INDICE DE CONTENIDOS

CAPITULO 1

INTRODUCCION

La familia de ordenadores Amiga consta de varios modelos, todos ellos diseñados con el mismo propósito - proporcionar al usuario un ordenador de bajo coste con las caracteristicas que normalmente estaban reservadas para los grandes ordenadores. Esto se consigue con el uso de hardware especialmente diseñado para proporcionar graficos y sonido avanzados.

Hay tres modelos fabricados de la familia Amiga: el A500, el A1000 y el A2000. Aunque con distintos precios y posibilidades, todos tienen un nucleo común de hardware que les hacen compatibles entre ellos. Este capitulo describe los componentes del hardware del Amiga y da un pequeño vistazo a las caracteristicas de los graficos y el sonido.

COMPONENTES DEL AMIGA

Estos son los componentes del Hardware del Amiga:

- Procesador principal de 16/32 bits Motorola MC68000. El Amiga tambien soporta el 68010, 68020 y el 68030.
- 512 KB de RAM interna, expansible a 1 MB en el A500 y A2000.
- 256 KB de ROM conteniendo un sistema operativo MULTITAREA en tiempo real con un soporte de rutinas de sonido, graficos y animación.
- Disco interno de 3.5 pulgadas con doble cara.
- Port de expansión de disco para conectar hasta tres unidades de discos mas, cada una de ellas puede ser indistintamente de 3.5 o 5.25 pulgadas y doble cara.
- Port serie RS232 completamente programable.
- Port paralelo CENTRONICS completamente programable.
- Un ratón de dos botones (tres opcional).
- Dos ports tipo CANON de 9 pines completamente reconfigurables como 2 ratones, 2 joysticks, 1 lapiz óptico, 2 paddles (ratón de bola orientada hacia arriba y que funciona como ratón), o 2 controladores de fabricación propia...
- Teclado profesional con 10 teclas de función, teclas de cursor y teclado numerico. (Mediante software soporta una gran variedad de teclados internacionales, pudiendose reprogramar totalmente)
- Dos ports para video compuesto y RGB analógico/digital simultaneos.
- Dos ports para salida Estereo de cuatro canales de sonido digital de 8 bits modulados en frecuencia (16 bits) y amplitud (6 bits)
- Opciones de expansión que permiten añadir RAM hasta 9 MB, unidades de disco adicionales (flexibles o duros), perifericos o coprocesadores.

EL MC68000 Y LOS CUSTOM CHIPS DEL AMIGA

El microprocesador Motorola 68000 es de 16/32 bits (32 internos y 16 externos). La velocidad del reloj del sistema para NTSC es 7.15909 MHz y en PAL es de 7.09379 MHz. Estas velocidades pueden variar usando un reloj externo para el sistema, como con un genlock. El 68000 tiene un espacio de direccionamiento de 16 MB (24 bits de bus de direcciones SIN OVERLAY). En el Amiga se usa 0.5 MB para memoria CHIP y 8.5 MB para FAST.

Ademas del 68000, el Amiga contiene el hardware para funciones especiales conocido como los "custom chips" que aumenta considerablemente la potencia del sistema. El termino "custom chips" se refiere a tres circuitos integrados que fueron diseñados especificamente para el Amiga. Estos tres chips (llamados Agnus, Paula y Denise) contienen cada uno la circuiteria para manejar un tipo de funciones como el video, el sonido, el acceso directo a memoria DMA, o los graficos. Entre otras funciones, los custom chips proporcionan lo siguiente:

- Graficos de alta resolución capaces de ser visualizados en pantallas PAL o NTSC standard generados por Bitplanes. Caracteristicas tipicas:

Resolución:	**Baja**	**Entrelazado**	**Media**	**Alta**
Tamaño en PAL:	320 x 256	320 x 512	640 x 256	640 x 512
Tamaño en NTSC:	320 x 200	320 x 400	640 x 200	640 x 400
Colores max.:	32	32	16	16

  Mediante HAM se visualizan 4096 colores simultaneamente y tambien se puede producir overescan (imagenes mas grandes que su tamaño tipico)

- Un coprocesador custom de imagen (Copper) que permite cambios en la mayoria de los registros de hardware en sincronización con la posición del rayo de imagen. Esto permite una gran cantidad de "trucos" tecnicos como visualizar una pantalla con dos paletas a la vez, dividir la pantalla en zonas horizontales (cada una de ellas con diferentes resoluciones, colores), generación de interrupciones hacia el 68000 sincronizadas con la posición del rayo de imagen, etc. El coprocesador puede dispararse varias veces por pantalla, en la mitad de las lineas, y en el comienzo o durante el intervalo de vertical blank. El Copper puede alterar directamente los registros de hardware de los custom chips, liberanndo al 68000 para otras tareas.

- 32 registros de color, cada uno de los cuales contiene 12 bits que se reparten 4 para el ROJO, 4 para el VERDE y 4 para el AZUL, dando sus intensidades. Esto genera una paleta de 4096 combinaciones por cada registro.

- 8 canales DMA para sprites de 16 bits de ancho con un maximo de 15 colores (cuando se agrupan dos). El sprite es el objeto grafico del Amiga mas facil de mover y que ademas es completamente independiente de la pantalla o playfield, siendo posible visualizarlos delante o detras del decorado. El ancho fijo son 16 bits y la altura es ilimitada. Despues de producir la última linea de un sprite, el mismo canal de DMA puede ser usado para producir otro sprite en cualquier otra parte de la pantalla, que este como minimo una linea horizontal por debajo del final del anterior sprite. De esta manera, es posible generar una gran cantidad de sprites volviendo a usar sus canales adecuadamente.

- Prioridad entre objetos controlable dinamicamente, con detección de colisiones. Esto significa que el sistema puede controlar la prioridad en la imagen entre los sprites y los bitplanes o playfields. Se puede controlar el objeto o los objetos que aparecen encima o debajo del decorado en cualquier momento. Ademas se puede usar el hardware para detectar colisiones y hacer responder al programa en consecuencia.

- BIT BLITTER para movimiento de datos a alta velocidad, adaptable a la animación sobre bitplanes. Ha sido diseñado para obtener los datos de tres canales fuente, combinarlos de una de 256 maneras, y almacenar el resultado en un cuarto canal destino. Esta es una de las situaciones donde el 68000 cede ciclos de memoria a un dispositivo DMA que puede hacer el trabajo mas eficientemente (se vera mas tarde). El bit Blitter, puede dibujar lineas en zonas de memoria rectangulares a un millón puntos por segundo, tambien hace el rellenado de areas (fill).

- El Audio consiste en cuatro canales digitales con volumen y frecuencia de reproducción programables independientemente. Los cuatro canales reciven su control y datos a traves del DMA. Una vez han empezado, cada canal producira una forma especifica de onda sin necesidad de NINGUNA intervención del 68000. Los canales estan agrupados dos en la parte derecha y los otros dos en la izquierda (en los sistemas estereo). Se puede enganchar dos canales para producir modulación en frecuencia y/o en amplitud.

- Lectura y escritura de los discos por pistas controladas por DMA. Esto significa que se puede leer 5600 bytes en una sola vuelta del disco (11 sectores de 512 bytes cada uno)

La memoria interna compartida por los custom chips y el 68000 se llama memoria CHIP. Los custom chips originales estaban diseñados para disponer de los primeros 512 KB. La nueva versión de FAT AGNUS es capaz de direccionar con un bit mas, es decir 1 MB de memoria.

Los modelos 500 y 2000 estan diseñados para aceptar el nuevo AGNUS (tienen un zocalo cuadrado) direccionando 1 MB como memoria chip. Sin embargo en el A1000 lo maximo son 512 KB.

Los custom chips y el 68000 comparten la memoria en base a intercambios de acceso a memoria. El 68000 sólo accede a la memoria en ciclos alternativos, quedando los restantes para otras actividades. Los custom chips usan estos ciclos libres, permitiendo al 68000 funcionar a su maxima velocidad la mayor parte del tiempo, porque en algunas ocasiones se "roban" ciclos de memoria del 68000, especialmente por el Copper y el Blitter que necesitan gran velocidad para trabajos que hacen mejor que el 68000. De esta manera el trabajo que se ha de realizar lo ejecuta el elemento del hardware mas adecuado para ello. Cada vez que se "roba" un ciclo, se bloquea sólo el acceso a la memoria CHIP. Siempre que se usa la ROM o la memoria FAST, el 68000 funciona a su maxima velocidad.

Otra de las caracteristicas del hardware del Amiga es la capacidad para controlar dinamicamente la parte de la memoria chip que se usa para imagen, sonido y sprites. El Amiga NO esta limitado a un area pequeña o predeterminada de RAM para la imagen. El sistema permite colocar bitplanes, listas de proceso de sprites, instrucciones del coprocesador, y bloques de sonido para los canales de audio en cualquier parte de la memoria CHIP.

Esta misma región de memoria puede ser accedida por el BLITTER. Esto significa, por ejemplo, que el usuario puede guardar imagenes parciales en distintas areas de memoria chip y usarlas para efectos de animación reemplazandolas rapidamente en la pantalla mientras se guarda y restaura el fondo de la misma. De hecho, el Amiga incluye un soporte de firmware (rutinas en la ROM) para crear animación y movimiento en los playfields.

VCR E INTERFACE DIRECTO DE CAMARA

Ademas de los conectores de video compuesto monocromo, y conexión RGB analogico/digital, puede conectarse tambien un genlock. El sistema es capaz de sincronizarse con una fuente externa de video reemplazando el color 0 (fondo) por el de la imagen exterior. Esto permite titulación de imagenes, sobreimpresión de graficos y animación. Tambien se acepta la entrada de disco laser de la misma manera.

PERIFERICOS

El almacenamiento en discos lo proporciona una unidad interna de 3.5 pulgadas con 80 pistas, doble cara, 11 sectores/pista y 512 bytes/sector (alrededor de 900000 bytes por disco). El controlador puede leer y escribir discos de 5.25 y 3.5 pulgadas de 320/360K IBM PC y 640/720K IBM PC. Tambien se puede añadir unidades externas de 3.5 o 5.25.

La circuiteria de algunos perifericos reside en Paula. Otros chips manejan las señales no asignadas a ninguno de los custom chips, incluyendo control de modem, status del disco, control del cabezal y motor del disco, posibilitamiento de ROM, entrada/salida paralelo, y teclado.

El Amiga incluye un port serie RS232 para dispositivos de entrada/salida.

Un teclado con 10 teclas de función, teclas cursor y teclado numerico. Para la maxima flexibilidad, se envian las señales de "tecla pulsada" y "tecla soltada".El Amiga dispone de una gran cantidad de teclados internacionales.

Se pueden conectar muchos tipos de controladores a los dos ports CANON. Se pueden usar ratones, joysticks, teclados numericos, track-balls, lapiz óptico, volantes tipo coche en cualquiera de los ports (excepto el lapiz óptico que sólo funciona en el port 2).

EXPANSION Y ADPTABILIDAD DEL SISTEMA

Los nuevos perifericos se pueden añadir facilmente en cualquiera de los modelos Amiga. Estos dispositivos se reconocen automaticamente y se usan por el software del sistema a traves de un procedimiento bien definido y documentado llamado AUTOCONFIG.

En los modelos A500 y A1000, los perifericos se añaden al conector de expansión de 86 pines, incluyendo FAST RAM externa. Las unidades de disco extra pueden ser añadidas al conector posterior del ordenador.

En el A2000 se puede hacer lo mismo que en el A500 y A1000, pero con la comodidad de que el conector de expansión es interno y esta repetido 7 veces lo que facilita en gran medida las ampliaciones de la maquina (coprocesadores, FAST RAMs, controladores de disco duro, video o ports de entrada/salida). Tambien hay espacio para dos unidades de disco y un disco duro. El A2000 tambien soporta la targeta BRIDGEBOARD que proporciona un IBM PC completo funcionando a la vez con el AMIGA, y que es completamente compatible con todos los programas de MS-DOS.

SOBRE LOS EJEMPLOS

Los ejemplos en este libro demuestran una manipulación directa del hardware del Amiga. No obstante, como regla general, no se puede acceder directamente, a no ser que el programa tenga el control completo del sistema, o haya avisado al sistema operativo para acceder exclusivamente a un area del hardware.

La mayor parte del hardware explicado en este manual, mas concretamente el Blitter, el Copper, los playfields, los Sprites, los CIA, el disco y el hardware de control del sistema, es de uso exclusivo de porciones del sistema operativo del Amiga. El hardware restante como el Audio, port paralelo y port serie, puede ser usado en aplicaciones que hayan solicitado su uso a traves del sistema operativo.

Antes de proceder a manipular directamente una parte del hardware en el entorno multitarea del Amiga, el programa debe asegurarse de que tiene acceso exclusivo a dicha parte por la libreria, dispositivo o recurso del sistema que controla su funcionamiento. Las funciones del sistema operativo para solicitar y recivir el control de las distintas partes del hardware son muy variadas y no estan dentro del propósito de este manual. Generalmente tales funciones, cuando existen, se pueden encontrar en la libreria, dispositivo o recurso que maneja esa parte del hardware (en el entorno multitarea). La siguiente lista ayuda a encontrar las funciones o mecanismos del sistema operativo que pueden existir para el acceso exclusivo al hardware explicado en este manual.

```
Copper, Playfield, Sprite, Blitter - graphics.library
System Control - graphics.library, exec.library (interrupciones)
Audio      - audio.device
Trackdisk  - trackdisk.device, disk.resource
Serial     - serial.device, misc.resource
Parallel   - parallel.device, cia.resource, misc.resource
Gameport   - input.device, gameport.device, potgo.resource
Keyboard   - input.device, keyboard.device
```

La mayoria de los ejemplos de este libro usan el fichero hw_examples.i (ver Apendice J) para definir los nombres de los registros. Hw_examples.i usa el fichero include del sistema hardware/custom.i para definir las estructuras de los chips y sus direcciones relativas. Los valores definidos en hardware/custom.i y hw_examples.i son offsets de la dirección base del espacio de los chips ($DFF000). En general, esta base esta definida como _custom y se resuelve durante el linking desde amiga.lib (_ciaa y _ciab se resuelven de la misma manera.)

Normalmente, la dirección base se carga en un registro de direcciones y los offsets dados por hardware/custom.i y hw_examples.i son usados por direccionamiento relativo al registro.

Nota: Los offsets de los registros son las direcciones que el Copper usa para referirse a dichos registros (no sus direcciones absolutas).

Por ejemplo, en assembler:

```
        INCLUDE"exec/types.i"
        INCLUDE"hardware/custom.i"

        XREF    _CUSTOM                 ; Referencia externa.

START:  LEA     _CUSTOM,A0              ; Usa A0 como base a los registros
        MOVE.W #$7FFF,INTENA(A0)        ; Desconecta todas las interrupciones
```

Los ficheros include del hardware se proporcionan con el compilador o el ensamblador. Los listados de estos ficheros se pueden encontrar en el libro Addison-Wesley Amiga ROM Kernel Manual "Includes and Autodocs". Generalmente, los nombres de las etiquetas en los ficheros include son muy similares a los nombres equivalentes de los registros de hardware con las siguientes diferencias:

- Los registros de direcciones que se componen de una palabra baja y otra alta, son listados como dos registros de hardware de 16 bits, conteniendo sus nombres un sufijo "L" o "H" para baja y alta respectivamente. En la etiqueta include se trata a todo el registro como una palabra larga (32 bits), y por tanto no contiene "L" ni "H".

- Los registros de hardware que tienen un nombre seguido de un sufijo numerico, estan definidos desde un registro base en los ficheros include. Por ejemplo, los registros del color en la lista del hardware (COLOR00, COLOR01, etc) se utilizan en los ficheros include refiriendose a la base color (color+0, color+2, etc.)

- Los ejemplos de como definir el offset correcto se pueden encontrar en el fichero hw_examples.i del Apendice J.

ALGUNOS CONSEJOS A LOS PROGRAMADORES DE NIVEL HARDWARE

El Amiga esta disponible en una gran variedad de modelos y configuraciones, y mas aún, con gran variedad de ampliaciones, perifericos y procesadores disponibles. Ademas, cada hardware standard Amiga incluye una unidad de discos, suministrada por un fabricante distinto y puede estar sujeta a cambios de sus caracteristicas y tiempos para realizar ciertas operaciones.

El sistema operativo del Amiga esta diseñado para funcionar en el hardware del Amiga, adaptandose a las diferentes configuraciones de RAM, y proporcionando una compatibilidad con futuras ampliaciones del hardware. Para la mayor compatibilidad, se recomienda tratar con el hardware a traves de las funciones del sistema operativo.

Si es necesario programar el hardware directamente, es responsabilidad propia el escribir un programa que funcione con las distintas configuraciones de memoria y modelos existentes. Tambien se debe asegurar el control exclusivo del hardware que se esta manipulando, y llevar un cuidado especial en los siguientes aspectos:

- No saltar a la ROM. Llevar cuidado con los ejemplos que utilizan direcciones en el rango de $F80000 a $FFFFFF. Estas son direcciones de la ROM y las rutinas de la ROM cambian de dirección con cada nueva versión del sistema operativo. La única manera aceptable para utilizar la ROM son las librerias, dipositivos o recursos disponibles.

- No modificar o depender del formato de estructuras del sistema privadas. Esto incluye el "pokeamiento" de listas del Copper, de memoria y de librerias.

- No depender de ninguna dirección fija a una estructura particular del sistema o tipo de memoria. Los módulos del sistema alojan dinamicamente su espacio de memoria cuando se inicializan. Las direcciones de las estructuras del sistema y buffers difieren con cada sistema operativo, cada modelo, y cada configuración, tanto como de la memoria libre y el stack del sistema. Recuerdese que todos los datos para acceso directo de los custom chips han de estar en CHIP RAM. Esto incluye graficos (bitplanes, sprites, bobs, etc), sonidos, buffers del disco y listas del Copper.

- No escribir datos espúreos o interpretar datos de bits o direcciones no definidas del espacio de los custom chips. Todos los bits sin definir deben ponerse a cero al escribirlos e ignorarlos al leerlos.

- No escribir datos pasado el final actual del espacio de los custom chips. Los custom chips podrian ser ampliados para proporcionar registros adicionales, o usar los bits no definidos de los registros ya existentes.

- Todos los custom chips son de sólo-lectura O sólo-escritura. No leer registros de sólo-escritura, ni escribir registros de sólo-lectura.

- No leer, escribir, o usar ninguno de los rangos de direcciones actualmente no definidos. El uso actual y futuro de estas areas esta reservado por Commodore y esta sujeto a cambio.

- Si se estan usando las librerias, dispositivos o recursos del sistema, se debe seguir el procedimiento adecuado. Los programadores assembler (y los escritores compiladores) deben introducir las funciones a traves de la base de las tablas de salto de la libreria (library base jump tables), con los argumentos pasados como palabras largas y la dirección base de la libreria en A6. Los resultados devueltos en D0 deben ser comprobados, y los contenidos de D0-D1/A0-A1 deben ser asumidos como nulos despues de la llamada al sistema.

  NOTA: La instrucción de assembler TAS no debe ser usada en ningún programa de Amiga. Esta instrucción asume un indivisible read-modify-write pero este puede ser anulado por el DMA del sistema. En su lugar se usan BSET y BCLR. Estas instrucciones producen una operación test y set que no son interrumpidas.

  TAS sólo se necesita para un sistema de múltiples CPUs. En un sistema de una sóla CPU, las intrucciones BSET y BCLR son identicas a TAS, porque el 68000 no interrumpe las instrucciones a la mitad. BSET y BCLR primero comprueban los bits y luego los activan.

- Todas las direcciones deben ser de 32 bits. No usar los 8 bits superiores para otros datos, y no usar variables con signo o calculos con signo para las direcciones. No ejecutar programas dentro del stack o usar programas que se auto-modifican porque pueden no funcionar debido a las caracteristicas de algunos microprocesadores 68xxx. Y nunca usar bucles de software dependientes de la velocidad del microprocesador o del reloj para los retardos temporizados. Ver Apendice F para información de cómo usar los temporizadores de los CIA.

  NOTA: Cuando se activa un registro de tipo STROBE, el cual responde a una lectura o escritura, (por ejemplo COPJMP2) se debe usar MOVE.W #$00, y no CLR.W. La instrucción CLR causa una lectura y una escritura (dos accesos) en un 68000, pero un sólo acceso en un 68020 y superiores. Esto dara diferentes resultados en diferentes microprocesadores.

Si se esta programando a nivel de hardware, se debe usar las especificaciones de interconexion adecuadas. Todo el hardware NO es igual. No suponer que los cortes de bajo nivel para velocidad o protección funcionaran en todas las unidades de disco, teclados, sistemas, o futuros sistemas. Comprobar los programas en modelos diferentes, con diferentes CPUs, sistemas operativos y configuraciones de RAM.

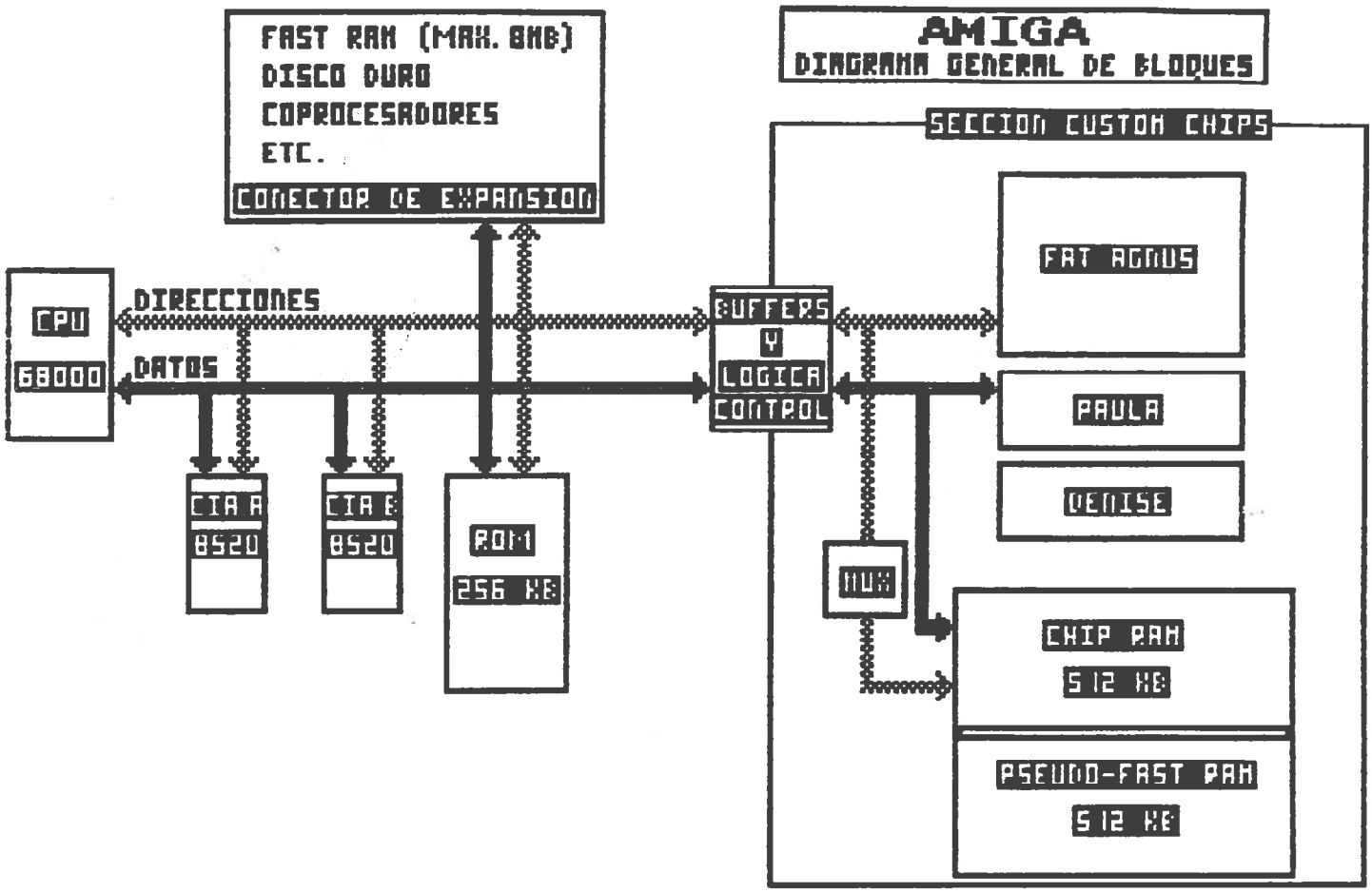


Figura 1-1: DIAGRAMA DE BLOQUES DE LA FAMILIA DE ORDENADORES AMIGA

CAPITULO 2

EL HARDWARE DEL COPROCESADOR

INTRODUCCION

El Copper es un coprocesador de propósito general que reside en uno de los custom chips del Amiga. Recive sus instrucciones por el acceso directo a memoria (DMA). El Copper puede controlar la totalidad del sistema de graficos, liberando al 68000 para ejecutar el programa lógico; tambien puede alterar directamente el contenido de la mayoria de los chips de control. Es una herramienta muy poderosa para dirigir las modificaciones durante la visualización de la pantalla y para controlar los cambios de los registros que ocurren durante los periodos del vertical blank (tiempo entre el final de una pantalla y el comienzo de la siguiente). Entre otras cosas, puede controlar la actualización de los registros, la reposición de los Sprites, los cambios en la paleta de color, la actualización de los canales de audio, y el control del Blitter.

Una de las utilidades del Copper es la capacidad de esperar (WAIT) a una posición especifica del rayo de imagen, y entonces mover datos (MOVE) en un registro del sistema. Durante el periodo de espera (WAIT), el Copper examina el contenido del registro de posición del rayo directamente. Esto significa que mientras el Copper esta esperando, no usa el bus hacia la memoria para nada en absoluto. Por tanto, el bus queda libre para los demas canales de DMA o para el 68000.

Cuando la condición WAIT se cumple, el Copper roba ciclos de memoria al Blitter o al 68000 para mover los datos especificados a los registros señalados.

El Copper es un procesador de dos ciclos que usa el bus sólo durante los ciclos de memoria impares. Esto evita la colisión con el Audio, el Disco, el Refresco (de memoria), los Sprites, y la mayoria de los accesos a las pantallas de baja resolución, los cuales usan sólo los ciclos de memoria impares. El Copper, por tanto, necesita unicamente prioridad sobre el 68000 y el Blitter (canal de DMA que maneja animación, dibujo de lineas, y rellenado de poligonos).

Como pasa con los demas canales de DMA en el Amiga, el Copper sólo puede recivir sus instrucciones del area CHIP de la memoria del sistema.

Sobre este capitulo

En este capitulo, se aprendera cómo usar el conjunto de instrucciones del Copper para organizar los cambios de registros en las pantallas y la puesta a punto de los registros de direcciones durante el intervalo del vetical blank. El capitulo muestra cómo organizar las instrucciones del Copper en Copper lists, cómo usar las Copper lists en modo entrelazado, y cómo usar el Copper con el Blitter. El Copper esta esplicado en este capitulo en forma general. Los capitulos que tratan con Playfields, Sprites, Audio, y el Blitter contienen sugerencias mas especificas para usar el Copper.

¿QUE ES UNA INSTRUCCION DEL COPPER?

Como cualquier coprocesador, el Copper añade su propio conjunto de instrucciones a las que ya tiene el 68000. El Copper solamente tiene tres con las cuales se puede hacer TODO lo anteriormente dicho

- WAIT, espera a una posición especificada por sus coordenadas x e y.
- MOVE, pone un valor en una dirección de los registros CUSTOM.
- SKIP, se salta la siguiente instrucción si el rayo ya ha alcanzado una determinada posición de la imagen.

Todas las instrucciones del Copper constan de 2 palabras (32 bits) en orden secuencial. Cada vez que el Copper trae una instrucción, trae ambas palabras. Debido a que el Copper sólo usa los ciclos impares, se necesitan cuatro ciclos de memoria por instrucción. La instrucción WAIT necesita tres ciclos de memoria impares (seis ciclos de memoria); toma un ciclo extra para abandonar su estado de espera.

Aunque el Copper puede afectar sólo los registros CUSTOM directamente, puede afectar a la memoria mediante una operación con el Blitter. Para mas información sobre cómo usar el Copper controlando al Blitter mirar las secciones: "Registros de Control" y "Usando el Copper con el Blitter."

Las instrucciones WAIT y MOVE se explican a continuación. La instrucción SKIP se describe en la sección "Temas Avanzados".

LA INSTRUCCION MOVE

La instrucción MOVE transfiere un dato desde la RAM a un registro. El dato transmitido es la segunda palabra de la instrucción MOVE; la primera contiene la dirección del registro. Este procedimiento se muestra con detalle en la sección "Sumario de instrucciones del Copper."

PRIMERA PALABRA DE LA INSTRUCCION (IR1)

Bit 0	Siempre puesto a 0
Bits 8-1	Dirección del registro (DA8-1)
Bits 15-9	No usados, deben ser 0.

SEGUNDA PALABRA DE LA INSTRUCCION (IR2)

Bits 15-0	Dato de 16 bits para ser transferido al registro

El Copper puede transmitir datos a los siguientes registros:

- Cualquier registro cuya dirección sea $20 o mayor.
- Cualquier registro cuya dirección este entre $10 y $20 si el bit danger del Copper esta a 1. Este bit esta en el registro de control del Copper, COPCON, que se explica en la sección "Registros de Control."
- El Copper no puede escribir en los registros inferiores a $10.

El Apendice B contiene todas las direcciones de los registros.

El siguiente ejemplo de instrucciones MOVE coloca un $21000 en el registro del bitplane 1 y un $25000 en el registro del bitplane 2.

```
DC.W    $00E0,$0002     ;Mueve $0002 al registro $0E0 (BPL1PTH)
DC.W    $00E2,$1000     ;Mueve $1000 al registro $0E2 (BPL1PTL)
DC.W    $00E4,$0002     ;Mueve $0002 al registro $0E4 (BPL2PTH)
DC.W    $00E6,$5000     ;Mueve $5000 al registro $0E6 (BPL2PTL)
```

Es mejor usar en los listados de asembler los nombres de los registros de hardware (que se encuentran en los ficheros 'include') que usar los registros que sus direcciones, ya que de esta manera los programas son mas facilmente adaptables a las nuevas versiones de hardware. Por ejemplo:

```
INCLUDE "hardware/custom.i"

DC.W    bplpt+$00,$0002   ;Mueve $0002 al registro $0E0 (BPL1PTH)
DC.W    bplpt+$02,$1000   ;Mueve $1000 al registro $0E2 (BPL1PTL)
DC.W    bplpt+$04,$0002   ;Mueve $0002 al registro $0E4 (BPL2PTH)
DC.W    bplpt+$06,$5000   ;Mueve $5000 al registro $0E6 (BPL2PTL)
```

Para el uso de los ejemplos de este manual, hemos definido un fichero include especial (ver Apendice J) que define todos los nombres de los registros del hardware basados en el fichero "hardware/custom.i". Esto se hizo para hacer los ejemplos mas faciles de leer desde el punto de vista del hardware. La mayoria de los ejemplos de este manual se han puesto para ayudar a explicar el hardware y son, en la mayoria de los casos, inútiles si no se modifican y un programa adecuado para hacerlos funcionar.

LA INSTRUCCION WAIT

La instrucción WAIT provoca al Copper una espera hasta que los contadores del rayo sean iguales (o mayores) que las coordenadas especificadas en la instrucción. Mientras espera, el Copper abandona el bus y no usa ciclos de memoria.

La primera palabra de la instrucción contiene las coordenadas verticales y horizontales del rayo. La segunda palabra contiene los bits que se usan para formar una "mascara" que indica al sistema que bits de la posición del rayo usar al hacer la comparación.

PRIMERA PALABRA DE LA INSTRUCCION (IR1)

Bits 15-8	Posición vertical del rayo (VP).
Bits 7-1	Posición horizontal del rayo (HP).
Bit 0	Siempre puesto a 1.

SEGUNDA PALABRA DE LA INSTRUCCION (IR2)

Bit 15	Bit de desconexión del Blitter-finished. Normalmente, esta a 1. (ver la sección "Temas avanzados")
Bits 14-8	Mascara de comparación de la posición vertical (VE)
Bits 7-1	Mascara de comparación de la posición horizontal (HE)
Bit 0	Siempre puesto a 0.

El siguiente ejemplo de instrucción Wait espera a la linea 150 ($96) ignorando la posición horizontal.

```
DC.W    $9601,$FF00        ;Espera a la linea 150, ignora HP.
```

Este otro ejemplo espera a la linea 255 y posición horizontal 254, esto no ocurrira nunca, y el Copper no parara hasta que empiece el siguiente intervalo de vertical blank.

```
DC.W    $FFFF,$FFFE        ; Espera a la linea 255,
                           ; H=254 (fin del la lista del Copper).
```

Para entender por que la posición VP=$FF HP=$FE nunca ocurrira, se debe mirar la operación de comparación del Copper y las restrinciones del tamaño de la posición. La linea número 255 es una linea valida, de hecho es el valor maximo que cabe en este campo. Como es el valor maximo, la siguiente empieza desde cero (la linea 256 aparecera como cero en la comparación). El número de linea no podra ser nunca mayor que $FF. La posición horizontal tiene un valor maximo de $E2. Esto significa que el número mas grande que aparecera en la comparación es $FFE2. Cuando se espera a $FFFE, la linea $FF sera alcanzada, pero la posición horizontal $FE nunca ocurrira. Por tanto, la posición $FFFE nunca sera alcanzada.

Si se intenta esperar una posición horizontal $FE (aunque nunca ocurrira), y poner un valor menor en el campo de la posición vertical. Esto no dara el resultado deseado. La operación de comparación espera que la posición del rayo sea igual o mayor que la indicada. Si la posición vertical no es $FF, entonces tan pronto como el número de linea sea mayor que el número entrado, la comparación dara positivo y la espera terminara.

Las siguientes notas de la posición horizontal y vertical se aplican a las instrucciónes WAIT y SKIP. La instrucción SKIP se explica mas tarde en la sección "Temas Avanzados".

POSICION HORIZONTAL DEL RAYO

La posición horizontal tiene un valor de $0 a $E2. El bit de menos significado no se usa en la comparación, por eso hay 113 posiciones disponibles para las operaciones del Copper. Esto corresponde a 4 pixels en baja resolución y 8 en alta resolución. El "horizontal blanking" cae en las posiciones de $0F a $35. La pantalla standard (320 pixels de ancha) tiene una porción horizontal sin usar desde $04 a $47 (durante la cual sólo se visualiza el color de fondo).

Todas las lineas no tienen la misma longitud en NTSC. Unas lineas son lineas largas (228 ciclos del reloj de color, 0-$E3), y las otras son de 227 ciclos de color. En PAL, son todas de 227 de largo. La pantalla ve todas estas lineas como 227.5 ciclos del reloj de color de largo, mientras el Copper ve las lineas alternativas como lineas largas y cortas.

POSICION VERTICAL DEL RAYO

La posición vertical puede ser resuelta a una linea, con un valor maximo de 255. Hay actualmente 262 posiciones posibles en NTSC y 312 en PAL. Pueden ocurrir algunas pequeñas complicaciones si se quiere que suceda algo en las últimas seis o siete lineas. Porque sólo hay ocho bits de resolución para la posición vertical (dando 256 posiciones differentes), una de las maneras mas simples para manejar esto se muestra a continuación.

Instrucción	Explicación
otras instrucciones...	
WAIT posición (0,255)	En este punto (256), el contador vertical funciona como 0 porque la comparación trabaja con los bits menos significativos del contador vertical.
WAIT cualquier posición horizontal con posición vertical de 0 a 5, cubriendo las últimas 6 lineas de la pantalla antes de que ocurra el vetical blank.	Asi el total de 256 + 6 = 262 lineas del rayo de la imagen durante las cuales, se pueden ejecutar las instrucciones del Copper.

Nota: Las posiciones verticales son como las horizontales - igual que hay lineas largas y cortas alternativamente, tambien hay campos largos y cortos (sólo en interlace). En NTSC los campos son de 262 y 263 lineas y en PAL de 312 y 313 lineas.

Esta alternancia de lineas y campos produce el patrón standard NTSC de cuatro campos, que se va repitiendo:

campo corto terminado en linea corta
campo largo terminado en linea larga
campo corto terminado en linea larga
campo largo terminado en linea corta
y vuelta a empezar...

Un ciclo horizontal es igual a un ciclo del sistema (el 68000 es el doble)

NTSC - 3.579.545 Hz
PAL - 3.546.895 Hz
con genlock- la frecuencia varia un ±2% sobre la del sistema.

LA MASCARA DE COMPARACION

Los bits 14-1 normalmente se encuentran a 1. El uso de estos bits se explica mas tarde en la sección "Temas Avanzados".

USANDO LOS REGISTROS DEL COPPER

Hay algunos registros y direcciones de tipo STROBE dedicadas al Copper:

- Registros de localización
- Direcciones de tipo STROBE (para comenzar el funcionamiento)
- Un registro de control

REGISTROS DE LOCALIZACION

El Copper tiene dos conjuntos de registros de localización:

COP1LCH	3 bits superiores con dirección a la primera Copper list.
COP1LCL	16 bits inferiores con dirección a la primera Copper list.
COP2LCH	3 bits superiores con dirección a la segunda Copper list.
COP2LCL	16 bits inferiores con dirección a la segunda Copper list.

En el acceso al hardware, a menudo se escribe a un par de registros que apuntan a ciertos datos. El registro con la dirección inferior tiene el nombre acabado en "H" y contiene 3 bits, los mas significativos de la dirección. El registro con la dirección mayor tiene el nombre acabado en "L" y contiene 15 bits, los menos significativos. Por lo tanto, se escriben direcciones de 18 bits cuando se mueve una palabra larga a un registro cuyo nombe acaba en "H". Esto es porque cuando se escriben palabras largas con el 68000, la palabra superior va en la dirección inferior.

En el caso de los registros de localización del Copper, se escribe la dirección a COP1LCH. A partir de ahora y para simplificar, nos referiremos a estas direcciones como COP1LC o COP2LC.

Los registros de localización del Copper son dos direcciones indirectas de salto usadas por el Copper. El Copper trae sus instrucciones usando su propio contador de programa, y lo va incrementando despues de cada lectura. Cuando se escribe a una dirección STROBE de salto, el registro de localización correspondiente se mete en el contador de programa del Copper. Esto obliga al Copper a saltar a una nueva localización, desde la cual procesara su siguiente instrucción. El procesamiento de instrucciones continua secuencialmente hasta que el Copper es interrumpido por otro STROBE de salto.

Nota: Al comienzo del vertical blank, se usa COP1LC automaticamente para reponer el contador de programa. Lo cual no afecta al Copper hasta que termina el vertical blank, que es cuando el Copper vuelve a empezar sus operaciones desde la dirección contenida en COP1LC

DIRECCIONES STROBE DE SALTO

Cuando se escribe a una dirección STROBE de salto del Copper, el Copper pone en su contador de programa el registro de localización correspondiente. El Copper puede escribir sus propios registros de localización y hacerlos funcionar (STROBE) para producir saltos programados. Por ejemplo, se puede MOVEr una dirección directa al registro COP2LC. Entonces, cualquier instrucción MOVE que apunte a COPJMP2 producira el salto.

Hay dos direcciones STROBE de salto:

COPJMP1	El Copper vuelve a empezar desde la dirección de COP1LC.
COPJMP2	El Copper vuelve a empezar desde la dirección de COP1LC.

REGISTRO DE CONTROL

El Copper puede acceder a algunos registros siempre, otros cuando un bit especial de control esta puesto a 1, y otros nunca. Los registros a los que el Copper puede acceder siempre estan numerados de $20 a $FF inclusive. Los que no puede acceder nunca van de $00 a $0F (Ver en el Apendice B los registros por orden de direcciones). El registro de control del Copper esta dentro de este grupo ($00-$0F). De esta manera, se necesita la acción deliberada del 68000 para permitir que el Copper escriba en determinados registros.

El registro de control del Copper llamado COPCON, contiene sólo un bit, el bit número 1. Este bit llamado CDANG (Copper DANGer bit) protege a todos los registros numerados entre $10 y $1F inclusive. Este rango incluye los registros de control del Blitter. Cuando CDANG esta a 0, estos registros no pueden ser escritos por el Copper. Cuando CDANG esta a 1, entonces si. Previniendo al Copper de acceder al Blitter se previene la perdida de control sobre el Copper (causada por una lista de instrucciones mal formada) por afectar accidentalmente la memoria del sistema.

NOTA: El bit CDANG es borrado despues de un reset.

PONIENDO JUNTAS LAS INSTRUCCIONES DEL COPPER EN UNA LISTA

La lista de instrucciones del Copper contiene todos los registros que se restauran durante el vertical blank y las modificaciones de registros que se hacen en la pantalla (si son necesarias). Si se esta planeando lo que sucedera durante la visualización de la pantalla, es mas facil si se piensa en cada aspecto diferente como un subsistema separado, como los playfields, los Sprites, el sonido, las interrupciones, y los demas. Se puden hacer listas separadas de las cosas que debe hacer cada subsistema en cada posición del rayo.

Cuando ya se han hecho las listas de todas las cosas que debe hacer el Copper, se deben juntar para formar una única CopperList por cada imagen. Tambien se puede crear la lista final directamente.

Por ejemplo, los punteros a los bit-planes y a los sprites se deben actualizar en cada imagen durante el vertical blank para que los datos sean correctos cuando comience la nueva pantalla. Por ejemplo:

```
WAIT   hasta la primera linea de la imagen.
MOVE   datos al registro del BITPLANE1
MOVE   datos al registro del BITPLANE2
MOVE   datos al registro del SPRITE 1
etc
```

Como otro ejemplo, los canales DMA de los sprites que cran objetos movibles pueden ser vueltos a usar múltiples veces en la misma imagen. Se puede cambiar el tamaño, coordenadas y grafico del registro; aunque normalmente se utiliza el mismo conjunto de colores al reutilizar sprites, tambien estos se pueden alterar esperando hasta la última linea donde se utiliza un registro, para en ella alterar sus colores (que ya no le afectaran a el, sino al siguiente):

```
WAIT   la primera la linea de la imagen.
MOVE   primer color1 a COLOR17
MOVE   primer color2 a COLOR18
MOVE   primer color3 a COLOR19
WAIT   última linea+1 del sprite anterior
MOVE   segundo color1 a COLOR17
MOVE   segundo color2 a COLOR18
MOVE   segundo color3 a COLOR19
etc
```

Cuando se crean CopperLists, la lista final ha de estar en el mismo orden en que el rayo dibuja la imagen. El rayo recorre la pantalla desde la posición (0,0) en la esquina superior izquierda, al final de la pantalla (226,262) en NTSC o (226,312) en PAL en la esquina inferior derecha. Por ejemplo: una instrucción que haga algo en la posición (0,100) deberia ir despues que una que lo hiciera en (0,60)

Nota: Cuando se le da forma a la instrucción WAIT, no siempre ha de ser en este orden estricto, ya que el Copper espera a una posición igual o mayor a la indicada.

Esto significa, por ejmplo, si se dan las siguientes instrucciones:

```
WAIT   posición (64,64)
MOVE   datos

WAIT   posición (60,60)
MOVE   datos
```

El Copper ejecutara los dos MOVEs, aunque en orden incorrecto. La especificación "mayor que" previene al Copper de esperar si el rayo ha pasado ya por la posición especificada. El efecto sería equivalente al del siguiente ejemplo:

```
WAIT  posición (60,60)
MOVE  datos

WAIT  posición (60,60)
MOVE  datos
```

En el instante del segundo WAIT de esta secuencia, los contadores del rayo ya son mayores que la posición especificada. Pero de todas maneras, el segundo MOVE se procesara. Esta última secuencia podria dejarse simplificada como la siguiente: (porque despues de un WAIT pueden haber muchos MOVES)

```
WAIT  posición (60,60)
MOVE  datos
MOVE  datos
```

UNA COPPERLIST COMPLETA DE EJEMPLO

El siguiente ejemplo muestra una CopperList completa. Esta lista es para dos bitplanes -uno en $21000 y el otro en $25000. Al principio los registros de color se cargan con unos valores, y en la linea 150 se sustituyen por otros:

Registro	Color (linea 0)	Color (linea 150)
COLOR00	blanco	negro
COLOR01	rojo	amarillo
COLOR02	verde	azul celeste
COLOR03	azul	magenta

La CopperList completa seria la siguiente:

```
;
; Notas:
;    1. Las CopperLists deben estar en CHIP RAM.
;    2. Las direcciones de los bitplanes de este ejemplo son arbitrarias
;    3. Las direcciones de los registros en las instrucciones MOVE son
;       offsets sobre la dirección base de los custom chips.
;    4. Como siempre, este ejemplo presupone que tiene el control completo
;       del hardware, y no plantea conflictos con el sistema operativo por
;       el uso del mismo hardware
;    5. Algunos de los ejemplos, toman direcciones de memoria para usarlas.
;       Normalmente se necesita reservar el tipo de memoria requerido
;       desde el sistema con AllocMem()
;    6. Como se explicaba antes, los programas de ejmplo estan escritos,
;       no para que funcionen como tales, sino para comprender mejor como
;       trabajan con el hardwre.
;    7. Los siguientes ficheros INCLUDE se necesitan en todos los ejemplos
;       de este capitulo
;
```

```
        INCLUDE "exec/types.i"
        INCLUDE "hardware/custom.i"
        INCLUDE "hardware/dmabits.i"
        INCLUDE "hardware/hw_examples.i"

COPPERLIST:
;
; Puesta a punto de los registros para dos bitplanes
;
        DC.W    BPL1PTH,$0002   ;Mueve $0002 al registro $0E0 (BPL1PTH)
        DC.W    BPL1PTL,$1000   ;Mueve $1000 al registro $0E2 (BPL1PTL)
        DC.W    BPL2PTH,$0002   ;Mueve $0002 al registro $0E4 (BPL2PTH)
        DC.W    BPL2PTL,$5000   ;Mueve $5000 al registro $0E6 (BPL2PTL)
;
; Carga los registros de color
;
        DC.W    COLOR00,$0FFF   ;Mueve blanco al registro $180 (COLOR00)
        DC.W    COLOR01,$0F00   ;Mueve rojo   al registro $180 (COLOR01)
        DC.W    COLOR02,$00F0   ;Mueve verde  al registro $180 (COLOR02)
        DC.W    COLOR03,$000F   ;Mueve azul   al registro $180 (COLOR03)
;
; Especifica 2 bitplanes de baja resolución
;
        DC.W    BPLCON0,$2200   ;2 planos de baja resolución, color on
;
; Espera a la linea 150
;
        DC.W    $9601,$FF00     ;Espera linea 150, ignora pos. horizontal
;
; Cambia los registros de color a mitad de pantalla
;
        DC.W    COLOR00,$0000   ;Mueve negro    al registro $180 (COLOR00)
        DC.W    COLOR01,$0FF0   ;Mueve amarillo al registro $180 (COLOR01)
        DC.W    COLOR02,$00FF   ;Mueve cyan     al registro $180 (COLOR02)
        DC.W    COLOR03,$0F0F   ;Mueve magenta  al registro $180 (COLOR03)
;
; Finaliza la CopperList esperando lo imposible
;
        DC.W    $FFFF,$FFFE     ;Espera VP=255, VH=254 (nunca ocurrira)
```

Para mas información ver el Capitulo 3, "El Hardware de los Playfields".

