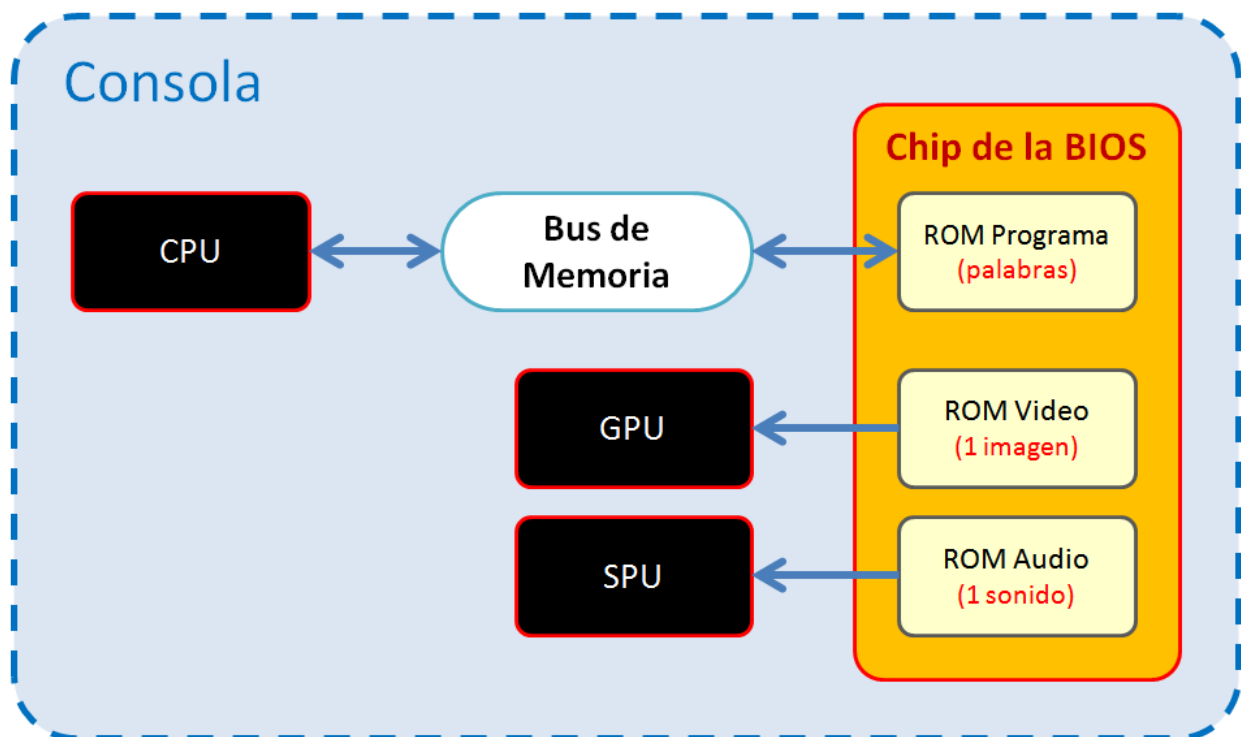
5 Chip de la BIOS

El chip de la BIOS, al igual que el de RAM, es principalmente un componente pasivo que gestiona el acceso a la ROM de la BIOS. Como tal, el chip de la BIOS no tiene variables internas ni puertos de control. Tampoco tiene necesidad de recibir señales de control, a menos que lo requieran detalles concretos de la implementación.

Las siglas BIOS significan Sistema Básico de Entrada/Salida, y es una forma común de llamar, en muchos dispositivos electrónicos, a un conjunto de rutinas instaladas en ROM que se encargan de gestionar operaciones de bajo nivel, como el arranque o la gestión de errores. En el caso de Vircon32 una ROM de BIOS es similar en estructura a una ROM de cartucho y contiene las mismas 3 partes (ROM de programa, ROM de video y ROM de audio), pero está sujeta a algunas restricciones adicionales.

5.1 Conexiones externas

Como la BIOS es sólo uno de los chips que forman la consola, no puede aislada. Esta figura muestra todas sus comunicaciones con otros componentes.



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

5.1.1 Bus de Memoria

El chip de BIOS está conectado como dispositivo esclavo al bus de memoria, con ID de dispositivo = 1. Esto permite al maestro del bus (la CPU) pedir operaciones de lectura o

escritura en las direcciones de memoria expuestas por el chip de BIOS. El rango y las propiedades de estas direcciones se cubren en secciones posteriores.

5.1.2 GPU y SPU

La GPU y la SPU necesitan acceder a la ROM de video y audio de la BIOS, respectivamente. Las conexiones de GPU y SPU se conciben como un acceso directo: cada chip puede leer libremente esos contenidos. Sin embargo, el mecanismo real para configurar y gestionar este acceso debe definirlo la implementación.