BUCLES Y BIFURCACIONES

Los bucles y las bifurcaciones en las CopperLists se explican en la sección posterior "Temas Avanzados"

ARRANCANDO Y PARANDO EL COPPER

Arrancando el Copper despues de un reset. En la conexión de corriente o en el reset de software, se debe inicializar uno de los registros de localización del Copper (COP1LC o COP2LC) y escribir en su correspondiente registro STROBE, antes de conectar el acceso del Copper por DMA. Esto asegura una dirección de arranque conocida y un estado tambien conocido. Normalmente, se usa COP1LC porque es este registro el que se reutiliza en cada vertical blank. La siguiente secuencia de instrucciones muestra como inicializar un registro de localización. Se supone que el usuario ya ha creado una CopperList con dirección "mycoplist."

```
; Instala la CopperList

      LEA     CUSTOM,a1           ; a1 = dirección de los custom chips
      LEA     MYCOPLIST(pc),a0    ; dirección de la CopperList
      MOVE.L  a0,COP1LC(a1)       ; escribe la dirección
      MOVE.W  COPJMP1(a1),d0      ; El Copper carga su PC desde COP1LC

; Entonces se conecta el DMA del Copper y de la imagen (raster)

      MOVE.W  #(DMAF_SETCLR!DMAF_COPPER!DMAF_RASTERDMAF_MASTER),DMACON(a1)
```

Ahora, si el contenido de COP1LC no se cambia, cada vertical blank que ocurra, el Copper volvera a comenzar desde la misma localización para cada imagen. Esto forma un bucle repetitivo, que si la lista esta bien formada, producira una pantalla estable.

PARANDO EL COPPER

No hay instrucción de parada en el Copper. Para asegurarse que no haga nada hasta el final de la imagen y el contador de programa (PC) empiece de nuevo con la CopperList, la última instrucción debe ser un WAIT a algo que nunca ocurra. El tipico WAIT es para VP=$FF y HP=$FE, ya que un HP mayor que $E2 es imposible. Cuando la imagen termina y comienza el vertical blank, el Copper automaticamente es llevado al comienzo de la CopperList, y esta instrucción final WAIT nunca llega a acabar.

Tambien se puede detener el Copper desconectando su canal de DMA. El registro DMACON controla todos los canales DMA. El Bit 7, COPEN, conecta el canal DMA del Copper cuando se pone a 1. Para mas información sobre el control del DMA, ver el Capitulo 7, "El Hardware de Control del Sistema."

TEMAS AVANZADOS

LA INSTRUCCION SKIP. Esta instrucción hace que el Copper salte la siguiente instrucción si los contadores del rayo son iguales o mayores que el valor especificado en la misma.

Los contenidos de las palabras de la instrucción SKIP se muestran abajo. Son identicos a los de la instrucción WAIT, excepto que el bit 0 de la segunda palbra es un 1 para identificarla como instrucción SKIP.

PRIMERA PALABRA DE LA INSTRUCCION (IR1)

Bits 15-8	Posición vertical del rayo (VP).
Bits 7-1	Posición horizontal del rayo (HP). Salta si el contador del rayo es igual o mayor que estos bits combinados (bits del 15 al 1)
Bit 0	Siempre puesto a 1.

SEGUNDA PALABRA DE LA INSTRUCCION (IR2)

Bit 15	Bit de desconexión del Blitter-finished. (ver "Usando el Copper con el Bliter mas tarde")
Bits 14-8	Mascara de comparación de la posición vertical (VE)
Bits 7-1	Mascara de comparación de la posición horizontal (HE)
Bit 0	Siempre puesto a 1.

Las notas sobre las posiciones horizontal y vertical de la instrucción WAIT también se aplican a la instrucción SKIP.

El siguiente ejemplo salta la instrucción siguiente si VP es mayor o igual a 100 ($64).

```
DC.W  $6401,$FF01    ; Si VP >= 100, salta (ignora HP)
```

BUCLES Y BIFURCACIONES DEL COPPER Y MASCARA DE COMPARACION

Se puede alterar el valor de los registros de localización en cualquier momento y construir bucles en las CopperLists. Antes de que se produzca el siguiente vertical blank, el registro COP1LC debe ser restaurado a su valor normal (comienzo de CopperList). El valor de COP1LC sera colocado en el contador de programa del Copper al comienzo del vertical blank.

Los Bits 14-1 de la segunda palabra (mascara) en las instrucciones WAIT y SKIP especifican los bits de las posiciones horizontal y vertical que se usan en la comparación con el rayo. La posición (en la primera palabra) y la mascara (en la segunda palabra) se juntan (función AND) y se compara con la posición actual de los contadores del rayo antes de que se produzca una acción posterior. Cada bit de posición de la primera palabra es usado para ser comparado con los contadores del rayo **si y solo si** el bit correspondiente en la mascara es 1. Por ejemplo, si sólo interesa el valor de los 4 últimos bits de la posición vertical, sólo se activaran los 4 últimos bits en la mascara de comparación, bits (11-8) de la segunda palabra.

No se pueden enmascarar todos los bits de los contadores del rayo. Si se mira la descripción de IR2 (segunda palabra) se observa que el bit 15 es el bit de desconexión del blitter-finished. Este bit ya no es parte de la mascara de comparación, y no tiene un significado propio en la instrucción WAIT. Por eso no se puede enmascarar este bit en las instrucciones WAIT y SKIP. Aunque muchas veces esta limitación nó plantea ninún problema, en el siguiente ejemplo si y se muestra como solucionarlo.

Este ejemplo hace que el Copper genere una interrupción cada 16 lineas. Podria parecer que la forma para hacer esto es usar una mascara de $0F y entonces comparar el resultado con $0F. Esta comparación daria resultado "verdadero" para $1F, $2F, $3F, etc. Como la comparación es para igual o mayor que, esto pareceria permitir detectar cada 16 lineas. Pero como el bit mayor no puede ser enmascarado, itervendria en las comparaciones. Cuando el Copper esta esperando a $0F y la posición vertical ya ha llegado a 128 ($80), la comparación daria siempre "verdadera". En este caso, el valor minimo de la comparación sera $80, que es siempre mayor que $0F, y la interrupción pasaria a producirse cada linea. Recuerdese que el Copper comprueba igual o mayor que.

En el siguiente ejemplo, la CopperList se ha hecho en forma de bucle. Los registros COP1LC y COP2LC los coloca el 68000 o una seción de CopperList anterior a esta. Tambien se supone que se ha instalado un programa para gestionar la interrupción que genera el Copper cada 16 lineas de modo no-entrelazado.

Cómo funciona: Ambos bucles son, en gran parte, lo mismo. En cada uno, el Copper espera a que la posición vertical sea $?F (? es cualquier número hexadecimal), y en ese momento el Copper genera una interrupción en el Hardware del Amiga. Para estar seguros de que el Copper no repita el bucle en la misma linea y genere otra interrupción, se le hace esperar a la posición horizontal $E2 despues de la interrupción ya que es la última HP para cada linea. Esto fuerza al Copper a la siguiente linea antes del WAIT. El bucle se ejecutá escribiendo al registro COPJMP1 haciendo que el Copper comience desde la dirección que se ha colocado en COP1LC.

El problema de la mascara descrito antes se soluciona separando la Copperlist en dos bucles, uno para VP<127 y el otro para VP>=127. Cuando VP>=127, el Copper se salta (SKIP) la instrucción que hace el primer bucle, haciendo que el Copper caiga en el segundo bucle. El segundo bucle es muy parecido al primero, excepto que se espera a $?F con el bit alto a 1 (1xxx1111 en binario). Esto se hace con el VP de las dos instrucciones WAIT. El segundo bucle se produce escribiendo en al registro COPJMP2. La lista se lleva a un WAIT al infinito cuando VP>=255 que terminara antes del vertical blank. Al final del vertical blank el sistema operativo restaura el contenido de COP1LC, haciendo que el primer bucle comience de nuevo.

Nota: El registro COP1LC se restaura al final del vertical blank por un handler grafico de interrupciones que forma parte de una cadena de "servers" de interrupciones del vertical blank. Mientras este server este intacto, COP1LC sera restaurado correctamente al final de cada vertical blank.

```
; Estos son los datos de la CopperList
; Se supone que COPPERL1 ha sido cargado en COP1LC y
; que COPPERL2 ha sido cargado en COP2LC por algún otro programa
;
COPPERL1:   DC.W    $0F01,$8F00     ; WAIT a VP=0xxx1111
            DC.W    INTREQ,$8010    ; Activa bit de interrupción del Copper

            DC.W    $00E3,$80FE     ; WAIT a VH=$E2 para que termine la
                                    ; linea antes de empezar otra vez

            DC.W    $7F01,$7FE01    ; SKIP si VP>=127 (se salta el bucle)
            DC.W    COPJMP1,$0      ; Fuerza el salto a COP1LC (bucle)

COPPERL2:   DC.W    $8F01,$8F00     ; Espera a VP=1xxx1111
            DC.W    INTREQ,$8010    ; Activa bit de interrupción del Copper

            DC.W    $80E3,$80FE     ; WAIT a VH=$E2 para que termine la
                                    ; linea antes de empezar otra vez

            DC.W    $FF01,$7FE01    ; SKIP si VP>=255 (se salta el bucle)
            DC.W    COPJMP2,$0      ; Fuerza el salto a COP2LC (bucle)

; Aqui se necesitaria otra lista para cubrir la totalidad de las 262
; lineas de NTSC o las 312 de PAL, ya que esta CopperList termina en 255

            DC.W    $FFFF,$FFFE     ; Final de la CopperList
```

USANDO EL COPPER EN MODO ENTRELAZADO

Una imagen en entrelazado tiene dos veces el número normal de lineas verticales en la pantalla. En NTSC pasa de 262 (no-lace) a 524 (interlace), en PAL pasa de 312 (no-lace) a 625 (interlace). En el modo interlace el rayo recorre la pantalla dos veces por imagen, durante la primera vez se visualizan las lineas impares, durante la segunda, las lineas pares entrelazandose con las primeras que aún estan en el fósforo de la pantalla y en la retina del ojo. La circuiteria que maneja esto trata la imagen entrelazada como dos imagenes separadas, una con las lineas impares y la otra con las pares. La figura 2-1 muestra como se almacena en memoria una pantalla entrelazada.

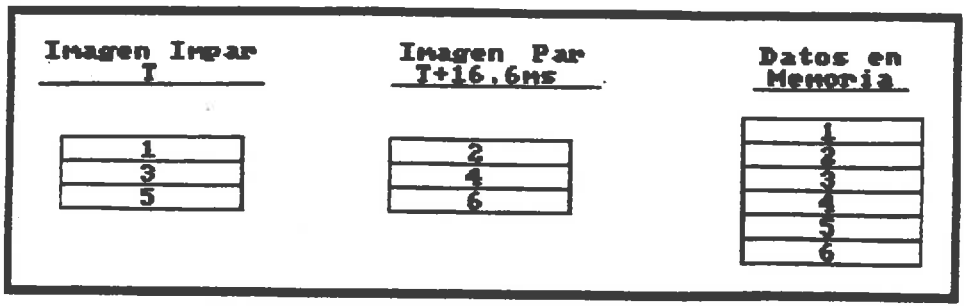


Figura 2-1: Bit-plane entrelazado en RAM

El sistema obtiene datos para visualizar el bit-plane usando punteros a la dirección de los datos en memoria. Como se puede ver, la dirección inicial para la imagen par es una linea mayor a la de la imagen impar. Por tanto, el puntero al bitplane debe contener diferentes valores para las imagenes alternativas de la pantalla entrelazada.

Simplemente, la organización de la información en memoria es la misma que la de la pantalla (las lineas pares e impares estan entrelazadas juntas). Esto se hace teniendo dos CopperList, una para cada imagen.

Para hacer que el Copper ejecute la lista correcta, se envia una interrupción al 68000 despues de la primera linea de la imagen. Cuando el 68000 ejecuta la interrupción, cambia el contenido de COP1LC para que apunte a la segunda lista. Despues, durante el vertical Blank, COP1LC volvera a ser apuntado hacia la primera lista.

Para mas información sobre las imagenes entrelazadas ver el Capitulo 3, "El Hardware de los Playfields".

USANDO EL COPPER CON EL BLITTER

Si se usa el Copper para poner a punto una secuencia de operaciones con el Blitter, debe esperar a la interrupción de "blitter-finished" antes de volverlo a utilizar, ya que cambiar los registros del Blitter mientras esta funcionando causa resultados impredicibles. Para este propósito, la instrucción WAIT incluye un bit adicional de control, llamdo BFD (blitter finished disable). Normalmente, este bit esta a 1 y WAIT sólo controla comparaciones con los contadores del rayo.

Cuando el bit BFD se pone a 0, se modifica la lógica de la instrucción WAIT. El Copper esperara hasta que la comparación con el rayo sea "verdad" y el Blitter haya finalizado. El Blitter habra finalizado cuando el indicador de blitter-finished sea 1. Este bit ha de utilizarse con cuidado, podria hacer que no se visualizaran las pantallas o evitar que los objetos se visualizaran correctamente. Para mas información sobre el uso del Blitter, ver el capitulo 6, "El Hardware del Blitter"

EL COPPER Y EL 68000

En aquellas ocasiones en que las instrucciones del Copper no sean suficientes, se pueden enviar interrupciones al 68000 para usar su conjunto de interrupciones. El 68000 puede mirar los flags del registro de interrupciones INTREQ para averiguar el que le ha interrumpido. Para interrumpir al 68000, se usa la instrucción MOVE del Copper para poner a 1 los bits 15 (SET/CLR, determina si los bits puestos a 1 deben ser activados o borrados) y 4 (COPEN, bit de interrupción del Copper al 68000) de INTREQ. Ver capitulo 7, "El Hardware de Control del Sistema" para mas información sobre las interrupciones.

SUMARIO DE LAS INSTRUCCIONES DEL COPPER

La siguiente tabla muestra un sumario del estado de los bits para cada instrucción del Copper. Ver en el Apendice A el sumario de todos los registros de los custom chips.

Tabla 2-1: Sumario de las instrucciones del Copper

	MOVE		WAIT		SKIP	
Bit	IR1	IR2	IR1	IR2	IR1	IR2
15	X	RD15	VP7	BFD	VP7	BFD
14	X	RD14	VP6	VE6	VP6	VE6
13	X	RD13	VP5	VE5	VP5	VE5
12	X	RD12	VP4	VE4	VP4	VE4
11	X	RD11	VP3	VE3	VP3	VE3
10	X	RD10	VP2	VE2	VP2	VE2
09	X	RD09	VP1	VE1	VP1	VE1
08	DA8	RD08	VP0	VE0	VP0	VE0
07	DA7	RD07	HP8	HE8	HP8	HE8
06	DA6	RD06	HP7	HE7	HP7	HE7
05	DA5	RD05	HP6	HE6	HP6	HE6
04	DA4	RD04	HP5	HE5	HP5	HE5
03	DA3	RD03	HP4	HE4	HP4	HE4
02	DA2	RD02	HP3	HE3	HP3	HE3
01	DA1	RD01	HP2	HE2	HP2	HE2
00	0	RD00	1	0	1	1

X - Sin usar, debe ser 0 para posterior compatibilidad.
IR1 - Primera palabra de la instrucción.
IR2 - Segunda palabra de la instrucción.
DA - Dirección de destino.
RD - Dato para mover al registro destino.
VP - Posición vertical del rayo.
HP - Posición horizontal del rayo.
VE - Mascara vertical de comparación.
HE - Mascara horizontal de comparación.
BFD - Bit de Blitter-finished (se activa a 0).

CAPITULO 3

EL HARDWARE DE LOS PLAYFIELDS

INTRODUCCION

La pantalla del Amiga consiste de dos partes basicas -los playfields, a los que a veces se les llama backgrounds, y los sprites, que son objetos graficos de facil movimiento. Este capitulo describe como acceder directamente a los registros de hardware para formar playfields.

SOBRE ESTE CAPITULO

Este capitulo comienza con un breve vistazo de las caracteristicas de los playfields, incluyendo las definiciones de algunos terminos fundamentales, y continua con los siguientes grandes temas:

- Formando un playfield simple, del mismo tamaño que la pantalla. Esta sección incluye conceptos que son fundamentales para formar playfields.

- Formando una pantalla de doble playfield (un playfield esta superpuesto al otro). Esto difiere de lo anterior en algunos detalles.

- Formando playfields de diferentes tamaños y visualizando parte del playfield (que es mas grande que la pantalla).

- Moviendo playfields mediante scrolls horizontales y/o verticales.

- Temas avanzados de ayuda de los playfields en situaciones especiales.

Para mas información sobre los Sprites, ver el Capitulo 4, "El Hardware de los Sprites". Tambien hay objetos móviles que forman parte del playfield llamados BOBs (Blitter OBject). Ver Capitulo 6, "El Hardware del Blitter".

CARACTERISTICAS DE LOS PLAYFIELDS

El Amiga genera la imagen de video mediante tecnicas de visualización por ratreo (RASTER). La imagen que se ve en la pantalla esta hecha de una serie de lineas horizontales visualizadas unas despues de las otras. Cada linea horizontal esta hecha de una serie de pixels. Las imagenes se crean definiendo uno o mas bit-planes en la memoria y poniendo sus bits a 1 o 0. La combinación de los unos y ceros determina los colores de cada pixel.

El rayo produce 262 lineas desde arriba a abajo, de las cuales 200 son visibles en la pantalla con un sistema NTSC. Con un sistema PAL, el rayo produce 312 lineas, de las cuales son visibles normalmente 256. Cada conjunto completo de lineas (262 en NTSC o 312 en RAL) es llamado FRAME o display field. El tiempo de la imagen, es decir el tiempo necesario para completar un FRAME es aproximadamente 1/60s en NTSC y 1/50s (2 decimas) en PAL. Entre cada FRAME el rayo traza las lineas que no son visibles en la pantalla y vuelve a la parte superior para producir un nuevo FRAME.

El area visible esta definida como una rejilla de pixels. Un pixel es el elemento mas pequeño de una imagen. Las figuras de la siguiente pagina muestran cómo se produce la imagen en la pantalla, qué es un pixel y la cantidad de ellos que caben en una sola linea.

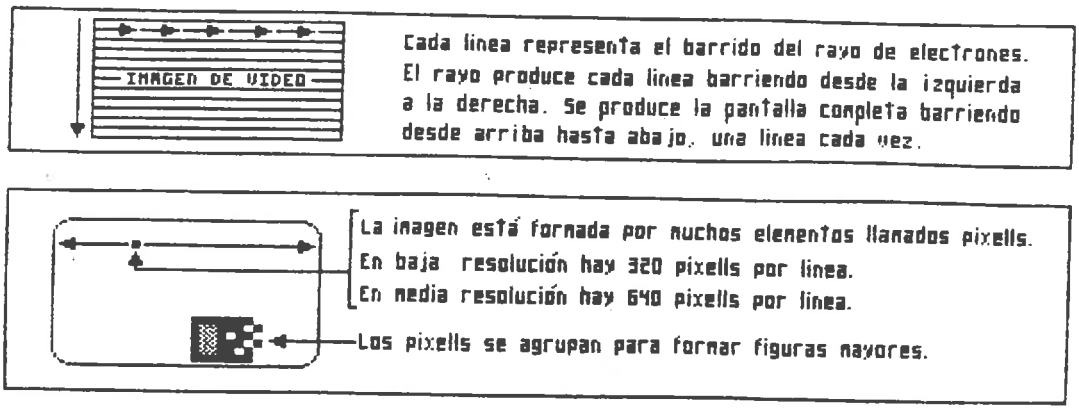


Figura 3-1: Cómo se produce la Imagen y ¿que es un pixel ?.

El Amiga ofrece la posibilidad de elegir en las resoluciones vertical y horizontal. La resolución horizontal puede ser ajustada para operar en modo de baja o alta resolución. La resolución vertical puede ser ajustada para operar en modo entrelazado o no-entrelazado.

- En modo de baja-resolución, un playfield normal tiene 320 pixelS de ancho.

- El modo de alta-resolución da mayor resolución horizontal -640 pixelS de ancho del playfield en la misma area de imagen.

- En modo no-entrelazado, un playfield normal en NTSC tiene una altura de 200 lineas. En PAL tiene una altura de 256 lineas.

- El modo entrelazado da mayor resolución vertical -400 lineas en la misma pantalla en NTSC o 512 en PAL.

Estos modos pueden combinarse, por tanto se puede tener, por ejemplo, una imagen entrelazada y con alta-resolución (640 x 400 NTSC o 640 x 512 PAL)

Nótese que las dimensiones referidas como "normal" en el parrafo anterior son dimensiones nominales y representan los valores normales que se suelen usar. La verdad es que se pueden visualizar playfields mayores; las dimensiones maximas se dan en la sección llamada "Bitplanes y Playfields de todos los tamaños". Tambien, aveces las dimensiones del playfield en memoria son mayores que las que se estan visualizando en la pantalla. Se puede decidir que parte de esta gran imagen sera visualizada especificando diferentes tamaños para la ventana de visualización.

Se puede hacer scroll a un playfield mas largo que la pantalla (moverlo suavemente hacia arriba o abajo). Tambien se puede hacer scroll horizontalmente a un playfield mas ancho que la pantalla (moviendolo de izquierda a derecha o viceversa). El scroll se explica en la sección llamada "Moviendo (scrolling) Playfields."

En el sistema de graficos del Amiga, se pueden tener 32 colores diferentes en un único playfield usando metodos normales. Se puede controlar el color de cada pixel independientemente, actuando sobre el bit o los bits que definen su combinación de 32 colores. Una imagen formada de esta manera se llama una imagen de mapeado de bits (bit-mapped display).

Por ejemplo, en un playfield de dos colores, el color de cada pixel esta determinado por si un sólo bit es 1 ó 0. Si el bit es 0, el pixel tiene un color definido por el usuario, si el bit es 1, el pixel tiene otro color. Para un playfield de cuatro colores, se construyen dos bit-planes en memoria. Cuando se visualiza el playfield, se superponen los dos bit-planes de manera que cada pixel tiene dos bits de profundidad. Se pueden combinar

hasta 5 bit-planes consiguiendo 32 colores (colores = 2^bitplanes, por ejemplo: 3 bit-planes-> 8 colores, 4 bitplanes-> 16 colores).

El color de un pixel esta siempre determinado por la combinación en binario de los bits que le definen. Cuando el sistema combina los bit-planes en la pantalla, la combinación formada para cada pixel corresponde al número de un registro de color. Este metodo de colorización de pixel se llama "color indirection". El Amiga tiene treinta y dos registros de color, cada uno conteniendo los bits para definir el color seleccionado por el usuario (de un total de 4096 colores posibles).

La Figura 3-2 muestra como la combinación de hasta 5 bit-planes selecciona uno de los 32 registros de color que se usa para visualizar cada pixel . Los bits del bit-plane de mayor orden ocupan los bits mas significativos en la combinación binaria.

Tambien se tiene la posibilidad de definir dos playfields separados, cada uno de ellos formado por hasta 3 bit-planes. Cada uno de los playfields usa un conjunto de 8 colores independientes. Esto se llama modo de doble playfield (dual-palyfield mode).

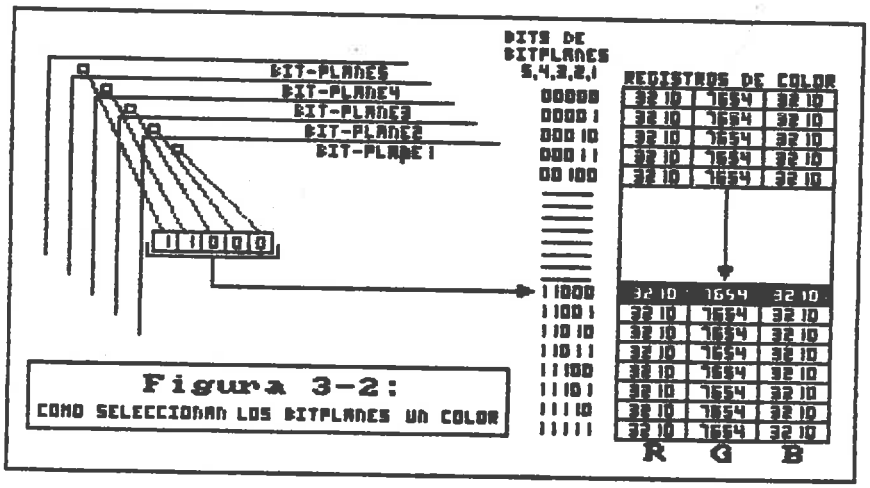


Figura 3-2:
COMO SELECCIONAN LOS BITPLANES UN COLOR

FORMANDO UN PLAYFIELD BASICO

Esta sección describe cómo acceder directamente a los registros de hardware para formar un único y basico playfield que sera del mismo tamaño que la pantalla. Aqui, "el mismo tamaño" significa que el palyfield sera del mismo tamaño que la imagen normal. Quedara un pequeño espacio entre el playfield y el borde de la pantalla, ya que normalmente son asi.

Para definir un playfield, se necesitan definir las siguientes caracteristicas:

- La altura y la anchura del playfield y el tamaño de la ventana de visualización (es decir, "cuanto" playfield aparecera en la pantalla)
- El color de cada pixel del playfield.
- La resolución Horizontal.
- La resolución vertical (entrelazado).
- La transmisión de los datos y el módulo, que indican al sistema la cantidad de datos que ha de poner por linea y cómo llevarlos desde la memoria a la pantalla.

Ademas, se necesita reservar la memoria que contendra el playfield, poner los punteros que le indican al sistema dónde se encuentran los datos en la memoria, y (opcionalmente) escribir una CopperList para manejar la visualización continuada del mismo playfield.

ALTURA Y ANCHURA DEL PLAYFIELD

Para crear un playfield que sea del mismo tamaño que la pantalla, se puede usar un tamaño de 320 o 640 pixels, dependiendo de la resolución que se elija. La altura puede ser de 200 o 400 lineas en NTSC, 256 o 512 en PAL, dependiendo de si se activa o no el modo entrelazado.

LOS BIT-PLANES Y EL COLOR

El color de un playfield se define por:

- Decidiendo cuantos colores se necesitan y como se coloreara cada pixel .
- Cargando los colores en los registros de color.
- Reservando la memoria para el número de bit-planes que se necesitaran y poniendo un puntero a cada bit-plane (un registro de hardware).
- Escribiendo instrucciones para poner un Valor en cada bit de los bit-planes y de esta manera dar los colores correctos a cada pixel .

Tabla 3-1: Colores en un único playfield:

Número de bit-planes:	1	2	3	4	5
Número de colores:	1-2	3-4	5-8	9-16	17-32

La Tabla de Color (o paleta) contiene 32 registros, y se puede cargar un color diferente en cada uno de ellos. Esta es la tabla simplificada:

Tabla 3-2: Porción de la Tabla de Color

Nombre	**Contenido**	**Significado**
COLOR00	12 bits	Color para el fondo y el borde.
COLOR01	12 bits	Color número 1.
COLOR02	12 bits	Color número 2.
.	.	.
.	.	.
COLOR31	12 bits	Color número 31.

El COLOR00 siempre esta reservado para el color de fondo. El color de fondo se ve en cualquier zona del playfield donde no haya objetos y tambien en el area del borde.

Nota: Si se usa genlock con entrada desde una camara, video o disco laser, el color de fondo se reemplazara por la imagen la fuente conectada.

Los doce bits para la selección del color permiten definir para cada uno de los 32 registros, uno de los 4096 colores posibles:

Tabla 3-3: Contenido de los Registros de Color

Bits:	15-12	11-8	7-4	3-0
Función:	Ninguna	ROJO	VERDE	AZUL

Tabla 3-4: Ejemplo de valores para los Registros de Color.

Contenido del registro:	$FFF	$6FE	$DB9	$000
Color resultante:	Blanco	Celeste	Tostado	Negro

Instrucciones de ejemplo para cargar los registros de color:

```
LEA     CUSTOM,a0               ; Obtiene la base de los custom chips
MOVE.W #$FFF,COLOR00(a0)        ; Carga el blanco en el registro 0
MOVE.W #$6FE,COLOR01(a0)        ; Carga el azul celeste en el registro 1
```

Nota: Los registros de color son de sólo-escritura. Solamente mirando a la pantalla es posible saber sus contenidos. Como practica general, para estos y otros registros de sólo-escritura, se suele hacer una copia de su contenido en RAM, de manera que cada vez que se modifican, tambien se actualiza la copia de RAM. De esta manera siempre se podra saber el contenido de cada registro.

Seleccionando el número de Bit-Planes. Despues de decidir cuantos colores se quieren y cuantos bit-planes se necesitan para proporcionar estos colores, se debe indicar al sistema cuantos bit-planes se van a usar.

El número de bit-planes se selecciona escribiendo su número en el registro BPLCON0 (Bit Plane Control Register 0). Los bits utilizados son el 14, 13 y 12, llamados BPU2, BPU1 y BPU0 (Bit Planes Used). La Tabla 3-5 muestra los valores a escribir en estos bits y cómo el sistema los asigna al número de bit-planes.

Tabla 3-5: Eligiendo el Numero de Bit-Planes

Valor	**Bit-Planes**	**Bitplanes implicados**
000	0 *	
001	1	Bit-Plane 1
010	2	Bit-Planes 1 y 2
011	3	Bit-Planes del 1 al 3
100	4	Bit-Planes del 1 al 4
101	5	Bit-Planes del 1 al 5
110	6	Bit-Planes del 1 al 6 **
111		Valor no usado

* Muestra únicamente el color de fondo; no se ven playfields.

** Unicamente se usan seis bit-planes en los modos de doble playfield y Hold-And-Modify (HAM), descritos en la sección llamada "Temas Avanzados"

Nota: Los bits en el registro BPLCON0 no pueden activarse independientemente. Para activar alguno de ellos, se deben reactivar todos los demas.

El siguiente ejemplo muestra como hacer que el sistema use 2 bit-planes de baja resolución.

```
MOVE.W    #$2200,BPLCON0+CUSTOM     ; Escribe los datos
```

Debido a que el registro BPLCON0 se usa para controlar otras caracteristicas de la imagen y los bits no son independientes, el ejemplo anterior desactivaria otros parametros que se describiran mas tarde en este capitulo.

- Se desactiva el modo HAM

- Se activa el modo de playfield-único.

- Se conecta el color del video compuesto (no aplicable en todos los modelos)

- Se desconecta el audio del genlock.

- Se desconecta el modo de Lapiz Optico.

- Se desconecta el modo entrelazado.

- Se desconecta la resincronización externa (genlock).

SELECCIONANDO LA RESOLUCION HORIZONTAL Y VERTICAL

Las imagenes de baja resolución son las mas apropiadas para televisores domesticos. El modo de baja resolución proporciona 320 pixels por cada linea. Los monitores monocromos de alta resolución y los monitores color pueden producir imagenes en el modo de alta resolución que proporciona 640 pixels por linea. Si se visualiza un objeto de baja resolución en el modo de alta resolución se vera la mitad de ancho.

Para modificar el modo de resolución horizontal, se actua sobre el bit 15, HIRES, del registro BPLCON0 (0-baja resolución, 1-Alta resolución). Nótese que en el modo de alta resolución, se puede tener sólo hasta 4 bit-planes en el playfield y por tanto, hasta 16 colores.

El modo entrelazado permite alojar el doble de datos para visualizarse en la misma area vertical que el modo no-entrelazado. Esto se hace doblando el número de lineas de la imagen. La siguiente tabla muestra el número de lineas para rellenar una pantalla normal, no overescan:

Tabla 3-6: Lineas de un Playfield Normal

	NTSC	PAL
No-entrelazado	200	256
Entrelazado	400	512

En el modo entrelazado, la circuiteria modifica el comienzo de las lineas en los campos alternativos a la posición normal mas media linea.

Aunque el modo entrelazado requiere un poco de trabajo extra poniendo a punto ciertos registros (como se vera mas tarde en esta sección), proporciona mayor precisión, sobre todo para algunos tipos de graficos. Comparando la linea diagonal de la Figura 3-4 en modo no-entrelazado con la que esta en modo entrelazado, se observa que desaparece gran parte del efecto de dientes de sierra en los bordes de la linea.

Cuando se usa el canal especial de DMA llamado Blitter para dibujar lineas o poligonos en un playfield entrelazado, este sera tratado mas como una imagen que como campos pares e imapares. Por lo tanto, se seguira teniendo el efecto de antialiassing en los bordes de los objetos dibujados.

Para conectar y desconectar el modo entrelazado, se actúa sobre el bit 2, LACE del registro BPLCON0 (0-> no-entrelazado, 1-> entrelazado). Como se explicaba antes en "seleccionando el número de bit-planes", los bits del registro BPLCON0 no se pueden activar independientemente.

El siguiente ejemplo especifica los modos de alta resolución y entrelazado:

```
MOVE.W    #$A204,BPLCON0+CUSTOM     ; Escribe los datos
```

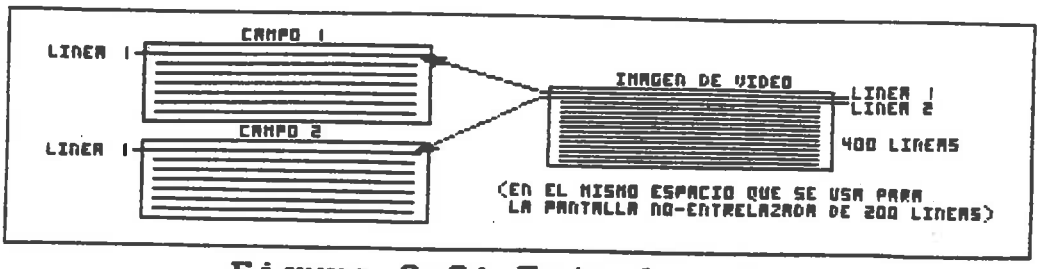


Figura 3-3: Entrelazado

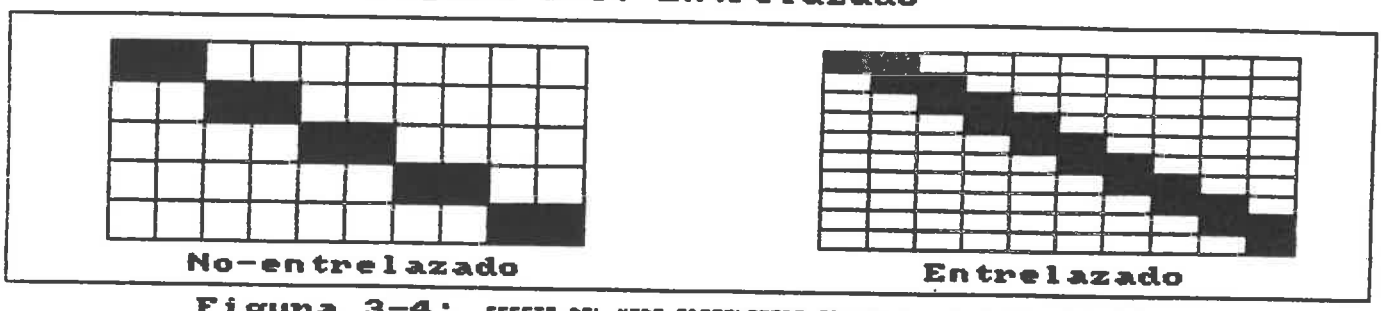


Figura 3-4: EFECTO DEL MODO ENTRELAZADO EN LOS BORDES DE LOS OBJETOS

El ejemplo anterior tambien actúa sobre los siguientes parametros que estan controlados por el registro BPLCON0:

- Se conecta el modo de alta resolución.
- Se usan dos bit-planes.
- Se desconecta el modo HAM.
- Se conecta el modo de único-playfield.
- Se conecta el modo de color en video compuesto.
- Se desconecta el audio del genlock.
- Se desconecta el lapiz óptico.
- Se conecta el modo entrelazado.
- Se desconecta la resincronización externa.

La cantidad de memoria que se necesita reservar para cada bit-plane depende de los modos de resolución que se hayan seleccionado, porque los playfields de alta resolución o entrelazados contienen mas datos y por tanto requieren bit-planes mayores.

RESERVANDO MEMORIA PARA LOS BITPLANES

Despues de seleccionar el número de bit-planes y especificar los modos de resolución, ya esta preparado para resevar memoria. Un bit-plane consiste en una secuencia de palabras en posiciones de memoria consecutivas. Cuando se esta trabajando con el sistema operativo, ha de usarse la función del sistema **AllocMem()** para quitar el bloque de memoria de la lista de memoria libre y hacerla disponible al programa. Si el sistema operativo ha sido desconectado, simplemente hay que reservar una zona de memoria para que únicamente sea usada por los bit-planes. Despues han de ponerse a punto los registros que apuntan a los bit-planes (BPLxPTH / BPLxPTL) para apuntar a la dirección de memoria donde comienza cada bit-plane (la esquina superior izquierda del mismo).

La Tabla 3-7 muestra la memoria que se necesita en los playfields basicos. Hay que hacer un balance entre los colores y la resolución que se necesitan con la memoria disponible que haya en el ordenador.

Tabla 3-7: Requerimientos de memoria por bit-plane.

Tamaño NTSC	Tamaño PAL	Modos de resolución	Bytes por bit-plane NTSC	Bytes por bit-plane PAL
320 x 200	320 x 256	Lo-RES no-LACE	8.000	8.192
320 x 400	320 x 512	Lo-RES LACE	16.000	16.384
640 x 200	640 x 256	Hi-RES no-LACE	16.000	16.384
640 x 400	640 x 512	Hi-RES LACE	32.000	32.768

Ejemplo de tamaño de un bit-plane en NTSC. Por ejemplo, usando una pantalla NTSC normal de baja resolución y modo no-entrelazado con 320 pixels de ancho y 200 lineas de alto, cada linea del bit-plane requiere 40 bytes (320 bits dividido entre 8 bits/byte = 40 bytes). Se multiplican las 200 lineas por los 40 bytes dando 8.000 bytes/bit-plane.

Un playfield de baja resolución y modo no-entrelazado con dos bitplanes requeriria 16.000 bytes de memoria. El area de memoria de cada bit-plane debe ser continua, es decir que se necesitaria tener dos bloques de 8.000 bytes disponibles. La Figura 3-5 muestra un bloque de memoria de 8.000 bytes organizado como 200 lineas de 40 bytes cada una, proporcionando 1 bit para pixel en la pantalla.

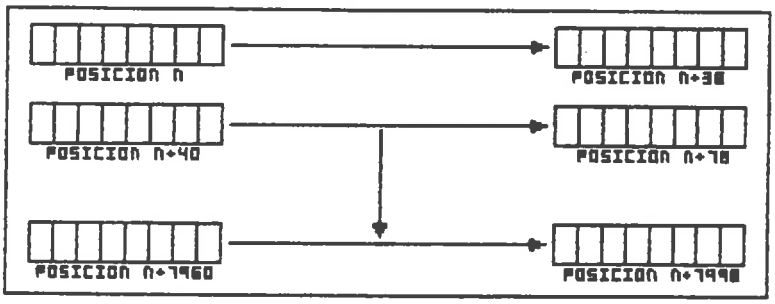


Figura 3-5: ORGANIZACION DE LA MEMORIA EN UN BIT-PLANE

El acceso a los bit-planes en memoria lo proporcionan dos registros de direcciones, BPLxPTH y BPLxPTL, por cada bit-plane (12 registros en total). La x del nombre equivale a cualquiera de los 6 bit-planes, por ejemplo BPL1PTH y BPL1PTL contendrian la dirección de comienzo del Bit-plane 1. Dichos pares de registros constan de 19 bits pero se puede direccionar al registro acabado en "PTH" con el 68000 como palabra larga (32 bits).

El siguiente ejemplo muestra como poner los punteros de bit-plane. Asumiendo dos bit-planes, uno en $21000 y el otro en $25000, el 68000 pondra en BPL1PT la dirección $21000 y en BPL2PT la dirección $25000, aunque esto normalmente es tarea del Copper.

```
LEA     CUSTOM,a0              ; Obtiene la base de los custom chips...
MOVE.L  $21000,BPL1PTH(a0)     ; Escribe la dirección del bit-plane 1
MOVE.L  $25000,BPL2PTH(a0)     ; Escribe la dirección del bit-plane 2
```

Tengase en cuenta que los requerimientos de memoria dados aqui son sólo para el playfield. Tambien se necesita reservar memoria para los sprites, el audio, la animación, y para los programas de aplicación. Para mas información sobre la reserva de memoria ver los capitulos de estos asuntos.

CODIFICANDO LOS BIT-PLANES PARA COLORIZACION CORRECTA

Despues de especificar el número de bit-planes y escribir los punteros a ellos, ya se pueden escribir los codigos de los registros de color en los bit-planes.

Un playfield de uno o dos colores. Para un playfield de 1 color, se necesita poner todo el bit-plane a "0" (borrarlo). El siguiente ejemplo llena un bit-plane de baja resolución con el color 0 (fondo). El bit-plane comienza en $21000 y es 8.000 bytes de largo.

```
        LEA     $21000,a0       ; dirección al bit-plane
        MOVE.W  #2000,d0        ; Escribir 2000 palabras largas= 8000 bytes
LOOP:   MOVE.L  #0,(a0)+        ; Escribe un cero
        DBRA    d0,LOOP         ; Decrementar contador y repetir hasta final
```

Para un playfield de dos colores, se define un bit-plane que este a "0" y se escribe un "1" donde se quiera que este el color del registro 1. El siguiente ejemplo es igual al anterior, execpto que el bit-plane se rellena con $FF00FF00 lo que da lugar a franjas de dos colores.

```
        LEA     $21000,a0       ; dirección al bit-plane
        MOVE.W  #2000,d0        ; Escribir 2000 palabras largas= 8000 bytes
LOOP:   MOVE.L  #$FF00FF00,(a0)+  ; Escribe un cero
        DBRA    d0,LOOP         ; Decrementar contador y repetir hasta final
```

Un Playfield de Tres o Mas colores. En estos casos se necesitan mas de un bit-plane, por tanto hay que definirlos de manera que cuando sus bits se combienen (siguiendo la prioridad de los bit-planes) cada pixel contenga el valor que apunte al registro de color adecuado. Esto es un poco mas complicado que con un solo bit-plane. La Figura 3-6 muestra un playfield de 4 colores, pero la idea basica es aplicable tambien para 5 bit-planes.

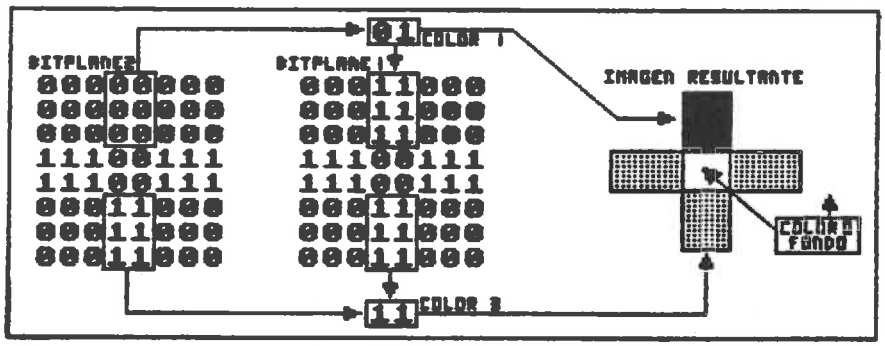


Figura 3-6: Combinacion de Bitplanes

Se han de colocar adecuadamente los bits de los bit-planes para que cada pixel de el color correcto. En un playfield único se pueden combinar hasta 5 bit-planes dando 32 colores posibles por cada pixel . Las tablas de selección de color se encuentran al final de este capitulo para las combinaciones de bits hechas para 4 o 5 bit-plane.

DEFINIENDO EL TAMAÑO DE LA VENTANA DE VISUALIZACION

Despues de que se haya definido completamente el playfield, se necesita definir el tamaño de la ventana de visualización, que es la zona de imagen que se ve en la pantalla. El ajuste de esta ventana afecta al borde, los sprites pero no al playfield. No se pueden visualizar objetos fuera de esta ventana.El tamaño del borde del playfield depende del tamaño de la ventana.