5.2 Memoria de la BIOS

Para ejecutar las rutinas de la BIOS, la CPU necesita leer las palabras de memoria almacenadas en la BIOS 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 chip de la BIOS está conectado al bus de memoria para poder exponer el contenido de su ROM de programa a la CPU. El chip de BIOS usa el ID de dispositivo = 1 dentro del bus de memoria, por lo que, para una BIOS cuya ROM de programa contiene N palabras, el rango de direcciones en memoria será:

Direcciones internas → De 0 a N – 1

Direcciones externas → De 10000000h a (10000000h + N – 1).

El tamaño de la ROM de programa de la BIOS variará para cada BIOS. Puede contener cualquier número de palabras desde 1 hasta el límite de tamaño de la ROM de programa de la BIOS, que es de 1024 x 1024.

5.2.1 Rendimiento de la memoria

No se indican niveles de rendimiento, pero se da por supuesto que los 3 componentes de la ROM de la BIOS son lo bastante rápidos como para ser capaces de lo siguiente:

- En cada ciclo, la memoria ROM de programa puede enviar a la CPU todas las palabras que pide a tiempo para que termine la instrucción en ese mismo ciclo.
- En cada frame, la memoria ROM de video puede enviar a la GPU todos los pixels que necesita a tiempo para terminar los comandos de dibujo en ese mismo frame.
- En cada frame, la memoria ROM de audio puede enviar a la SPU todos los samples que necesita a tiempo para terminar la generación de sonido en ese mismo frame.

5.3 Requisitos para una BIOS

Además de los requisitos generales aplicables a todas las ROM de programa, video y audio, el contenido de una BIOS de Vircon32 está sujeto a varios requisitos más restrictivos. Se enumeran a continuación, agrupados en una subsección para cada tipo:

5.3.1 Requisitos para el contenido de una BIOS

- Su ROM de video debe existir y contener exactamente 1 textura.
- Su ROM de audio debe existir y contener exactamente 1 sonido.
- Su ROM de audio tiene un límite de tamaño de 1024 x 1024 samples.
- Su ROM de programa tiene un límite de tamaño de 1024 x 1024 palabras.
- Su ROM de programa debe contener una rutina de arranque que será llamada inmediatamente después de cada encendido o reinicio de la consola. El punto de entrada de esta rutina debe estar en la dirección 4 de la ROM de programa.
- Su ROM de programa debe contener una rutina de gestión de errores que será llamada cuando la CPU active cualquier error hardware. El punto de entrada de esta rutina debe estar en la dirección 0 de la ROM de programa.

5.3.2 Requisitos para la rutina de arranque de una BIOS

- Debe definir una fuente de texto en las regiones 0 a 255 de la textura de la BIOS. Ver sección 5.4 para más detalles sobre esta fuente.
- Debe definir la región 256 de la textura de la BIOS para que sea un único píxel blanco con opacidad total, es decir: sus 4 componentes RGBA serán 255.
- Mostrar algún tipo de pantalla de arranque con imagen y sonido es opcional, pero recomendable. Puede ayudar a descartar fallos de funcionamiento en ciertas implementaciones sin necesidad de usar un cartucho.
- Como último paso debe comprobar si hay un cartucho presente y, en ese caso, transferir la ejecución al inicio de su ROM de programa.
- Si por el contrario el cartucho está ausente, lo comunicará por pantalla.







5.3.3 Requisitos para el manejo de errores de una BIOS

- Debe informar en pantalla, al menos, del tipo de error hardware que ha ocurrido.
- Mostrar información ampliada sobre el error (por ejemplo, el valor del registro Puntero a Instrucción) es opcional, pero recomendable.
- Una vez que termine su procesado, debe detener la ejecución parando la CPU.

5.4 La fuente de texto de la BIOS

La fuente de texto de la BIOS tiene 256 caracteres, definidos en las regiones 0 a 255 de la textura de la BIOS. Cada una debe tener 10 pixels de ancho x 20 pixels de alto, y su foco debe estar en su píxel superior izquierdo. Como la BIOS siempre está instalada, todos los programas pueden asumir que la fuente de la BIOS está disponible para escribir texto.