El playfield descrito en esta sección es del mismo tamaño que el area de visualización y tambien que la ventana. Esto no es siempre asi; a veces la ventana es menor que el "gran playfield" definido memoria (el raster). Este tipo de ventana permite visualizar solo una parte del playfield o moverlo por la ventana. Tambien se pueden definir ventanas mas grandes que los playfields, pero todo esto se explica en la sección "Bitplanes y Ventanas de visualización de todos los tamaños".

El tamaño de la Ventana se define escribiendo las posiciones horizontales y verticales donde comienza y acaba la ventana en los registros de la Ventana. La resolución de estos registros es la misma que el modo baja resolución tanto horizontal como vertical. Cada posición de la pantalla define la posición horizontal y vertical de un pixel , especificada por sus coordenadas (x,y). Aunque las coordenadas comienzan en (0,0) en la esquina superior izquierda de la imagen, normalmente se usan como coordenadas iniciales ($81,$2C) hex. o (129,44) dec. tanto en NTSC como en PAL.

El hardware permite especificar una posición de inicio anterior a ($81,$2C), pero parte de la imagen podria no ser visible. La diferencia entre la posición absoluta de inicio (0,0) y la normal ($81,$2C) es debida a la forma en que estan diseñados muchos monitores. Para superar la distorsión que se puede producir en los bordes extremos de la pantalla, el rayo rastrea un area mayor que la pantalla. La posición ($81,$2C) los playfields de tamaño normal, dejando un borde de 8 pixels de baja resolución alrededor del area de la Ventana. La Figura 3-7 muestra la relación entre la ventana de visualización normal, el area visible de la pantalla y el area en realidad cubierta por el rayo.

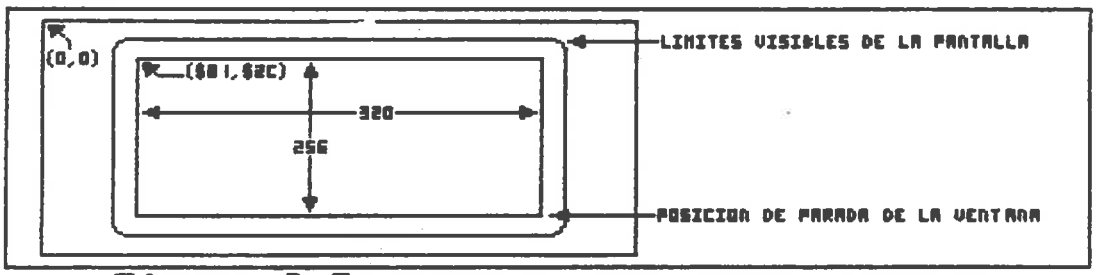


Figura 3-7: COLOCACION DE LA PANTALLA RESPECTO A LA IMAGEN REAL

Poniendo la Posición de comienzo de la ventana de visualización. La posición de inicio ($81,$2C) encentra la pantalla en la mayoria de las pantallas. Si se seleccionan los modos de alta resolución (640 pixels horiz.) o de entrelazado (400 lineas NTSC, 512 PAL) la posición de inicio no cambia. La posición de inicio siempre se interpreta en baja resolución y no-entrelazado. En otras palabras, se selecciona una posición de inicio que representa las coordenadas correctas en modo de baja resolución y no-entrelazado.

El registro DIWSTRT (Display Window Start) controla la posición de inicio de la ventana. Este registro contiene ambas coordenadas, HSTART y VSTART respectivamente. El siguiente ejemplo activa DIWSTRT con sus valores normales ($81,$2C):

```
LEA     CUSTOM,(a0)        ; Base de los custom chips...
MOVE.W  #$2C81,DIWSTRT     ; Dirección de incio de la ventana.
```

Poniendo la Posición de parada de la ventana de visualización. Tambien se necesita poner esta otra posición, que corresponde a la esquina inferior derecha de la ventana. Tambien se interpreta siempre en baja resolución.

El registro DIWSTOP (Display Window Stop) controla dicha posición. Tambien contiene ambas coordenadas y en el mismo orden HSTOP y VSTOP. Las siguientes instrucciones muestran como actuar sobre este registro asumiendo que la posición de inicio es ($81,$2C). Notese que el valor de HSTOP es el valor real-256 ($100) y por tanto esta restringido a la parte derecha de la pantalla. El valor normal para HSTOP es ($1C1) y se escribe como ($C1) tanto en NTSC como en PAL.

El valor de VSTOP esta restringido a la parte inferior de la pantalla. Esto se consigue en el hardware forzando el MSB (bit mas significativo) de VSTOP a ser el complemento del siguiente MSB, permitiendo de esta manera una posición mayor de 256 ($100) usando sólo 8 bits. El valor normal de VSTOP es ($F4) en NTSC y ($2C) en PAL.

El siguiente ejmplo pone $F4 en VSTOP y $C1 en HSTOP:

```
LEA       CUSTOM,(a0)          ; Base de los custom chips...
MOVE.W    #$F4C1,DIWSTRT       ; Dirección de parada de la ventana.
```

Tabla 3-9: Sumario de DIWSTRT y DIWSTOP

		-Valores Normales-		-Valores Posibles-	
		NTSC	PAL	MIN	MAX
DIWSTRT:	VSTRT	$2C	$2C	$00	$FF
	HSTRT	$81	$81	$00	$FF
DIWSTOP:	VSTOP	$F4	$2C($12C)	$80	$7F($17F)
	HSTOP	$C1	$C1	$00($100)	$FF($1FF)

INDICANDO AL SISTEMA COMO TRAER Y VISUALIZAR LOS DATOS

Despues de decidir el tamaño y la posición de la ventana de visualización, se necesita indicar al sistema la posición de la pantalla en la que han de colocarse los datos traidos de la memoria. Para hacer esto, se escriben las posiciones horizontales donde comienzan y acaban las lineas en los registros de "data-fetch". Estos registros tienen una resolución de 4 pixels (en lugar de 1 como los de la ventana). Cada posición es cuatro pixels mayor que la anterior: pos 0 -> pixel 0, pos 1 -> pixel 4, etc.

Las posiciones de inicio de ventana y de inicio de linea (data-fetch) estan en interacción y se recomienda usar una resolución de software de 16 pixels (8 ciclos de reloj en baja resolución y 4 en alta) ya que el hardware necesita un tiempo antes de que pueda comenzar a visualizar despues del primer "data-fetch". Como resultado hay una diferencia entre el valor de comienzo de ventana y el de comienzo de "data-fetch" de 4.5 ciclos de reloj (del reloj de color). El valor normal para DDFSTRT en baja resolución es $38 y en alta es $3C.

La resolución de hardware para el inicio de ventana es dos veces la del inicio de "data-fetch":

$$\frac{\$81}{2} - 8.5 = \$38 \qquad \frac{\$81}{2} - 4.5 = \$3C$$

La relación entre el inicio y la parada de data-fetch es:

```
DDFSTRT = DDFSTOP-(8*(cantidad de palabras-1))  ; Baja resolución.
DDFSTRT = DDFSTOP-(4*(cantidad de palabras-2))  ; Alta resolución.
```

El valor normal para DDFSTOP en baja resolución es $D0 y en alta es $D4.

El siguiente ejemplo pone DDFSTRT a $38 y DDFSTOP a $D0 para el playfield:

```
LEA     CUSTOM,a0               ; Base de los custom chips
MOVE.W  #$0038,DDFSTRT(a0)      ; Escribe a DDFSTRT
MOVE.W  #$00D0,DDFSTOP(a0)      ; Escribe a DDFSTOP
```

Tambien se necesita indicar al sistema exactamente que bytes de memoria pertenecen a cada linea horizontal. Para hacer esto, se especifica el valor del MODULO. El módulo indica los bytes de memoria que hay entre la última palabra de una linea y la primera palabra de la siguiente linea. Por tanto, el módulo permite que el sistema convierta los datos del bit-plane que estan en forma lineal en unaforma rectangular (cada linea secuencial debajo de la anterior). Para un playfield basico, donde el playfield de memoria es del mismo tamaño que la ventana de visualización, el módulo es cero porque el area de memoria contiene exactamente el mismo número de bytes que se quieren visualizar en la pantalla. Las Figuras 3-8 y 3-9 muestran el trazado de un bit-plane en memoria y cómo estar seguro de que se visualizan los datos correctos.

Los registros de dirección de bit-plane (BPLxPTH y BPLxPTL) son usados por el sistema para llevar los datos a la pantalla. Estos punteros son dinamicos, es decir que una vez que comienza el data-fetch (transmisión de los datos de la imagen), los punteros se incrementan continuamente para apuntar a la siguiente palabra para ser enviada (se envian 2 bytes a la vez). Cuando se cumple la condición de "fin de linea" (definida por el registro DDFSTOP) se añade el módulo a los punteros de los bit-planes, ajustando el puntero a la primera palabra a enviar a la siguiente linea.

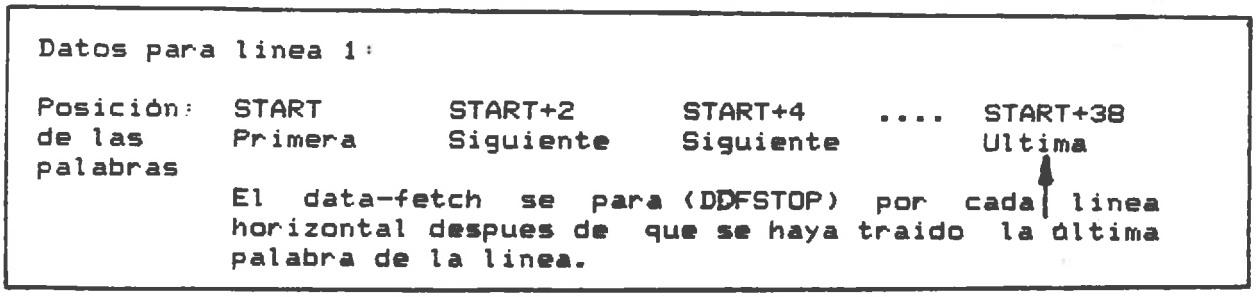


Figura 3-8: Data-fetch para la primera linea con MODULO=0

Despues de que haya sido visualizada la primera linea, los punteros de BPLxPTH y BPLxPTL contienen el valor START+40. El módulo (en este caso 0) se suma al valor actual de los registros para que cuando vuelva a comenzar la visualización de la siguiente linea, los registros apunten a la posición correcta. Los datos de la siguiente linea comienzan en START+40

Figura 3-9: Data-fetch para la segunda linea con MODULO=0

Nótese que los punteros siempre contienen un número par porque los datos se envian a la pantalla como palabras.

Hay dos registros del módulo -BPL1MOD para los bit-planes impares y BPL2MOD para los pares. Esto permite diferenciar los módulos para cada playfield en el modo de doble playfield. Normalmente BP1MOD y BPL2MOD son iguales.

El siguiente ejemplo pone el módulo a 0 para un playfield de un solo bit-plane y baja resolución (bit-plane 1 = bit-plane impar).

```
        MOVE.W    #0,BPL1MOD+CUSTOM    ; Pone el módulo a 0
```

El Data-fetch en modo de alta resolución necesita visualizar 80 bytes por linea en lugar de 40.

El Módulo en modo Entrelazado se necesita redefinir porque este modo usa dos diferentes campos de visualización en un único playfield. Durante la primera visualización, se envian a la pantalla las lineas impares; y durante la segunda visualización, se envian las lineas pares.

Los bit-planes para una pantalla normal en modo entrelazado son de 400 lineas NTSC (512 PAL), en lugar de 200 NTSC o 512 PAL. Suponiendo que el playfield en memoria fuera de 320 pixels de ancho, los datos para la imagen entrelazada comenzarian en las siguientes posiciones (en bytes):

Linea 1	START
Linea 2	START+40
Linea 3	START+80
Linea 4	START+120

Y asi las demas lineas. Por tanto se usa un módulo de 40 para saltarse las lineas correspondientes al otro campo. Para los campos impares, los registros comenzarian con START y para los campos pares con START+40. Se suele usar el Copper para manejar los registros en modo entrelazado.

VISUALIZANDO Y REVISUALIZANDO EL PLAYFIELD

La visualización del playfield comienza cuando se activan los punteros de bit-plane y se conecta el acceso DMA de los bit-planes escribiendo un uno en el bit BPLEN del registro DMACON (DMA CONtrol). Ver capitulo 7, "El Hardware de Control del Sistema" para las instrucciones de este registro.

Cada vez que se revisualiza el playfield, se han de volver a poner los registros de los bit-planes a su valor inicial porque estos registros van incrementando su valor a medida que visualizan las palabras del bit-plane y debe ser redireccionado a la primera palabra para la siguente visualización. Se suelen escribir instrucciones al Copper para manejar la revisualización como parte de las tareas en el vertical blank.

CONECTANDO LA VISUALIZACION EN COLOR

El A1000 tiene una salida de video compuesto en color que requiere que se active el bit 9 de BPLCON0 para crear la señal de color. En el A500 y A2000 esta señal de color no se puede generar (si no es con hardware adicional).

Nota: Este bit de color burst no afecta a la señal RGB que es generada independientemente de la salida de la señal de video compuesto.

SUMARIO DE PLAYFIELD BASICO

1. Definir las caracteristicas del playfield.

a) Especificar la altura en lineas:

- 200 en NTSC, 256 en PAL en modo no entrelazado
- 400 en NTSC, 512 en PAL en modo entrelazado

b) Especificar el ancho en pixels:

- 320 en modo de baja resolución
- 640 en modo de alta resolución

c) Especificar el color para cada pixel:

- Cargar los colores deseados en los registros de color.
- Definir el color de cada pixel en terminos del valor en binario que apunta al registro de color deseado.
- Construir los bit-planes.
- Poner los registros de los bit-planes:
  * bits 12-14 de BPLCON0 -número de bit-planes
  * BPLxPTH -puntero al comienzo del bit-plane en memoria

d) Especificar la resolución:

- Baja resolución (320 pixels) el bit 15 de BPLCON0 a 0.
- Alta resolución (640 pixels) el bit 15 de BPLCON0 a 1.

e) Especificar modo entrelazado:

- No-entrelz. (200 NTSC, 256 PAL) el bit 2 de BPLCON0 a 0.
- Entrelazado (400 NTSC, 512 PAL) el bit 2 de BPLCON0 a 1.

2. Reservar memoria. Para calcular los bytes necesarios para todos los bit-planes, se usa la siguiente formula:

memoria = bytes por linea * lineas de la imagen * número de bit-planes

3. Definir el tamaño de la ventana de visualización.

- Escribir la posición de inicio de la ventana en DIWSTRT:
  HSTRT en bits 0-7, VSTRT en bits 8-15

- Escribir la posición de parada de la ventana en DIWSTOP:
  HSTOP en bits 0-7, VSTOP en bits 8-15

4. Definir el Data-Fetch. Poner los registros DDFSTRT y DDFSTOP.

- Para DDFSTRT, poner la posición horizontal como se indica en "Poniendo la posición de inicio de la ventana"

- Para DDFSTOP, poner la posición horizontal como se indica en "Poniendo la posición de parada de la ventana"

5. Definir el Módulo. 0-> no-entrel, 40-> entrelazado en BPL1MOD y BPL2MOD.

6. **Escribir la CopperList para manejar la Revisualización.**

7. **Conectar la salida de color.** Para el A1000: poner a 1 el bit 9 de BPLCON0 para la imagen en color en un monitor de video compuesto. Esto no afecta a la salida de RGB. Sólo el A1000 tiene salida de video compuesto en color, los demas modelos lo tienen en blanco y negro.

EJEMPLOS PARA FORMAR UN PLAYFIELD BASICO

Los siguientes ejemplos muestran como poner los registros y escribir las CopperLists para dos playfields diferentes.

El primer ejemplo crea un playfield de 320 x 200 con un bit-plane, que esta colocado en $21000, La CopperList se coloca en $20000. Este ejemplo necesita el fichero include "hw_examples.i", en el Apendice J.

```
        LEA     CUSTOM,a0               ; Base de los custom chips
        MOVE.W  #$1200,BPLCON0(a0)      ; Un bit-plane, conecta color
        MOVE.W  #0,BPLCON1(a0)          ; Valor de scroll horizontal a 0
        MOVE.W  #0,BPL1MOD(a0)          ; Modulo a 0  para bit-planes impares
        MOVE.W  #$0038,DDFSTRT(a0)      ; Inicio de data fetch a $38
        MOVE.W  #$00D0,DDFSTOP(a0)      ; Parada de data fetch a $D0
        MOVE.W  #$2C81,DIWSTRT(a0)      ; Inicio de ventana en ($81,$2C)
        MOVE.W  #$F4C1,DIWSTOP(a0)      ; Parada de ventana en ($C1,$F4)
        MOVE.W  #$0F00,COLOR00(a0)      ; Color de fondo en rojo
        MOVE.W  #$0FF0,COLOR01(a0)      ; Color de "tinta" en amarillo
;
; Se llena el bit-plane con $FF00FF00 para producir rayas
;
        MOVE.L  #$21000,a1              ; Comienzo del bit-plane
        MOVE.L  #$FF00FF00,d0           ; Valor para rellenar el bit-plane
        MOVE.W  #2000,d1                ; 2000 palabras largas = 8000 bytes

LOOP:   MOVE.L  d0,(a1)+                ; Escribe la palabra larga
        DBRA    d1,LOOP                 ; Decrementa el contador y repite
;
; Prepara la CopperList en $20000
;
        MOVE.L  #$20000,a1              ; Comienzo de la CopperList
        LEA     COPPERL(pc),a2          ; Apunta a los datos de la CopperList
CLOOP:  MOVE.L  (a2)+,(a1)+             ; Mueve una palabra larga
        CMPI.L  #$FFFFFFFE,(a2)         ; Mira si es el final de la CopperList
        BNE     CLOOP                   ; Repite hasta que este toda la lista
;
; Direcciona al Copper hacia la CopperList
;
        MOVE.L  #$20000,COP1LCH(a0) ; Escribe al registro del Copper
        MOVE.W  COPJMP1(a0),d0      ; Fuerza al Copper a $20000
;
  Comienza el DMA

        MOVE.W #(DMAF_SETCLR!DMAF_COPPER!DMAF_RASTER!DMAF_MASTER),DMACON(a0)
                ; Conecta el DMA del Copper y de los bit-planes
        BRA     ....                    ; Salta a la siguiente tarea

; Esta es la CopperList
;
        DC.W    BPL1PTH,$0002           ; Mueve $0002 a la dirección $0E0
        DC.W    BPL1PTL,$1000           ; Mueve $1000 a la dirección $0E2
        DC.W    $FFFF,$FFFE             ; Final de Copper List
```

El segundo ejemplo crea un playfield de un bit-plane en alta resolución y entrelazado. Tambien necesita el fichero include "hw_examples.i"

```
        LEA     CUSTOM,a0               ; Base de los custom chips
        MOVE.W  #$9204,BPLCON0(a0)      ; Un bit-plane, Hi-res, lace, color
        MOVE.W  #0,BPLCON1(a0)          ; Valor de scroll horizontal a 0
        MOVE.W  #80,BPL1MOD(a0)         ; Modulo a 80  para bit-planes impares
        MOVE.W  #$003C,DDFSTRT(a0)      ; Inicio de data fetch a $3C
        MOVE.W  #$00D4,DDFSTOP(a0)      ; Parada de data fetch a $D4
        MOVE.W  #$2C81,DIWSTRT(a0)      ; Inicio de ventana en ($81,$2C)
        MOVE.W  #$F4C1,DIWSTOP(a0)      ; Parada de ventana en ($C1,$F4)
;
; Pone los registros de color
;
        MOVE.W  #$000F,COLOR00(a0)      ; Color de fondo en azul
        MOVE.W  #$0FFF,COLOR01(a0)      ; Color de "tinta" en blanco
;
; Prepara un bit-plane en $20000
;
        LEA     #$20000,a1              ; Apunta al bit-plane
        LEA     CHARLIST(pc),a2         ; a2 apunta a la lista de caracteres
        MOVE.W  #400,d1                 ; Escribe 400 lineas de datos
        MOVE.W  #20,d0                  ; Escribe 20 palabras largas por linea

L1:     MOVE.L  (a2)+,(a1)+             ; Mueve una palabra larga
        DBRA    d0,L1                   ; Decrementa y repite hasta el final

        MOVE.W  #20,d0                  ; Repone el valor del contador
        ADDQ.L  #4,a2                   ; Apunta a siguiente palabra en lista
        CMPI.L  #$FFFFFFFF,(a2)         ; Final de la lista?
        BNE     L2
        LEA     CHARLIST(pc),a2         ; Si, pone a2 al inicio de la lista
L2:     DBRA    d1,L1                   ; Decrementa y repite hasta el final
;
; Comienzo de DMA
;
        MOVE.W  #(DMAF_SETCLR!DMAF_RASTER!DMAF_MASTER),DMACON(a0) ;  Conecta
                el DMA de los bit-planes, del Copper no.

; Debido a que este ejemplo no tiene CopperList, se basa en un
; bucle de espera al intervalo del vertical blank. Cuando llega
; se comprueba el bit LOF de VPOSR (imagen larga). Si LOF=0,
; entonces es una imagen corta y los punteros de bit-plane se
; colocan apuntando a $20050. Si LOF=1, entonces es una imagen
; larga y los punteros de bit-plane se colocan a $20000.
; Esto mantiene las imagenes largas y cortas en la relación correcta

VLOOP:  MOVE.W  INTREQR(a0),d0          ; Lee interrupciones requeridas
        AND.W   #$0020,d0               ; Mascara a todas menos vertical blank
        BEQ     VLOOP                   ; Espera al vertical blank
        MOVE.W  #$0020,INTREQ(a0)       ; Borra la interrupción
        MOVE.W  VPOSR(a0),d0            ; Lee bit LOF en el bit 15 de D0
        BPL     VL1                     ; Si LOF=0, salta
        MOVE.L  #$20000,BPL1PTH(a0)     ; LOF=1, apunta a $20000
        BRA     VLOOP                   ; Vuelve al bucle

VL1:    MOVE.L  #$20050,BPL1PTH         ; LOF=0, apunta a $20050
        BRA     VLOOP                   ; Vuelve al bucle

; Lista de caracteres
;
CHARLIST:   DC.L    $18FC3DF0,$3C6666D8,$3C66C0CC,$667CC0CC
            DC.L    $7E66C0CC,$C36666D8,$C3FC3DF0,$00000000
            DC.L    $FFFFFFFF
```

FORMANDO UNA PANTALLA DE DOBLE PLAYFIELD

Para mas flexibilidad en el diseño del fondo de la imagen, se pueden usar dos playfields en lugar de uno solo. En este modo, un playfield se visualiza encima del otro. Por ejemplo, en un juego (Figura 3-10), la acción se desarrolla en el playfield de detras, mientras que los indicadores van en el de delante. De esta manera se puede modificar el contenido de uno de los dos playfields sin tener que afectar necesariamente al otro. Ademas se pueden mover independientemente.

El doble playfield es similar al de uno solo, excepto en estos aspectos:

- Cada playfield en este modo solo puede tener hasta 3 bit-planes.
- Los colores de cada playfield (hasta 7 mas transparente) se toman de registros de color diferentes.
- Se debe poner a 1 un bit para indicar el modo de doble playfield.

En la Figura 3-10, uno de los colores de cada playfield es "transparente" (el color 0 en el playfield 1, y el color 8 en el playfield 2). El color transparente se usa para visualizar el contenido del otro playfield en las partes en que dicho color se utiliza.

En el modo de doble playfield, cada uno esta formado de hasta 3 bit-planes. Los registros del color 0 a 7 son del playfield 1, dependiendo de cuantos bit-planes se usen y los registros del color 8-15 son del playfield 2.

Figura 3-10: Imagen de doble Playfield

ASIGNACION DE BIT-PLANES EN MODO DE DOBLE PLAYFIELD

El hardware agrupa a los bit-planes imapares (1,3,5) para ser usados en el playfield 1. De la misma manera, se agrupan los bit-planes pares (2,4,6) para usarse en el playfield 2. Los bit-planes se asignan alternativamente como muestra la Figura 3-11.

Nota: En el modo de alta resolución, solo se pueden tener hasta 2 bit-planes por playfield (1 y 3 para el primero y 2 y 4 para el segundo)

LOS REGISTROS DE COLOR EN MODO DE DOBLE PLAYFIELD

Cuando se usa este modo, el hardware interpreta las combinaciones de los bit-planes 1, 3 y 5 como los números de color del playfield 1. Los bits mas significativos son los del bit-plane 5 y los menos significativos son los del bit-plane 1. Las combinaciones se muestran en la Tabla 3-10.

El hardware interpreta las combinaciones de los bit-planes 2, 4 y 6 como los números de color del playfield 2. Los bits mas significativos son los del bit-plane 6 y los menos significativos son los del bit-plane 2. Las combinaciones se muestran en la Tabla 3-11.

La combinación 000 selecciona el modo transparente, para mostrar el color de otro objeto (el otro playfield, un sprite o el color de fondo) que esta detras del playfield.

Tabla 3-10,11y12: Combinaciones de bit-planes.

Comb. bits	Color Playfield 1	Color Playfield 2	Modo de Resolución
000	Transp.	Transp.	
001	COLOR 1	COLOR 9	BAJA
010	COLOR 2	COLOR10	y
011	COLOR 3	COLOR11	ALTA
100	COLOR 4	COLOR12	
101	COLOR 5	COLOR13	Sólo
110	COLOR 6	COLOR14	BAJA
111	COLOR 7	COLOR15	

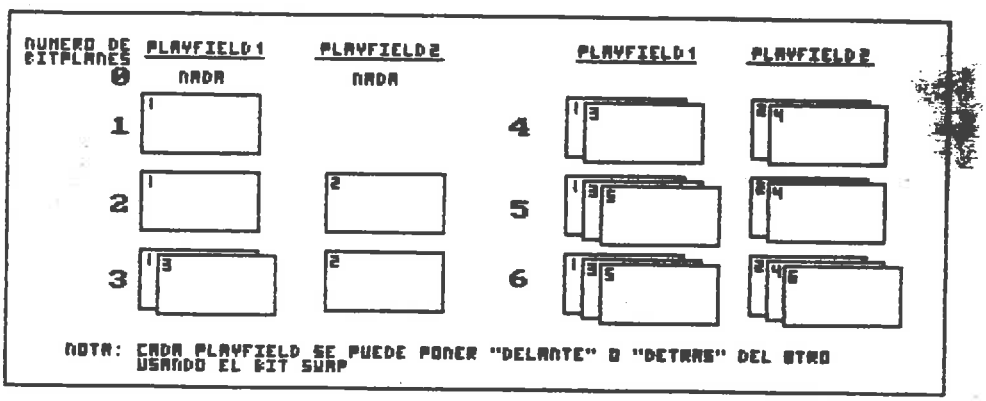


Figura 3-11: DISTRIBUCION DE BIT-PLANES EN MODO DE DOBLE PLAYFIELD

PRIORIDAD Y CONTROL DEL DOBLE PLAYFIELD

Uno de los playfields tiene prioridad para visualizarse sobre el otro (normalmente el primero) y se selecciona mediante el bit llamado PF2PRI (bit 6 del registro BPLCON2). Cuando PF2PR=1, el playfield 2 tiene prioridad sobre el playfield 1 y cuando PF2PR=0, la tiene el playfield 1.

Tambien se puede controlar la prioridad relativa de los playfields y los sprites. Ver el Capitulo 7, "El Hardware de Control del Sistema"

Se pueden controlar los dos playfields separadamente como sigue:

- Pueden tener diferente tamaño en memoria, y visulizarse diferentes porciones de los mismos.

- Pueden hacer scroll independientemente.

Nota: Se debe tener especial cuidado cuando se hace scroll en un playfield y se mantiene quieto al otro. Cuando se hace scroll en playfields de baja resolución se debe traer un palabra mas que el ancho del playfield que se intenta mover (dos palabras en alta resolución) para proporcionar nuevos datos a la imagen cuando tiene lugar el scroll. Sólo hay un registro DDFSTRT y otro DDFSTOP que estan compartidos por los dos playfields. Si se quiere hacer scroll en un playfield y mantener quieto al otro, los registros DDFSTRT y DDFSTOP se deberan ajustar para el playfield que hace el scroll y tambien los punteros de los bit-planes y el módulo del playfield que permanecera quieto, restando a los punteros -2 (-4 para alta resolución) y poniendo el módulo a -2 (-4 para alta resolución).

ACTIVACION DEL MODO DE DOBLE PLAYFIELD

Escribiendo un 1 a DBLPF (bit 10 del registro BPLCON0) se selecciona el modo de doble play-field. Seleccionando este modo, el hardware cambia la forma de interpretar los bit-planes agrupando a los impares en un grupo y a los pares en el otro.

SUMARIO DE DOBLE PLAYFIELD

Los pasos para definir dobles playfields son los mismos que para playfields únicos. Solamente los siguientes pasos difieren en los dobles playfields:

- **Cargar los colores en los registros.** Hay que tener en cuenta que los colores 0-7 son del playfield 1 y los colores 8-15 son del playfield 2 (si hay tres bit-planes por playfield)
- **Construir bit-planes.** Recordar que el playfield 1 se compone de los bit-planes 1, 3 y 5 y el playfield 2 de los bit-planes 2, 4 y 6.
- **Poner los registros del modulo.** Escribir BPL1MOD y BPL2MOD segun se vayan a usar los bit-planes impares y los pares.

Se añaden los siguientes pasos:

- **Definir la prioridad.** Escribir un 1 en PF2PRI (bit 6 de BPLCON2) para que el playfield 2 tenga prioridad sobre el playfield 1 y un 0 para lo contrario.
- **Activar el doble playfield.** Poner DBLPF a 1 (bit 10 de BPLCON0)

BIT-PLANES Y VENTANAS DE VISUALIZACION DE TODOS LOS TAMAÑOS

Ya se ha visto como crear playfields únicos y dobles en los cuales el playfield en memoria era del mismo tamaño que la ventana de visualización. Esta sección muestra como definir y usar un playfields mas grandes que la ventana, como definir la ventana para que sea mayor o menor que los playfields normales, y como mover la ventana en un playfield grande.

CUANDO EL PLAYFIELD ES MAYOR QUE LA VENTANA

Si se diseña un playfield mayor que la ventana, se debe elegir que parte del mismo sera visualizada. La forma de visualizar una porción de un gran playfield es distinta que en los playfields basicos en lo siguiente:

- Si el playfield es mas ancho que la ventana, se deberan calcular los nuevos módulos (nunca sera 0).
- Se necesitara reservar mas memoria para un playfield mayor.

Especificando el módulo. Para un playfield mas ancho que la ventana, se debera calcular un módulo para que se lleven las palabras correctas a cada linea de la imagen. Como ejemplo, el ancho standard de la ventana es 320 (40 bytes por linea). El playfield en memoria es dos veces este valor (80 bytes de ancho). Y se quiere visualizar la mitad izquierda del playfield. La Figura 3-12 muestra la relación del playfield y la zona a visualizar.

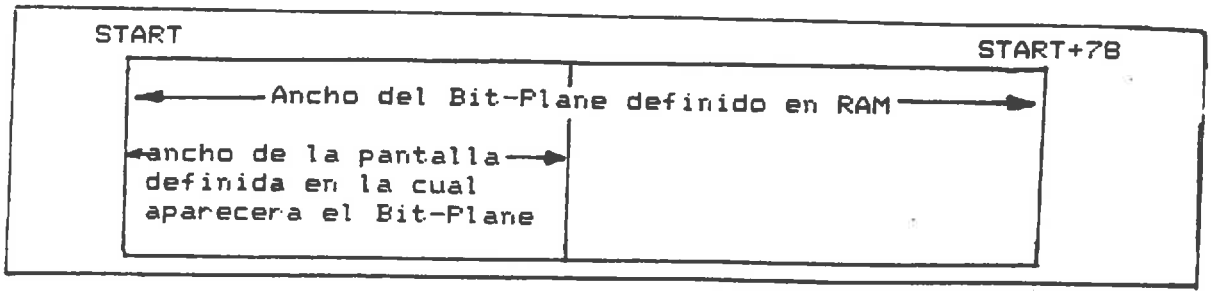


Figura 3-12: Playfield mayor que la Imagen.

Datos para linea 1:

Posición de las palabras	START Primera	START+2 Siguiente	START+4 Siguiente	....	START+38 Ultima

El data-fetch se para (DDFSTOP) por cada [illegible] horizontal despues de que se haya traido la [illegible]ima palabra de la linea.

Figura 3-13: Data-fetch para la primera linea con MODULO=40

En este punto, BPLxPTH y BPLxPTL contienen el valor START+40. El módulo que es 40, se suma al valor actual de los registros para que cuando vuelva a comenzar la visualización de la siguiente linea, los registros apunten a los datos que corresponden a esa linea, como se muestra en la Figura 3-14.

Datos para linea 2:

Posición de las palabras	START+80 Primera	START+82 Siguiente	START+84 Siguiente	....	START+118 Ultima

Figura 3-14: Data-fetch para la segunda linea con MODULO=40

Para visualizar la parte derecha del playfield, se puede hacer una rutina o una CopperList que durante el vertical blank ponga los registros del bit-plane a START+40 en lugar de START, dejando el módulo igual, 40. Esto se muestra en las Figuras 3-15 y 3-16.

Datos para linea 1:

Posición de las palabras	START+40 Primera	START+42 Siguiente	START+44 Siguiente	....	START+78 Ultima

Figura 3-15: Data-fetch para la primera linea, mitad derecha.

Ahora, BPLxPTH y BPLxPTL contienen el valor START+80. El módulo (40) se suma al valor de los registros que apunten a la dirección correcta.

Datos para linea 2:				
Posición de las palabras:	START+120 Primera	START+122 Siguiente	START+124 .... Siguiente	START+158 Ultima

Figura 3-16: Data-fetch para la segunda, mitad derecha.

Recuerdese que en el modo de alta resolución, se necesitan el doble de bytes que en baja resolución (80 bytes de ancho en lugar de 40).

Especificando el Data-Fetch. Los registros de data-fetch indican las posiciones de comienzo y final donde han de llevarse los datos en cada linea horizontal. Esto se hace de la misma manera que en la sección llamada "Formando un Playfield Basico"

Reserva de memoria. Para imagenes grandes, se necesitan bloques grandes de memoria. La formula de siempre:

memoria = bytes por linea * lineas de la imagen * número de bit-planes

Por tanto el ejemplo de esta sección necesitaria 32.000 bytes (80*200*2). Aunque esto sólo incluye el playfield: el sonido, los sprites, la animación o los programas tambien necesitan memoria aparte.

Poniendo la posición de comienzo de la ventana. La posición de comienzo de la ventana son las coordenadas vertical y horizontal de su esquina superior izquierda. Estas coordenadas (VSTART y HSTART) se mantienen en el registro DIWSTRT. Los ocho bits asignados a HSTART permiten 256 posiciones. Por tanto se puede dejar la zona de comienzo en cualquier posición de este rango.

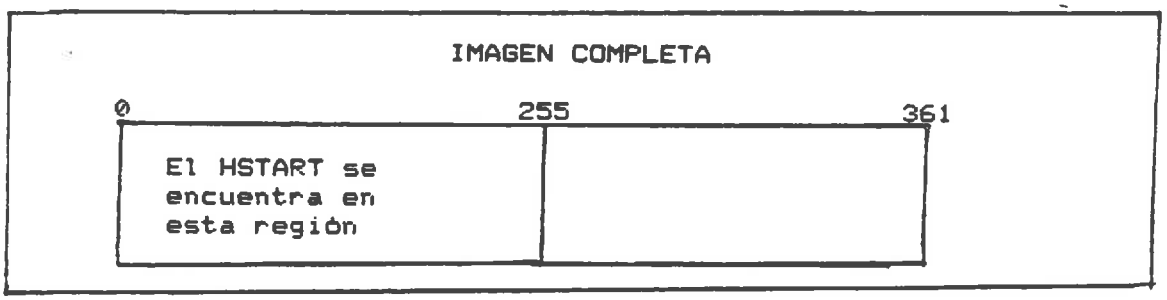


Figura 3-17: Posición horizontal de comienzo de Ventana.

Los ocho bits de VSTART permiten tambien 256 posiciones:

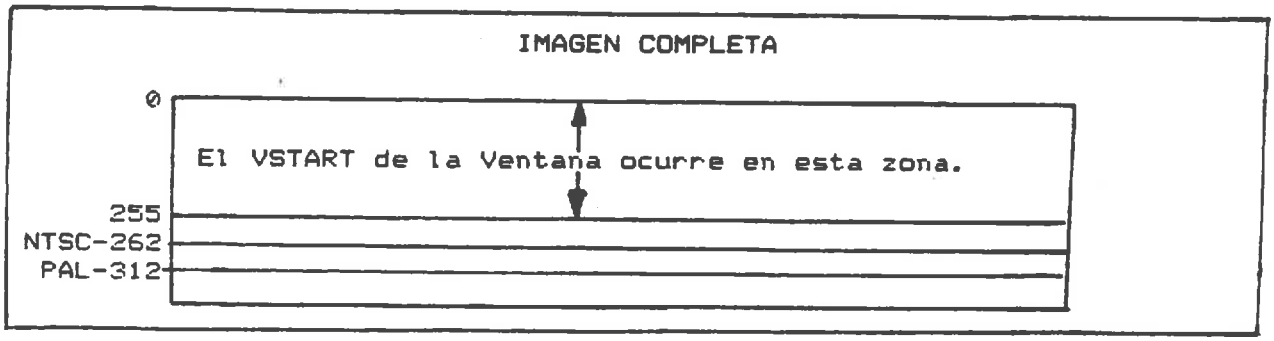


Figura 3-18: Posición vertical de comienzo de Ventana.

Recuerdese que los valores para la posición de comienzo corresponden al modo de baja resolución y no-entrelazado. En el modo entrelazado la ventana de visualización debe tener un número de lineas pares de altura para que ambos campos tengan el mismo número de lineas.

Para escribir el registro DIWSTRT, el valor de VSTART va en los bits 8-15 y el de VSTOP en los bits 0-7.

Poniendo la posición de parada de la ventana. La posición de parada de la ventana son las coordenadas vertical y horizontal de su esquina inferior derecha. Estas coordenadas (VSTOP y HSTOP) estan en el registro DIWSTOP. Ver las notas de la sección "Formando un Playfield Basico" sobre este registro.

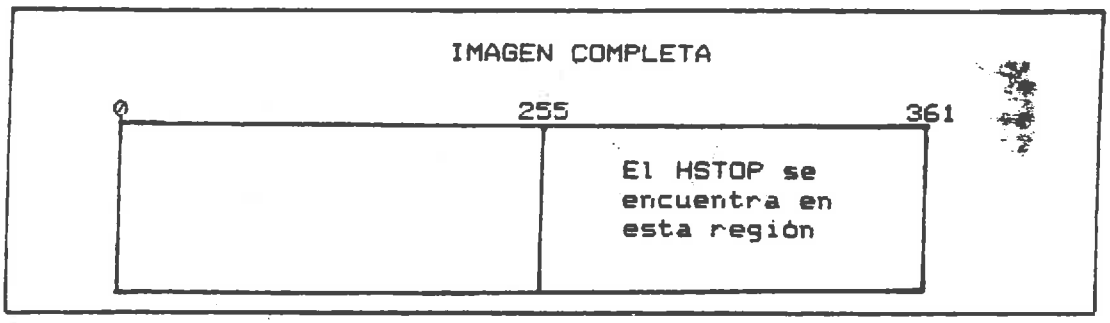


Figura 3-19: Posición horizontal de parada de Ventana.

El valor debe representar la posición en baja resolución y no-entrelazado.

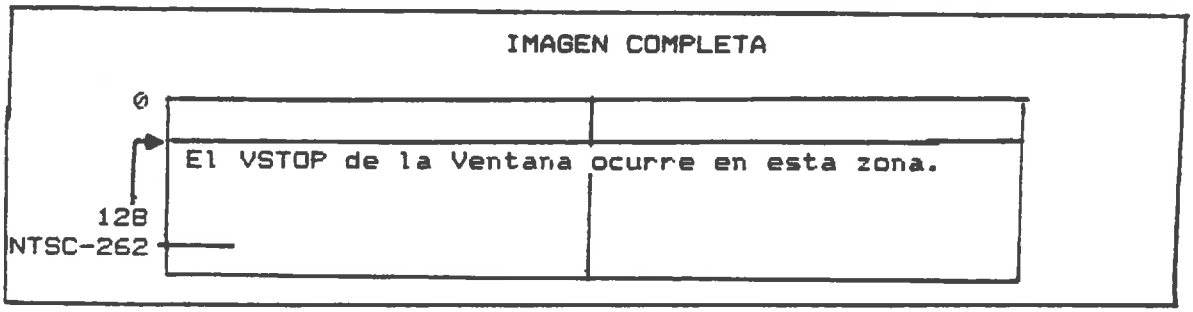


Figura 3-20: Posición vertical de parada de Ventana.

Para poner la posición vertical de parada de la ventana, escribir HSTOP en los bits 0-7 y VSTOP en los bits 8-15 del registro DIWSTOP.

TAMAÑO MAXIMO DE LA VENTANA DE VISUALIZACION

El tamaño maximo de la ventana de un playfield esta determinado por el maximo número de lineas y el maximo número de columnas. Verticalmente, las restricciones son simples. Ningún dato puede visualizarse en el area del vertical blank. La siguiente tabla muestra el area vertical disponible:

Tabla 3-13: Maxima area vertical disponible

Vertical Blank	NTSC	PAL
Comienzo	0	0
Fin	$15	$1D(29)

	NTSC	Entrelazado	PAL	Entrelazado
Lineas disponibles	241	483 (525-21*2)	283	567 (625-29*2)

Horizontalmente, la situación es similar. Estrictamente hablando, el hardware pone un limite derecho (DDFSTOP) de $D8 y un limite izquierdo (DDFSTRT) de $18. Esto da un maximo de 25 palabras visualizadas en baja resolución. En alta resolución el maximo es 49 palabras, debido a que el limite derecho continua a $D8 y sólo se visualiza una palabra en este limite. Pero en la realidad el "horizontal blanking" limita la imagen a 368 pixels de baja resolución (23 palabras). Estos números son iguales para NTSC y PAL. Ademas, si se usa DDFSTRT menor a $38 se desconectaran algunos sprites.

Tabla 3-14: Ancho maximo de la imagen

	Lo-Res	Hi-Res
DDFSTRT (standard)	$38	$3C
DDFSTOP (standard)	$D0	$D4
DDFSTRT (limite)	$18	$18
DDFSTOP (limite)	$D8	$D8
Max palabras	25	49
Max pixels	368	736

MOVIENDO LOS PLAYFIELDS (SCROLL)

Si se quiere que un playfield de fondo se mueva, se puede diseñar un playfield mayor que la ventana y moverlo. Si se usa doble playfield, se podran mover independientemente.

En el scroll vertical, el playfield se movera suavemente por la pantalla. Todo lo que se necesita hacer es incrementear o decrementear progresivamente la dirección inicial de los punteros de bit-planes con el tamaño de una linea horizontal del playfield. Esto tiene el efecto de visualizar en cada FRAME una parte un poco superior o inferior del playfield.

En el scroll horizontal, el playfield se mueve de un lado al otro. Este scroll se consigue de manera distinta al anterior -se necesita visualizar una palabra mas por cada linea y retardar la visualización de estos datos.

Para estos tipos de scroll, el manejo de los punteros y los registros de data-fetch es recomendable hacerlo con el Copper durante el vertical blank.

SCROLL VERTICAL

Se puede mover un playfield hacia arriba y abajo en la ventana. Cada vez que se visualiza el playfield, los punteros de los bit-planes comienzan en un posición un poco mayor o menor del gran playfield. Cuando el valor de los punteros aumenta, se ve mas de la parte inferior del playfield y parece haber ocurrido un scroll hacia arriba. Cuando el valor de los punteros disminuye, se ve mas de la parte superior del playfield y parece haber ocurrido un scroll hacia abajo. En un sistema NTSC, con una ventana de 200 lineas, cada paso puede ser como minimo de 1/200 de la pantalla (1/256 PAL). En entrelazado el minimo es 1/400 (1/512 PAL) si se hace una manipulación inteligente de los punteros, pero se recomienda que el scroll se haga de dos lineas para mantener la relación entre los campos pares e impares.

Un playfield necesita para el scroll vertical, bit-planes tan altos como se quiera hacer el scroll, software que calcule los punteros y una CopperList que maneje los resultados. Si por ejemplo se quisiera mover un playfield una linea hacia arriba, antes de que comenzara a visualizarse, habria que incrementar los punteros para que aparecieran en la linea inferior cada vez. Para un playfield normal de baja resolución con módulo=0, los punteros se incrementarian 40 cada vez.

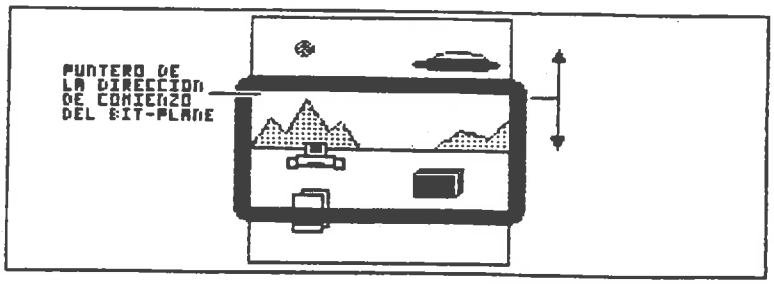


Figura 3-21: Scroll Vertical

SCROLL HORIZONTAL

Tambien se pueden mover los playfields de derecha a izquierda y viceversa. Se puede controlar la velocidad del scroll especificando la cantidad de pixels de retardo. El retardo significa que se trae una palabra extra que no se visualiza inmediatamente. La palabra extra se coloca justo a la izquierda del borde de la ventana y antes del data-fetch normal. Como la imagen se desplaza hacia la derecha, los bits de la palabra extra aparecen en la pantalla a la izquierda de la ventana a la vez que los bits de la parte derecha desaparecen. Por cada pixel de retardo, los datos de la imagen se desplazan un pixel a la derecha. A mayor retardo, mayor velocidad de scroll. Se puede tener hasta 15 pixells de retardo. En alta resolución el scroll se produce a 2 pixels . La Figura 3-22 muestra como se combinan el retardo y la palabra extra para producir el scroll.

Para preparar los bit-planes para un scroll horizontal se necesita:

- Definir los bit-planes lo suficientemente anchos para el scroll.
- Poner los registros de data-fetch correctamente para las lineas horizontales, incluyendo la palabra extra.
- Poner los bits de retardo.
- Poner el módulo para que los punteros de los bit-planes comiencen en la posición adecuada.
- Escribir la CopperList para manejar los cambios en el vertical blank.

Especificando el data-fetch en el scroll horizontal. El data-fetch normal comienza en $38. Si se ha de hacer scroll horizontal, habra de comenzar una palabra antes (DDFSTRT=$30), aunque esto desconectara al sprite 7. Recuerdese que estos cambios afectan a ambos playfields.

Especificando el módulo en el scroll horizontal. Como siempre, el módulo es dos veces menos que la diferencia entre la dirección de la siguiente palabra y la dirección de la que ya se ha visualizado. Como ejemplo de scroll horizontal, se asume que el ancho de la ventana es 40 bytes y el del playfield es 80. Debido a que el scroll necesita 2 bytes extra, la longitud de cada linea sera de 42 bytes.

Nota: Llevar una palabra extra para el scroll desconecta algunos sprites.

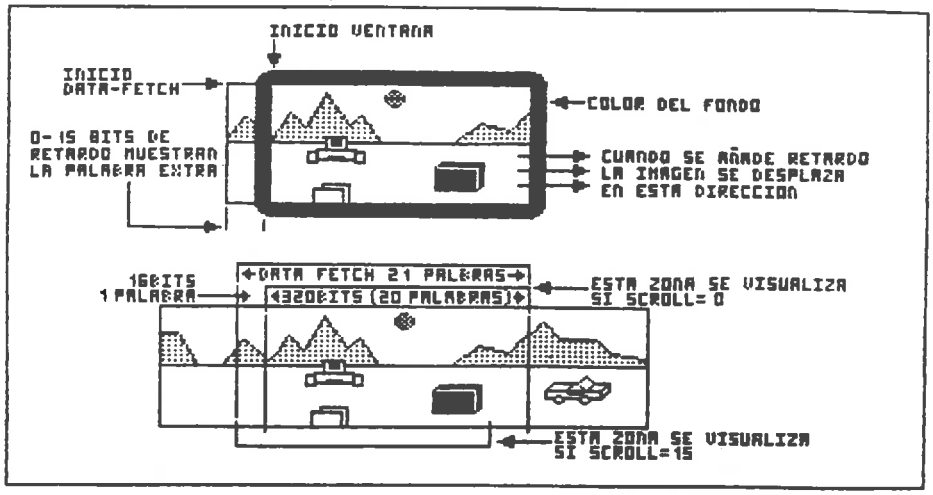


Figura 3-22: Scroll Horizontal

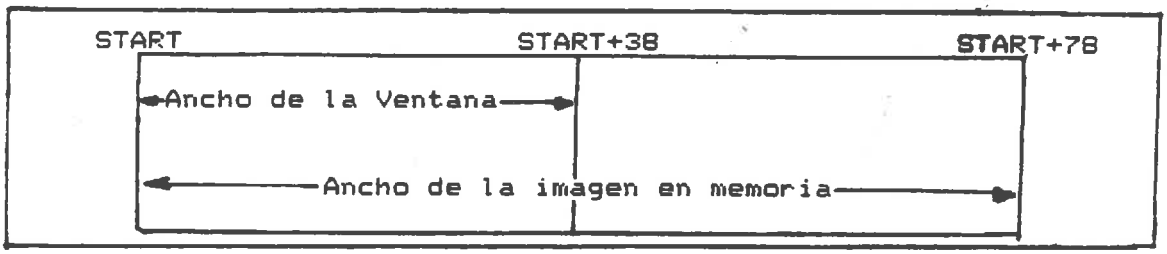


Figura 3-23: Playfield mayor que la Ventana.

Datos para linea 1:					
Posición de las palabras:	START Primera	START+2 Siguiente	START+4 Siguiente	....	START+40 Ultima

Figura 3-24: Datos para linea 1 -Scroll Horizontal

En este punto, los registros contienen el valor START+42, añadiendo el módulo (38) da la dirección correcta a la siguiente linea.

Datos para linea 2:					
Posición de las palabras:	START+80 Primera	START+82 Siguiente	START+84 Siguiente	....	START+120 Ultima

Figura 3-25: Datos para linea 2 -Scroll Horizontal

En los BPLxMOD se pone un módulo según el tamaño de los bit-planes.

Especificando la cantidad de retardo. Este valor lo controlan los bits 7-0 del registro BPLCON1 que controla a ambos playfields por separado; los bits 3-0 para el playfield 1 (bit-planes 1,3,5) y los bits 7-4 para el playfield 2 (bit-planes 2,4,6).

Nota: Siempre se utilizan todos los bits, si se tiene un solo playfield hay que poner los bits 3-0 igual que los 7-4.

SUMARIO DE SCROLL DE PLAYFIELDS

Los pasos para un playfield con posibilidad de scrolls son los mismos que para definir un playfield basico, excepto por los siguientes pasos:

- **Definir el data-fetch.** Para llevar una palabra extra y comenzar 16 pixels antes del comienzo normal (sin scroll).

- **Definir el módulo.** El módulo es dos veces menor que sin scroll.

Se añaden los siguientes pasos:

- **Para scroll vertical, actualizar los punteros de los bit-planes según la cantidad del incremento del scroll.** actualizar BPLxPTH y BPLxPTL durante el vertical blank.

- **Para scroll horizontal, especificar el retardo.** Utilizar los bits 7-0 de BPLCON1 para tener de 0 a 15 bits de retardo (desplazamiento).

TEMAS AVANZADOS

En esta sección se describen características que se usan menos o son opcionales.

INTERACCIONES ENTRE LOS PLAYFIELDS Y OTROS OBJETOS

Los playfields comparten la pantalla con los sprites. El Capitulo 7, "El Hardware de Control del Sistema" muestra cómo los playfields pueden tener diferentes prioridades relativas respecto a los sprites y cómo los playfields pueden colisionar (ocupar la misma posición) que los sprites o el otro playfield.

MODO HOLD-AND-MODIFY (CONGELAR Y MODIFICAR)

Este es un modo especial que permite producir hata 4.096 colores al mismo tiempo en la pantalla. Normalmente, para cada valor formado por la combinación de los bit-planes se selecciona un registro de color cuyo contenido se carga en el circuito de salida del color para que el pixel se envie a la pantalla. Por tanto, cada pixel tiene el color del registro que ha seleccionado.

Sin embargo, en el modo HOLD-AND-MODIFY (HAM), el valor del circuito de color se congela, y uno de los tres componentes del color (rojo, verde, o azul) se modifica por los bits de ciertos bit-planes preseleccionados. Despues de la modificación, el pixel se envia a la pantalla.

El modo HAM permite hacer gamas de color muy precisas y sombreados en la pantalla. Por ejemplo, se puede cargar los primeros 16 registros con 16 colores distintos y mediante el uso del HAM hacer una gama de 16 tonos elevando la luminosidad de cada uno de los registros (de esta manera habrian 256 colores en pantalla). Hay que tener en cuenta que en un pixel de HAM sólo puede cambiar uno de los tres componentes del color cada vez. Por tanto, el efecto tiene un control limitado.

En modo HAM, se usan los seis bit-planes. Los planos 5 y 6 indican el modo de tratar a los planos 1-4, según sigue:

- Si la combinación de los bit-planes 6 y 5 da el valor 00, se produce la selección de color de manera normal. Por tanto los bit-planes 4-1, en este orden seleccionan uno de los primeros 16 registros de color.

- Si la combinación de los planos 6-5 es 01, el color del pixel anterior (a la izquierda) se congela y se modifica. La combinación de los planos 4-1 se usa para modificar el valor de los 4 bits "azules" del pixel anterior.

- Si la combinación de los planos 6-5 es 10, el color del pixel anterior (a la izquierda) se congela y se modifica. La combinación de los planos 4-1 se usa para modificar el valor de los 4 bits "rojos" del pixel anterior.

- Si la combinación de los planos 6-5 es 11, el color del pixel anterior (a la izquierda) se congela y se modifica. La combinación de los planos 4-1 se usa para modificar el valor de los 4 bits "verdes" del pixel anterior.

Usando el modo HAM, es posible obtener cualquier color únicamente definiendo un registro de color (COLOR 00 o color de fondo). Se trata la pantalla como una modificación del color original, como en el siguiente programa.

Los siguientes bits de BPLCON0 son necesarios para el modo HAM:

- Bit 11, HOMOD debe ser 1.
- Bit 10, DBLPF debe ser 0 (solamente un playfield)
- Bit 15, HIRES debe ser 0 (solamente baja resolución)
- Bits 14-12, BPU2-BPU0 deben ser 101 o 110 (5 o 6 bit-planes)

El siguiente ejemplo genera un playfield de 6 bit-planes en modo HAM. Los 32 registros de color se cargan con negro para demostrar que los colores se generan en modo HAM.

```
; Primero se preparan los registros de control
;
     LEA       CUSTOM,a0                ; Base de los custom chips...
     MOVE.W    #$6A00,BPLCON0(a0)       ; Seis bit-planes y modo HAM
     MOVE.W    #$0,BPLCON1(a0)          ; Scroll horizontal=0
     MOVE.W    #$0,BPL1MOD(a0)          ; Módulo planos impares=0
     MOVE.W    #$0,BPL2MOD(a0)          ; Módulo planos  pares=0
     MOVE.W    #$0038,DDFSTRT(a0)       ; Inicio de data-fetch
     MOVE.W    #$00D0,DDFSTOP(a0)       ; Parada de data-ferch
     MOVE.W    #$2C81,DIWSTRT(a0)       ; Inicio de Ventana
     MOVE.W    #$F4C1,DIWSTOP(a0)       ; Parada de Ventana
;
; Todos los registros de color = negro para demostrar el HAM
;
     MOVE.W    #32,d0                   ; Inicializa el contador
     MOVE.W    CUSTOM+COLOR00,a1        ; Primer registro de color
CREGLOOP:
     MOVE.W    #$0000,(a1)+             ; Mete color negro al registro
     DBRA      d0,CREGLOOP              ; Decrementa contador y repite bucle
;
; Rellena los seis bit-planes con una trama facilmente reconocible
;
; NOTA: Esto es sólo para uso como ejemplo. Normalmente para estos
;       bit-planes deberia reservarse bloques de memoria CHIP del sistema.
;
     MOVE.W    #$2000,d0                ; 2000 palabras largas por bit-plane
     MOVE.L    #$21000,a1               ; a1 apunta al bit-plane 1
     MOVE.L    #$23000,a1               ; a1 apunta al bit-plane 2
     MOVE.L    #$25000,a1               ; a1 apunta al bit-plane 3
     MOVE.L    #$27000,a1               ; a1 apunta al bit-plane 4
     MOVE.L    #$29000,a1               ; a1 apunta al bit-plane 5
     MOVE.L    #$2B000,a1               ; a1 apunta al bit-plane 6
```

```
FPLLOPP:
        MOVE.L    #$55555555,(a1)+      ; Llena el bit-plane1 con $55555555
        MOVE.L    #$33333333,(a2)+      ; Llena el bit-plane2 con $33333333
        MOVE.L    #$0F0F0F0F,(a3)+      ; Llena el bit-plane3 con $0F0F0F0F
        MOVE.L    #$00FF00FF,(a4)+      ; Llena el bit-plane4 con $00FF00FF
        MOVE.L    #$CF3CF3CF,(a5)+      ; Llena el bit-plane5 con $CF3CF3CF
        MOVE.L    #$3CF3CF3C,(a6)+      ; Llena el bit-plane6 con $3CF3CF3C
        DBRA      d0,FPLLOOP            ; Decrementa contador y repite bucle

;  Prepara una CopperList en $20000
;
;  NOTA: Como los bit-planes, la CopperList deberia ser reservada de
;        la memoria CHIP del sistema.

        MOVE.L    #$20000,a1            ; apunta al destino de CopperList
        LEA       COPPERL(pc),a2        ; apunta a la posición actual
CLOOP:  MOVE.W    (a2),a1)+             ; Mueve una palabra larga
        CMPI.L    #$FFFFFFFE,(a2)+      ; Comprueba fin de CopperList
        BNE       CLOOP                 ; Bucle hasta final de CopperList

;  Apunta a la CopperList
;
        MOVE.L    #$20000,COP1LCH(a0)   ; Carga CopperList en el registro
        MOVE.W    COPJMP1(a0),d0        ; Comienza la CopperList
;
;  Comienza DMA
;
        MOVE.W    #$8380,DMACON(a0)     ; Conecta Bit-planes y Copper

        BRA       .... siguiente cosa para hacer ....
;
;  CopperList para seis bit-planes. Bit-plane 1 esta en $21000; 2 en $23000;
;  3 en $25000; 4 en $27000; 5 en $29000 ; 6 en $2B000.
;
;  NOTA: Estas direcciones de bit-planes son sólo de ejemplo.
;
COPPERL:
        DC.W      BPL1PTH,$0002         ; puntero Bit-plane 1 = $21000
        DC.W      BPL1PTL,$1000
        DC.W      BPL2PTH,$0002         ; puntero Bit-plane 2 = $23000
        DC.W      BPL2PTL,$3000
        DC.W      BPL3PTH,$0002         ; puntero Bit-plane 3 = $25000
        DC.W      BPL3PTL,$5000
        DC.W      BPL4PTH,$0002         ; puntero Bit-plane 4 = $27000
        DC.W      BPL4PTL,$7000
        DC.W      BPL5PTH,$0002         ; puntero Bit-plane 5 = $29000
        DC.W      BPL5PTL,$9000
        DC.W      BPL6PTH,$0002         ; puntero Bit-plane 6 = $2B000
        DC.W      BPL6PTL,$B000
        DC.W      $FFFF,$FFFE           ; WAIT imposible (fin)
```

FORMANDO UNA PANTALLA CON DIFERENTES PLAYFIELDS

La libreria grafica (graphics.library) da la posibilidad de tener una pantalla con diferentes ViewPorts, cada uno con sus propios colores y resolución. Ver el Amiga ROM Kernel Manual para mas información.

Nota: Esto lo consigue el sistema mediante una CopperList que restaura para cada ViewPort sus punteros, registros de control, colores, etc. en la posición en que comienzan, dividiendo así la pantalla en distintas zonas.

USANDO UNA FUENTE EXTERNA DE VIDEO

Uno de los perifericos opcionales del Amiga es el genlock. El genlock permite "mezclar" la imagen del Amiga con una fuente de video externa (camara, magnetoscopio, o disco laser). Cuando se usa el genlock, el color de fondo se sustituye por la imagen de la fuente externa. Para mas informacion ver las instrucciones del genlock.

SUMARIO DE LOS REGISTROS DEL PLAYFIELD

Esta sección reune todos los registros usados en este capitulo y el significado de sus bits. El sumario de los registros de color esta en la siguiente sección. Ver Apendice A para el sumario de TODOS los registros.

BPLCON0 -Bit Plane Control

Nota: Los bits de este registro no se pueden manipular por separado.

Bit 0 - Sin usar

Bit 1 - ERSY (external synchronization enable)
1 -> Sincronización externa conectada (permite genlock)
0 -> Sincronización externa desconectada

Bit 2 - LACE (interlace enable) 0 -> no entrelazado, 1-> entrelazado

Bit 3 - LPEN (light pen enable) 0 -> desconectado, 1 -> lapiz óptico

Bits 4-7 Sin usar (deben ser 0)

Bit 8 - GAUD (genlock audio enable)
1 -> Sonido genlock conectado
2 -> Sonido genlock desconectado (durante el vertical blank, este bit sale al genlock por el pin 14, ZD)

Bit 9 - COLOR_ON (color enable)
1 -> color en video compuesto conectado
0 -> color en video compuesto desconectado

Bit 10 - DBLPF (double-playfield enable)
1 -> modo de doble playfield
0 -> modo de playfield único

Bit 11 - HOMOD (hold-and-modify enable) 0 -> no HAM, 1 -> si HAM

Bit 14,13,12 - BPU2,BPU1,BPU0 (number of bit-planes)
000 -> sólo color de fondo
001 -> 1 bit-plane, PLANO 1
010 -> 2 bit-planes, PLANOS 1y2
011 -> 3 bit-planes, PLANOS 1-3
100 -> 4 bit-planes, PLANOS 1-4
101 -> 5 bit-planes, PLANOS 1-5
110 -> 6 bit-planes, PLANOS 1-6
111 -> no usado

Bit 15 - HIRES (high-resolution enable) 0 -> baja res, 1 -> alta res.

BPLCON1 -Bit Plane Control

Bits 3-0 -PH1H(3-0) Delay o retardo del playfield 1
Bits 7-4 -PH2H(3-0) Delay o retardo del playfield 2
Bits 15-8 -sin usar

BPLCON2 -Bit Plane Control

Bits 0-5 Prioridad de sprites y playfields.

Bit 6 -PF2PRI (double-playfield priority)
0-> prioridad del playfield 1 sobre el 2
1-> prioridad del playfield 2 sobre el 1

Bits 7-15 sin usar

DDFSTRT -Data-fetch Start (inicio visualización de datos)

Bits 15-8 sin usar.

Bits 7-2 -Posición del pixel H8-H3 (H3 sólo en alta resolución)

Bits 1-0 sin usar

DDFSTOP -Data-fetch Stop (fin visualización de datos)

Bits 15-8 sin usar.

Bits 7-2 -Posición del pixel H8-H3 (H3 sólo en alta resolución)

Bits 1-0 sin usar

BPLxPTH -Bit-plane pointer (palabra alta de la dirección al bit-plane x)

BPLxPTL -Bit-plane pointer (palabra baja de la dirección al bit-plane x)

DIWSTRT -Display Window Start (coordenadas vert. y horz. de inicio)

Bits 15-8 -VSTART (V7-V0).

Bits 7-0 -HSTART (H7-H0).

DIWSTOP -Display Window Stop (coordenadas vert. y horz. de parada)

Bits 15-8 -VSTOP (V7-V0).

Bits 7-0 -HSTOP (H7-H0).

BPL1MOD -Bit-plane Modulo (módulo para bit-planes impares, playfield 1)

BPL2MOD -Bit-plane Modulo (módulo para bit-planes pares, playfield 2)

SUMARIO DE SELECCION DE COLOR

Esta sección contiene los sumarios de la selección de color incluyendo los contenidos de los registros de color, ejemplos de color, y diferencias en la selección de color en los modos de baja y alta resolución.

CONTENIDO DE LOS REGISTROS DE COLOR

Tabla 3-15: Contenido de los registros de color (sólo escritura)

Bits:	15-12	11-8	7-4	3-0
Contenido:	nada (0)	ROJO	VERDE	AZUL

EJEMPLOS DE COLORES PARA LOS REGISTROS

Tabla 3-16: Valores para los registros y color resultante

Valor	Color	Valor	Color
$FFF	Blanco	$1FB	Agua clara
$D00	Rojo ladrillo	$6FE	Azul celeste
$F00	Rojo	$6CE	Azul claro
$FB0	Rojo-naranja	$00F	Azul
$F90	Naranja	$61F	Azul brillante
$FB0	Naranja dorado	$06D	Azul oscuro
$FD0	Amarillo cadmio	$91F	Púrpura
$FF0	Amarillo limón	$C1F	Violeta
$BF0	Verde lima	$F1F	Magenta
$8E0	Verde claro	$FAC	Rosa
$0F0	Verde	$DB9	Tostado
$2C0	Verde oscuro	$C80	Marrón
$0B1	Verde bosque	$A87	Marrón oscuro
$0BB	Verde-azul	$CCC	Gris claro
$0DB	Agua	$999	Gris medio
		$000	Negro

SELECCION DE COLOR

Tabla 3-17: Selección de color en todos los modos.

(5,4,3,2,1)	HAM (4,3,2,1)		Registro
00000	0000 *	000 **	0 Fondo
00001	0001 *	001	1
00010	0010 *	010	2
00011	0011 *	011	3
00100	0100 *	PF1 100 *	4
00101	0101 *	101 *	5
00110	0110 *	110 *	6
00111	0111 *	111 *	7
01000	1000 *	000 **	8
01001	1001 *	001	9
01010	1010 *	010	10
01011	1011 *	PF2 011	11
01100	1100 *	100 *	12
01101	1101 *	101 *	13
01110	1110 *	110 *	14
01111	1111 *	111 *	15
10000 *	-	-	16
10001 *	-	-	17
10010 *	-	-	18
10011 *	-	-	19
10100 *	-	-	20
10101 *	-	-	21
10110 *	NO SE	NO SE	22
10111 *	USAN	USAN	23
11000 *	EN	EN	24
11001 *	ESTE	ESTE	25
11010 *	MODO	MODO	26
11011 *	-	-	27
11100 *	-	-	28
11101 *	-	-	29
11110 *	-	-	30
11111 *	-	-	31

* Color sólo disponible en baja resolución.
** Color transparente (doble playfield)
Nota: HAM, planos 6,5: 00 -> selección normal, 01 -> sustitución del Azul
10 -> sustitución del rojo y 11 -> sustitución del verde.

CAPITULO 4

EL HARDWARE DE LOS SPRITES

INTRODUCCION

Los Sprites son objetos del hardware que se crean y se mueven independientemente del playfield. Junto con los playfields, los sprites forman la pantalla del Amiga. Se pueden crear efectos de animación mas complejos usando el blitter, que se describe en el capitulo 6, "El Hardware del Blitter". Los sprites se generan en la pantalla por 8 canales especiales de DMA. Los sprites normales son de 16 pixels de ancho y cualquier número de lineas de alto. Los pixels de los sprites pueden adoptar 3 colores diferentes + transparente (muestra los que hay detras del pixel). Para sprites mas anchos o con mas colores se pueden combinar dos o mas de ellos.

Los canales de DMA de los sprites pueden reutilizarse muchas veces en la misma imagen. Por tanto, no se esta limitado a tener sólo ocho sprites en la pantalla al mismo tiempo.

SOBRE ESTE CAPITULO

Este capitulo explica los siguientes temas:

- Definición del tamaño, dibujo, color y posición de los sprites.
- Visualización y movimiento de los sprites.
- Combinación de sprites para ser mas anchos o tener mas colores.
- Reutilización de los canales DMA múltiples veces para tener en una misma imagen mas de 8 sprites al mismo tiempo.

FORMANDO UN SPRITE

Para formar un sprite, se debe definir primeramente y despues crear una structura con sus datos en memoria. Para definir el sprite se especifican las siguientes caracteristicas:

- Ancho en la pantalla hasta 16 pixels.
- Altura ilimitada.
- Cualquier dibujo.
- Una combinación de tres colores + transparente.
- Cualquier posición en la pantalla.

POSICION EN LA PANTALLA

La posición de un sprite esta definida por sus coordenadas X,Y respecto su pixel superior izquierdo y en modo de baja resolución y no-entrelazado. La definición de las coordenadas X e Y en el sistema aparece en la Figura 4-1. Observese que la posición (0,0) queda fuera del area visible de la imagen.

La cantidad de area visible tambien esta afectada por el tamaño de la ventana de visualización (definida por los valores de DDFSTRT, DDFSTOP, DIWSTRT y DIWSTOP). Ver el capitulo 3, "El Hardware de los playfields para mas información.

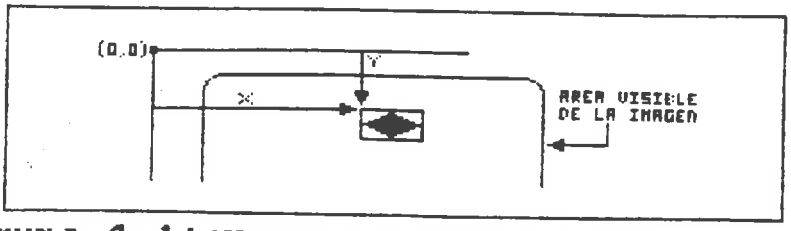


Figura 4-1: DEFINICION DE LA POSICION DE UN SPRITE EN LA PANTALLA

Posición Horizontal. La coordenada X de un sprite puede ser cualquier pixel de la pantalla entre 0 y 447. Aunque para que sea visible, tendra que estar dentro de los limites de la ventana de visualización. En los ejemplos de este capitulo se usa una ventana de 64 a 383 horizontal (320 pixels de ancha). Se pueden usar ventanas mayores o menores, pero antes se recomienda leer el capitulo "El Hardware de los Playfields". En realidad la imagen es mucho mayor, pero normalmente no es visible completamente.

Si se especifica un valor X que este fuera de la ventana, entonces una parte o todo el sprite no aparecerá en la pantalla. Esto a veces es deseable; por ejemplo para ocultar los sprites (clipped).

Para hacer aparecer a un sprite en su posición horizontal correcta dentro de la ventana de visualización, simplemente se añade su offset al valor X. En el ejemplo siguiente, esto se consigue añadiendo 64 a la X. Por ejemplo, para hacer que el pixel superior izquierdo del sprite aparezca en la posición 94 dentro de la ventana se haria el siguiente calculo:

Posición X + offset horizontal de la ventana = 94 + 64 = 158

Por tanto 158 sera el valor de X a escribir en la estructura de datos.

NOTA: La posición X representa al lugar del primer pixel a la izquierda del sprite aunque sea transparente y no aparezca en la imagen.

Si el sprite de la Figura 4-2 se coloca en la posición X 158, la imagen contenida en el apareceria 4 pixels despues, en 162 ya que los cuatro primeros pixels son transparentes y dejarian ver el fondo.

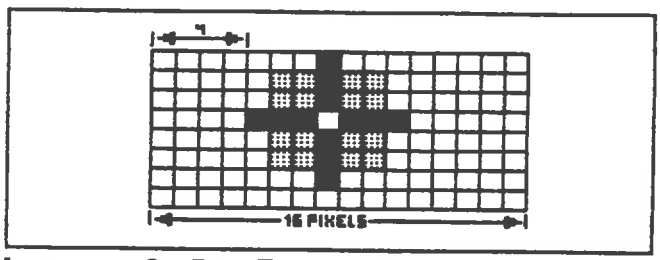


Figura 4-2: Posicion de Sprites

Posición Vertical. Se puede seleccionar cualquier posición vertical desde la linea 0 a la 262 para el borde superior del sprite. En los ejemplos de este capitulo, se usa una ventana NTSC con posiciones verticales desde la linea 44 a la 243. Esto permite la altura normal de 200 lineas en modo no-entrelazado. Si se especifica una posición vertical menor que 44 la parte superior del sprite no aparecera en la pantalla.

Para hacer que el sprite aparezca en la posición Y correcta, se suma el valor Y al offset vertical. Usando los números siguientes se suma 44 a la posición Y 25 para que este dentro de la ventana de visualización.

Posición Y + offset vertical de la ventana = 25 + 44 = 69

Sprites ocultos (clipped). Como se indicaba antes, los sprites pueden estar parcialmente o totalmente ocultos si estan en los límites de la ventana o los han sobrepasado. Los valores de 64 (horizontal) y 44 (vertical) son los normales para una imagen centrada en NTSC. Ver el capitulo 3, "El Hardware de los Playfields" para mas información sobre los offsets en PAL. Si se eligen otros offsets, los sprites seran ocultados en ellos.

TAMAÑO DE LOS SPRITES

Los sprites son de 16 pixels de ancho y de cualquier altura desde una linea hasta toda la pantalla. Tambien se puede mover un sprite "infinito" verticalmente para mostrar sólo una parte de el cada vez (scroll).

El tamaño de los sprites esta basado en los pixels de baja resolución y no-entrelazado. Los sprites son independientes de los modos de resolución de los playfields, de manera que si se cambia el modo del playfield no se produce ningún efecto sobre el sprite.

FORMA DE LOS SPRITES

Un sprite puede tener cualquier forma que quepa en el ancho de 16 pixels. La forma del sprite se define especificando que pixels aparecen en el sprite. Por ejemplo, las Figuras 4-3 y 4-4 muestran una forma utilizando X. La primera figura muestra cómo se dibujaria en el papel. La segunda figura muestra la nave espacial dentro del ancho de 16 pixels del sprite. Los 0s alrededor de la nave espacial marcan la parte del sprite que no se utiliza y es por eso transparente.

```
        X X
      X X X X X X
  X X X X X X X X X X
  X X X X X X X X X X
      X X X X X X
        X X
```

Figura 4-3: Forma de nave espacial.

```
0 0 0 0 X X 0 0 0 0 0 0 0 0 0 0
0 0 X X X X X X 0 0 0 0 0 0 0 0
X X X X X X X X X X 0 0 0 0 0 0
X X X X X X X X X X 0 0 0 0 0 0
0 0 X X X X X X 0 0 0 0 0 0 0 0
0 0 0 0 X X 0 0 0 0 0 0 0 0 0 0
```

Figura 4-4: Sprite con forma de nave espacial

En este ejemplo, la parte mas ancha de la forma es de 10 pixels y esta colocada a la izquierda del sprite. Cuando la forma es mas estrecha que el sprite, se puede elegir la parte del sprite en la que estara situada. Esta forma en concreto podria comenzar en cualquier pixel desde 2 a 7.

COLOR DE LOS SPRITES

Cuando se usan los sprites individualmente (es decir que no estan "conectados" como se explica en la sección "Sprites Conectados"), cada pixel puede ser de 3 colores o transparente. Los colores se seleccionan de forma similar a los playfields (Figura 4-5).

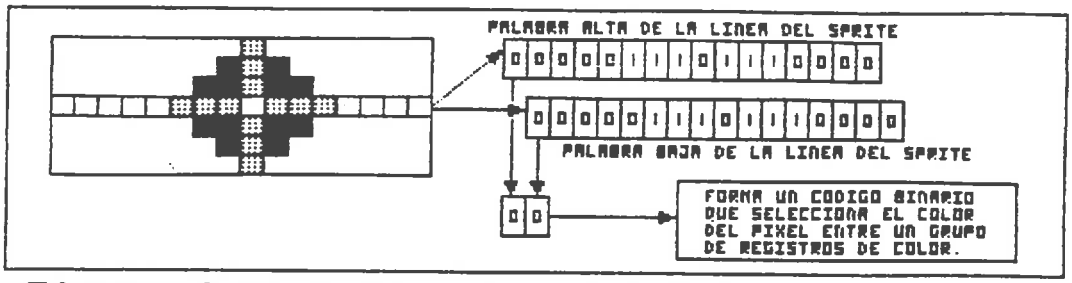


Figura 4-5: Definicion del Color en Sprites

Los 0s y 1s en las dos palabras de datos que definen cada linea del sprite, forman un número binario. Este número binario apunta a uno de los cuatro registros de color asignados a ese canal DMA. Los ocho sprites usan los registros 16-31 estando organizados en cuatro pares de sprites como se muestra en la Figura 4-6.

Nota: El valor del primer registro de color de cada grupo se ignora en los sprites. Cuando los sprites seleccionan estos registros se usa el color "transparente".

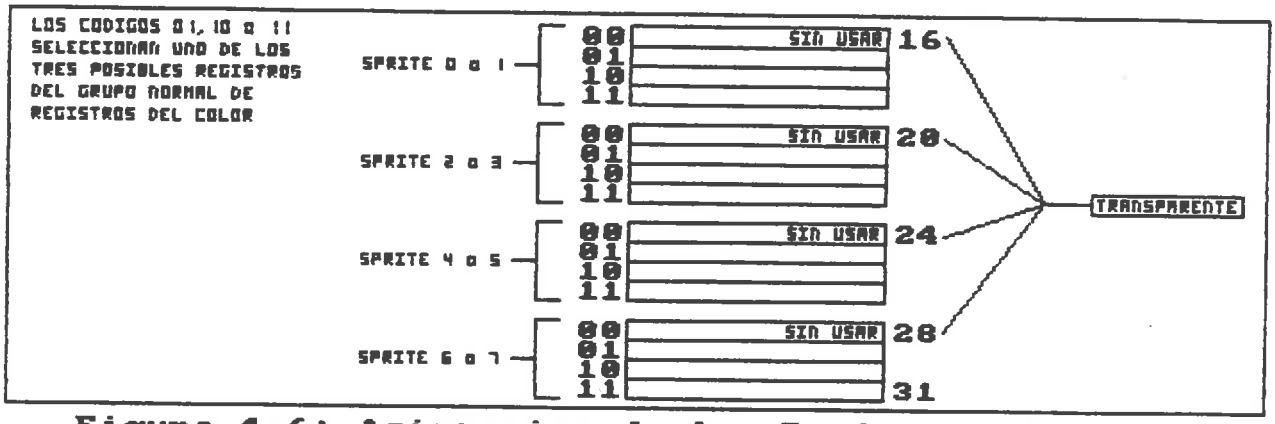


Figura 4-6: Asignacion de los Registros de Color

Si se necesitan unos colores determinados en un sprite se necesitara cargar los registros de color con esos colores (como se explica en el capitulo 3, "El Hardware de los Playfields")

El número binario 00 es especial en este esquema de color. Un pixel cuyo valor sea 00 pasa a ser transparente y muestra el color de cualquier sprite o playfield que tuviera menor prioridad en la imagen. Un objeto de prioridad menor aparece "detras" de un objeto de mayor prioridad. Cada sprite tiene un número fijo de prioridad con respecto a los otros sprites. Se puede variar la prioridad de los sprites respecto a los playfields (ver capitulo 7, "El Hardware de Control del Sistema" para mas información)

DISEÑANDO UN SPRITE

Para diseñar el sprite, es conveniente dibujarlo primero en papel. Para mostrar los distintos colores se utilizan los números del 0 al 3. Por ejemplo, la nave espacial anterior podria ser esto:

```
0000122332210000
0001223333221000
0012223333222100
0001223333221000
0000122332210000
```

El siguiente paso es convertir los números 0-3 en números binarios, que se usaran para crear las palabras que forman el sprite. La siguiente sección muestra como hacer esto.

CONSTRUYENDO LA ESTRUCTURA DE DATOS

Despues de diseñado el sprite, se necesita construir su estructura de datos, que es una serie de palabras (16 bits) en una zona continua de memoria. Las primeras palabras contienen la posición y la información de control y las demas contienen la descripción del color (la forma). Para crear la estructura de datos de un sprite se necesita:

- Escribir la posición horizontal y vertical en la primera palabra de control.
- Escribir la posición vertical de parada en la segunda palabra de control.
- Traducir los números decimales 0-3 del color a números binarios. Usar los valores binarios para construir las palabras descriptoras del color y escribirlas en memoria.
- Escribir las palabras de control que indican el final de la estructura de sprites.

Nota: Los datos de los sprites, al igual que con todos los demas datos utilizados por los custom chips, deben estar situados en la memoria CHIP. Tambien hay que asegurarse de que las palabras de las estructuras de los sprites estan alineadas en posiciones pares.

La Tabla 4-1 muestra la estructura de datos del sprite con la posición y función de cada palabra:

Tabla 4-1: Estructura de Datos del Sprite.

Posición	Palabra	Función
N	Palabra 1 de control	Posición horizontal y vertical
N+2	Palabra 2 de control	Posición de parada vertical
N+4	Descriptor de color bajo	Bits del color para linea 1
N+6	Descriptor de color alto	Bits del color para linea 1
N+8	Descriptor de color bajo	Bits del color para linea 2
N+10	Descriptor de color alto	Bits del color para linea 2
	.	
	.	
	.	
	Final de datos	Dos palabras que indican el siguiente uso del sprite

Todas las direcciones de memoria en los sprites son direcciones pares. Se necesita la cantidad de memoria CHIP continua para obtener dos palabras de control, dos palabras por cada linea horizontal y dos palabras de final.

La Figura 4-7 muestra como funciona la estructura de datos en el sprite.

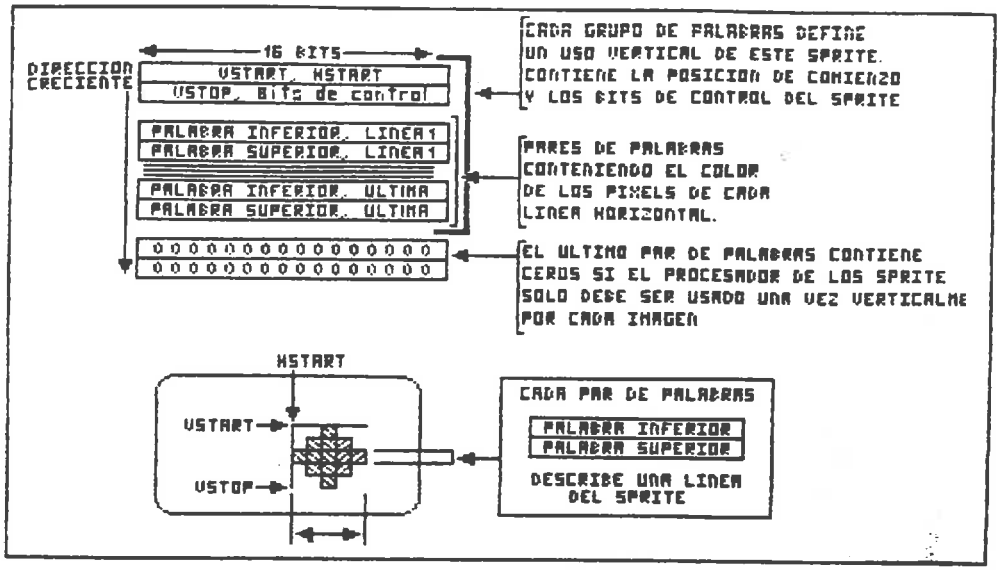


Figura 4-7: Trazado de la Estructura de Datos

Palabra 1 de control del Sprite: SPRxPOS

Esta palabra contiene la posición vertical (VSTART) y horizontal (HSTART) donde comienza el sprite. Los Bits 15-8 contienen los 8 bits inferiores de VSTART y los bits 7-0 los 8 bits superiores de HSTART.

Palabra 2 de control del Sprite: SPRxCTL

Esta palabra contiene la posición vertical de parada (la posición POSTERIOR a la última linea visualizada). Tambien contiene bits que permiten la conexión de sprites (attachament) que se describe mas tarde.

SPRxCTL

Bits 15-8	Los 8 bits inferiores de VSTOP
Bit 7	Usado en Attachment
Bits 6-3	Sin usar (deben ser 0)
Bit 2	Bit superior de VSTART
Bit 1	Bit superior de VSTOP
Bit 0	Bit inferior de HSTART

El valor (VSTOP - VSTART) define cuantas lineas de alto tendra el sprite cuando sea visualizado.

Palabras Descriptoras del Color del Sprite.

Se usan dos palabras para describir el color de cada linea horizontal en el sprite; la palabra de mayor orden y la palabra de menor orden. Para calcular cuantas palabras se necesitaran, se multiplica el número de lineas del sprite por 2. Los bits de la palabra de mayor orden corresponden a los bits superiores de los números binario del color de cada pixel; los bits de la palabra de menor orden corresponden a los bits inferiores de los números binario del color de cada pixel. Por ejemplo si el color de un pixel fuera el 2 (en binario 10) la palabra de mayor orden contendria un 1 y la de menor orden un 0.

Para formar las palabras descriptoras del color, se necesita primero tener un dibujo del sprite, mostrando el color de cada pixel con un número 0-3 (como en la figura de la nave espacial)

Despues, se traduce cada número del dibujo a binario. Abajo se muestra la primera linea en binario. Los números binarios se representan verticalmente con el bit inferior en palabra inferior y el bit superior en la palabra superior.

```
0 0 0 0 1 2 2 3 3 2 2 1 0 0 0 0  <- Primera linea

0 0 0 0 0 1 1 1 1 1 1 0 0 0 0 0  <- Palabra superior para linea 1
0 0 0 0 1 0 0 1 1 0 0 1 0 0 0 0  <- Palabra inferior para linea 1
```

De esta manera, se traduce cada linea del sprite a binario. Ver Figura 4-7. Cada uno de los números binarios formado por las dos palabras de cada linea se refiere a uno de los registros de color correspondiente al canal DMA que se esta utilizando. El canal de sprites 0, por ejemplo, toma sus colores de los registros 17-19. En la Tabla 4-2 se muestra la correspondencia entre los números binarios y los registros de color del canal 0 de sprites DMA.

Tabla 4-2: Registros de Color de los Sprites

Número binario	Número de registro
00	Transparente
01	17
10	18
11	19

Recuerdese que el número binario 00 significa siempre transparente y nunca se refiere a un color, excepto en el color de fondo (registro 00).

Palabras de final-de-datos

Cuando la posición vertical del contador del rayo es igual al VSTOP de **SPRxCTL**, las dos palabras siguientes se llevan a los registros de control en lugar de los registros de color. Estas dos palabras son interpretadas por el hardware de la misma manera que las palabras originales que se utilizaron al principio en los registros de control. Si el VSTART del nuevo registro de control es menor que la posición actual del rayo, el sprite no se volvera a usar en esa imagen. Para estar seguros, se usa el valor 0 en ambas palabras para indicar el final de la utilización de un sprite. La reutilizción de los sprites se explica mas tarde.

La siguiente estructura de datos es para el sprite de nave espacial. Que esta situado en V=65 y H=128 en la parte visible de la pantalla.

```
SPRITE:  DC.W   $6D60,$7200      ;VSTART, HSTART, VSTOP
         DC.W   $0990,$07E0      ;Primer par de palabras descriptoras
         DC.W   $13C8,$0FF0
         DC.W   $23C4,$1FF8
         DC.W   $13C8,$0FF0
         DC.W   $0990,$07E0
         DC.W   $0000,$0000      ;Final de los datos del sprite
```

VISUALIZACION DE UN SPRITE

Despues de construir la estructura de datos, se necesita indicar al sistema cómo visualizarlo. Esta sección describe la visualización de los sprites en modo "automatico". En este modo, una vez el canal DMA comienza a tomar y visualizar los datos, la visualización continua hasta que se detecta la posición de VSTOP. El modo manual se describe mas tarde en este capitulo.

Se usan los siguientes pasos para visualizar el sprite:

1. Decidir cual de los ocho canales DMA para sprites se va a usar (asegurandose de que ese canal este disponible)

2. Poner a punto los punteros del sprite para indicar al sistema dónde encontrar los datos del sprite.

3. Conectar el DMA de los sprites, si no lo estaba ya.

4. Para cada re-visualización de la imagen, re-escribir los punteros del sprite durante el vertical blank.

CUIDADO: Si se desconecta el DMA de los sprites cuando se esta visualizando uno (despues de VSTART pero antes de VSTOP), el sistema continuara visualizando la linea del sprite donde quedó cortado el suministro de datos por DMA. Esto provoca que aparezca una barra vertical en la pantalla. Se recomienda que el DMA de los sprites se desconecte solamente durante el vertical blank o durante alguna porción de la pantalla en la que se este SEGURO de que no se esta visualizando ningún sprite.

SELECCIONANDO UN CANAL DE DMA Y PONIENDO A PUNTO LOS PUNTEROS

Al decidir el canal de DMA se va a usar, se deben tener en cuenta los colores asignados al sprite y su prioridad en la pantalla.

Cada canal DMA de sprites usa dos punteros para leer los datos y las palabras de control del sprite. Durante el vertical blank antes de que se visualice el sprite por primera vez, se necesita escribir la dirección de memoria de sus datos en los estos punteros. Los punteros de los sprites se llaman SPRxPTH y SPRxPTL, dónde la "x" es el número de canal DMA de sprite. SPRxPTH contiene los 3 bits superiores de la dirección de memoria de la primera palabra de los daots datos y SPRxPTL contiene los 16 bits inferiores de la dirección. El bit menos significativo de SPRxPTL se ignora, porque las direcciones han de ser pares. Por tanto, sólo se usan 15 bits de SPRxPTL. Como es normal, se puede utilizar SPRxPTH con palabras largas (32 bits).

En el siguiente ejemplo el 68000 inicializa los punteros del sprite 0 con la dirección $20000. Normalmente, esto lo realiza el Copper.

```
MOVE.L #$20000,SPR0PTH+CUSTOM ;Escribe $20000 al puntero del sprite 0
```

Estos punteros son dinamicos; son incrementados por el canal DMA primero para apuntar a las palabras de control, despues a las palabras de datos, y finalmente a las palabras de final-de-datos. Despues de leer la información de control del sprite y guardarla en otros registros, se procede a leer las palabras descriptoras del color (parte grafica del sprite). Estas palabras se guardan en los registros de datos del sprite que son usados por el canal DMA para visualizar los datos en la pantalla. Para mas información sobre cómo los canales DMA de los sprites manejan la visualización, ver la sección posterior "Detalles del Hardware".

ACTUALIZANDO LOS PUNTEROS DE DIRECCIONES

Para un único FRAME (60 por segundo en NTSC y 50 en PAL), el sistema lee automaticamente la estructura de datos y produce el sprite en la pantalla con los colores que se habian especificado en los registros de color reservados para los sprites. Si se quiere que el sprite se visualice continuamente en todos los FRAMEs, se deben re-escribir los punteros del

sprite durante el vertical blank. Esto es necesario, porque durante la visualización de la imagen, los punteros se van incrementando para apuntar a los datos que se visualizan en cada linea a medida que avanza la imagen.

La re-escritura es parte de la rutina del vertical blank, que suele ser efectuada por las instrucciones de una CopperList.