El mapeo de caracteres para la fuente de la BIOS se basa en la página de códigos 1252 de Windows (también conocida como Latin-1), y en su subconjunto ISO 8859-1. La siguiente tabla indica qué carácter debe definirse en cada región, de 00h a FFh. Los 4 dígitos en azul en cada celda indican el punto de código Unicode del carácter en esa posición.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	no usado	no usado	no usado	no usado	no usado	no usado	no usado	no usado	no usado	(HT) 0009	(LF) 000A	no usado	no usado	(CR) 000D	 2B1A	 25AF
10	(vacío) 2591	 2591	 2592	 2593	 2588	— 2500	 2502	J 2518	┐ 2510	└ 250C	L 2514	┌ 251C	└ 2524	┐ 2534	T 252C	└ 253C
20	(SP) 0020	! 0021	" 0022	# 0023	\$ 0024	% 0025	& 0026	' 0027	(0028) 0029	* 002A	+ 002B	, 002C	- 002D	. 002E	/ 002F
30	0 0030	1 0031	2 0032	3 0033	4 0034	5 0035	6 0036	7 0037	8 0038	9 0039	: 003A	; 003B	< 003C	= 003D	> 003E	? 003F
40	@ 0040	A 0041	B 0042	C 0043	D 0044	E 0045	F 0046	G 0047	H 0048	I 0049	J 004A	K 004B	L 004C	M 004D	N 004E	O 004F
50	P 0050	Q 0051	R 0052	S 0053	T 0054	U 0055	V 0056	W 0057	X 0058	Y 0059	Z 005A	[005B	\ 005C] 005D	^ 005E	_ 005F
60	` 0060	a 0061	b 0062	c 0063	d 0064	e 0065	f 0066	g 0067	h 0068	i 0069	j 006A	k 006B	l 006C	m 006D	n 006E	o 006F
70	p 0070	q 0071	r 0072	s 0073	t 0074	u 0075	v 0076	w 0077	x 0078	y 0079	z 007A	{ 007B	 007C	} 007D	~ 007E	no usado
80	€ 20AC	no usado	, 201A	f 0192	” 201E	... 2026	† 2020	‡ 2021	^ 02C6	% 2030	Š 0160	‹ 2039	Œ 0152	no usado	Ž 017D	no usado
90	no usado	` 2018	´ 2019	“ 201C	” 201D	• 2022	— 2013	— 2014	~ 02DC	™ 2122	š 0161	› 203A	œ 0153	no usado	ž 017E	ÿ 0178
A0	no usado	ı 00A1	¢ 00A2	£ 00A3	¤ 00A4	¥ 00A5	¦ 00A6	§ 00A7	¨ 00A8	© 00A9	ª 00AA	« 00AB	¬ 00AC	- 00AD	® 00AE	¯ 00AF
B0	° 00B0	± 00B1	² 00B2	³ 00B3	´ 00B4	µ 00B5	¶ 00B6	· 00B7	¸ 00B8	¹ 00B9	º 00BA	» 00BB	¼ 00BC	½ 00BD	¾ 00BE	¿ 00BF
C0	À 00C0	Á 00C1	Â 00C2	Ã 00C3	Ä 00C4	Å 00C5	Æ 00C6	Ç 00C7	È 00C8	É 00C9	Ê 00CA	Ë 00CB	Ì 00CC	Í 00CD	Î 00CE	Ï 00CF
D0	Ð 00D0	Ñ 00D1	Ò 00D2	Ó 00D3	Ô 00D4	Õ 00D5	Ö 00D6	× 00D7	Ø 00D8	Ù 00D9	Ú 00DA	Û 00DB	Ü 00DC	Ý 00DD	Þ 00DE	ß 00DF
E0	à 00E0	á 00E1	â 00E2	ã 00E3	ä 00E4	å 00E5	æ 00E6	ç 00E7	è 00E8	é 00E9	ê 00EA	ë 00EB	ì 00EC	í 00ED	î 00EE	ï 00EF
F0	ð 00F0	ñ 00F1	ò 00F2	ó 00F3	ô 00F4	õ 00F5	ö 00F6	÷ 00F7	ø 00F8	ù 00F9	ú 00FA	û 00FB	ü 00FC	ý 00FD	þ 00FE	ÿ 00FF

Los colores usados en esta tabla se interpretan de la siguiente forma:

	Blanco: Caracteres imprimibles normales. Dibujados como su código Unicode.
	Rojo: Caracteres sin usar. Deberían estar todos vacíos.
	Amarillo: Espacios en blanco. El espacio debería estar vacío; el resto pueden estarlo también, o dibujarse como elija la implementación.
	Verde: Estas posiciones corresponden a códigos de control no imprimibles en la página de códigos 1252, pero la fuente de la BIOS los reemplaza por un conjunto de caracteres diseñados para dibujar marcos y rejillas sólo con texto.

5.5 La BIOS estándar

Pueden existir múltiples implementaciones de la BIOS, creadas por distintos autores. Todas ellas pueden considerarse BIOS válidas siempre que cumplan los requisitos que establece este documento. Sin embargo hay sólo una BIOS estándar, que se denomina “Vircon32 standard BIOS” en el archivo ROM. Otras implementaciones de BIOS deben usar nombres diferentes para que implementaciones y usuarios puedan distinguirlas de la BIOS estándar.

Se recomienda que todas las implementaciones de Vircon32, aún si usan una BIOS personalizada por defecto, incluyan también la BIOS estándar como opción. De esta forma todas las implementaciones tendrán los medios para coincidir en la forma de manejar errores hardware, en caso de que los desarrolladores necesiten recurrir a esto para pruebas o depuración.

(Fin de la parte 7)