EJEMPLO DE VISUALIZACION DE UN SPRITE

Este ejemplo visualiza el sprite "nave espacial" en la posición V=65, H=128. Recuerdese incluir el fichero "hw_examples.i" del Apendice J.

```
; Primero, se prepara un bit-plane
;
        LEA     CUSTOM,a0               ;a0 apunta a los custom chips
        MOVE.W  #$1200,BPLCON0(a0)      ;1 bit-plane, color on
        MOVE.W  #$0000,BPLMOD1(a0)      ;Modulo = 0
        MOVE.W  #$0000,BPLCON1(a0)      ;Scroll Horizontal=0
        MOVE.W  #$0024,BPLCON2(a0)      ;Sprites con prioridad sobre playfield
        MOVE.W  #$0038,DDFSTRT(a0)      ;Inicio de data-fetch
        MOVE.W  #$00D0,DDFSTOP(a0)      ;Final  de data-fetch
        MOVE.W  #$2C81,DIWSTRT(a0)      ;Inicio de la ventana de visualización
        MOVE.W  #$F4C1,DIWSTOP(a0)      ;Final  de la ventana de visualización

;
; Puesta a punto de los registros de color
;
        MOVE.W  #$0FFF,COLOR00(a0)      ;Color de fondo = blanco
        MOVE.W  #$0000,COLOR01(a0)      ;Color de tinta = negro
        MOVE.W  #$00F0,COLOR17(a0)      ;Color17 = amarillo
        MOVE.W  #$00FF,COLOR18(a0)      ;Color18 = azul celeste
        MOVE.W  #$0F0F,COLOR19(a0)      ;Color19 = magenta

;
; Pone la CopperList en $20000
;
        MOVE.L  #$20000,a1              ;A1 apunta al destino de la CopperList
        LEA     COPPERL(pc),a2          ;A2 apunta a la CopperList actual
CLOOP:
        MOVE.L  (a2),(a1)+              ;Mueve una palabra larga
        CMP.L   #$FFFFFFFE,(a2)+        ;Comprueba si es el final de la CopperL
        BNE     CLOOP                   ;Y si no, repite el bucle

;
; Mueve el sprite a $25000
;
        MOVE.L  #$25000,a1              ;A1 apunta al destino del sprite
        LEA     SPRITE(pc),a2           ;A2 apunta al sprite actual
SPRLOOP:
        MOVE.L  (a2),(a1)+              ;Mueve una palabra larga
        CMP.L   #0,(a2)+                ;Comprueba si es el final del sprite
        BNE     SPRLOOP                 ;Y si no, repite el bucle

;
; Ahora se escribe un sprite falso en $30000, debido a que cuando se activa
; el DMA de los sprites se activa para todos a la vez. Los sprites que no
; se van a usar se apuntan hacia este falso sprite.
;
        MOVE.L  #$00000000,$30000       ;Lo escribe
;
; Se apunta el Copper hacia la CopperList
;
        MOVE.L  $20000,COP1LC(a0)
```

```
;
; Rellena el bit-plane con $F0F0F0F0
;
     MOVE.L #$21000,a1              ;A1 apunta al bit-plane
     MOVE.W #1999,d0                ;2000-1 (por dbf) palabras largas(8000 b)
FLOOP:
     MOVE.L #$F0F0F0F0,(a1)+        ;Mueve la palabra larga
     DBF    d0,FLOOP                ;Decrementa y repite hasta el final
;
; Comienza DMA
;
     MOVE.W d0,COPJMP1(a0)          ;Hace que el Copper inicie la CopperList
     MOVE.W #$83a0,DMACON(a0)       ;conecta DMA: bit-planes, sprites, Copper
     RTS                            ;Vuelve
;
; Esta CopperList maneja 1 bit-plane (en $21000) y 8 sprites.
; El sprite 0 esta en $25    los demas son falsos y estan en $30000
;
COPPERL:
     DC.W   BPL1PTH,$0002           ;Bit-plane en $21000
     DC.W   BPL1PTL,$1000
     DC.W   SPR0PTH,$0002           ;Sprite 0 en $25000
     DC.W   SPR0PTL,$5000
     DC.W   SPR1PTH,$0003           ;Sprite 1 en $30000
     DC.W   SPR1PTL,$0000
     DC.W   SPR2PTH,$0003           ;Sprite 2 en $30000
     DC.W   SPR2PTL,$0000
     DC.W   SPR3PTH,$0003           ;Sprite 3 en $30000
     DC.W   SPR3PTL,$0000
     DC.W   SPR4PTH,$0003           ;Sprite 4 en $30000
     DC.W   SPR4PTL,$0000
     DC.W   SPR5PTH,$0003           ;Sprite 5 en $30000
     DC.W   SPR5PTL,$0000
     DC.W   SPR6PTH,$0003           ;Sprite 6 en $30000
     DC.W   SPR6PTL,$0000
     DC.W   SPR7PTH,$0003           ;Sprite 7 en $30000
     DC.W   SPR7PTL,$0000
     DC.W   $FFFF,$FFFE             ;Fin de CopperList
;
; Datos para el grafico de la nave. Aparece en V=65 y H=128
;
SPRITE:  DC.W   $6D60,$7200         ;VSTART, HSTART, VSTOP
         DC.W   $0990,$07E0         ;Primer par de palabras descriptoras
         DC.W   $13C8,$0FF0
         DC.W   $23C4,$1FF8
         DC.W   $13C8,$0FF0
         DC.W   $0990,$07E0
         DC.W   $0000,$0000         ;Final de los datos del sprite
```

MOVIENDO UN SPRITE

Un sprite generado en modo automatico se puede mover especificando una posición diferente en la estructura de datos. Para cada FRAME, se vuelven a leer los datos y se vuelve a visualizar el sprite. Entonces, si se cambia la posición en los datos antes de que el sprite sea re-visualizado, aparecera en la nueva posición y parecera que se ha movido.

Se debe tener cuidado con no mover el sprite (cambiar la palabra de control) al mismo tiempo que el sistema utiliza sus datos para posicionar el sprite en pantalla. Si se hace esto, el sistema podria tomar la posición para el inicio del sprite de un FRAME y la posición de parada del sprite del otro FRAME. Esto causaria un parpadeo y ensuciaria la pantalla. Por tanto, se debe cambiar el contenido de las palabras de control únicamente cuando el sistema no esta intentando leerlos. Normalmente, el vertical blank es el periodo ideal, y esta es una de las tareas mas que suele manejar el Copper durante el vertical blank, como en el próximo ejemplo.

Como los sprites se mueven por la pantalla, pueden colisionar con otros o con alguno de los dos playfields. Se puede usar el hardware para detectar estas colisiones y explotar esta posibilidad para efectos especiales. Ademas, se puede usar la detección de colisión para mantener un objeto móvil dentro de unos limites en la pantalla. La detección de colisión se describe en el Capitulo 7, "El Hardware del Control del Sistema".

En este ejemplo de movimiento de un sprite, la nave espacial va rebotando por la pantalla, cambiando de dirección cuando choca contra un borde.

Los datos de la posición del sprite (VSTART, VSTOP) estan en $25000 y VSTOP en $25002. Se escribe a estas posiciones para mover el sprite. Una vez durante cada FRAME, VSTART se incrementa (o decrementa) en 1 y HSTART en 2. Despues se calcula un nuevo VSTOP, que sera el nuevo VSTART+6.

```
        MOVE.B #151,d0              ;Inicializa el contador horizontal
        MOVE.B #194,d1              ;Inicializa el contador vertical
        MOVE.B #64,d2               ;Inicializa la posición horizontal
        MOVE.B #44,d3               ;Inicializa la posición vertical
        MOVE.B #1,d4                ;Inicializa el incremento horizontal
        MOVE.B #1,d5                ;Inicializa el incremento vertical
;
;Aqui se espera al comienzo de un nuevo FRAME
;Esto asegura una imagen sin parapadeos.
;
        LEA    CUSTOM,a0            ;a0 apunta a la base de los custom chips
VLOOP:
        MOVE.B VHPOSR(a0),d6        ;lee la posición vertical del rayo
        CMP.B  #$20,d6              ;Compara con el final de la pantalla PAL
        BNE.S  VLOOP                ;Repite bucle hasta final de FRAME

; Alternativamente se puede usar la siguiente rutina:
;       MOVE.W INTREQR(a0),d6       ;Lee los requerimientos de interrupción
;       AND.W  #$0020,d6            ;Enmascara todo menos el bit del vblank
;       BEQ    VLOOP                ;Repite el bucle hasta que sea 1
;       MOVE.W #$0020,INTREQ(a0)    ;El vblank esta a 1, se pone a 0
;
; Notese que esto sólo funciona si se desconecta la interrupción
; de vertical blank (no se recomienda para largos periodos

        ADD.B  d4,d2                ;incrementa el valor horizontal
        SUBQ.B #1,d0                ;decrementa el valor horizontal
        BNE    L1
        MOVE.B #151,d0              ;Contador agotado, se pone a 151
        EOR.B  #$FE,d4              ;Se niega el valor de incremento
L1:     MOVE.B d2,$25001            ;Escribe el nuevo valor HSTART
        ADD.B  d5,d3                ;incrementa el valor vertical
```

```
        SUBQ.B  #1,d1           ;Decrementa el contador vertical
        BNE     L2
        MOVE.B  #194,d1         ;Contador agotado, se pone a 194
        EOR.B   #$FE,d5         ;Se niega el valor de incremento
L2:     MOVE.B  d3,$25000       ;Escribe el nuevo valor VSTART
        MOVE.B  d3,d6           ;Se debe calcular ahora el nuevo VSTOP
        ADD.B   #6,d6           ;VSTOP es VSTART+6 en este ejemplo
        MOVE.B  d6,$25002       ;Escribe el nuevo VSTOP
        BRA     VLOOP           ;Bucle sin final
```

CREANDO SPRITES ADICIONALES

Para usar sprites adicionales, se debe crear una estructura de datos para cada uno y hacerlos aparecer en la pantalla como se explicaba en la sección anterior, utilizando los punteros SPR1PTH y SPR1PTL para el canal 1 DMA de sprites, los punteros SPR2PTH y SPR2PTL para el canal 2, etc.

NOTA: Cuando se conecta el DMA para un sprite, se esta conectando en realidad para los ocho, el modo automatico. Por tanto no se necesita repetir este paso cuando se usan sprites adicionales.

Una vez los canales DMA de sprites estan conectados, deben ser inicializados los ocho punteros de sprites a sprites reales o a sprites nulos. Un sprite sin inicializar puede causar apariciones espúreas.

Recuerdese que algunos sprites pueden quedar inútiles cuando se usan mas ciclos de DMA para visualizar la pantalla, por ejemplo cuando una imagen mas ancha de lo normal o con scroll horizontal. (Ver Figura 6-9: Reparto de Ciclos de Reloj en DMA).

Tambien, recuerdese que cada par de sprites toma sus colores de diferentes de registros de color, como se muestra en la Tabla 4-3.

Tabla 4-3: Registros de Color para pares de Sprites

Número Sprite	Registros de Color
0 y 1	17 - 19
2 y 3	21 - 23
4 y 5	25 - 27
6 y 7	29 - 31

PRIORIDAD DE LOS SPRITES

Cuando se tiene mas de un sprite en pantalla, se necesita tener en cuenta sus prioridades relativas, esto es, cuales sprites aparecen delante o detras de los otros. Cada sprite tiene una prioridad fija respecto a los otros. El sprite de número inferior tiene la prioridad mayor y aparece delante de todos los demas sprites; el sprite de número mayor tiene la prioridad menor. Esto se muestra en la Figura 4-8.

Ver el Capitulo 7, "El Hardware de Control del Sistema" para mas información sobre las prioridades.

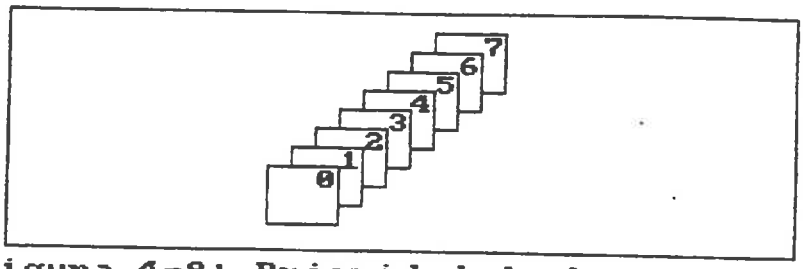


Figura 4-8: Prioridad de los Sprites

RE-USANDO LOS CANALES DMA DE LOS SPRITES

Cada uno de los ocho canales DMA de los sprites puede producir mas de una imagen controlable independientemente. Puede haber veces en las que se quiera mas de ocho objetos, o haber quedado con menos de ocho objetos porque se hayan conectado (attached) algunos sprites para producir mas colores, objetos mas grandes o superpuestos para producir imagenes mas complejas. Se puede re-usar cada canal DMA muchas veces en el mismo FRAME, como muestra la Figura 4-9.

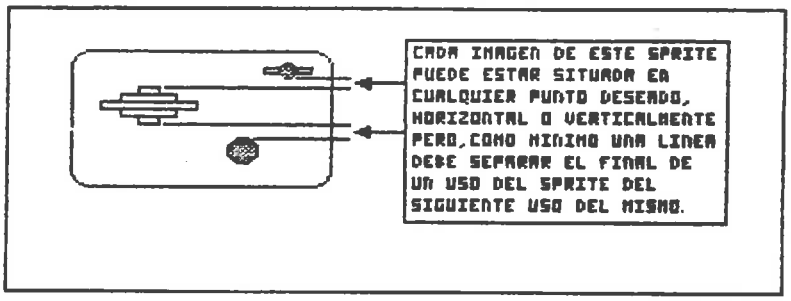


Figura 4-9: TIPICO EJEMPLO DE RE-USO DE UN SPRITE

En el uso de un solo sprite, se colocaban dos palabras a 0 al final de la estructura de datos para parar el canal DMA y que no continuara llevando datos a ese sprite durante el resto del FRAME. Para re-usar un canal DMA, se reemplaza este par de palabras a 0 por otra estructura completa de sprite, que describe el re-uso del canal DMA en un posición inferior en la pantalla al primer uso. Ahora se deben colocar las dos palabras a 0 al final de la estructura que contiene todos los usos del canal DMA. Por ejemplo, la Figura 4-10 muestra la estructura de datos que genera el dibujo anterior.

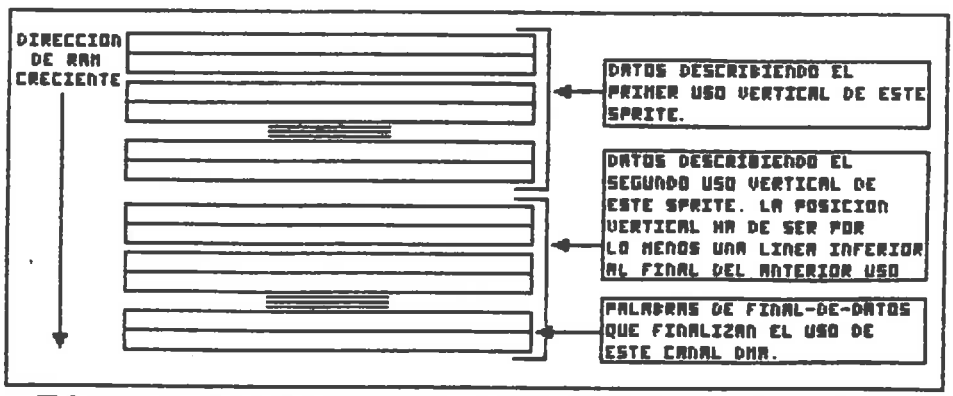


Figura 4-10: ESTRUCTURA DE DATOS TIPICA PARA RE-USO DE SPRITES

Las únicas restricciones en el re-uso de sprites en un FRAME es que la última linea de un uso del sprite debe estar separada de la primera linea del siguiente uso del sprite como mínimo en una linea horizontal. Esta restricción es necesaria porque sólo hay reservados dos ciclos DMA por linea para cada uno de los ocho sprites. El canal de sprites necesita el tiempo de la linea de separación para obtener la palabra de control que describe el siguiente uso del sprite.

El siguiente ejemplo visualiza el sprite de nave espacial y despues la re-visualiza como un objeto diferente. Sólo se muestra la lista de datos por que es lo único que cambia. Sin embargo, el sprite queda mejor con los registros de color puestos como en el ejemplo.

```
          LEA     CUSTOM,a0
          MOVE.W  #$0F00,COLOR17(a0)     ;Color 17 = Rojo
          MOVE.W  #$0FF0,COLOR18(a0)     ;Color 18 = Amarillo
          MOVE.W  #$0FFF,COLOR19(a0)     ;Color 19 = Blanco
SPRITE:   DC.W    $6D60,$7200
          DC.W    $0990,$07E0
          DC.W    $13C8,$0FF0
          DC.W    $23C4,$1FF8
          DC.W    $13C8,$0FF0
          DC.W    $0990,$07E0
          DC.W    $8080,$8D00            ;VSTART, HSTART, VSTOP para el
          DC.W    $1818,$0000            ;nuevo sprite
          DC.W    $7E7E,$0000
          DC.W    $7FFE,$0000
          DC.W    $FFFF,$2000
          DC.W    $FFFF,$2000
          DC.W    $FFFF,$3000
          DC.W    $FFFF,$3000
          DC.W    $7FFE,$1800
          DC.W    $7FFE,$0C00
          DC.W    $3FFC,$0000
          DC.W    $0FF0,$0000
          DC.W    $03C0,$0000
          DC.W    $0180,$0000
          DC.W    $0000,$0000            ;Final de datos del sprite
```

SPRITES SUPERPUESTOS

Para objetos móviles mas complejos o mas grandes, los sprites se pueden superponer. La superposición significa que los sprites tienen la misma o estan casi la misma posición en la pantalla. Una posición cercana (un sprite al lado del otro) puede dar como resultado un objeto de un ancho mayor a 16 pixels.

La prioridad del hardware de imagen hace que cuando dos sprites estan en la misma posición uno de ellos aparezca "detras" del otro. La prioridad mayor corresponde al sprite 0 y va descendiendo sucesivamente hasta el sprite 7 que tiene la prioridad menor. Por tanto, cuando se diseñan imagenes con sprites superpuestos, hay que asegurarse que el sprite de "delante" tiene un número menor al sprite de "detras". En la Figura 4-11, por ejemplo, la reja debe esta generada por el sprite 0, y el mono por el sprite 1 que tiene menor prioridad y por tanto aparecera "detras" de la reja cuando coincidan en la misma posición de la pantalla.

Se pueden crear objetos mas anchos de 16 pixels, poniendo un sprite al lado de otro. Por ejemplo, la Figura 4-12 muestra el sprite de nave espacial y como puede ser dos veces mas ancho usando dos sprites puestos uno junto al otro.

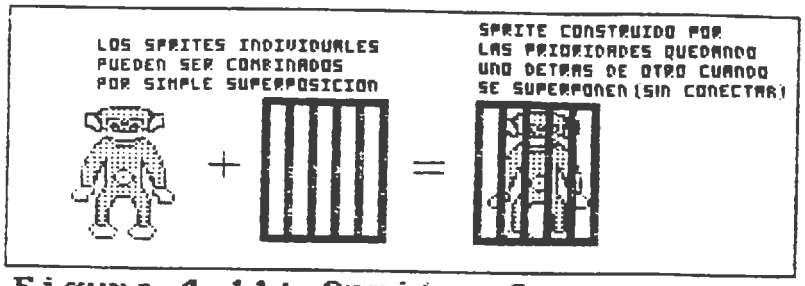


Figura 4-11: Sprites Superpuestos

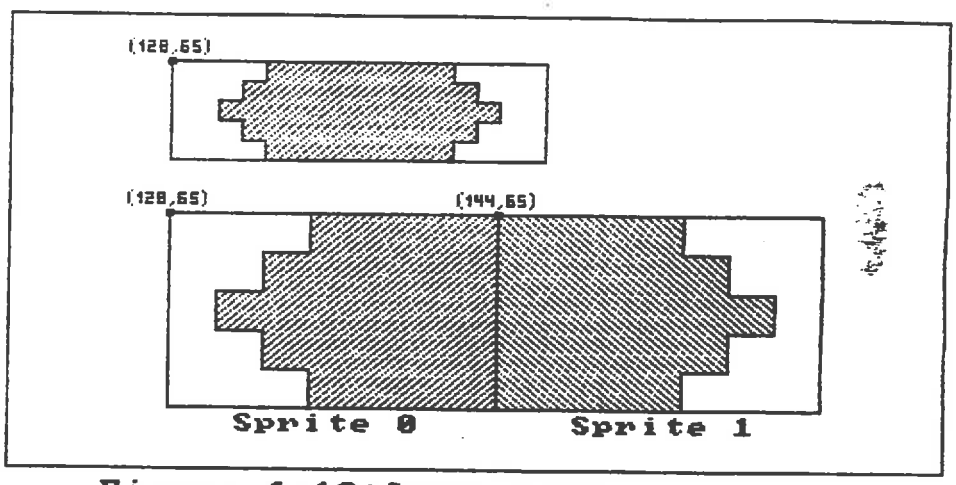


Figura 4-12: Poniendo los sprites uno al lado del otro

SPRITES CONECTADOS (ATTACHED)

Se puede crear un tipo de sprites que tiene 15 colores posibles (mas transparente) en lugar de 3 (mas transparente), conectando dos sprites simples. Para crear sprites compuestos, se debe:

- Usar dos canales por sprite, creando dos sprites del mismo tamaño y colocados en la misma posición.

- Poner a 1 el bit ATTACH en la segunda palabra de control del sprite.

Los 15 colores se seleccionan de todos los registros de color disponibles para los sprites -registros del 17 al 31. Esta cantidad extra de color es posible porque en este modo cada pixel contiene cuatro bits en lugar de dos como en el sprite simple. Por ejemplo, si se usan los canales DMA de sprites 0 y 1, la palabras descriptoras de alto y bajo orden para la linea 1 de ambas estructuras se combinan en la linea 1 del objeto compuesto.

Los sprites se pueden conectar en las siguientes combinaciones:

Sprite 1 con Sprite 0
Sprite 3 con Sprite 2
Sprite 5 con Sprite 4
Sprite 7 con Sprite 6

Cualquiera o todas estas conexiones pueden estar activas durante el mismo playfield. Como ejemplo, supóngase que se desea tener mas colores en el sprite de nave espacial y se estan usando los canales DMA de sprites 0 y 1. Habrán 5 colores más transparente en este sprite.

```
0000154444510000
0001564444651000
0015676446765100
0001564444651000
0000154444510000
```

La primera linea en este sprite requiere las cuatro palabras de datos mostradas en la Tabla 4-4 para formar los números de selección del color correctos.

El sprite de número mayor (el 1 en este ejemplo) contiene los bits de mayor orden del número binario (3 y 2). El sprite de número menor contiene los bits de menor orden (1 y 0).

Tabla 4-4: Palabras de Datos para la Linea 1 de la Nave espacial

Pixel:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Sprite1 AO	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Sprite1 BO	0	0	0	0	0	1	1	1	1	1	1	0	0	0	0	0
Sprite0 AO	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Sprite0 BO	0	0	0	0	1	1	0	0	0	0	1	1	0	0	0	0
Linea 1	0	0	0	0	1	5	4	4	4	4	5	1	0	0	0	0

Nota: AO-> Palabra alto orden, BO-> Palabra bajo orden

Ver en la Figura 4-7 el orden en que estas palabras se almacenan en memoria. Recuerdese que estos datos estan contenidos en DOS estructuras de datos.

Tabla 4-5: Registros de Color en Sprites Conectados

Decimal	Binario	Registro de Color
0	0000	16 (Transparente)
1	0001	17
2	0010	18
3	0011	19
4	0100	20
5	0101	21
6	0110	22
7	0111	23
8	1000	24
9	1001	25
10	1010	26
11	1011	27
12	1100	28
13	1101	29
14	1110	30
15	1111	31

La conexión sólo es efectiva cuando el bit ATTACH (bit 7 de la segunda palabra de control) esta puesto a 1 en la estructura de datos del sprite impar (sprite 1 en este ejmplo).

Cuando los sprites se mueven, la CopperList debe mantener ambos sprites en la misma posición relativa entre ellos. Si no se mantienen juntos en la pantalla, sus pixels cambiaran de color. Cada sprite tendra tres colores mas transparente, perc los colores seran diferentes que si fueran sprites sin conectar. La selección de color para el sprite de número inferior sera de los registros de color 17-19. La selección de color para el sprite de número mayor sera de los registros de color 20, 24 y 28.

La siguiente estructura de datos es para la nave espacial de seis colores construida mediante sprites conectados.

```
SPRITE0  DC.W   $6D60,$7200          ;VSTART = 65, HSTART = 128
         DC.W   $0C30,$0000
         DC.W   $1818,$0420
         DC.W   $342C,$0E70
         DC.W   $1818,$0420
         DC.W   $0C30,$0000
         DC.W   $0000,$0000          ;Fin sprite 0

SPRITE1  DC.W   $6D60,$7280          ;Igual que sprite 1 excepto el bit
         DC.W   $07E0,$0000          ;ATTACH puesto a 1
         DC.W   $0FF0,$0000
         DC.W   $1FF8,$0000
         DC.W   $0FF0,$0000
         DC.W   $07E0,$0000
         DC.W   $8080,$0000          ;Fin sprite 1
```

MODO MANUAL

Siempre es mejor cargar los registros de los sprites automaticamente utilizando los canales DMA. Pero, a veces, es útil cargar estos registros directamente desde uno de los procesadores. Los sprites pueden ser activados "manualmente" siempre que no vayan a ser usados por un canal DMA. El mismo sprite que esta mostrando una imagen mediante DMA en la parte inferior de la pantalla, puede ser cargado manualmente para mostrar una barra vertical desde la parte superior. Para utilizar manualmente los sprites es conveniente desconectar primero sus canales DMA.

Los sprites se visualizan manualmente escribiendo a los registros SPRxDATB y SPRxDATA en este orden. Se escribe SPRxDATA despues porque su dirección "arma" el sprite para ser visualizado en la siguiente comparación horizontal. Los datos escritos seran visualizados en cada linea, en la posición horizontal dada en la porción "H" de los registros de posición SPRxPOS y SPRxCTL. Si no se modifican los datos, el resultado sera una barra vertical. Si los datos son recargados en cada linea, se podra producir un sprite completo.

El sprite puede ser terminado ("desarmado") escribiendo al registro SPRxCTL. Si se escribe al registro SPRxPOS, se puede mover manualmente el sprite horizontalmente en cualquier momento, como en el uso normal de sprites.

DETALLES DEL HARDWARE DE LOS SPRITES

Los sprites se generan en la circuiteria de la Figura 4-13. Esta figura muestra en forma de bloques cómo un par de palabras de datos llegan a ser pixels visualizados en la pantalla.

Los elementos para la visualización de los sprites se explican a continuación:

- Registros de datos de los sprites. Los registros SPRxDATA y SPRxDATB mantienen los patrones de bits que definen una linea horizontal de un sprite por cada uno de los ocho sprites. Una linea es 16 pixels de ancha, y cada linea esta definida por dos palabras para proporcionar una selección entre tres colores y transparente.

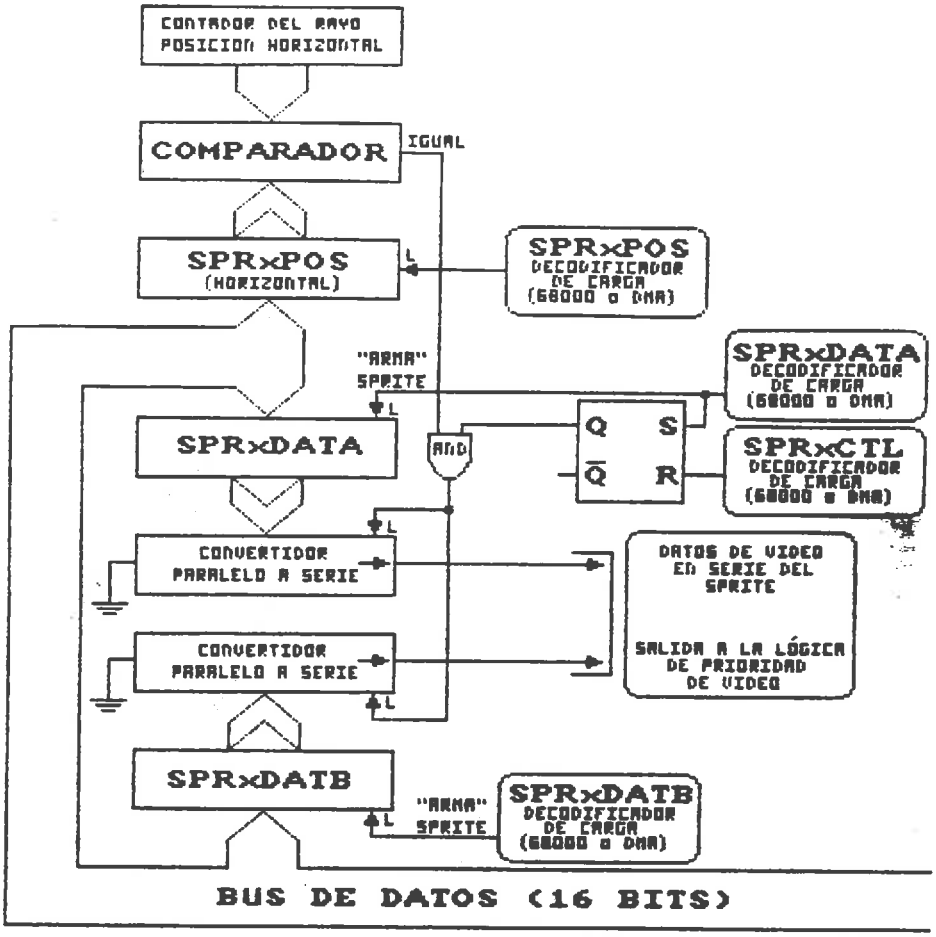


Figura 4-13: Circuiteria de Control de Sprites

- Convertidores paralelo a serie. Cada uno de los 16 bits del patron de de bits del sprite es enviado individualmente al circuito de selección de color a la vez que el pixel asociado con ese bit se visualiza en la pantalla.

  Inmediatamente despues de que los datos son transferidos desde los registros de datos de los sprites, cada convertidor paralelo-a-serie comienza a enviar los bits fuera del convertidor a la vez que los va desplazando, el mas significativo primero. El envio ocurre una vez durante el tiempo de un pixel de baja resolución y continua hasta que los 16 bits hayan sido transferidos al circuito de visualización. El desplazamiento y la salida de datos no comienzan de nuevo hasta que el convertidor no sea cargado por los registros de datos.

  Debido a que la imagen del monitor esta producida por un rayo de electrones que recorre la pantalla de izquierda a derecha, la imagen de bits de los datos corresponde exactamente a la imagen que aparece en realidad en la pantalla (los datos mas significativos a la izquierda)

- Datos de Video en serie del sprite. Los datos del sprite van al circuito de prioridad para establecer la prioridad entre sprites y playfields.

- Registros de posición de los sprites. Estos registros, llamados SPRxPOS, contienen la posición horizontal (X) y vertical (Y) por cada uno de los ocho sprites.

- Registros de control de los sprites. Estos registros, llamados SPRxCTL, contienen la posición de parada por cada uno de los ocho sprites y el bit ATTACH para indicar si el sprite esta conectado.

- Contador del Rayo. El contador del rayo indica al sistema la posición actual del rayo de electrones que esta produciendo la imagen.

- Comparador. Este dispositivo compara la posición del rayo con la posición Y del registro SPRxPOS. Si el rayo ha alcanzado la posición en la cual va a aparecer el sprite, el comparador envia una señal de carga a los convertidores paralelo-a-serie y comienza la visualización del sprite.

La Figura 4-13 muestra lo siguiente:

- Escribiendo al registro de control del sprite se desconecta la circuiteria del comparador horizontal. Esto evita que el sistema envie datos desde los registros de datos al convertidor o a la pantalla.

- Escribiendo al registro A del sprite se conecta el comparador horizontal. Esto permite la salida a la pantalla cuando la posición horizontal del rayo es igual al valor del registro de posición.

- Si el comparador esta conectado, el sprite sera enviado a la pantalla, con el pixel mas a la izquierda del sprite en la posición definida en la porción horizontal de SPRxPOS.

- Tanto tiempo como el comparador continue conectado, los contenidos actuales de los registros de datos seran enviados a la pantalla en la posición horizontal seleccionada (en cada linea).

- Los datos de los registros de datos de los sprites no cambian. Deben ser re-escritos por el usuario o por los canales DMA.

Estos componentes producen la visualización automatica como sigue:

Cuando los sprites estan en modo DMA, el registro de direcciones del sprite (compuesto por SPRxPTH y SPRxPTL que dan 18 bits) se usa para leer las primeras dos palabras de la estructura de datos del sprite. Estas palabras contienen la posición de comienzo y parada del sprite. A continuación, estas palabras se escriben en SPRxPOS y SPRxCTL. Despues de esto, el valor de los punteros apuntan a la primera palabra de datos (palabra inferior de datos para la linea 1 del sprite)

Al escribir en el registro SPRxCTL se desconecta el sprite. Ahora el canal DMA del sprite esperara hasta que el valor vertical del contador sea el mismo que el que se encuentra en la parte VSTART del registro SPRxPOS. Cuando estos valores coinciden, el sistema conecta el acceso a los datos del sprite.

El canal DMA del sprite examina los contenidos de VSTOP (de SPRxCTL, que es la posición de una linea posterior a la última del sprite) y VSTART (de SPRxPOS) para ver cuantas lineas de datos del sprite van a ser enviadas. se envian 2 palabras por linea, que son escritas en los registros de datos del sprite. La primera palabra se guarda en SPRxDATA y la segunda en SPRxDATB.

El acceso y el almacenamiento por cada linea horizontal ocurre durante el periodo de horizontal blank, lejos del comienzo de la parte izquierda de la imagen. Esto arma los comparadores horizontales del sprite y les permite comenzar la salida de los datos del sprite cuando la posición horizontal del rayo coincide con el valor guardado en HSTART que es parte de SPRxPOS.

Si la operación VSTOP - VSTART es igual a cero, no se producira salida de datos. El siguiente par de palabras sera enviado, pero no se alamacenara en los registros de datos del sprite, sino en SPRxPOS y SPRxCTL.

Cuando el sprite se usa sólo una vez por cada FRAME, el par final de palabras, que siguen a las palabras descriptoras del color, se carga automaticamente como el siguiente contenido de SPRxPOS y SPRxCTL. Para parar el sprite despues del primer bloque de datos, el par de palabras debe contener ceros.

Por tanto, si se ha formado un patrón de sprite en memoria, el mismo patrón sera reproducido como pixels automaticamente bajo el control DMA, linea por linea.

SUMARIO DE REGISTROS DE LOS SPRITES

Hay ocho conjuntos completos de registros para describir los sprites. Cada conjunto consta de cinco registros. Sólo se describen aqui los registros del sprite 0. Los demas son lo mismo, excepto el nombre del registro, porque cambia el número.

PUNTEROS

Los punteros son registros que usa el sistema para apuntar a los datos que van a ser usados en ese momento. Durante la visualización de la pantalla, los registros se van incrementando para apuntar a los datos que se necesitan en ese momento a medida que va progresando la imagen. Pero, estos registros deben ser actualizados durante el comienzo del vetical blank.

SPR0PTH y SPR0PTL Este par de registros contiene la dirección de 18 bits de los datos para el sprite 0. Los nombres para los otros sprites son:

SPR1PTH	SPR2PTH	SPR3PTH	SPR4PTH	SPR5PTH	SPR6PTH	SPR7PTH
SPR1PTL	SPR2PTL	SPR3PTL	SPR4PTL	SPR5PTL	SPR6PTL	SPR7PTL

REGISTROS DE CONTROL

SPR0POS es el registro de posición del sprite 0. La palabra escrita en este registro controla la posición de la pantalla en la cual aparecera el sprite (su esquina superior izquierda).

NOTA: Los sprites tienen una resolución de 320 por 200 NTSC (320 por 256 PAL). La resolución de los sprites es independiente de la de los playfields.

Posiciones de los bits:

- Bits 15-8 Posición vertical (V7-V0)
- Bits 7-0 Posición Horizontal (H8-H1).

NOTA: Este registro normalmente sólo lo utiliza su propio canal DMA. Ver los detalles sobre la organización de los datos de los sprites. Este registro es normalmente controlado directamente por DMA.

SPR0CTL es el registro que controla la información que es usada para controlar el proceso de llegada de datos al sprite. Normalmente sólo lo utiliza su propio canal DMA. Posiciones de los Bits:

- Bits 15-8 posición vertical de parada del sprite (V7-V0).
- Bit 7: ATTACH. Este es valido sólo para los sprites impares. Indica que los sprites 0,1 (2,3 o 4,5 o 6,7) seran considerados como uno solo y por tanto tendran 4 bits por pixel. El sprite impar contiene los bits de mayor orden. Durante este modo, los sprites conectados se mueven juntos bajo el control del procesador. Esto permite mas cantidad de colores. Los sprites conectados continuan teniendo movimiento independiente, aunque sólo se utiliza esta mayor cantidad de color cuando sus puntos se superponen.
- Bits 6-3 reservados para uso futuro (deben ser 0)
- Bit 2 es el bit V8 de la posición vertical.
- Bit 1 es el bit V8 del stop vertical.
- Bit 0 es el bit H0 de la posición horizontal.

Los nombres para los otros sprites son:

SPR1POS	SPR2POS	SPR3POS	SPR4POS	SPR5POS	SPR6POS	SPR7POS
SPR1CTL	SPR2CTL	SPR3CTL	SPR4CTL	SPR5CTL	SPR6CTL	SPR7CTL

REGISTROS DE DATOS

Los siguientes registros, aunque estan definidos en el espacio de direccionamiento del procesador, normalmente sólo los usa el Copper y los canales DMA. Son los registros que mantienen los datos obtenidos por los ciclos de DMA.

SPR0DATA	SPR1DATA	SPR2DATA	SPR3DATA	SPR4DATA	SPR5DATA	SPR6DATA	SPR7DATA
SPR0DATB	SPR1DATB	SPR2DATB	SPR3DATB	SPR4DATB	SPR5DATB	SPR6DATB	SPR7DATB

SUMARIO DE LOS REGISTROS DEL COLOR DE LOS SPRITES

Tabla 4-6: Registros de Color para los Sprites

Sprite	Valor	Color
0 o 1	00	Transparente
	01	17
	10	18
	11	19
2 o 3	00	20 *
	01	21
	10	22
	11	23
4 o 5	00	24 *
	01	25
	10	26
	11	27
6 o 7	00	28 *
	01	29
	10	30
	11	31

* Transparente en sprites simples

En los sprites simples, cada uno de ellos utiliza los 3 registros que tiene asignado, como se muestra en la tabla, pero en los sprites conectados o compuestos se utilizan los 15 registros en el objeto formado por los dos sprites simples, quedando como transparente la combinación 00.

INTERACIONES ENTRE LOS SPRITES Y OTROS OBJETOS

Los playfields comparten la pantalla con los sprites. El Capítulo 7, "El Hardware de Control del Sistema" muestra cómo los playfields pueden tener diferentes prioridades respecto a los sprites y cómo los playfields pueden colisionar (superponerse) con los sprites o con los otros playfields.

CAPITULO 5

EL HADWARE DEL AUDIO

INTRODUCCION

Este Capitulo muestra cómo acceder directamente al hardware del Audio para producir sonidos. Los temas principales en este capitulo son:

- Un pequeño vistazo a cómo el ordenador produce el sonido.
- Cómo producir sonidos estables, cambiantes y efectos mas complejos.
- Cómo usar los canales de audio para efectos especiales, utilizandolos en estereo o usando un canal para modular a otro.
- Cómo producir sonido de alta calidad, dentro de las limitaciones del sistema.

Una sección al final del capitulo muestra los valores a usar para crear notas musicales en una escala musical bien ajustada.

Este capitulo no es una guia sobre la sintesis de sonido por ordenador; una descripción completa requeriria un documento mucho mayor. El propósito del capitulo es mostrar cómo se pueden usar las caracteristicas del Amiga. La producción de sonido por ordenador es mucho mas compleja, y normalmente requiere una gran parte de pruebas y paciencia por parte del usuario -las instrucciones de este capitulo explican como crear sonidos y reproducirlos, despues se deben reajustar los parametros y volverlos a probar y asi sucesivamente.

Se recomiendan los siguientes libros para mas información sobre la creación de música con ordenadores:

- Wayne A. Bateman, Introduction to Computer Music (New York: John Wiley and Sons, 1980).
- Hal Chamberlain, Musical Applications of Microprocessors (Rochelle Park, New Jersey: Haydem, 1980).

INTRODUCCION A LA GENERACION DE SONIDO

El sonido viaja a traves del aire hasta los timpanos de los oidos como un ciclo repetido de variaciones en la presión del aire, u ondas de sonido. Los sonidos pueden ser representados como graficas que muestran cómo la presión del aire varia sobre el tiempo. Los atributos de un sonido, como se oye, estan relacionados con la forma de la onda. Si la onda es regular y repetitiva, el sonido sera un tono estable, como una sola nota musical. Cada repetición de la onda se llama ciclo de sonido. Si la onda es irregular, el sonido tendra poca tonalidad o ninguna, como un fuerte choque o un arroyo de agua. Según las veces que se repita la onda (su frecuencia) tendra un efecto sobre su tono; los sonidos con mayores frecuencias tienen tonos mayores. Los Humanos podemos oir sonidos que tengan una frecuencia entre 20 y 20000 ciclos por segundo. La amplitud de la onda (los puntos mas altos en la grafica), esta relacionada con la fuerza del sonido. Finalmente, la forma en general de la onda determina las cualidades del tono, o timbre. La Figura 5-1 muestra un tipo particular de onda, llamada onda senoidal, que representa un ciclo de un tono simple.

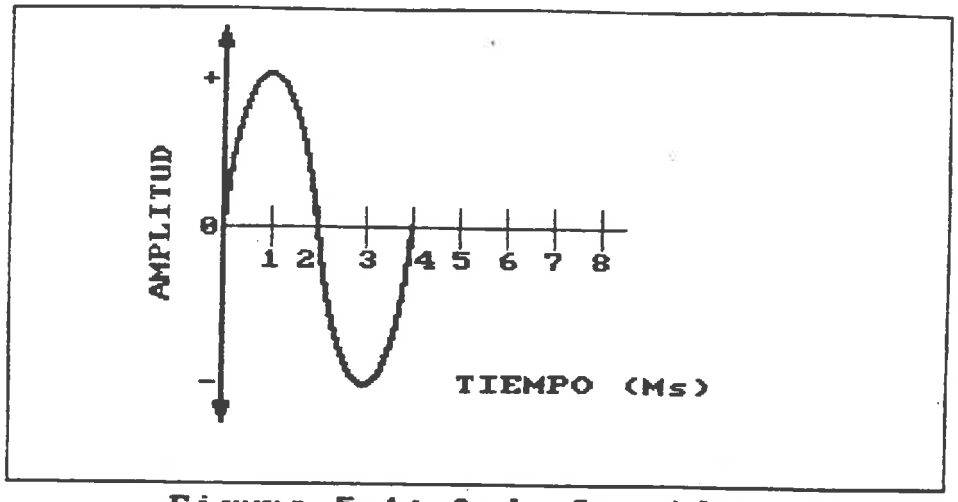


Figura 5-1: Onda Senoidal

En los dispositivos electrónicos de grabación y reproducción del sonido, los atributos de los sonidos estan representados por los parametros de la amplitud y la frecuencia. La frecuencia es el número de ciclos por segundo, y la unidad mas común de frecuencia es el Hertz (Hz), que es 1 ciclo por segundo. Los valores mas grandes, o altas frecuencias, se miden en KiloHertz (KHz) o MegaHertz (MHz).

La frecuencia esta fuertemente relacionada con el tono percibido del sonido. Cuando la frecuencia se incrementa, el tono aumenta. Esta relación es exponencial. Un incremento de 100 Hz a 200 Hz produce una gran subida del tono, pero un incremento de 1000 Hz a 1100 Hz casi no se nota. El tono musical se representa en octavas. Un tono es una octava mayor que otro cuando la frecuencia del primero es el doble que la del segundo.

El segundo parametro que define la onda es la amplitud. En un circuito electrónico, la amplitud esta relacionada con el voltage o la corriente del circuito. Cuando la señal va al altavóz, la amplitud se expresa en vatios (Watts = Volts * Ampers). La intensidad del sonido percibido se mide en decibelios (db). El oido humano tiene un rango de 120 db; 1b es la intensidad del sonido audible mas debil. Apróximadamente cada 10 db el sonido dobla su amplitud, y 1 db es el cambio mas pequeño en la amplitud que se nota en un sonido de fuerza media. El volumen, que es la amplitud de la señal del sonido cuando sale del circuito, corresponde logaritmicamente al nivel de decibelios.

Los parametros de frecuencia y amplitud de una onda senoidal son completamente independientes. Cuando el sonido se oye, sin embargo, hay una interacción entre la fuerza y el tono. Los sonidos de bajas frecuencias decrecen en fuerza mucho mas rapidamente que los de altas frecuencias.

El tercer atributo del sonido, el timbre, depende de la presencia o ausencia de tonos añadidos, o armónicos. Cualquier onda compleja es en realidad una mezcla de ondas senoidales de diferentes amplitudes, frecuencias y fases (el punto donde comienza la onda en el eje del tiempo). Estas ondas senoidales que la componen se llaman armónicos. Una onda cuadrada, por ejemplo, tiene un número infinito de armónicos.

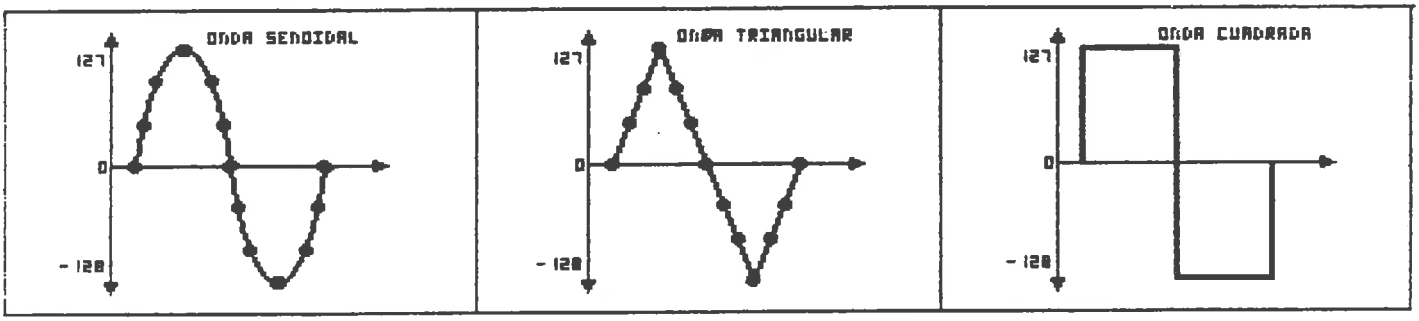
En suma, todos los sonidos estables se pueden describir por su frecuencia, amplitud total, y las amplitudes relativas de sus armónicos. Los equivalentes audibles de estos parametros son el tono, la fuerza y el timbre, respectivamente. Un sonido cambiante es un sonido estable cuyos parametros cambian sobre el tiempo.

En la producción electrónica del sonido, un dispositivo analógico, como una grabadora, graba las ondas del sonido y sus frecuencias ciclicas como la representación continua de las variaciones de la presión del aire. Despues la grabadora reproduce el sonido enviando las ondas a un amplificador dónde se transforman en ondas analógicas de voltage que van a un altavóz y se transforman en vibraciones de la presión del aire que el oyente percibe como sonido.

Un ordenador no puede almacenar la información de la ondas analógicas. En la produción del sonido por ordenador, la onda tiene que ser representada por una cadena de números finita. Esta transformación se hace dividiendo el eje del tiempo en la grafica de una onda en segmentos iguales, cada uno de los cuales representa una porción de tiempo tan pequeña que no se aprecian casi los cambios en la onda. Cada uno de los puntos resultantes se llama sample (muestra). Estos samples se almacenan en memoria, y se pueden reproducir en la frecuencia que se desee. El ordenador lleva los samples a un convertidor digital-analógico (DAC), que los convierte en una onda analógica de voltage. Para producir el sonido, las ondas analógicas se envian primero a un amplificador y despues a un altavóz.

La Figura 5-2 muestra un ejemplo de onda senoidal, cuadrada y triangular, y tambien la tabla de samples correspondiente a cada una.

Nota: Las ilustraciones no estan hechas a escala y hay menos puntos en las ondas que los que hay en las tablas. Los valores del eje de la amplitud 127 y -128 representan los limites superiores e inferiores relativos del sistema de sonido del Amiga.



TIEMPO	SENOIDAL	TRIANGULAR	CUADRADA
0	0	0	127
1	39	20	127
2	75	40	127
3	103	60	127
4	121	80	127
5	127	100	127
6	121	80	127
7	103	60	127
8	75	40	127
9	39	20	127
10	0	0	-128
11	-39	-20	-128
12	-75	-40	-128
13	-103	-60	-128
14	-121	-80	-128
15	-127	-100	-128
16	-121	-80	-128
17	-103	-60	-128
18	-75	-40	-128
19	-39	-20	-128

Figura 5-2: Valores de Amplitud Digitalizados

EL HARDWARE DE SONIDO DEL AMIGA

El Amiga tiene cuatro canales de hardware para el sonido, que se pueden programar independientemente para producir efectos complejos de sonido. Tambien se pueden conectar los canales de manera que un canal module el sonido de otro en frecuencia y/o en amplitud, o combinar dos canales para efectos en estereo.

Cada canal de sonido incluye un convertidor digital-analógico de ocho bits manejado por un canal DMA (acceso directo a memoria). El DMA de audio puede obtener dos samples durante cada linea horizontal. Para los tonos estables simples, el DMA puede hacer que la onda se repita automaticamente, tambien se puede programar todo tipo de efectos de sonido.

Hay dos metodos de producción basica de sonido en el Amiga -el modo de generación automatica del sonido (DMA) y el modo de generación directa (sin DMA). Cuando se usa el modo automatico, el sistema obtiene los datos automaticamente por DMA.

FORMANDO Y REPRODUCIENDO UN SONIDO

Esta sección muestra cómo crear un sonido estable simple y hacer que se oiga. Algunos conceptos basicos que se aplican en la generación de sonido en Amiga se introducen en esta sección.

Para producir un tono estable, se siguen los siguientes pasos basicos:

1. Decidir que canal se va a usar.
2. Definir la onda y crear la tabla de samples en memoria.
3. Poner los registros que indican al sistema dónde encontrar los datos y la longitud de los mismos.
4. Seleccionar el volumen al que se va a poner el sonido.
5. Seleccionar el periodo de "sampleado", o velocidad de salida de los samples.
6. Poner en marcha el DMA para ese canal de sonido.

DECIDIR QUE CANAL USAR

El Amiga tiene cuatro canales de sonido. Los canales 0 y 3 estan conectaoos al jack de salida izquierdo (L. AUDIO). Los canales 1 y 2 estan conectados al jack de salida derecho (R. AUDIO). Se debe seleccionar el canal en la parte que se quiere que aparezca el sonido.

CREANDO LOS DATOS DE LA ONDA

La onda usada como ejemplo en esta sección es una simple onda senoidal, que produce un tono puro. Normalmente sólo se define un ciclo completo de la onda en memoria. Para un sonido estable, un sonido que no cambie, los valores del comienzo de la onda, los puntos finales y la inclinación o declive de los datos al comienzo y al final deben estar aproximadamente en la misma posición. Esto asegura que una repetición continua de la onda sonará como una cadena continua de sonido.

Los datos del sonido estan organizados como un bloque de elementos de ocho bits; cada elemento es un sample de la onda que esta en el rango de +127 a -128. Cada palabra de datos que se obtiene por DMA consiste de dos samples.

Como ejemplo, el conjunto de datos mostrados abajo producen una aproximación muy cercana a la onda senoidal.

Nota: Los datos almacenados en orden de direcciones de bytes con el primer valor de amplitud digitalizado en la dirección de byte mas baja, el segundo en la siguiente dirección de byte, y asi sucesivamente. Tambien, notese que el primer byte de datos debe estar en dirección par. Esto es porque el DMA del audio obtiene una palabra (16 bits) cada vez y luego la usa como dos bytes separados.

Para usar el canal de audio 0, se escribe la localización de "datos-sonido" en AUD0LC, dónde los datos del sonido estan organizados como se muestra abajo. Para simplificar, en tabla pone "AUDxLC" en lugar de "AUDxLCH" y "AUDxLCL". Para que los canales DMA del sonido puedan obtener los datos, la dirección de la tabla a la cual apunta AUD0LC debe estar en la memoria CHIP.

Tabla 5-1: Bloque de Datos de sonido para el Canal 0

datos-sonido ->	AUD0LC *	100	98
	AUD0LC+2	92	83
	AUD0LC+4	71	56
	AUD0LC+6	38	20
	AUD0LC+8	0	-20
	AUD0LC+10	-38	-56
	AUD0LC+12	-71	-83
	AUD0LC+14	-92	-83
	AUD0LC+16	-100	-98
	AUD0LC+18	-92	-83
	AUD0LC+20	-71	-56
	AUD0LC+22	-38	-20
	AUD0LC+24	0	20
	AUD0LC+26	38	56
	AUD0LC+28	71	83
	AUD0LC+30	92	98

* AUD0LC es una dirección par

INFORMANDOLE AL SISTEMA SOBRE LOS DATOS

En orden a obtener los datos del sonido para el canal de audio, el sistema necesita conocer dónde estan situados y que longitud tienen, en palabras.

Los registros de localización AUDxLCH y AUDxLCL contienen los 3 bits superiores y los 15 bits inferiores, respectivamente, de la dirección de comienzo de los datos del sonido. Como estos dos registros estan contiguamente, escribiendo una palabra larga en AUDxLXH se mueve la dirección de los datos del sonido en ambos registros. La "x" en los nombres de registros sustituye al número del canal de audio dónde ocurrira la salida del sonido. Los canales estan numerados 0, 1, 2 y 3.

Estos registros son registros de localización, y se distinguen de los de posición en que sólo se necesita especificar su contenido una vez; no es necesario restaurar sus valores iniciales cuando se desea que el canal de audio repita la misma onda. Cada vez que el sistema obtiene la última palabra del area de datos, usa el contenido de estos registros de localización para encontrar de nuevo el comienzo de los datos. Asumiendo que la primera palabra de datos comienza en la localización "datos-sonido" y se esta usando el canal 0, los registros se utilizarian asi:

```
LEA     CUSTOM,a0               ; Base de los custom chips
LEA     DATOSSONIDO,a1          ; Dirección "datos-sonido"
MOVE.L  a1,AUD0LCH(a0)          ; Pone la dirección en el registro
```

La longitud de los datos es el número de samples de la onda dividido entre dos, o el número de palabras del conjunto de datos. Usando el conjunto de samples de antes, la longitud de los datos es 16 palabras. Esta longitud se escribe en el registro de longitud de datos de este canal. El registro de longitud se llama AUDxLEN, donde la "x" se refiere al número del canal. El registro AUD0LEN se pone a 16 como se muestra abajo.

```
LEA       CUSTOM,a0                ; Base de los custom chips
MOVE.L    #16,AUD0LCH(a0)          ; Pone la longitud en el registro
```

SELECCIONANDO EL VOLUMEN

El volumen que se pone aqui es el volumen total para todo el sonido que llega del canal de audio. Las fuerzas relativas de los sonidos, que son importantes a la hora de combinar notas, estan determinadas por la ampliud de la onda. Hay un registro de seis bits por cada canal de audio. Para controlar el volumen del sonido que saldra por el canal de audio seleccionado, se escribe el valor deseado en el registro AUDxVOL, donde "x" es el número del canal. Se pueden especificar valores de 0 a 64. Estos valores corresponden a niveles de decibelios. Al final del capitulo hay una tabla que muestra los valores de decibelios para los 65 niveles de volumen.

Para un sonido a volumen 64, con valores maximos en los datos de -128 a 127, el voltage de salida estara entre +0.4 y -0.4 voltios.

Tabla 5-2: Algunos Valores del Volumen

Volumen	**Decibelios**	
64	0	(volumen maximo)
48	-2.5	
32	-6.0	
16	-12.0	(12 db menos que el volumen
0	-----	a al nivel maximo)

Para cualquier puesta del volumen de 0 a 64, se escribe el valor a los bits 5-0 de AUD0VOL. Por ejemplo:

```
LEA       CUSTOM,a0                ; Base de los custom chips
MOVE.L    #48,AUD0VOL(a0)          ; Pone el volumen en el registro
```

Los decibelios se muestran como valores negativos sobre un maximo de 0 porque esta es la forma en que los dispositivos de grabación, como una grabadora, muestran el volumen de grabación. Normalmente, la grabadora tiene un indicador que muestra el 0 como el nivel de grabación óptimo. Cualquier nivel inferior a este significa menos cantidad de volumen.

SELECCIONANDO LA VELOCIDAD DE SALIDA DE LOS DATOS

El tono del sonido producido por la onda depende de su frecuencia. Para indicar al sistema que frecuencia ha de usar, se necesita especificar el periodo de "sampleado". Este periodo indica el número de ciclos del reloj del sistema, o intervalos de tiempo, que deben pasar entre cada sample para alimentar al convertidor digital-analógico del canal de audio. Hay un registro del periodo por cada canal. El valor del registro del periodo se usa por sistema de cuenta atras; cada vez que el registro cuenta atras hasta 0, se envia otro sample del bloque de datos al convertidor digital-analógico. En unidades, el valor del periodo representa los ciclos del reloj por sample. El valor minimo que se usa en NTSC es 124 y en PAL 123 y el maximo 65535. Para sonido de alta calidad, hay otros detalles sobre el periodo de "sampleado" que se veran en la sección "Producciendo Sonido de Alta Calidad".

Nota: Un periodo bajo corresponde a una frecuencia del sonido alta y un periodo alto corresponde a una frecuencia baja.

Limitaciones en la Selección del Periodo de Sampleado.

El periodo de "Sampleado" esta limitado por el número de ciclos DMA que estan reservados para los canales de audio. Cada canal tiene un slot reservado por cada linea horizontal, por eso puede obtener dos samples (una palabra) por cada linea. El siguiente calculo da la maxima velocidad de "sampleado" en samples por segundo para un sistema NTSC:

2 samples/linea * 262.5 lineas/frame * 59.94 frames/segundo = 31.469

Pero el número 31.469 samples/segundo es un maximo teórico porque el hardware esta diseñado para llegar a 28.867 samples/segundo. El intervalo de temporización del sistema es de 279.365 ns (nanosegundos), o 0.279365 µs (microsegundos). Este valor maximo de 28.867 samples/segundo corresponde a 34.642 µs por sample (1/28.867 = 0.000034642). La fórmula para calcular el periodo de "sampleado" es esta:

$$\text{Valor del periodo} = \frac{\text{intervalo de sampleado}}{\text{intervalo del reloj}} = \frac{\text{constante del reloj}}{\text{samples/segundo}}$$

Por tanto, el periodo minimo resulta de dividir 34.642 µs/sps entre el número de µs por intervalo:

$$\text{Periodo minimo} = \frac{34.642\ \mu s/\text{sample}}{0.279365\ \mu s/\text{intervalo}} = 124\ \text{intervalos reloj/sample}$$

o:

$$\text{Periodo minimo} = \frac{3579545\ \text{ciclos/segundo}}{28867\ \text{samples/segundo}} = 124\ \text{ciclos/sample}$$

Por lo tanto, se debe escribir en el registro de periodo un valor como minimo de 124 para asegurar que el sistema DMA del audio podra proporcionar el siguiente sample cuando acabe el periodo del anterior. Si el periodo es menor a 124, cuando el ciclo llegue a 0, el DMA del audio no tendra suficiente tiempo para obtener el siguiente sample y se reutilizara el anterior.

28867 samples/segundo es tambien el maximo para los sistemas PAL (aunque el periodo minimo en PAL es 123).

Velocidades del Reloj

	NTSC	PAL	unidades
Constante del Reloj	3579656	3546895	ciclos/segundo (Hz)
Intervalo del Reloj	0.279365	0.281937	µs/intervalo

Nota: El intervalo del Reloj se deriva de la constante del reloj: **intervalo = 1/constante** despues se pasa el resultado a µs.

Especificando el Valor del Periodo

Despues de haber seleccionado el intervalo entre los samples, se debe calcular el valor que se pondra en el registro del periodo usando la formula anterior. Por ejemplo, se quiere producir ura onda senoidal de 1KHz, usando una tabla de ocho samples (4 palabras) como en la Figura 5-3.

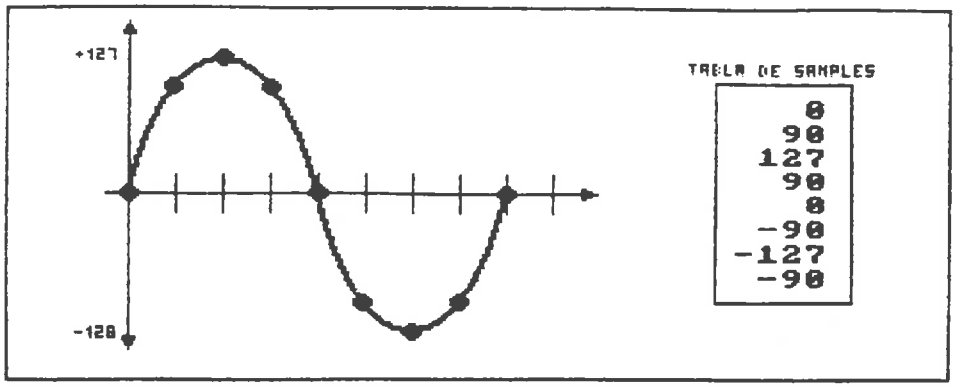


Figura 5-3: Ejemplo de Onda Senoidal

Para que la serie de ocho samples se oiga a 1KHz (1000 ciclos/segundo), cada ciclo completo sale a 1/1000 segundo . Por lo tanto, cada sample saldra a 1/8 del tiempo de toda la onda. Esto se traduce a 1000 µs por onda o 125 µs por sample. Para producir bien esta onda el periodo debe ser:

$$\text{Valor Periodo} = \frac{125\ \mu s/\text{sample}}{0.279365\ us/\text{intervalo}} = 447\ \text{intervalos/sample}$$

Despues el valor del periodo ha de escribirse en el registro AUDxPER, donde "x" es el número del canal. Por ejemplo las siguientes lines escriben el valor 447 en el registro AUD0PER:

```
LEA     CUSTOM,a0              ; Base de los custom chips
MOVE.L  #447,AUD0PER(a0)       ; Pone el periodo en el registro
```

Para que el sonido sea de alta calidad, evitando el efecto de "aliasing distortion", se deben observar las limitaciones del periodo que se explican en la sección "Produciendo Sonido de Alta Calidad".

Para ver la relación entre el periodo y el tono musical, ver la sección al final del capitulo que contiene una lista de la escala musical ajustada.

REPRODUCIENDO LA ONDA

Despues de haber definido la localización de los datos, la longitud, el volumen y el periodo, ya se puede reproducir la onda poniendo en marcha el DMA del canal de audio. Esto inicia la salida del sonido. Una vez puesto en marcha, el DMA continua hasta que el usuario lo pare. Por tanto, la onda se reproduce indefinidamente, produciendo un tono estable. El sistema usa el valor de los registros de localización cada vez que repite la onda.

Para que cualquier proceso DMA pueda funcionar, el bit DMAEN del registro DMACON debe estar a 1. Cuando DMAEN y AUDxEN estan a 1, el DMA comienza para el canal x.

Tabla 5-3: DMA y bits de conexión de Canales Audio

Registro DMACON

Bit	Nombre	Función
15	SET/CLR	A 1 se puede escribir en los otros bits y a 0 se pueden borrar.
9	DMAEN	A 0 desconecta todos los canales DMA.
3	AUD3EN	Conexión canal 3.
2	AUD2EN	Conexión canal 2.
1	AUD1EN	Conexión canal 1.
0	AUD0EN	Conexión canal 0.

Por ejemplo, si se esta usando el canal 0, entonces se escribe un 1 en el bit 9 para permitir el DMA y un 1 en el bit 0 para conectar el canal de audio 0:

```
LEA       CUSTOM,a0
MOVE.L    #(DMAF_SETCLR!DMAF_AUD0!DMAF_MASTER),DMACON(a0)
```

PARANDO EL DMA DEL AUDIO

Se puede parar un canal en cualquier momento escribiendo un 0 en el bit AUDxEN. Pero no se puede continuar la salida del sonido en el mismo punto escribiendo un 1 en el bit. Al conectar el canal de audio siempre comienza el sonido desde el principio del bloque de samples. Si el canal sólo se desconectara durante menos de dos periodos de "sampleado" si que continuaria en el mismo punto en que se quedó.

```
LEA       CUSTOM,a0
MOVE.L    #(DMAF_AUD0),DMACON(a0)
```

SUMARIO

Estos pasos son necesarios para producir un tono estable:

1. Definir la onda.
2. Crear el bloque de datos conteniendo los pares de samples. Normalmente, un bloque de datos contiene la definición de una onda.
3. Activar los registros de localización: AUDxLCH y AUDxLCL
4. Activar el registro de longitud, AUDxLEN, con el número de palabras de datos que deben ser llevadas por el canal antes de que se repita la onda.
5. Avtivar el registro del volumen, AUDxVOL.
6. Activar el registro del periodo, AUDxPER.
7. Iniciar el DMA escribiendo un 1 en el bit 9, DMAEN y un 1 en el bit del canal que se quiere activar, AUDxEN. (tambien bit 15 debe ser 1)

EJEMPLO

En este ejemplo, que recoge todos los segmentos de programas mostrados por este capitulo, se hace sonar una onda senoidal por el canal 0. Este ejemplo asume que se tiene acceso exclusivo al hardware del Audio, y puede no funcionar correctamente en el sistema multitarea.

```
        LEA       CUSTOM,a0              ; Base de los custom chips
        LEA       SINE,a1                ; Dirección datos de la onda
        MOVE.L    a1,AUD0LCH(a0)         ; Pone la dirección en el registro
        MOVE.L    #4,AUD0LCH(a0)         ; Pone la longitud en el registro
        MOVE.L    #64,AUD0VOL(a0)        ; Usa el volumen maximo
        MOVE.L    #447,AUD0VOL(a0)       ; Pone el periodo
        MOVE.L    #(DMAF_SETCLR!DMAF_AUD0!DMAF_MASTER),DMACON(a0)
        RTS
SINE:   DC.B      0,90,127,90,0,-90,-127,-90
        END
```

PRODUCIENDO SONIDOS COMPLEJOS

Ademas de los tonos simples, se pueden producir sonidos mas complejos, como diferentes notas musicales unidas en una melodia, diferentes notas sonando al mismo tiempo, sonidos modulados, samples digitalizados de una fuente de sonido externa (hace falta un digitalizador), sonido sintetico, etc.

UNIENDO TONOS

Los tonos se unen escribiendo los registros de localización y longitud, comenzando la salida del audio, y re-escribiendo los registros para preparar la siguiente onda que se desea conectar a la primera. Esto se hace facil por la temporización de las interrupciones del audio y la existencia de registros de back-up (copia). El canal DMA lee los registros de localización y longitud antes de comenzar la salida del sonido. Una vez se han leido los registros originales, se puede cambiar sus valores sin afectar a la operación que ya se habia iniciado. Por lo tanto, se puede reescribir los contenidos de los registros, comenzar la salida del audio, y despues re-escribirlos para que suene la segunda onda a continuación de la primera.

Las interrupciones ocurren inmediatamente despues de que el canal DMA haya leido los registros de localización y longitud y haya guardado sus valores en los registros de back-up. Una vez la interrupción haya ocurrido, se puede re-escribir los registros con la localización y la longitud del siguiente segmento de la onda. Esta combinación de registros de back-up y interrupciones temporizadas permite manipular los registros justo despues de que hayan sido leidos, haciendo que la salida del sonido sea continua y suave.

Si no se re-escriben los registros, la onda se repetira. Cada vez que el contador de longitud llega a cero, se vuelven a leer los registros de localización y longitud con los mismos valores (o los nuevos) para continuar la salida del sonido.

Ejemplo.

Este ejemplo detalla la acción del sistema DMA del audio paso a paso.

Supóngase que se quiere unir una onda senoidal y otra triangular, de manera que se repitan alternativamente. La siguiente secuencia muestra la acción del programa al igual que la interación con el sistema DMA del audio. El ejemplo supone que el periodo, volumen y longitud de los datos es el mismo para las dos ondas.

Programa de la Interrupción

Si (Onda=triangulo) escribe en AUD0LCL dirección onda senoidal.

Si no, Si (Onda=Senoidal) escribe en AUD0LCL dirección onda triangular.

Programa Principal

1. Pone el volumen, periodo y longitud
2. Escribe en AUD0LCL la dirección de la onda senoidal
3. Comienza DMA
4. Continua con cualquier otra cosa...

Respuesta del Sistema

Tan pronto como comienza el DMA,

a. Copia el contenido AUD0LEN al registro back-up de la longitud.

b. Copia el contenido AUD0LCL al registro back-up de la localización para ser usado como puntero a la tabla de samples.

c. Crea una interrupción al 68000 indicandole que ha terminado de hacer las copias de los registros.

d. Comienza a enviar datos al convertidor digital-analogico via DMA.

CREANDO MULTIPLES TONOS AL MISMO TIEMPO

Se puede hacer sonar multiples tonos usando los cuatro canales independientemente o sumando los samples en tablas y haciendolas sonar por un canal.

Debido a que los cuatro canales son programables independientemente, cada canal tiene su propio bloque de datos; por tanto se puede reproducir un sonido diferente por cada canal.

MODULANDO EL SONIDO

Para producir efectos de sonido mas complicados, se puede usar un canal de sonido para modular a otro. Esto incrementa el rango y tipo de efectos que se pueden hacer. Se puede modular la frecuencia y/o la amplitud de un canal.

La modulación en amplitud afecta al volumen de la onda. Se utiliza a veces para producir efectos de vibrato o de tremolo. La modulación en frecuencia afecta al periodo de la onda. Aunque la onda basica continua siendo la misma, el tono varia por la modulación de la frecuencia.

El sistema usa un canal para modular a otro cuando se conectan dos canales. Los bits de conexión del registro ADKCON controlan cómo se interpretan los datos de un canal de audio (ver la tabla de mas abajo). Normalmente, cada canal produce un sonido cuando esta conectado. Si el bit de conexión de un canal se pone a uno, el canal cesa de producir sonido y sus datos se utilizan para modular el sonido del siguiente canal en número. Cuando un canal se usa como modulador, las palabras de su bloque de datos ya no se tratan como bytes. En cambio, se usan como palabras moduladoras. Las palabras del canal modulador se escriben en los registros correspondientes del canal modulado cada vez que el registro de periodo del modulador llega a 0.

Para modular sólo la amplitud de la salida del audio, se debe conectar un canal como modulador de volumen. El bloque de datos del modulador se define como una serie de palabras que contienen la información del volumen en los bits 6-0, mientras que los bits 15-7 quedan sin usar.

Para modular sólo la frecuencia, se debe conectar un canal como modulador del periodo. El bloque de datos del modulador se define como una serie de palabras que contienen la información del periodo bits 15-0.

Si se quiere modular el periodo y la amplitud del mismo canal, se debe conectar el canal como mudulador de periodo y volumen. Por ejemplo, si se usa el canal 0 para modular el periodo y la amplitud del canal 1, se activan dos bits de conexión -el bit 0 para modular el volumen y el bit 4 para modular el periodo. Cuando el periodo y el volumen son modulados, las palabras del bloque de datos del canal modulador se definen alternativamente como modulador sobre el volumen y el periodo.

Tabla 5-4: Interpretación de los Datos en Modo de Conexión

Palabras de Datos	Independiente (sin modular)	Modulando Per. y Vol.	Modulando Sólo Periodo	Modulando Sólo Volumen
Palabra 1	dato - dato	volumen	periodo	volumen
Palabra 2	dato - dato	periodo	periodo	volumen
Palabra 3	dato - dato	volumen	periodo	volumen
Palabra 4	dato - dato	periodo	periodo	volumen

Las longitudes de los bloques de datos del canal modulador y el canal modulado son completamente independientes.

El sistema conecta los canales en un orden determinado, como muestra la Tabla 5-5. Para conectar un canal como modulador, se activa su bit de conexión poniendolo a 1. Si se conecta la modulación del periodo o el volumen para un canal, la salida de sonido de ese canal queda desconectada: el canal quedara conectado al siguiente canal en numero, como muestra la Tabla 5-5. Debido a que los canales conectados siempre modulan al siguiente canal, no se puede conectar el canal 3 porque quedaria desconectado sin modular a ninguno.

Tabla 5-5: Conexionado de Canales para Modulación

Registro ADKCON

Bit	Nombre	Función
7	ATPER3	Canal 3 desconectado.
6	ATPER2	Canal 2 modula periodo de 3.
5	ATPER1	Canal 1 modula periodo de 2.
4	ATPER0	Canal 0 modula periodo de 1.
3	ATVOL3	Canal 3 desconectado.
2	ATVOL2	Canal 2 modula volumen de 3.
1	ATVOL1	Canal 1 modula volumen de 2.
0	ATVOL0	Canal 0 modula volumen de 1.

PRODUCIENDO SONIDO DE ALTA CALIDAD

Cuando se quiere producir sonido de alta calidad hay que tomar en cuenta los siguientes factores:

- Transiciones de la onda.
- Velocidad de "sampleado".
- Eficiencia.
- Evitar el efecto de "Aliasing Distortion"
- Limitaciones del filtro de agudos.

CREANDO LAS TRANSICIONES DE LA ONDA

Para evitar sonidos desagradables cuando se cambia de una onda a otra, se necesita hacer las transiciones suaves. Los "cliks" se pueden evitar asegurandose que las ondas comienzan en el mismo punto (o casi el mismo) en que termina la anterior. Los "pops" se pueden evitar haciendo que la onda comience desde el punto 0 de la amplitud. Los "dzumps" se pueden evitar haciendo que la amplitud media de cada onda tenga el mismo valor aproximado. La amplitud media es la suma de los bytes que componen la onda dividida entre el número de bytes de la onda.

VELOCIDAD DE SAMPLEADO

Si se necesita una precisión alta en la frecuencia de salida, puede pasar que la frecuencia que se desea producir este entre dos posibles velocidades de sampleado, pero no lo suficientemente cercana a ninguna de ellas. En estos casos, se puede ajustar la longitud del bloque de datos para poder alterar la velocidad de "sampleado".

Para frecuencias mayores, se puede necesitar bloques de datos con mas de un ciclo completo de la onda para producir la frecuencia deseada con mas precisión, como se muestra en la Figura 5-4.

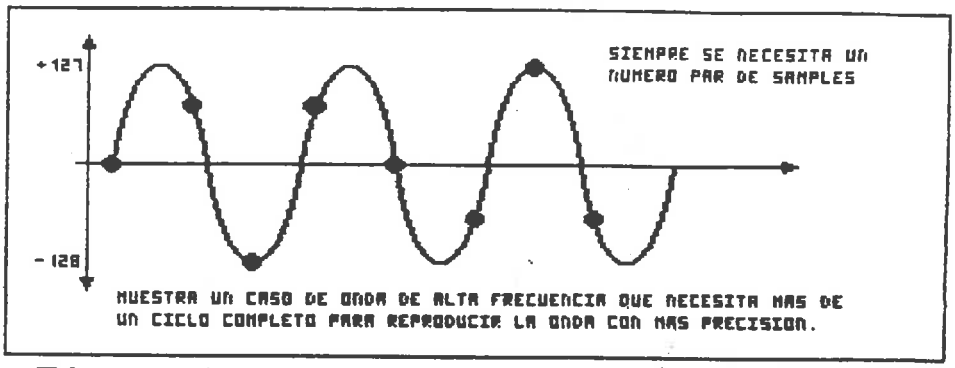


Figura 5-4: Onda con Ciclos Multiples

EFICIENCIA

Cuando se maneja el DMA del audio se produce un aumento en la actividad del sistema. Si se esta intentando producir un sonido sintetico que sea continuo y suave, se debe evitar en lo que sea posible este aumento de actividad del sistema. Basicamente, cuanto mas grande sea el bufer de audio que se le proporciona al sistema, menos veces necesitara interrumpir al 68000 para restaurar los punteros al siguiente buffer, por tanto se requerira menos interación con el sistema. Si sólo hay un buffer de ondas, el hardawre restaura los punteros automaticamente y no se produce el aumento de actividad en el sistema.

La sección "Uniendo Tonos" mostraba cómo se podian unir los finales de los tonos respondiendo a las interrupciones y cambiando los valores de los registros de localización para que los tonos se oyeran juntos. Si el sistema esta muy cargado, es posible que la respuesta a la interrupción no sucediera en el momento para asegurar una transición suave. Sin embargo, es aconsejable utilizar la tabla mayor posible dónde se necesite un sonido suave (en la que ya esten unidos los dos tonos). Esto aprovecha las ventajas del audio por DMA y minimiza el número de interrupciones a las que tiene que responder el 68000.

REDUCCION DEL RUIDO

Para reducir los niveles de ruido y producir un sonido perfecto, se debe usar el rango completo de -128 a +127 cuando se crea una onda. Esto reduce en gran medida el ruido (error de cuantización) que se añadiria a la señal si se usaran menos bits de precisión. El ruido de cuantización esta causado por la introducción de un error por redondeo o aproximación. Si se intenta producir una señal, como una onda senoidal, se puede representar la amplitud de cada sample con unos pocos digitos de precisión. La diferencia entre el número real y la aproximación es el error de redondeo, o ruido.

Doblando la amplitud, se crea la mitad del ruido porque el tamaño de los pasos de la onda sigue siendo el mismo y es por lo tanto una fracción mas pequeña de la amplitud.

En otras palabras, si se intenta representar una onda usando por ejemplo, un rango de sólo +3 a -3, el tamaño del error en la salida sera considerablemente mayor que si se usa un rango de +127 a -128 para crear la misma señal. Proporcionalmente, el valor digital usado para representar la amplitud de la onda tendra un error menor. A medida que se incrementa los niveles posibles del sample se decrementa el tamaño relativo de cada paso, y por tanto, decrece el tamaño del error.

Para producir sonidos flojos, tambien se define la onda en el rango completo, pero luego se ajusta el volumen. Esto mantiene el mismo nivel de precisión (relación señal-ruido) para los sonidos flojos y para los fuertes.

ALIASING DISTORTION

Cuando se crea una onda, se produce un efecto cuando la velocidad de sampleado se combina con la frecuencia que se quiere producir. Esto produce dos frecuencias adicionales, una en la velocidad de sampleado mas la frecuencia deseada y otra en la velocidad de sampleado menos la frecuencia deseada. Este fenómeno se denomina "aliasing distortion".

El "aliasing distortion" se elimina cuando la frecuencia de sampleado excede los 7 KHz. Esto pone la frecuencia adicional fuera del rango del filtro de agudos, eliminando las frecuencias no deseadas. La Figura 5-5 muestra las frecuencias que deja pasar el filtro de agudos del sistema.

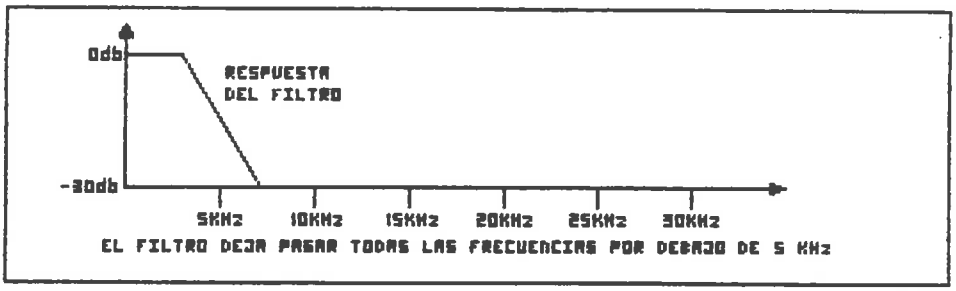


Figura 5-5: Frecuencias del filtro de agudos

La Figura 5-6 muestra que se puede usar una velocidad de "sampleado" de 12 KHz para producir una onda de 4 KHz. Ambas frecuencias quedan fuera del alcance del filtro de agudos, como muestran los siguientes calculos:

12 + 4 = 16 KHz 12 - 4 = 8 KHz

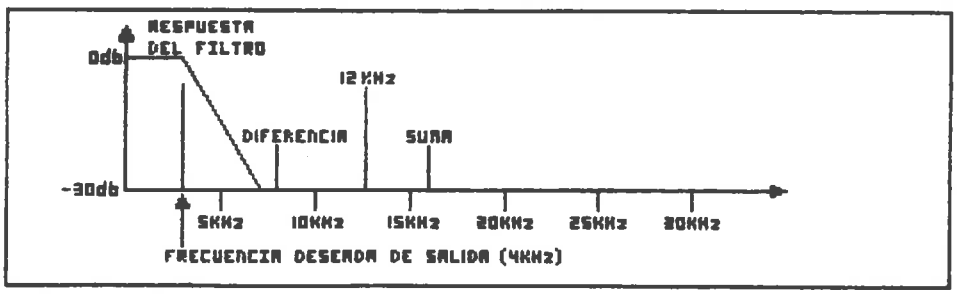


Figura 5-6: Sonido sin ruido (Sin Aliasing Distortion)

Se puede ver en la Figura 5-7 que es inaceptable usar una velocidad de sampleado de 10 KHz para producir una onda de 4 KHz. Una de las frecuencias añadidas (10 - 4) queda dentro del rango del filtro, y por lo tanto se oiria junto con la onda.

Todo esto da la siguiente ecuación, mostrando que la velocidad de sampleaado debe exceder la frecuencia de salida como minimo en 7 KHz, para que el filtro elimine las frecuencias añadidas:

Velocidad de sampleado minima = Componente de frecuencia mas alto + 7 KHz

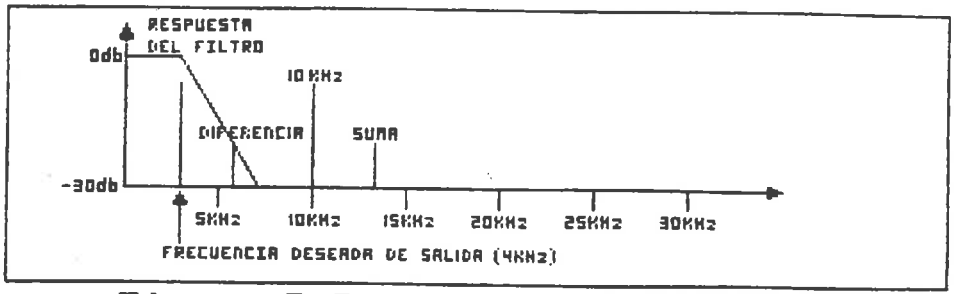


Figura 5-7: Aliasing Distortion

En la ecuación se utiliza "componente de frecuencia mas alta" porque en una onda compleja pueden haber múltiples elementos de frecuencias, en lugar de la onda senoidal pura.

EL FILTRO DE AGUDOS

El sistema incluye un filtro de agudos que elimina el "aliasing distortion" como se explicaba antes. Este filtro se activa a los 4 KHz y comienza a atenuar gradualmente (eliminar) la señal. Generalmente, las frecuencias mayores de 7 KHz ya no se oyen bien. Por lo tanto, se obtiene una respuesta mayor en el rango de frecuencias de 0 a 7 KHz. Si se estan creando frecuencias entre 0 y 7 KHz, se debe seleccionar un periodo de sampleado no inferior a 14 KHz, que corresponde al periodo de sampleado entre 124 y 256.

En un periodo de sampleado de 320, se comienza a perder los valores de alta frecuencia entre 0 y 7 KHz, como se muestra en la Tabla 5-6.

Tabla 5-6: Periodo de "sampleado" y relación con la Frecuencia

	Periodo	Velocidad	Maxima Frecuencia
Maximo	124	29 KHz	7 KHz
Minimo para 7 KHz	256	14 KHz	7 KHz
Muy bajo para 7 KHz	320	11 KHz	4 KHz

En la mayoria de los A2000 y A500 hay un bit de control que desconecta el filtro de agudos. Este bit de control es el mismo bit del CIA.A 8520 que controla la luz del LED rojo de "power" en el Amiga. Cuando se atraviesa el filtro, el sonido puede mejorar para algunas aplicaciones, pero es conveniente un filtro externo para eliminar las frecuencias parasitas.

USANDO LA SALIDA DE AUDIO DIRECTA (SIN DMA)

Es posible crear sonido escribiendo una palabra de datos para el audio cada vez en el registro de salida del sonido, en lugar de que el DMA obtenga las palabras de una lista de memoria. Este metodo de control del audio es mas intensivo para el microprocesador y no se recomienda para usos normales.

Para usar la salida directa del audio, no se debe conectar el DMA para el canal que se va a usar; esto cambia la temporización de las interrupciones. La interrupción normal ocurre despues de que se haya leido una dirección de datos; en la salida directa del audio, la interrupción ocurre despues de que se haya enviado una palabra al convertidor digital-a-analógico.

A diferencia del audio por DMA, en el audio directo, si no se escribe una nueva palabra en el registro antes de dos periodos de sampleado, la salida del sonido se detiene. El último valor continua en el convertidor digital-a-analógico. El volumen y el periodo se introducen como es usual.

LA ESCALA MUSICAL BIEN AJUSTADA

La Tabla 5-7 da una aproximación a la escala exacta para una octava cuando el tamaño del sample es de 16 bytes. La columna del periodo da el valor que se ha de introducir en el registro del mismo nombre. El registro AUDxLEN se debe poner a 8 (16 bytes = 8 palabras). El sample debe corresponder a un sólo ciclo de la onda.

Tabla 5-7: Octava bien ajustada para un Sample de 16 bytes.

Periodo en NTSC	Periodo en PAL	Nota		Frecuencia Ideal	Frecuencia real en NTSC	Frecuencia real en PAL
254	252	A	LA	880.0 Hz	880.8 Hz	879.7 Hz
240	238	A#	LA#	932.3 Hz	932.2 Hz	931.4 Hz
226	224	B	SI	987.8 Hz	989.9 Hz	989.6 Hz
214	212	C	DO	1046.5 Hz	1045.4 Hz	1045.7 Hz
202	200	C#	DO#	1108.7 Hz	1107.5 Hz	1108.4 Hz
190	189	D	RE	1174.7 Hz	1177.5 Hz	1172.9 Hz
180	178	D#	RE#	1244.5 Hz	1242.9 Hz	1245.4 Hz
170	168	E	MI	1318.5 Hz	1316.0 Hz	1319.5 Hz
160	159	F	FA	1396.9 Hz	1398.3 Hz	1394.2 Hz
151	150	F#	FA#	1480.0 Hz	1481.6 Hz	1477.9 Hz
143	141	G	SOL	1568.0 Hz	1564.5 Hz	1572.2 Hz
135	133	G#	SOL#	1661.2 Hz	1657.2 Hz	1666.8 Hz

La tabla anterior muestra los valores del periodo para usar con un sample de 16 bytes para generar tonos en la segunda octava con C sobre la mitad. Para generar los tonos de las octavas inferiores, hay dos metodos que se pueden usar, doblar el valor del periodo o doblar el tamaño del sample.

Cuando se dobla el periodo, el tiempo entre cada sample se dobla y asi el sample se hace el doble de largo. Esto significa que la frecuencia del tono generado se queda en la mitad, que da la siguiente la siguiente octava inferior. Por tanto, si C se reproduce con el periodo 214, entonces si se utiliza el mismo sample con el periodo 428, C sonaria en la siguiente octava inferior.

De la misma manera, cuando se dobla el tamaño del sample, se hace el doble de larga la reproducción del mismo y la frecuencia del tono generado estara en la siguiente octava inferior. Por tanto, si se tiene un sample de 8 bytes y otro de 16 bytes de la misma onda reproduciendose a la misma velocidad, el sample de 16 bytes sera una octava inferior al de 8 bytes.

Un sample para una escala bien ajustada representa tipicamente un ciclo completo de la onda. Para prevenir el "aliasing distortion" con estos samples se debe usar un periodo sólo en el rango 124-256. Los periodos de 124 a 256 corresponden a las frecuencias de reproducción en el rango 14000-28000 samples/segundo que hace el uso mas efectivo del filtro de agudos del Amiga. Para continuar en este rango se necesitan diferentes tamaños de samples por cada octava.

Si no se pueden usar diferentes samples por cada octava, entonces se tendra que ajustar el periodo sobre el rango completo 124-65536. Es mas facil de programar, pero puede producir ruido de alta frecuencia en el tono resultante. Leer la sección "Aliasing Distortion" para mas información.

Los valores de la Tabla 5-7 fueron generados usando esta fórmula.

$$\text{Frecuencia} = \frac{\text{Constante del Reloj}}{\text{Bytes del sample} * \text{Periodo}} = \frac{3579545}{16 * \text{Periodo}} = 880.8 \text{ Hz}$$

La constante del reloj en un sistema NTSC es de 3579545 ciclos/segundo. En un sistema PAL, la constante del reloj es de 3546895 ciclos/segundo. El número de bytes de la fórmula es el número de bytes de un solo ciclo de la onda. (La constante del reloj es el resultando de dividir el reloj del sistema entre 2. Este valor varia si se usa un reloj externo, como un genlock).

La siguiente tabla de samples representa una simple onda triangular.

Sample de 256 Bytes

```
  0   2   4   6   8  10  12  14  16  18  20  22  24  26  28  30
 32  34  36  38  40  42  44  46  48  50  52  54  56  58  60  62
 64  66  68  70  72  74  76  78  80  82  84  86  88  90  92  94
 96  98 100 102 104 106 108 110 112 114 116 118 120 122 124 126
128 126 124 122 120 118 116 114 112 110 108 106 104 102 100  98
 96  94  92  90  88  86  84  82  80  78  76  74  72  70  68  66
 64  62  60  58  56  54  52  50  48  46  44  42  40  38  36  34
 32  30  28  26  24  22  20  18  16  14  12  10   8   6   4   2
  0  -2  -4  -6  -8 -10 -12 -14 -16 -18 -20 -22 -24 -26 -28 -30
-32 -34 -36 -38 -40 -42 -44 -46 -48 -50 -52 -54 -56 -58 -60 -62
-64 -66 -68 -70 -72 -74 -76 -78 -80 -82 -84 -86 -88 -90 -92 -94
-96 -98 -100 -102 -104 -106 -108 -110 -112 -114 -116 -118 -120 -122 -124 -126
-128 -126 -124 -122 -120 -118 -116 -114 -112 -110 -108 -106 -104 -102 -100 -98
-96 -94 -92 -90 -88 -86 -84 -82 -80 -78 -76 -74 -72 -70 -68 -66
-64 -62 -60 -58 -56 -54 -52 -50 -48 -46 -44 -42 -40 -38 -36 -34
-32 -30 -28 -26 -24 -22 -20 -18 -16 -14 -12 -10  -8  -6  -4  -2
```

Sample de 128 Bytes

```
  0   4   8  12  16  20  24  28  32  36  40  44  48  52  56  60
 64  68  72  76  80  84  88  92  96 100 104 108 112 116 120 124
128 124 120 116 112 108 104 100  96  92  88  84  80  76  72  68
 64  60  56  52  48  44  40  36  32  28  24  20  16  12   8   4
  0  -4  -8 -12 -16 -20 -24 -28 -32 -36 -40 -44 -48 -52 -56 -60
-64 -68 -72 -76 -80 -84 -88 -92 -96 -100 -104 -108 -112 -116 -120 -124
-128 -124 -120 -116 -112 -108 -104 -100 -96 -92 -88 -84 -80 -76 -72 -68
-64 -60 -56 -52 -48 -44 -40 -36 -32 -28 -24 -20 -16 -12  -8  -4
```

Sample de 64 Bytes

```
  0   8  16  24  32  40  48  56  64  72  80  88  96 104 112 120
128 120 112 104  96  88  80  72  64  56  48  40  32  24  16   8
  0  -8 -16 -24 -32 -40 -48 -56 -64 -72 -80 -88 -96 -104 -112 -120
-128 -120 -112 -104 -96 -88 -80 -72 -64 -56 -48 -40 -32 -24 -16  -8
```

Sample de 32 Bytes

```
  0  16  32  48  64  80  96 112 128 112  96  80  64  48  32  16
  0 -16 -32 -48 -64 -80 -96 -112 -128 -112 -96 -80 -64 -48 -32 -16
```

Sample de 16 Bytes

```
  0  32  64  96 128  96  64  32   0 -32 -64 -96 -128 -96 -64 -32
```

VALORES DE DECIBELIOS PARA LOS RANGOS DEL VOLUMEN

Tabla 5-8: Valores de decibelios y Rango del Volumen

Volumen	Decibelos	Volumen	Decibelios
64	0.0	32	-6.0
63	-0.1	31	-6.3
62	-0.3	30	-6.6
61	-0.4	29	-6.9
60	-0.6	28	-7.2
59	-0.7	27	-7.5
58	-0.9	26	-7.8
57	-1.0	25	-8.2
56	-1.2	24	-8.5
55	-1.3	23	-8.9
54	-1.5	22	-9.3
53	-1.6	21	-9.7
52	-1.8	20	-10.1
51	-2.0	19	-10.5
50	-2.1	18	-11.0
49	-2.3	17	-11.5
48	-2.5	16	-12.0
47	-2.7	15	-12.6
46	-2.9	14	-13.2
45	-3.1	13	-13.8
44	-3.3	12	-14.5
43	-3.5	11	-15.3
42	-3.7	10	-16.1
41	-3.9	9	-17.0
40	-4.1	8	-18.1
39	-4.3	7	-19.2
38	-4.5	6	-20.6
37	-4.8	5	-22.1
36	-5.0	4	-24.1
35	-5.2	3	-26.6
34	-5.5	2	-30.1
33	-5.8	1	-31.1
		0	-infinito

LA MAQUINA DEL ESTADO DEL AUDIO

Para la explicación de los varios estados, se hace referencia a la Figura 5-8. Hay una maquina del estado del audio por cada canal. La maquina tiene ocho estados y funciona con la constante del reloj (3.58 MHz en NTSC). Tres de los estados no se usan y sólo transfieren al estado de espera (000). Uno de los caminos hacia fuera del estado de espera esta diseñado para la operación de manejo por interrupciones (el 68000 proporciona los datos), y el otro camino esta diseñado para la operación de manejo por DMA (el chip "Agnus" proporciona los datos).

En la operación de manejo por interrupciones, la transferencia al bucle principal (estados 010 y 011) ocurre inmediatamente despues de que el 68000 haya escrito los datos. En el estado 010 el byte superior va al convertidor digital-a-analógico, y en el estado 011 es el byte inferior. Las transiciones como 010 -> 011 -> 010 ocurren cuando el contador del periodo cuenta hacia atras hasta 1. El contador del periodo se re-carga en estas transiciones. Durante el tiempo que el 68000 este respondiendo a las interrupciones, la maquina continua en el bucle principal. Si no, entra en el estado de espera. Las interrupciones se generan en la transición 011 -> 010 para obtener una nueva palabra.

En la operación de manejo por DMA, ocurre la transición al estado 001 cuando se conecta el DMA del Audio, al ocurrir esta transición se envia una petición de DMA a Agnus. Debido a la forma en que Agnus maneja el DMA, la primera palabra se debe ignorar. Se entra al estado 101 cuando llega esta palabra; se ha enviado ya una petición de la siguiente palabra. Cuando llegan los datos, se entra al estado 010 y el bucle principal continua hasta que se desconecta el DMA. La longitud del contador decrece una vez por cada palabra que llega. Cuando finaliza, se envia una petición a Agnus de restauración junto con la petición normal de mas datos. Esto indica a Agnus que tiene que restaurar los punteros al comienzo de la tabla de datos. Tambien se recarga el contador de longitud y se envia una petición de interrupción. La petición se produce justo cuando la última palabra de la onda comienza a salir del convertidor digital-a-analógico.

Las peticiones de DMA y de restauración se transfieren a Agnus una vez por cada linea horizontal, y los datos llegan 14 ciclos del reloj despues (un ciclo = 280 ns; 14 ciclos = 3920 ns).

Cuando se conectan los canales para producir modulación, las cosas ocurren con algunas diferencias. En la modulación de amplitud (volumen), las peticiones ocurren como en la operación normal (transición 011 -> 010). En la modulación de frecuencia (periodo) las peticiones se producen en la transición 010 -> 011. Cuando se modulan frecuencia y amplitud, las peticiones se producen en ambas transiciones.

Si la velocidad de sampleado se pone mucho mas alta que la velocidad normalmente maxima (29 KHz), los dos samples del registro de buffer se repiten. Si se desconecta el filtro de agudos y se pone el volumen al maximo ($40), se puede utilizar esto para producir portadoras moduladas hasta 1.79 Mhz (ondas de radio en AM). La modulación se coloca en la memoria, con mas valores en los bytes pares y menos valores en los bytes impares.

Los simbolos usados en el diagrama de estado se explican en la siguiente lista. Los nombres en mayúsculas indican señales externas; los nombres en minúsculas señales locales.

AUDxON	Conecta DMA, "x" indica el número de canal (señal de DMACON)
AUDxIP	Interrupción del audio pendiente (entrada al canal desde la circuiteria de interrupciones.
AUDxIR	Interrupción de solicitud del audio (salida del canal a la circuiteria de interrupciones)
intreq1	Solicitud de interrupción que se combina con intreq2 para formar AUDxIR
intreq2	Preparado para solicitud de interrupción. La solicitud sale despues de la transición 011 -> 010 en la operación normal.
AUDxDAT	Señal de carga de datos. Carga 16 bits en el canal de audio.
AUDxDR	El audio solicita a Agnus una palabra de datos del DMA.
AUDxDSR	El audio solicita a Agnus que restaure los punteros.
dmasen	Conecta la solicitud de restauración.
percntrld	Re-carga el contador de periodo desde el latch que utiliza el procesador con AUDxPER (tambien podria ser desde la modulación)
percount	Cuenta hacia atras una vez en el contador del periodo.
perfin	Contador de periodo finalizado (valor=1)

lencntrld	Re-carga el contador de longitud desde el latch de back-up.
lencount	El contador de longitud cuenta hacia atras una vez.
lenfin	Contador de longitud finalizado (valor=1)
volcntrld	Re-carga el contador del volumen del latch de back-up.
pbufld1	Carga el buffer de salida congelando el latch de AUDxDAT
pbufld2	Como pbufld1, pero sólo durante la transición 010 -> 011 con modulación en frecuencia.
AUDxAV	Modulación en amplitud. Envia datos al latch del volumen del siguiente canal en lugar de al convertidor D->A.
AUDxAP	Modulación en frecuencia. Envia datos al latch del periodo del siguiente canal en lugar de al convertidor D->A.
penhi	Los 8 bits superiores de datos van al convertidor D->A.
napnav	/AUDxAV * /AUDxAP + AUDxAV -no hay modulación, o si no, modulación en amplitud. Condición para el DMA normal o las interrupciones.
sq2,1,0	El nombre de los flip-flops de estado, MSB a LSB.

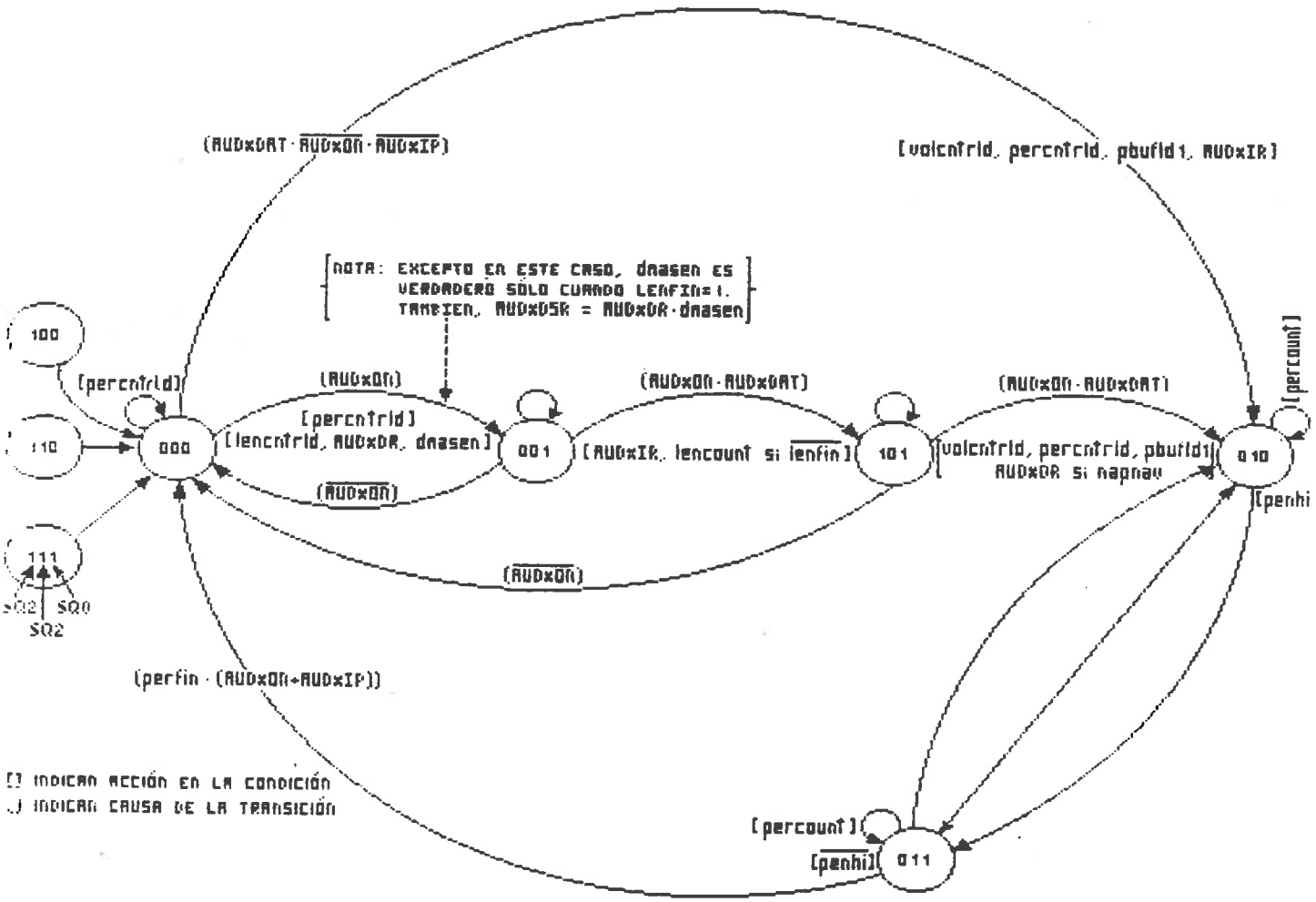


Figura 5-8: Diagrama del Estado del Audio

CAPITULO 6

EL HARDWARE DEL BLITTER

INTRODUCCION

El Blitter es uno de los dos coprocesadores del Amiga. Parte del chip "Agnus", se usa para copiar bloques rectangulares de memoria y para dibujar lineas. Cuando se copia memoria, es aproximadamente dos veces mas rapido que el 68000, capaz de mover hasta cuatro megabytes por segundo. Puede dibujar lineas hasta un millón de pixels por segundo.

En modo de movimiento de bloques, el Blitter puede ejecutar cualquier operación lógica desde tres areas fuentes, puede desplazar de uno a quince bits de hasta dos de las fuentes, puede rellenar figuras perfiladas, puede enmascarar la primera y la última palabra de cada linea. En modo de lineas, se puede imponer cualquier patrón a la linea (a puntos, a rayas, etc.), o tambien se puede dibujar la linea de modo que sólo se ponga un pixel por linea horizontal.

El Blitter sólo puede acceder a memoria CHIP -la porción de memoria accesible por el hardware de visualización. Intentar usar el Blitter para leer o escribir memoria FAST o cualquier otro tipo no-CHIP puede terminar en la destrucción de los contenidos de la memoria CHIP.

Un "blit" es una sola operación del Blitter -quizas el dibujo de una linea o el movimiento de un bloque de memoria. Un blit se ejecuta inicializando los registros con los valores apropiados y poniendo en marcha el Blitter al escribir el registro BLTSIZE. Cómo el Blitter es un coprocesador asincrono, el 68000 continua en marcha mientras se ejecuta el blit.

MANEJO DE LA MEMORIA

El Blitter es un word-blitter (Blitter de palabras), no un bit-blitter. Todos los datos obtenidos, modificados, y escritos lo son en palabras de 16 bits. Mediante una programación cuidadosa, el Blitter puede hacer operaciones de tipo bit.

El Blitter esta muy indicado para operaciones graficas. Como ejemplo, una pantalla de 320 por 200 con 16 colores esta organizada como cuatro bit-planes de 8000 bytes cada uno. Cada bit-plane consiste en 200 filas de 40 bytes, o 20 palabras. (El concepto palabra se suele utilizar como palabra de 16 bits)

CANALES DE DMA

El Blitter tiene cuatro canales DMA -tres canales fuente, llamados A, B y C, y un canal destino, llamado D. Cada uno de estos canales tiene un puntero separado de direcciones, registros de módulo y datos y un bit de conexión. Dos tienen registros de desplazamiento, y uno tiene un registro de mascara para la primera y última palabra. Los cuatro comparten un sólo registro de tamaño para el "blit".

Los registros de puntero de direcciones se componen cada uno de dos palabras, llamadas BLTxPTH y BLTxPTL. (Aqui y mas tarde, al referirnos a un registro, la "x" sera el nombre del canal, A, B, C o D). Las dos palabras de cada registro son adyacentes en el espacio de direccionamiento del 68000, con la dirección de la palabra superior primero, por eso se pueden escribir de una vez como palabra larga (32 bits). Los registros de puntero se pueden escribir con cualquier dirección en bytes, pero el bit inferior se ignora porque el Blitter sólo trabaja con palabras. En los ordenadores de 512 KB de memoria CHIP se ignoran los 13 bits mas significativos. En futuros ordenadores habra mas memoria CHIP y se ignorara menos bits.

Cualquier dirección valida, par y en el espacio de memoria CHIP se puede escribir siempre en estos registros.

NOTA: Se debe escribir sólo ceros en los bits sin usar de los registros. Estos bits podrían usarse en versiones posteriores de los custom chips. Si se escribe valores distintos a cero en estos bits podría causarse resultados impredicibles en futuros ordenadores.

Cada uno de los canales DMA se puede conectar o desconectar independientemente. Los bits de conexión son los bits SRCA, SRCB, SRCC, y DEST del registro de control cero (BLTCON0).

Cuando se desconecta un canal, no se ejecutaran ciclos de memoria para ese canal, en un canal fuente quedara un valor constante en el registro de datos de ese canal que podra ser usado por el Blitter en todos los ciclos. Para este propósito, cada una de las tres areas fuente tiene un registro de datos precargable, llamado BLTxDAT.

Las imagenes en memoria se guardan en memoria en una forma lineal; cada palabra de datos de una linea esta colocada en la dirección que es mayor en uno a la palabra de su izquierda. Cada linea es una continuación "mas uno" de la linea anterior.

20	21	22	23	24	25	26
27	28	29	30	31	32	33
34	35	36	37	38	39	40
41	42	43	44	45	46	47
48	49	50	51	52	53	54
55	56	57	58	59	60	61

Figura 6-1: Cómo se Almacenan las Imagenes en Memoria

El mapa de la Figura 6-1 representa un bit-plane de una imagen en la dirección de palabra 20 hasta la 61. Cada una de estas direcciones corresponde a una palabra (16 pixels) de un bit-plane. Si la imagen fuera de 16 colores, se necesitarian cuatro bit-planes como este en la mememoria, y cuatro operaciones de copia (movimiento) se necesitarian para mover completamente la imagen.

El Blitter es muy eficiente copiando estos bloques porque sólo se necesita indicar la dirección inicial (20), la dirección de destino, y el tamaño del bloque (alto=6, ancho=7). Entonces movera los datos automaticamente, una palabra cada vez, cuando el bus de datos este disponible. Cuando la transferencia se complete, el Blitter avisara al 68000 con un bit y una interrupción.

Nota: Esta operación de copia (movimiento) se realiza en la memoria y puede o puede no cambiar la porción usada para la imagen.

Todos los "blits" de copia de datos se ejecutan como rectangulos de palabras, con un ancho y alto dados. Los cuatro canales DMA usan un solo registro de tamaño, llamado BLTSIZE, en el que se encuentran el ancho y el alto. El ancho puede tomar un valor de 1 a 64 palabras (16 a 1024 bits). El alto puede ir de 1 a 1024 filas. El ancho se guarda en los 6 bits inferiores del registro BLTSIZE. Si se coloca un cero, se entendera como un ancho de 64 palabras. Este es el único parametro del Blitter que se da en palabras. El alto se guarda en los 10 bits superiores del registro BLTSIZE, un cero representa un alto de 1024 filas. Por lo tanto, el blit mas grande posible en el Blitter actual del Amiga es de 1024 por 1024 pixels. Aunque, en las operaciones de desplazamiento de bits y enmascaramiento se necesita una palabra extra por linea reduciendo el ancho maximo a 1008 pixels.

El Blitter tambien tiene unas facilidades, llamadas módulos, para acceder a imagenes mas pequeñas que el bit-plane completo. Cada uno de los cuatro canales DMA tiene un registro módulo de 16 bits llamado BLTxMOD. Cada vez que se obtiene una palabra (o se escribe) para un canal, el puntero de direcciones se incrementa en dos bytes (una palabra). Despues de que cada linea del blit se complete, el valor de 16 bits de cada canal se suma (con signo) a cada puntero de direcciones. (La linea esta definida por el ancho del registro BLTSIZE).

Nota: Los valores del módulo estan en bytes, no en palabras. Pero como el Blitter sólo puede operar con palabras, el bit menos significativo se ignora. El signo del módulo se extiende al ancho de los registros de direcciones. Los módulos negativos pueden ser utiles de muy diversas formas, como repetir una linea poniendo el módulo al mismo ancho de la linea, en negativo.

Como ejemplo, se supone que se quiere operar en una sección de un bit-plane de 320 por 200 que comienza en la linea 13, byte 12 (estan tomados desde 0) y la sección es de 10 bytes de ancha. Se deberia inicializar el registro de puntero con la dirección del bit-plane mas 40 * 13 (número de lineas), mas 12 (bytes) para obtener la posición correcta en el bit-plane. Se deberia poner el ancho a 5 palabras (10 bytes). Al final de cada linea, tendrian que saltarse 30 bytes para obtener el comienzo de la siguiente linea, por eso se usaria un módulo de 30. En general, el ancho (en palabras) por dos mas el módulo (en bytes) debe ser igual al ancho completo, en bytes, del bit-plane que contiene la imagen.

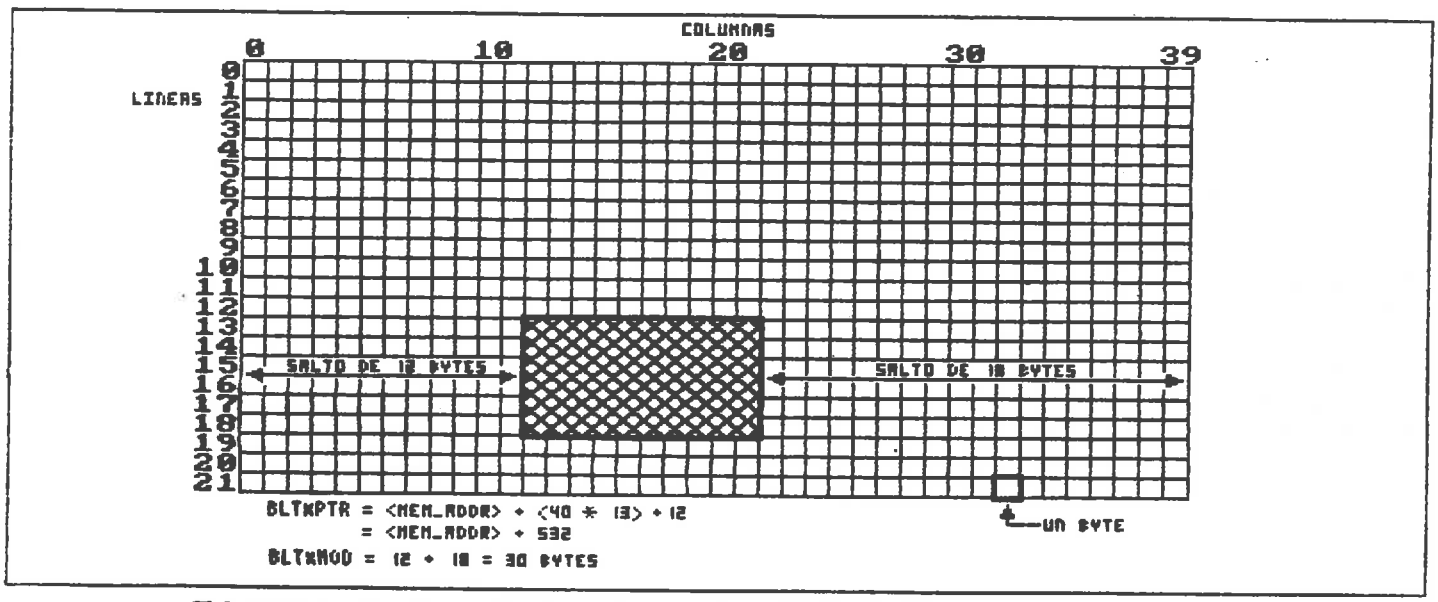


Figura 6-2: Calculos de BLTxPTR y BLTxMOD

Nota: El Blitter se puede usar para procesar bloques lineales en lugar de rectangulares poniendo el ancho o el alto de BLTSIZE a 1.

Debido a que cada canal DMA tiene su propio registro de módulo, se pueden mover datos entre bit-planes de diferentes tamaños. Esto es muy útil cuando se mueven imagenes pequeñas en grandes bit-planes.

GENERADOR DE FUNCIONES

El Blitter puede combinar los datos procedentes los tres canales DMA fuente en una de 256 maneras diferentes para generar los valores que se almacenan en el canal DMA destino. Las fuentes podrían ser un bit-plane con cada una de las imagenes por separado. Como cada una de estas fuentes es una región rectangular compuesta por muchos puntos, se ejecuta la misma operación lógica en cada uno de los puntos. Por lo tanto, cuando se define la operación lógica del Blitter sólo es necesario considerar lo que sucedera para todas las combinaciones posibles de un solo bit de cada una de las fuentes.

Hay ocho combinaciones posibles de valores de los tres bits, para cada uno de los cuales se necesita especificar el bit correspondiente de destino como cero o uno. Esto se puede mostrar en una tabla de verdad, como la siguiente. Estan listados los tres canales fuente, y sus valores posibles de un sólo bit para cada uno de ellos.

A	B	C	D	Posición BLTCON0	Miniterm
0	0	0	?	0	$\overline{A}\overline{B}\overline{C}$
0	0	1	?	1	$\overline{A}\overline{B}C$
0	1	0	?	2	$\overline{A}B\overline{C}$
0	1	1	?	3	$\overline{A}BC$
1	0	0	?	4	$A\overline{B}\overline{C}$
1	0	1	?	5	$A\overline{B}C$
1	1	0	?	6	$AB\overline{C}$
1	1	1	?	7	ABC

Esta información se reune de una forma standard, el byte de control LF del registro BLTCON0. Este byte programa al Blitter para ejecutar una de las 256 operaciones lógicas posibles sobre las tres fuentes en un "blit".

Para calcular el byte de control LF, se ha de rellenar la tabla de verdad con los valores que se desean para D, y leer el valor de la función desde la parte superior de la tabla.

Por ejemplo, si se quisiera que el bit D fuera 1 cuando el bit correspondiente de A sea 1 o el bit correspondiente de B sea 1, se deberían rellenar las cuatro últimas posiciones de la tabla con 1, la tercera, cuarta, septima y octava posición a 1 (bits 4 a 7), y los dos primeros (primera y segunda) a 0, porque ni el A ni el B son 0. Después, se leería la tabla de verdad desde la parte superior: 11111100, o $FC.

Otro ejemplo, un byte de control LF de $80 (10000000) daría como resultado 1 en los puntos del rectangulo que formaría D cuando las tres fuentes A, B y C fueran todas 1 (ABC = 1, bit 7 de LF = 1). Todos los demás puntos del rectangulo, que corresponderían a las otras combinaciones de A, B y C, serían 0. Esto es porque los bits de LF del 0 al 6, que especifican la salida de D para estas situaciones, serían 0.

CREANDO EL BYTE DE CONTROL LF CON MINITERMS

Una de las formas de crear el byte de control LF es con ecuaciones lógicas. Cada una de las filas de la tabla de verdad corresponde a una "miniterm", que es una asignación particular a los bits de A, B y C. Por ejemplo, la primera miniterm es ABC, o "no A y no B y no C". La última es ABC.

Nota: Dos terminos que son adyacentes corresponden a la operación "y" (and), dos terminos que estan separados por "+" corresponden a la operación "o" (or). "y" tiene mayor prioridad, por eso AB + BC es lo mismo que (AB) + (BC)

Cualquier función se puede escribir como la suma de miniterms. Si se quisiera calcular la función dónde D es 1 cuando el bit A es 1 y el bit C es 0, o cuando el bit B es 1, se podría dejar como AC+B, o "A y no C o B". Porque "1 y A" es "A":

$$D = A\bar{C} + B$$

$$D = A(1)\bar{C} + (1)B(1)$$

Como A o $\bar{A}$ es 1 ($1 = A + \bar{A}$), y de la misma forma para B, y C; se puede extender la ecuación anterior mas aún:

$$D = A(1)\bar{C} + (1)B(1)$$

$$D = A(B+\bar{B})\bar{C} + (A+\bar{A})B(C+\bar{C})$$

$$D = AB\bar{C} + A\bar{B}\bar{C} + AB(C+\bar{C}) + \bar{A}B(C+\bar{C})$$

$$D = AB\bar{C} + A\bar{B}\bar{C} + ABC + AB\bar{C} + \bar{A}BC + \bar{A}\bar{B}\bar{C}$$

Despues simplificando, quedaría con cinco miniterms:

$$A\bar{C}+B = AB\bar{C} + A\bar{B}\bar{C} + ABC + \bar{A}BC + \bar{A}\bar{B}\bar{C}$$

Esto corresponde a las posiciones de BLTCON0 de los bits 6, 4, 7, 3, y 2, de acuerdo con la tabla de verdad, estos bits se pondrían a uno y los demas a cero.

El amplio rango de operaciones lógicas permite las mas sofisticadas tecnicas graficas. Por ejemplo, se puede mover la imagen de un coche a traves de las imagenes de unos edificios que ya se encuentran en el decorado con unos pocos "blits". Para producir este efecto se necesitaría dibujar las imagenes del coche, los edificios (decorado), y la mascara del coche que contiene los bits a uno donde el coche no es transparente. Esta mascara se podría visualizar como la sombra del coche desde una fuente de luz en la misma posición que el observador.

Nota: La mascara del coche sólo necesitaría ser un bit-plane independientemente de la profundidad (número de bit-planes) del decorado. Esta mascara se usara para cada uno de los bit-planes del fondo.

Para mover el coche, primero se guardara la imagen del fondo dónde se pondra el coche. Despues se copiara el coche en esta posición con otro "blit". La imagen ya esta lista para visualizarse. Para crear la siguiente imagen, se restaurara la anterior imagen del fondo, se guardara la siguiente porción del fondo donde se volvera a colocar el coche, y se volvera a dibujar el coche, usando tres blits separados. (Esta tecnica trabaja mejor con "blits" sincronizados con el rayo o doble bufferización).

Para guardar el fondo temporalmente, se copia un rectangulo del decorado (del canal A, por ejemplo) a un buffer de almacenamiento (usando el canal D). En este caso, la función que se usaría sería "D = A", la función standard de copia. En la Tabla 6-1 se puede ver que esta función corresponde al código $F0.

Para dibujar el coche, se usara el canal A para la mascara, el canal B para el coche, el canal D para el decorado y el canal D para escribir la nueva imagen.

Nota: Se debe incluir el décorado en la operación antes de escribir sobre el, porque sólo se necesita modificar parte de las palabras del mismo, y no hay forma de escribir palabras a medias.

Cuando se esta escribiendo el coche sobre el decorado se usaria una función, que cuando los bits de la mascara (canal A) fueran 1, se dejarian pasar los bits del coche del canal B y cuando los bits de A fueran 0, se dejarian pasar los bits del decorado original del canal C. La función correspondiente, comunmente llamada función de "cookie-cut", es AB+AC, que corresponde al código $CA para LF.

Para restaurar el decorado y preparar la siguiente imagen, se copiaria la información guardada en el primer paso, con la función standard de copia.

Si se desplazan los datos y la mascara a la nueva posición y se repiten los tres pasos anteriores continuamente, el coche aparentara moverse sobre el decorado (los edificios).

Nota: Este no es el metodo mas efectivo de animación, dependiendo de la aplicación, pero la función "cookie-cut" se utiliza comunmente.

Tabla 6-1: Tabla de los Valores para Miniterms mas Frecuentes

Ecuación	Código LF	Ecuación	Código LF
$D = A$	$F0	$D = AB$	$C0
$D = \bar{A}$	$0F	$D = A\bar{B}$	$30
$D = B$	$CC	$D = \bar{A}B$	$0C
$D = \bar{B}$	$33	$D = \bar{A}\bar{B}$	$03
$D = C$	$AA	$D = BC$	$88
$D = \bar{C}$	$55	$D = B\bar{C}$	$44
$D = AC$	$A0	$D = \bar{B}C$	$22
$D = A\bar{C}$	$50	$D = \bar{A}\bar{C}$	$11
$D = \bar{A}C$	$0A	$D = A + \bar{B}$	$F3
$D = \bar{A}\bar{C}$	$05	$D = \bar{A} + \bar{B}$	$3F
$D = A + B$	$FC	$D = A + \bar{C}$	$F5
$D = \bar{A} + B$	$CF	$D = \bar{A} + \bar{C}$	$5F
$D = A + C$	$FA	$D = B + \bar{C}$	$DD
$D = \bar{A} + C$	$AF	$D = \bar{B} + \bar{C}$	$77
$D = B + C$	$EE	$D = AB + \bar{A}C$	$CA
$D = \bar{B} + C$	$BB		

CREANDO EL BYTE DE CONTROL LF CON DIAGRAMAS DE VENN

Otra forma de llegar a una función particular es a traves del uso de diagramas de Venn.

1. Para seleccionar la función D=A (que es, Destino = fuente A), se seleccionan únicamente los miniterms que estan totalmente englobados en el circulo A de la Figura 6-3. Este conjunto de miniterms 7, 6, 5 y 4. Se escriben como un conjunto de unos para las miniterms seleccionadas y ceros para las no seleccionadas.

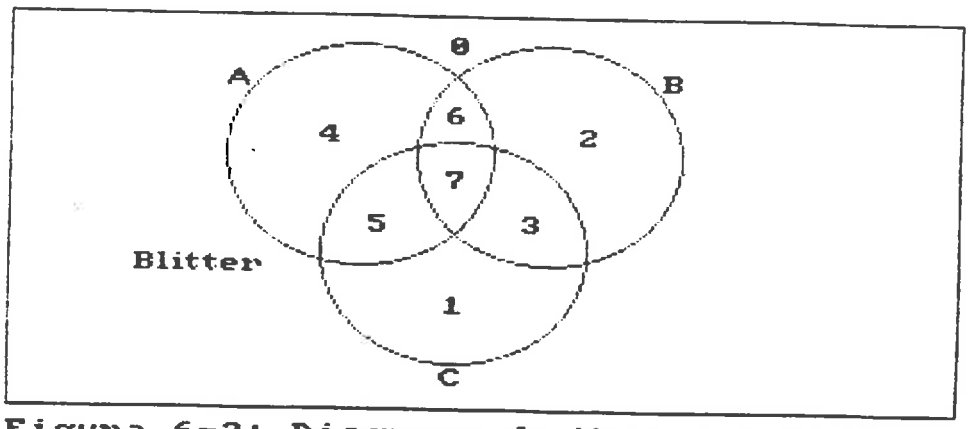


Figura 6-3: Diagrama de Venn del Blitter

```
Número de miniterm          7 6 5 4 3 2 1 0
Miniterms seleccionadas     1 1 1 1 0 0 0 0   --> $F0
```

2. Para seleccionar una función que es la combinación de dos fuentes, se cogen las miniterms de ambos circulos (su intersección). Por ejemplo, la combinación AB (A y B) se representa como el area común de los circulos A y B, o miniterms 7 y 6.

```
Número de miniterm          7 6 5 4 3 2 1 0
Miniterms seleccionadas     1 1 0 0 0 0 0 0   --> $C0
```

3. Para usar una función que es la inversa (no), de una de las fuentes como A, se toman todas las miniterms que no estan incluidas en el circulo representado por A de la Figura anterior. En este caso, se tomarian las miniterms 0, 1, 2 y 3.

```
Número de miniterm          7 6 5 4 3 2 1 0
Miniterms seleccionadas     0 0 0 0 1 1 1 1   --> $0F
```

4. Para sumar miniterms, han de sumarse sus valores según la operación lógica "o". Por ejemplo, la ecuación AB + BC sería.

```
Número de miniterm          7 6 5 4 3 2 1 0
AB                          1 1 0 0 0 0 0 0
BC                          1 0 0 0 1 0 0 0
--------------------------------------------
AB + BC                     1 1 0 0 1 0 0 0   --> $F0
```

DESPLAZAMIENTOS Y MASCARAS

Hasta ahora sólo se ha utilizado el Blitter para mover palabras de la memoria de un sitio a otro y combinarlas con operaciones lógicas. Esto es suficiente para mover imagenes graficas, mientras que las imagenes permanezcan en la misma posición relativa al inicio de la palabra. Si la imagen del coche tuviera su primer pixel de la izquierda en el segundo bit desde el comienzo izquierdo de la palabra, se podria dibujar facilmente en cualquier posición de la pantalla dónde este pixel comenzara a dos pixels desde el comienzo de la palabra. Pero frecuentemente se necesita dibujar el coche desplazado a izquierda o derecha unos pocos pixels. Con este fin, los canales A y B tienen un registro desplazador que puede desplazar la imagen entre 0 y 15 bits.

Esta operación de desplazamiento es completamente libre; no se necesita mas tiempo para un "blit" que realiza desplazamiento que para uno que no lo

realiza, al contrario que los desplazamientos con el 68000. El desplazamiento es normalmente hacia la derecha. Este desplazador permite el movimiento de las imagenes en los limites del pixel, aunque se accede a 16 pixels a la vez por cada dirección.

Por tanto si los datos que llegan se desplazan a la derecha, ¿que es lo que se esta colocando desde la izquierda para desplazar a dichos datos?. En la primera palabra del "blit", se introducen ceros en el desplazamiento; para cada palabra siguiente del mismo "blit", se introducen los datos que habian quedado fuera en la palabra anterior.

El valor de desplazamiento para el canal A se introduce en los bits del 15 al 12 del registro BLTCON0; el desplazamiento para B se introduce en los bits del 15 al 12 de BLTCON1. En la mayoria de las operaciones se usa el mismo desplazamiento en ambos canales. Para desplazamientos mayores de 15 bits, se carga el registro de puntero del canal D con una dirección mayor; un desplazamiento de 100 bits requeriria que el canal D se avanzara en 6 palabras (100/16), y se desplazara 4 bits a la derecha (100 = 6 * 16 + 4).

Como ejemplo, imaginese que se esta ejecutando un "blit" de dos palabras de ancho y tres de alto, y se esta usando un desplazamiento de 4 bits. Para una mayor simplicidad, se esta ejecutando una copia estricta de A a D. La primera palabra que se escribiria en D seria la primera palabra obtenida de A, desplazada 4 bits a la derecha con ceros en los primeros 4 bits de la izquierda. La segunda palabra seria la segunda palabra obtenida de A, desplazada a la derecha, con los 4 bits inferiores de la palabra anterior introducidos en los primeros 4 bits. A continuación, se escribiria la primera palabra de la segunda fila obtenida de A, desplazada cuatro bits, con los cuatro bits menos significativos de la palabra anterior en la izquierda. Esto continuaria hasta que terminase el "blit".

En los "blits" con desplazamiento, por lo tanto, se introduciran ceros para la primera palabra de la primera fila. En todas las demas filas el Blitter introducira los bits que quedaron fuera en la palabra anterior. En la mayoria de las aplicaciones graficas, esto no es lo adecuado. Por esta razón, el Blitter tiene la posibilidad de enmascarar la primera y la última palabra de cada linea que llega de el canal A. Por tanto, es posible extraer unos datos rectangulares de una fuente cuyos limites derecho e izquierdo estan en los limites de la palabra. Estos dos registros se llaman BLTAFWN y BLTALWN, que son las mascaras del canal A para la primera y última palabra de cada linea. Cuando no se usan, se deben dejar a $FF para que asi dejen pasar los datos del canal fuente tal como son.

Nota: Los Fonts del Amiga estan grabados en forma de bit-map comprimido. Los caracteres individuales se extraen usando el Blitter, enmascarando asi los bits no deseados. El caracter se puede colocar entonces en cualquier posición de la palabra.

Estas mascaras se aplican al canal fuente mediante la operación "y" (and), antes de que se apliquen los desplazamientos. Sólo cuando hay un bit 1 en la primera palabra de mascara se dejara pasar al bit correspondiente del canal fuente. La primera palabra de cada linea se enmascara con BLTAFWN, y la última palabra con BLTALWN. Si el ancho de la linea es de una sola palabra, se aplicaran ambas mascaras simultaneamente.

Las mascaras tambien son útiles para extraer un determinado número de "columnas" de un bit-plane. Por ejemplo, se tiene un rectangulo en la pantalla que contiene texto y graficos de 23 pixels de ancho. El primer pixel del rectangulo es el primer pixel de su bit-plane, y el bit-plane es de ancho dos palabras. Se desea llevar este rectangulo a un bit-plane de 320 por 200 comenzando en la posición 5, sin estorbar a lo que quede fuera del rectangulo.

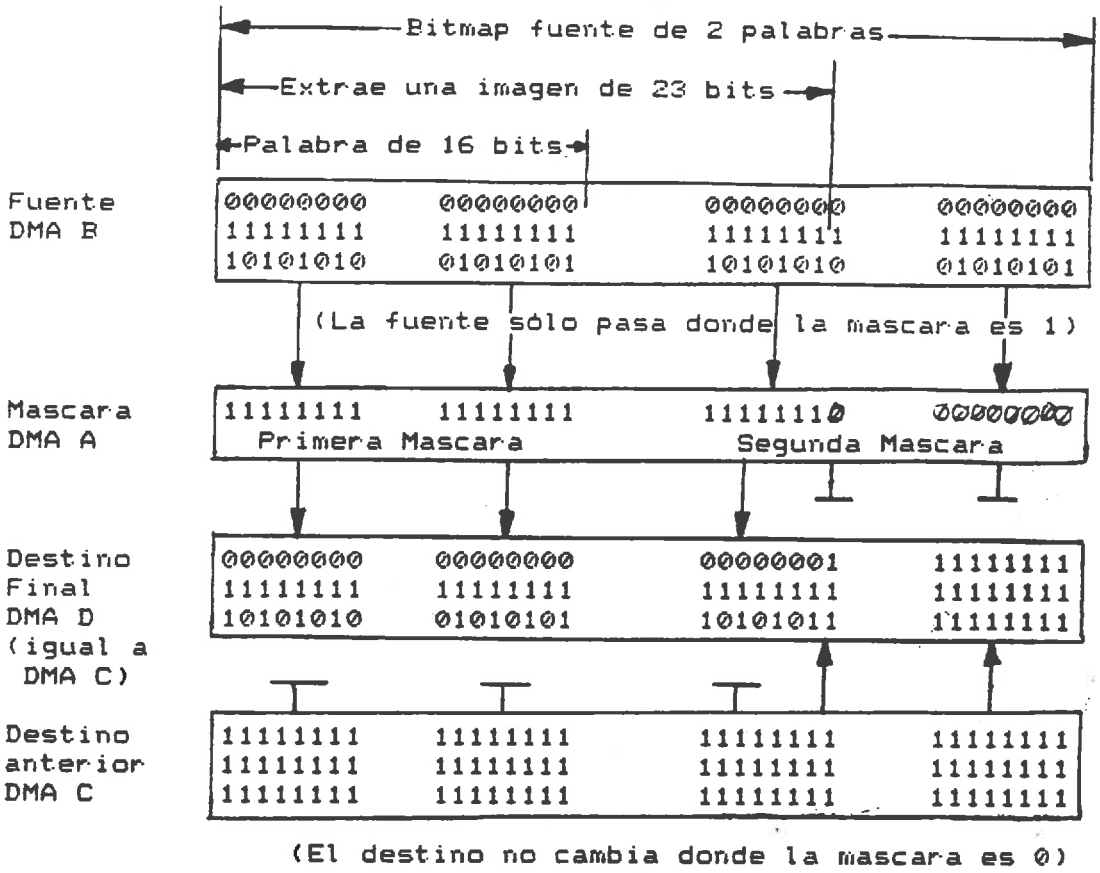


Figura 6-4: EXtracción de un Rango de Columnas

Para hacer esto, se apunta el canal DMA B al bitmap que contiene la imagen fuente, y el canal DMA D al bitmap de la pantalla. Se usa un desplazamiento de 5. Tambien se apunta el canal DMA C al bitmap de la pantalla. Se usa un "blit" de dos palabras de ancho. Lo que se necesita es una simple operación de copia, excepto que no se desea alterar los cinco primeros bits de la primera palabra, y los cuatro últimos bits (2 veces 16, menos 23, menos 5) de la última palabra. El Canal DMA A sirve de ayuda. Se carga el registro de datos A con $FFFF (todo unos), y se usa una mascara de la primera palabra con los cinco bits mas significativos a cero ($07FF) y la mascara de la última palabra con los cuatro bits menos significativos a cero ($FFF0). Pero no se conecta el canal DMA A, sólo los canales B, C y D, porque se quiere usar el canal como una mascara simple de lineas. Tambien se quiere que los datos de B (fuente) sólo pasen donde el canal A sea 1 (miniterm AB) y que pase el contenido original del canal destino (para el canal C) donde A sea 0 (miniterm AC), formando la clasica función de "cookie-cut" con AB+AC, que corresponde al código $CA.

Nota: Aunque el canal A esta desconectado, se usa en la función lógica y se carga su registro de datos. Cuando se desconecta un canal simplemente se desconecta la obtención de datos para ese canal; todas las demas operaciones se siguen ejecutando, sólo con un valor constante guardado el registro de datos del canal.

Una alternativa pero que es una forma mas aguda de conseguir el mismo efecto es usar un desplazamiento para A de 5, una mascara de primera palabra de todo unos, y una mascara de última palabra con los nueve bits menos significativos a cero. Todos los demas registros continuan igual.

Nota: Hay que asegurarse de cargar los registros de datos inmediatos del Blitter sólo despues de poner los desplazamientos en BLTCON0/BLTCON1, si se cargan primero los registros de datos dara unos resultados impredictibles. Por ejemplo, si se deja el desplazamiento de B a 4, despues se carga el registro de datos de B con 1 y se cambia el desplazamiento de B a 2, el resultado seria "1<<4", no "1<<2". El acto de cargar uno de los registros de datos "dibuja" los datos a la maquina y los desplaza.

MODO DESCENDIENTE

El "blit" standard de copia de memoria trabaja correctamente si la fuente no esta superpuesta con el destino. Si se desea mover una imagen una linea hacia abajo (incrementando su dirección), se estara en un problema -!la segunda linea seria sobreescrita antes de que se hubiera podido copiarla! El Blitter tiene un modo especial de funcionamiento -modo descendiente- que resuelve este problema satisfactoriamente.

El modo descendiente se conecta poniendo a uno el bit 1 de BLTCON1 (llamado BLITREVERSE). Si se usa el modo descendiente los punteros de direcciones se decrementaran en dos (bytes) en lugar de incrementarse en dos por cada palabra obtenida. Ademas, los modulos se restaran en lugar de sumarse. Los desplazamientos se produciran hacia la izquierda, en lugar de hacia la derecha, la mascara de primera palabra enmascarara la última palabra de la linea (que sera la primera palabra obtenida), y la mascara de la última palabra enmascarara la primera palabra de la linea.

Por lo tanto, para una copia standard de memoria, la única diferencia para la puesta a punto del Blitter (suponiendo que no hay desplazamientos ni mascaras) es inicializar los registros punteros de direcciones para apuntar a la última palabra del bloque, en lugar de la primera palabra. Los valores del módulo, el tamaño del "blit", y los demas parametros seran los mismos.

Nota: Este postdecremento difiere del predecremento del 68000, dónde un registro de direcciones se inicializa para apuntar a la palabra despues de la última, en lugar de la última palabra.

El modo descendiente es necesario tambien para el rellenado de areas, que se explicara en una sección posterior.

COPIANDO REGIONES ARBITRARIAS

Uno de los usos mas comunes del Blitter es mover rectangulos arbitrarios de datos desde un bit-plane a otro, o a diferentes posiciones dentro de un bit-plane. Estos rectangulos estan normalmente en coordenadas de bit arbitrarias, por lo tanto son necesarios desplazamientos y mascaras. Hay mas complicaciones. Se puede necesitar leer muchas veces esta sección y experimentar mucho antes de que se entienda todo lo que se explica en esta sección.

Una imagen fuente que ocupe sólo dos palabras, cuando se copia con unos determinados desplazamientos, puede ocupar tres palabras. El rectangulo de 23 pixels de ancho anterior, por ejemplo, cuando se desplaza 12 bits, ocupara tres palabras. Alternativamente, una imagen que ocupa tres palabras puede caber en dos con unos determinados desplazamientos. Bajo todas estas circunstancias, el tamaño del "blit" debera ponerse al mayor de estos dos valores, para que la fuente y el destino quepan en el tamaño del "blit". Debera aplicarse la mascara adecuada para eliminar todos los datos no deseados.

Algunas lineas generales para copiar una región arbitraria

1. Usar el canal DMA A, desconectado, cargando con $FFFF y los valores adecuados de mascara y desplazamiento, para enmascarar la función "cookie-cut". Usar el canal B para obtener los datos fuente, el canal C para obtener los datos destino, y el canal D para escribir los datos destino. Usar la función "cookie-cut", $CA.

2. Si hay desplazamiento, siempre usar el modo ascendiente si se desplaza hacia la derecha, y usar el modo descendiente si se desplaza hacia la izquierda.

   Nota: Estos desplazamientos son los desplazamientos de la posición del bit del borde mas a la izquierda dentro de una palabra, en lugar de desplazamientos absolutos, como se explicaba anteriormente.

3. Si la fuente y el destino se superponen, usar el modo ascendiente si el destino tiene una dirección inferior (esta mas arriba en la imagen) y si no, usar el modo descendiente.

4. Si la fuente ocupa mas palabras que el destino, usar el mismo valor de desplazamiento para el canal A que para el canal fuente B y poner las mascaras del canal A como si se estuviera enmascarando a los datos del canal fuente B.

5. Si el destino ocupa mas palabras que la fuente, usar un desplazamiento de cero para el canal A y poner la primera y la última palabra como si se estuvieran enmascarando los datos del canal D.

6. Si la fuente y el destino ocupan el mismo número de palabras, usar el canal A para enmascarar la fuente, como en el 4, o el destino como en la linea 5.

Nota: Las condiciones 2 y 3 pueden ser contradictorias si, por ejemplo, se intenta mover una imagen un pixel hacia abajo y a la derecha. En este caso, se deberia usar el modo descendiente para que el destino no sobreescriba a la fuente antes de que se use la fuente, pero se deberia usar el modo ascendiente para el desplazamiento a la derecha. En algunas situaciones, es posible evitar la linea general 2 con una mascara adecuada. Pero otras veces enmascarar la primera o la última palabra puede no ser suficiente; puede ser necesario enmascarar mas de 16 bits en uno de los dos extremos. En este caso, se puede construir una mascara en memoria para una sola linea, y conectar el canal DMA A para obtener explicitamente esta mascara. Al poner el valor del módulo de A con el valor negativo del ancho de la mascara, la mascara sera obtenida repetidamente para cada linea.

MODO DE RELLENADO DE AREAS

Ademas de la copia de datos, el Blitter puede ejecutar simultaneamente una operación de rellenado durante la copia. La operación de rellenado tiene sólo una restricción -el area a rellenar se debe dibujar primeramente mediante lineas simples de un solo bit por cada linea horizontal. Hay un modo especial de dibujo de lineas para ejecutar esta operación. Se usa un "blit" standard de copia (o cualquier otro "blit", ya que el rellenado de areas se realiza despues de todos los desplazamientos, mascaras y las combinaciones lógicas de las fuentes). Debe usarse el modo descendiente. Se pone a uno el bit de conexión-rellenado-inclusivo (FILL_OR, o bit 3) o el bit de conexión-rellenado-exclusivo (FILL_XOR, o bit 4) de BLTCON1. El modo de rellenado inclusivo rellena entre las lineas, dejando las lineas intactas. El modo de rellenado exclusivo rellena entre las lineas, dejando

las líneas del borde derecho o las zonas rellenas pero borrando las líneas del borde izquierdo. El rellenado exclusivo deja los patrones rellenos un pixel más cerca que los mismos patrones rellenos con el modo inclusivo.

Por ejemplo, el patrón 00100100-00011000

rellenado inclusivo quedaría: rellenado exclusivo quedaría:

00111100-00011000 00011100-00001000

(Por supuesto, los rellenados se ejecutan siempre en palabras de 16 bits)

Hay otro bit (FILL_CARRYIN, o bit 3 de BLTCON1) que sirve para rellenar el area exterior a las lineas. Para el ejemplo anterior el resultado sería:

11100111-11111111 (inclusivo) 11100011-11101111 (exclusivo)

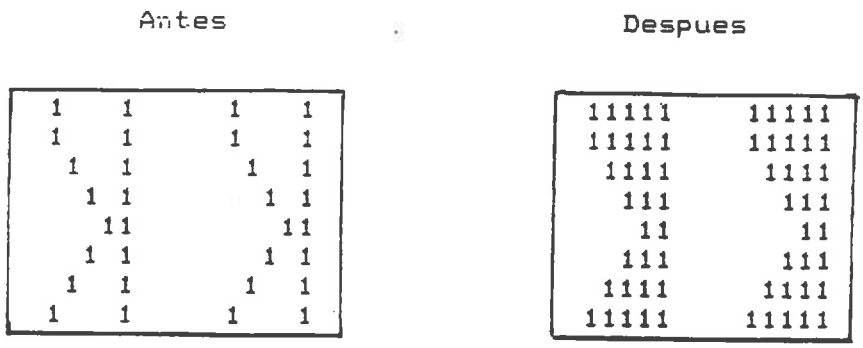


Figura 6-5: Uso del bit FCI - El bit esta a 0.

Si el bit FCI es 1 en lugar de 0, se rellena el area exterior a las lineas con unos y el area interior a las lineas se deja a ceros.

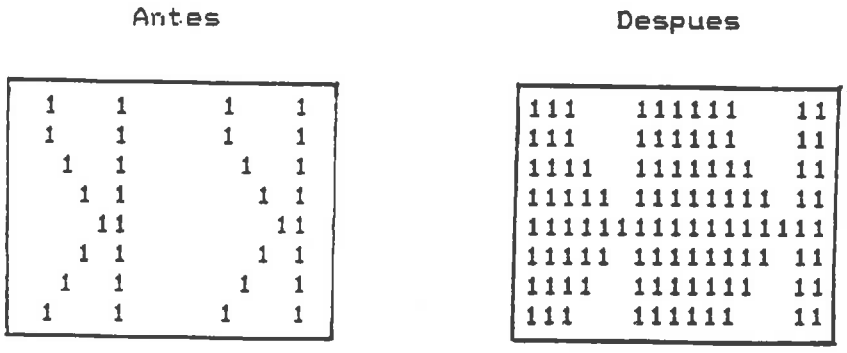


Figura 6-6: Uso del bit FCI - El bit esta a 1.

Si se desean producir vertices de un sólo punto, se debera usar el modo de rellenado exclusivo. La Figura 6-7 muestra cómo se producen los vertices de un sólo punto usando el modo de rellenado exclusivo.

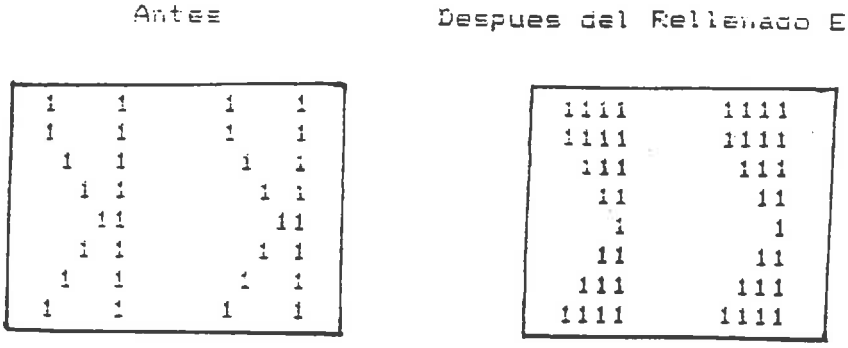


Figura 6-7: Ejemplo de vertice de un sólo punto

El Blitter usa el bit FILL_CARRYIN (FCI) como el estado inicial del rellenado para el primer bit de cada linea (a la derecha por ser modo descendiente). Por cada "1" del area fuente, el Blitter cambia el estado del rellenado, rellenando o no rellenando de esta manera los espacios con unos. Esto continua para cada linea hasta que se alcanza el borde izquierdo del "blit", en el cual termina el rellenado.

INDICADOR DE FINALIZACION DEL BLITTER

Cuando se escribe al registro BLTSIZE comienza el "blit". Sin embargo el microprocesador no se detiene mientras el Blitter esta trabajando; pueden funcionar los dos a la vez, y esto proporciona la mayor parte de la gran velocidad de proceso del Amiga. Pero por esto mismo se necesita tener un cuidado especial cuando se usa el Blitter.

En el registro DMACONR hay un indicador de finalización del Blitter, tambien llamado "indicador de Blitter ocupado" que es el bit DMAF_BLTDONE. Este indicador esta a uno cuando hay un "blit" ejecutandose.

Nota: Si un "blit" acaba de comenzar pero se le ha cerrado el acceso a memoria debido, por ejemplo, a los ciclos destinados a la visualización de la imagen, este bit podria no haber sido puesto a 1 aún. El microprocesador, por otra parte, puede estar funcionando completamente libre en la memoria FAST o en su buffer interno, por tanto continuara teniendo ciclos de memoria.

La solución es leer una dirección de memoria CHIP o un registro del HARDWARE con el microprocesador antes de comprobar el bit. Esto se puede hacer facilmente con la secuencia:

```
BTST.B   #DMAB_BLTDONE-8,DMACONR(a0)
BTST.B   #DMAB_BLTDONE-8,DMACONR(a0)
```

Dónde a0 esta cargado con la dirección de los registros del hardware ($DFF000). La primera "comprobación" del Bit de finalización del Blitter podria no devolver el resultado correcto, pero la segunda si.

Nota: Fat Agnus pone a 1 el bit de "Blitter ocupado" tan pronto como se escribe el registro BLTSIZE para comenzar el "blit", en lugar de esperar a que el Blitter obtenga su primer ciclo DMA. Sin embargo, no todos los ordenadores usan estos chips nuevos, y por tanto es mejor confiar en el metodo anterior de comprobacion.

LA MULTITAREA Y EL BLITTER

Cuando esta en ejecución un "blit", no se debe escribir a ninguno de los registros del Blitter. Para los detalles de la organización de los accesos al Blitter en el sistema, consultar el "ROM Kernel Manual". En particular, leer la explicación sobre las funciones **OwnBlitter()** y **DisownBlitter()**. Después de habernos "apropiado" del Blitter, el "blit" tardara un tiempo en completarse, por lo tanto habra de comprobarse el indicador de "finalización del Blitter" antes de usarlo otra vez. Se recomienda el uso de la función de la ROM **WaitBlit()**.

Se debe comprobar el indicador de "finalización del Blitter" antes de usar los resultados de un "blit". El "blit" puede no estar completado, y por tanto los datos pueden no estar preparados adn. Esto puede evitar que hayan fallos en los programas ("bugs"), porque un 68000 puede ser lo suficientemente lento para que el Blitter finalice sin necesidad de comprobar el indicador de "finalización del Blitter", mientras que un 68020, quizas funcionando en su buffer interno, puede ser capaz de llegar a los datos antes de que el Blitter haya terminado de escribirlos.

Supongase que se tiene una rutina que visualiza un rectangulo de texto temporalmente en la pantalla. Esta rutina deberia reservar un bloque de memoria para guardar el contenido original de la pantalla mientras se esta visualizando el rectangulo de texto, y despues visualizar el texto. Cuando se fuera a terminar, la rutina tendria que reponer el contenido original de la pantalla y liberar la memoria reservada. Si la memoria se libera antes de que se compruebe el indicador de "finalización del Blitter", algún otro proceso podria reservar ese mismo bloque de memoria y escribir nuevos datos en el antes de que el "blit" hubiera acabado, ensuciando la fuente del Blitter y, por lo tanto, la porción de la imagen que se estaba restaurando.

INDICADOR DE INTERRUPCION

El Blitter tambien tiene un indicador de interrupción que se pone a 1 cuando termina el "blit". Este indicador, INTF_BLIT, puede generar una interrupción al 68000 si se conecta. Para mas información sobre las interrupciones, ver el Capitulo 7 "El Hardware de Control del Sistema".

INDICADOR DE CERO

Hay un indicador de cero que se puede examinar para averiguar si la operación lógica seleccionada ha resultado 0 para todos los bits del canal destino, aunque estos bits resultantes no hayan sido escritos al canal DMA D por estar desconectado. Esta caracteristica es útil para detectar colisiones, ejecutando la operación lógica "y" (and) de dos fuentes para comprobar si se superponen. Si las imagenes no se superponen, el indicador de cero continuara siendo 1.

El indicador de cero sólo es valido despues de que el Blitter haya completado su operación y se puede leer del bit DMAF_BLTNZERO del registro DMACONR.

REGISTRO DE "TUBERIA"

El Blitter ejecuta muchas operaciones en cada ciclo -desplazamientos y enmascaramiento de las palabras fuente, combinaciones lógicas de las fuentes, y rellenado de areas y detección de cero en la salida. Para permitir que todas estas operaciones se realicen rapidamente, el Blitter esta unido por una "tuberia". Esto significa que en lugar de ejecutar todas

las operaciones anteriores en un ciclo del Blitter, las operaciones se reparten en dos ciclos del Blitter (aquí la palabra "ciclo" se usa para simplificar, y no corresponde a los ciclos de DMA). Para aclarar esto, podemos imaginar al Blitter como dos chips conectados en serie. Cada ciclo, llega un nuevo grupo de datos, y el primer chip ejecuta sus operaciones en los datos. Entonces pasa los datos "a mitad de procesar" al segundo chip para que se completen durante el siguiente ciclo, cuando el primer chip ya esta ocupado con en el siguiente grupo de datos. Cada grupo de datos necesita dos "ciclos" para atravesar los dos chips, por tanto un grupo de datos se puede "bombear" a traves de la "tuberia" en cada ciclo si se hace solapadamente.

Lo que todo esto quiere decir es que se obtiene el primer grupo de dos palabras de los canales fuente antes de que se escriba la primera palabra en el canal destino. Esto permite que se desplace una imagen como maximo una palabra hacia la derecha usando el modo ascendiente, por ejemplo, aunque normalmente las partes del canal destino serian sobreescritas antes de que el canal fuente las hubiera podido obtener.

Tabla 6-2: Secuencia Tipica de Ciclos del Blitter

Código en BLTCON0	Canales Activos	Secuencia de Ciclos
F	A B C D	A0 B0 C0 -- A1 B1 C1 D0 A2 B2 C2 D1 D2
E	A B C	A0 B0 C0 A1 B1 C1 A2 B2 C2
D	A B D	A0 B0 -- A1 B1 D0 A2 B2 D1 -- D2
C	A B	A0 B0 -- A1 B1 -- A2 B2
B	A C D	A0 C0 -- A1 C1 D0 A2 C2 D1 -- D2
A	A C	A0 C0 A1 C1 A2 C2
9	A D	A0 -- A1 D0 A2 D1 -- D2
8	A	A0 -- A1 -- A2
7	B C D	B0 C0 -- -- B1 C1 D0 -- B2 C2 D1 -- D2
6	B C	B0 C0 -- B1 C1 -- B2 C2
5	B D	B0 -- -- B1 D0 -- B2 D1 -- D2
4	B	B0 -- -- B1 -- -- B2
3	B	B0 -- -- C1 D0 -- C2 D1 -- D2
2	C D	C0 -- C1 -- C2
1	D	D0 -- D1 -- D2
0	NINGUNO	-- -- -- --

Notas:

- No esta avtivado el modo de rellenado
- El Blitter tiene acceso exclusivo al bus de datos.
- Se esta ejecutando un "blit" de tres palabras
- En el funcionamiento normal se obtinen los datos de los canales fuente dos veces antes de que este disponible el primer dato para el canal DMA D. Esto es debido a la "tuberia" interna. Se debe tener un cuidado especial cuando la fuente y el destino se superponen.

Nota: Esta tabla sólo quiere ser una ilustración del orden tipico de los ciclos del Blitter sobre el Bus de datos. Los ciclos del Bus se efectuan dinamicamente (sobre la marcha) basandose en el modo de operación del Blitter; la competencia con la actividad del micro-procesador, los bit-planes, y los demas canales DMA; y otros factores. Commodore Amiga no garantiza la precisión de esta tabla en los modelos actuales y futuros. Nosotros nos reservamos el derecho a hacer mejoras en el producto o cambios en el diseño de este area sin notificarlo.

MODO DE LINEAS

Además de todas las funciones descritas anteriormente, el Blitter puede dibujar lineas de distintos patrones. El modo de trazado de lineas se selecciona poniendo a uno el bit 0 (LINEMODE) de BLTCON1, que modifica el significado de otros bits de BLTCON0 y BLTCON1. En el modo de trazado de lineas, el Blitter puede dibujar lineas de hasta 1024 pixels de largo, las puede dibujar de diversas formas, con diversas texturas, y las puede dibujar tambien de una manera especial para rellenar areas.

Muchos de los registros del Blitter sirven para otros propósitos en el modo de trazado de lineas. Consultar el Apendice A para descripciones mas detalladas del uso de estos registros y los bits de control en el modo de trazado de lineas.

En el modo de lineas, el Blitter dibuja una linea de un punto a otro, que se puede ver como un vector. El vector director puede estar en cualquiera de los siguientes ocho octantes. (En el siguiente diagrama, se usa la convención standard del Amiga, con la X incrementandose hacia la derecha y la Y hacia abajo). El número entre paréntesis es el número del octante; el número de orden representa el valor que se debera poner en los bits 4-2 del registro BLTCON1.

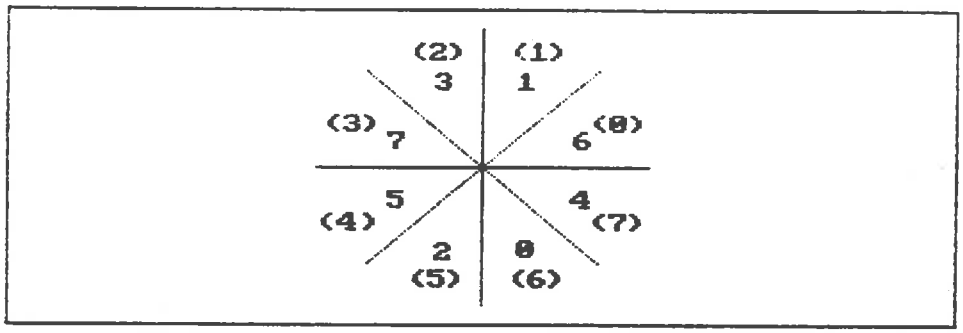


Figura 6-8: Octantes para Dibujo de Lineas

El dibujo de las lineas basado en los octantes es una simplificación de las simetrias entre X y -X, Y y -Y. La siguiente Tabla muestra los números de los octactes y sus correspondientes valores.

Tabla 6-3: Código de Bits para Octantes de Dibujo de Lineas.

Bits de BLTCON1 (4 3 2)	Octante Número
1 1 0	0
0 0 1	1
0 1 1	2
1 1 1	3
1 0 1	4
0 1 0	5
0 0 0	6
1 0 0	7

Ahora se inicializan los bits 4-2 de BLTCON1 de acuerdo con la anterior Tabla. Despues, se introducen las variables dx y dy, que son el valor absoluto de la diferencia entre las coordenadas X e Y, y las coordenadas del punto final de la linea, respectivamente.

```
dx = abs (x2 - x1)        dx = abs (y2 - y1)
```

Ahora, se modifican si son necesarias (cuando dx es mayor que dy).

si (dx < dy) --> temp = dx, dx = dy, dy = temp

Alternativamente, se puede calcular dx y dy como sigue:

```
dx = max (abs (x2 - x1), abs (y2 - y1));
dx = min (abs (x2 - x1), abs (y2 - y1));
```

Estos calculos tienen el efecto de "normalizar" la linea en el octante 0; aunque en realidad ya se ha informado al Blitter del octante que va a usar, y entonces ya no dificulta para dibujar la linea.

Se inicializa el registro A de direcciones con 4 * dy - 2 * dx. Si este valor es negativo, se pone a 1 el bit del signo (SIGNFLAG de BLTCON1), si no se pondria a cero. Se inicializa el registro A de módulo con 4 * (dy - dx) y el registro B de módulo con 4 * dy.

El registro A de datos se debe cargar con $8000. Ambas mascaras se deben poner a $FFFF. El valor A de desplazamiento se debe poner a la coordenada X del primer punto (x1) módulo 15 (resto de dividir: x1 / 15).

El registro B de datos se debe poner con el patrón de la linea, si no se quiere ninguno, o $FFFF para una linea sólida. El valor del desplazamiento se debe poner al número de bit en el cual empieza el patrón (0 significa el bit menos significativo).

Los registros C y D de direcciones se deben inicializar apuntando a la palabra que contiene el primer pixel de la linea; los registros C y D de módulo se deben poner con el ancho del bit-plane en bytes.

Los bits SRCA, SRCC, y DEST de BLTCON0 se deben poner a 1, y el bit SRCB se debe poner a 0. El bit OVFLAG se debe poner a 0. Si solo se desea un bit por cada linea orizontal se debera poner a 1 el bit ONEDOT de BLTCON1; si no se debera poner a 0.

La función lógica continda. El canal DMA C representa la fuente original, el canal A el bit que se activa en la linea, y el canal B el patrón a dibujar. Por lo tanto, para dibujar una linea la función $AB + \overline{A}C$ es la mas común. Para dibujar la linea usando el modo exclusivo (XOR), para que se pueda borrar facilmente la linea dibujandola otra vez, se puede usar la función $AB\overline{C} + \overline{A}C$.

Se pone la altura del "blit" con la longitud de la linea que es dx + 1. El ancho se debe poner en todos los dibujos de lineas a 2. (Por supuesto, el registro BLTSIZE no se debe escribir hasta el final, cuando se hayan inicializado todos los demas registros).

SUMARIO DE REGISTROS PARA EL MODO DE LINEAS

Primer calculo:

La linea va desde (x1,y1) hasta (x2,y2)

```
dx = max (abs (x2 - x1), abs (y2 - y1));
dx = min (abs (x2 - x1), abs (y2 - y1));
```

Puesta a punto de los registros:

BLTADAT = $8000
BLTBDAT = Textura del patrón de la linea ($FFFF para linea sólida)

```
BLTAFWM = $FFFF
BLTALWM = $FFFF

BLTAMOD = 4 * (dy - dx)
BLTBMOD = 4 * dy
BLTCMOD = Ancho del bit-plane en bytes
BLTDMOD = Ancho del bit-plane en bytes

BLTAPT = (4 * dy) - (2 * dx)
BLTBPT = Sin usar
BLTCPT = Dirección de la palabra que contiene el primer pixel de la linea
BLTDPT = Dirección de la palabra que contiene el primer pixel de la linea

BLTCON0 bits 15-12 = x1 módulo 15
BLTCON0 bits SRCA, SRCC, y SRCD = 1
BLTCON0 bit SRCB = 0
si modo :OR de lineas:
     entonces, byte LF de BLTCON0 = ABC + AC
     si no,    byte LF de BLTCON0 = AB + AC

BLTCON1 bit LINEMODE = 1
BLTCON1 bit OVFLAG = 0
BLTCON1 bits 4-2 = Número de octante de la tabla
BLTCON1 bits 15-12 = Bit inicial para la textura (0 = menos significativo)
si (((4 * dy) - (2 * x)) < 0):
     entonces, bit SIGNFLAG de BLTCON1 = 1
     si no,    bit SIGNFLAG de BLTCON1 = 0

Si un pixel/linea:
     entonces, bit ONEDOT de BLTCON1 = 1
     si no,    bit ONEDOT de BLTCON1 = 0

BLTSIZE bits 15-6 = dx + 1
BLTSIZE bits 5-0 = 2
```

NOTA: El registro BLTSIZE el último, porque pone en marcha el "blit".

VELOCIDAD DEL BLITTER

La velocidad del Blitter depende completamente de que canales DMA esten conectados. Se puede usar un canal DMA como una constante, pero a menos que este conectado, no gastara ciclos extras. El ciclo minimo del Blitter es de cuatro ciclos de memoria; el maximo son ocho ciclos de memoria. El uso del registro A es siempre libre. El uso del registro B siempre añade dos ciclos de memoria al ciclo del Blitter. El uso del C o el D es libre, pero el uso de ambos añade otros dos ciclos de memoria. Por lo tanto, un ciclo de copia, usando A y D, gasta cuatro ciclos de memoria por cada ciclo de copia; un ciclo de copia usando B y D gasta seis ciclos de memoria, y la copia normal con todos los registros gasta ocho ciclos de memoria. En el modo de lineas, cada punto gasta ocho ciclos de memoria.

La velocidad del reloj del sistema es de 7.16 MHz en NTSC, y 7.09 MHz en PAL. El reloj del Blitter es el reloj del sistema. Para calcular el tiempo total de un "blit" en microsegundos, excluyendo la puesta a punto y la contención de DMA, se usa la ecuación:

$$t = \frac{n * H * W}{7.16} \text{ (NTSC)} \qquad t = \frac{n * H * W}{7.09} \text{ (PAL)}$$

Dónde t es el tiempo en microsegundos (µs), n es el número de ciclos de memoria, y H y W son el ancho y el alto (en palabras) del "blit", respectivamente.

Por ejemplo, para copiar un bit-plane de 320 por 256 a otro bit-plane, se usarían los canales A y D. Se necesitarían cuatro ciclos de memoria por cada uno del Blitter, para un total de:

$$t = \frac{4 * 256 * 20}{7.09} = 2888.5 \text{ us (millonesimas de segundo)}$$

Estos calculos no toman en cuenta el tiempo de puesta a punto del Blitter, que es el tiempo necesario para calcular y cargar los registros del Blitter y ponen en marcha el "blit". Tambien se ignora la contención de DMA.

LAS OPERACIONES DEL BLITTER Y EL DMA DEL SISTEMA

Las operaciones del Blitter afectan al funcionamiento del resto del sistema. Las siguientes secciones explican cómo el funcionamiento del sistema por la prioridad de acceso directo a memoria del Blitter, los "slots" de tiempo del DMA, la compartición del bus con el 68000 y el hardware de visualización, las operaciones del Blitter y el Copper, y los diferentes tamaños de los playfileds.

El Blitter ejecuta sus diversas operaciones de obtención de datos, modificación y almacenamiento a traves de secuencias de DMA, y comparte los accesos a memoria con otros dispositivos del sistema. Cada dispositivo que accede a la memoria tiene un nivel de prioridad asignado, que indica su importancia relativa a los demas dispositivos.

Los DMAs del disco, del audio, de visualización, y de los Sprites tienen el nivel mas alto de prioridad. El DMA de visualización tiene prioridad sobre el de los Sprites sólo en alunos casos. Cada uno de estos cuatro dispositivos tiene reservado un grupo de "slots" de tiempo durante cada linea horizontal. Si un dispositivo no pide uno de los "slots" que tiene reservado, el "slot" queda libre para otros usos. Estos dispositivos tienen la maxima prioridad porque los ciclos de DMA perdidos podrian ocasionar perdidas de datos, ruido en la salida del sonido, o interrupciones en la imagen.

El Copper tiene el siguiente nivel de prioridad porque tiene que ejecutar operaciones al mismo tiempo que se visualiza la imagen para permanecer sincronizado con el rayo de electrones que barre la pantalla.

Las prioridades mas bajas estan asignadas al Blitter y al 68000, en este orden. El Blitter tiene mas prioridad que el 68000 porque ejecuta las operaciones de copia de datos, modificación, y dibujo mucho mas rapidamente que el 68000.

Durante una linea horizontal (unos 63 us), hay 227.5 ciclos del reloj de color, o ciclos de acceso a memoria. Un ciclo de memoria dura aproximadamente unos 280 ns. El total de 227.5 ciclos por linea horizontal incluye un tiempo de visualización y otro de no-visualización. De este tiempo total, hay 226 ciclos disponibles para ser utilizados por los diversos dispositivos que necesitan acceso a la memoria.

Los "slots" de tiempo estan reservados como sigue:

4 ciclos para el "refresco" de memoria	
3 ciclos para el DMA del disco	(hasta 91.5 K por segundo)
4 ciclos para el DMA del audio	(2 bytes por canal)
16 ciclos para el DMA de los Sprites	(2 palabras por canal)
80 ciclos para el DMA de los Bit-planes	(los "slots" pares o impares, según el tamaño del bit-plane)

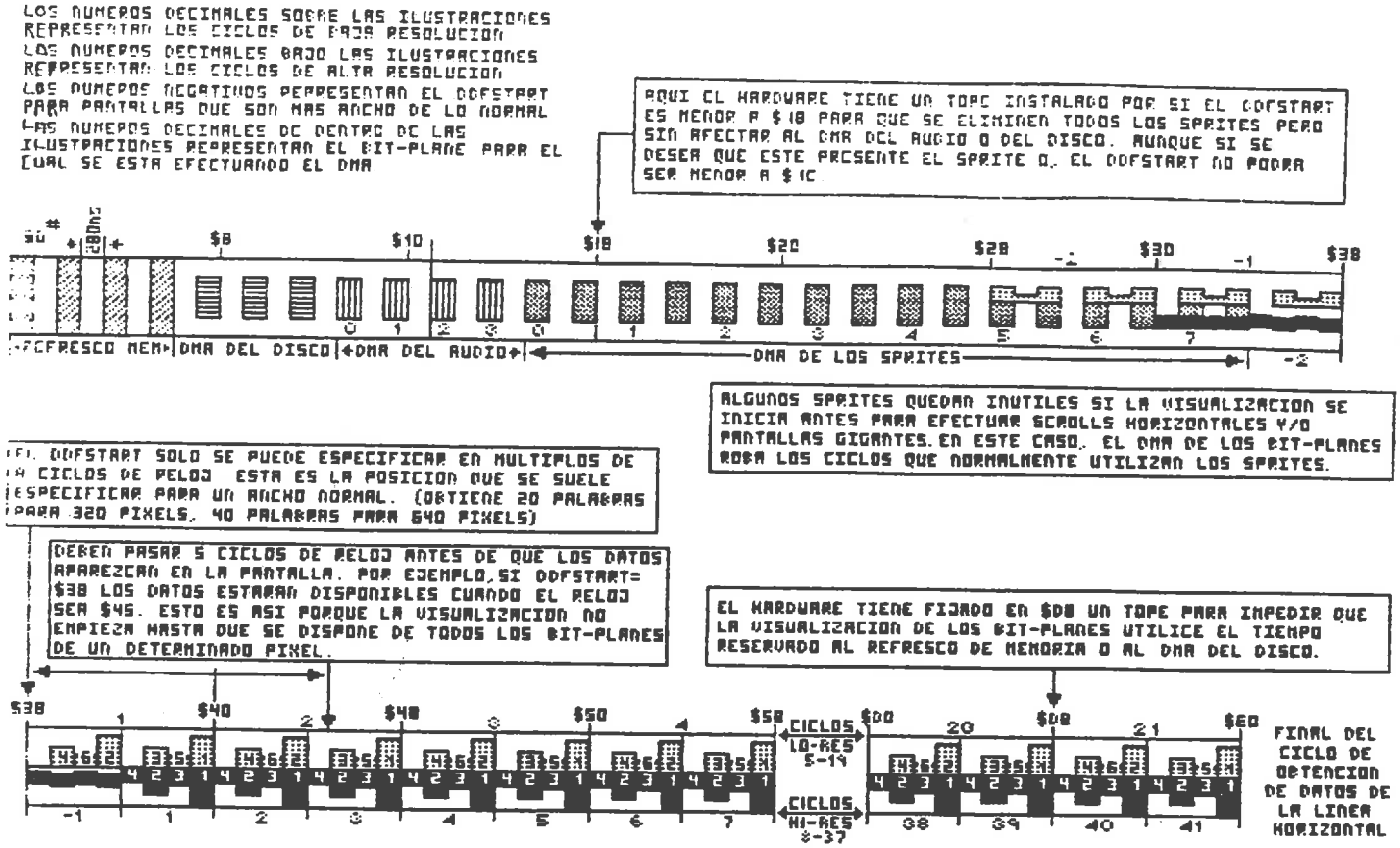


Figura 6-9: Los Slots de Tiempo del DMA

El 68000 sólo usa los ciclos pares de DMA. El 68000 utiliza la mitad del tiempo de una instrucción haciendo operaciones internas y la otra mitad accediendo a memoria. Por lo tanto, al reservar los ciclos de memoria alternativamente para el 68000 es como si este tuviera siempre el acceso a memoria, y por tanto funcionara a su maxima velocidad.

Algunas instrucciones del 68000 no funcionan correctamente con este sistema de los ciclos pares y provocan la perdida de ciclos. Si se pierden los ciclos, el 68000 se ve obligado a esperar hasta el siguiente "slot" disponible de memoria antes de continuar. Sin embargo, la mayoria de las instrucciones no provocan perdidas de ciclos, y por tanto el 68000 funciona a su maxima velocidad si no hay interferencias con el DMA del Blitter.

Nota: La instrucción del 68000 test-and-set (TAS, "comprobar y activar") no debe ser usada nunca en el Amiga; el ciclo indivisible read-modify-write (leer-modificar-escribir) que se usa sólo en esta instrucción no cabe en un "slot" de DMA.

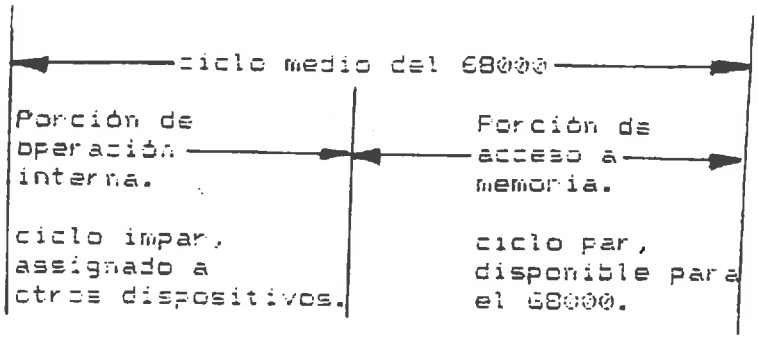


Figura 6-10: Ciclo normal del 68000

Si la imagen contiene 4 o menos bit-planes de baja resolución, el 68000 tendra asegurado los ciclos alternativos de memoria (si esta preparado para utilizar el ciclo y es el elemento de mayor prioridad en ese momento). Sin embargo, si hay mas de 4 bit-planes, el DMA de los bit-planes comenzara a robar ciclos del 68000 durante la visualización.

Durante el tiempo de visualización para una imagen de 6 bit-planes (baja resolución, 320 pixels de ancho), el DMA de los bit-planes tomara 120 "slots" de tiempo por cada linea horizontal. Como se puede ver en la Figura 6-11, el DMA de los bit-planes roba el 50% de los slots libres que habria usado el 68000 si sólo se hubieran visualizado 4 bit-planes.

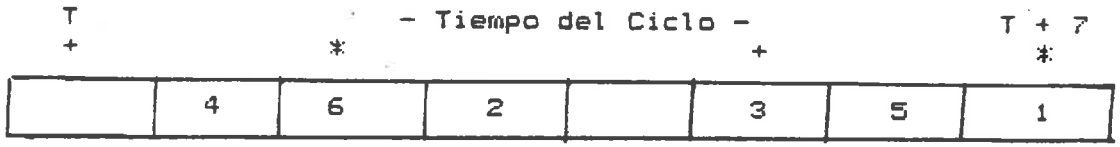


Figura 6-11: Slots de Tiempo usados en una imagen de 6 Bit-Planes

Si se especifican 4 bit-planes de alta resolución (640 pixels de ancho), el DMA de los bit-planes necesitara TODOS los "slots" de tiempo disponibles durante el tiempo de visualización para obtener las 40 palabras de datos de que se compone cada una de las lineas de los cuatro bit-planes (40 * 4 = 160 "slots" de tiempo). Esto efectivamente bloquea al 68000 (al igual que al Blitter y al Copper) de acceder a la memoria durante la visualización, excepto durante el vertical y el horizontal blank.

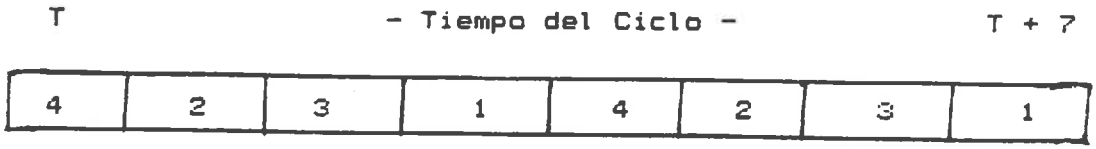


Figura 6-12: Slots de Tiempo usados en una imagen de Alta Resolución

Cada linea horizontal en una imagen normal, contiene 320 pixels en modo de baja resolución o 640 pixels en modo de alta resolución. Por lo tanto, se obtendran 20 o 40 palabras durante el tiempo de visualización de la linea horizontal. Si se desea desplazar (scroll) un playfield, se debera obtener una palabra extra de la memoria por cada linea.

El tamaño de la imagen es ajustable (ver el Capitulo 3, "El Hardware de los Playfields"), y el DMA de los bit-planes tiene prioridad sobre el DMA de los Sprites. Como se muestra en la Figura 6-9, las imagenes muy anchas pueden bloquear uno o mas Sprites, especialmente con los Scrolls.

Como se mencionaba antes, el Blitter normalmente tiene una mayor prioridad que el 68000 para los ciclos DMA. Hay ciertos casos, sin embargo, que el Blitter y el 68000 pueden compartir ciclos de memoria. Si se diese el caso, el Blitter robaria TODOS los ciclos de memoria disponibles. La visualización, el disco y el audio tienen prioridad sobre el Blitter, por lo que no se le permite al Blitter bloquearles el acceso a memoria. Dependiendo del modo del DMA del Blitter, comunmente llamado bit de "Blitter-nasty" (Blitter desagradable), el 68000 se vera bloqueado de los accesos al bus. Este bit se llama DMAF_BLITHOG y esta en el registro DMACON.

Si DMAF_BLITHOG es un 1, el Blitter se apoderara del bus por cada ciclo de memoria disponible. Esto podria ser potencialmente todos los ciclos.

Si DMAF_BLITHOG es un 0, el distribuidor de DMA observara las peticiones de ciclo de memoria del 68000. Si el 68000 no se satisface en tres ciclos consecutivos de memoria, el Blitter abandonara el bus por un ciclo.

DIAGRAMA DE BLOQUES DEL BLITTER

- La Figura 6-13 muestra la construcción basica de bloques para un solo bit de una operación de 16 bits de ancha del Blitter. No esta incluido el Hardware de dibujo de lineas.

- La esquina superior izquierda muestra como la primera -y la última- palabra de mascara se aplica al dato procedente de la fuente A. Cuando el "blit" se limita a una palabra de ancho, se aplican ambas mascaras.

- Los desplazadores (arriba a la derecha y en la mitad a la izquierda) muestran cómo se toman los 16 bits de datos de una determinada posición de un registro de 32 bits, basado en los valores de desplazamiento de A y B contenidos en los registros BLTCON0 y BLTCON1.

- El generador de miniterms (en la mitad a la derecha) ilustra cómo los bits de selección miniterm permiten o inhiben el uso de una miniterm especifica.

- El dibujo muestra cómo trabaja la operación de rellenado sobre los datos generados por las combinaciones de miniterms. Las operaciones de rellenado pueden ser ejecutadas simultaneamente con otras operaciones lógicas complejas.

- Al fondo, el dibujo muestra cómo se puede evitar que los datos generados por el canal D se escriban en una posición de memoria destino usando uno de los bits de control del Blitter.

- No se incluye en este diagrama la lógica de detección de cero, que examina cada bit generado por el canal D. Si se generara cualquier bit a 1, esta lógica indicaria que el area del "blit" contiene como minimo un bit a 1 (la detección de cero es falsa).

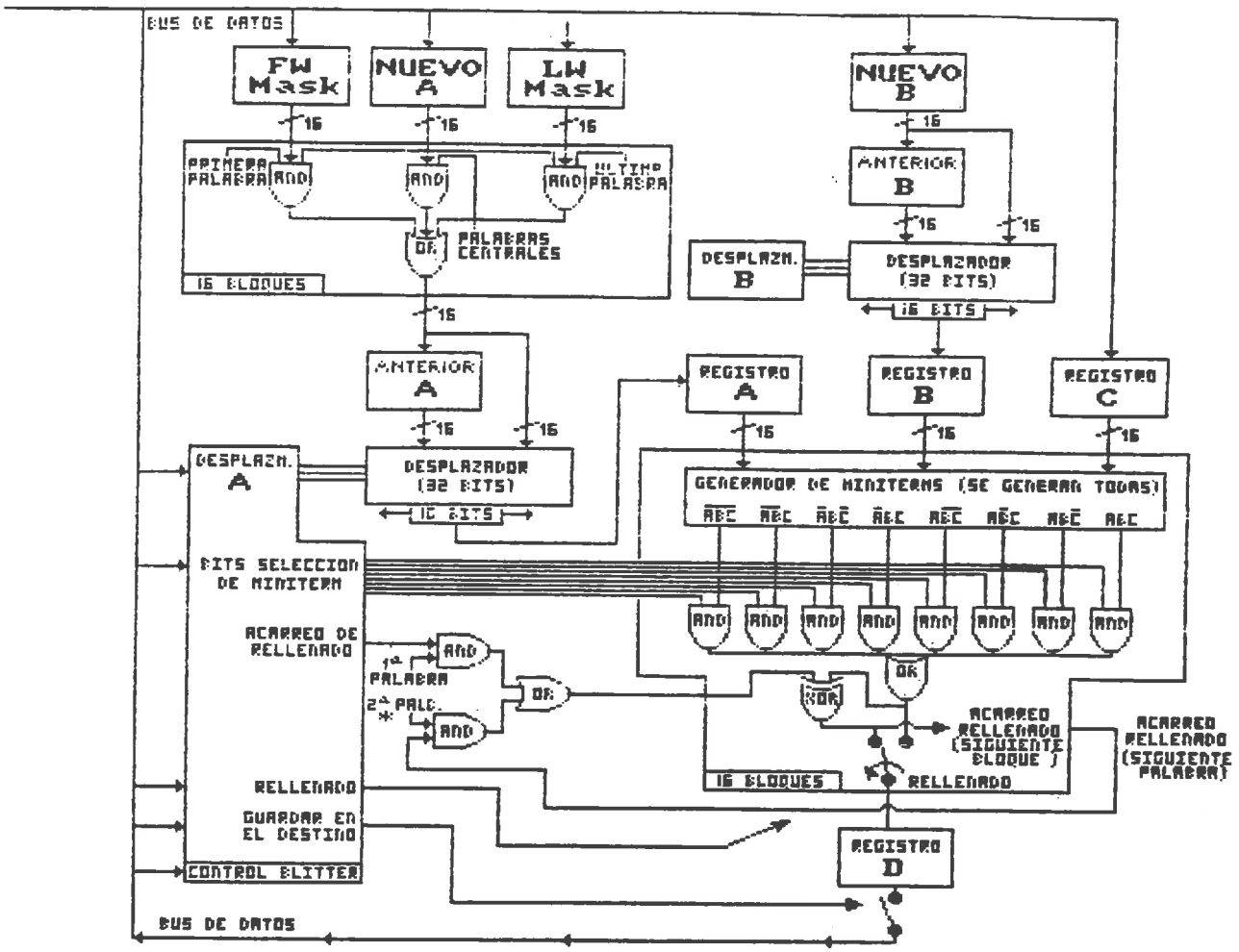


Figura 6-13: Diagrama de Bloques del Blitter

PUNTOS CLAVE DEL BLITTER

Esta es una lista de algunos de los puntos clave que se deben recordar cuando se programa al Blitter.

- Escribir BLTSIZE el último, cuando se escribe este registro se pone en marcha el "blit".

- Los módulos y los punteros estan en bytes; el ancho esta en palabras y el alto en pixels. El bit menos significativo de todos los punteros y módulos se ignora.

- El orden de las operaciones en el Blitter es enmascaramiento, desplazamiento, combinación lógica de fuentes, rellenado de areas, y activación del indicador de cero.

- En el modo ascendiente, el Blitter incrementa los punteros, suma los módulos, y desplaza hacia la derecha.

- En el modo descendiente, el Blitter decrementa los punteros, resta los módulos, y desplaza hacia la izquierda.

- El rellenado sólo funciona correctamente en el modo descendiente.

- Comprobar BLTDONE antes de escribir nuevamente los registros del Blitter o usar los resultados de un "blit".

- Los desplazamientos se efectuan sobre los datos inmediatos tan pronto como estos se cargan en los registros del Blitter.

EJEMPLO: CLEAR MEM (BORRA MEMORIA)

```
    Ejemplo de Blitter     --- borrado de memoria

      include  'exec/types.i'
      include  'hardware/custom.i'
      include  'hardware/dmabits.i'
      include  'hardware/blit.i'
      include  'hardware/hw_examples.i'

      xref     _custom

    Espera a que se complete el "blit" anterior

waitblit:
      btst.b   #DMAB_BLTDONE-8,DMACONR(a1)
waitblit2:
      btst.b   #DMAB_BLTDONE-8,DMACONR(a1)
      bne      waitblit2
      rts

;
    Esta rutina usa un efecto residual del Blitter. Cuando cada uno
    de los "blits" se completa, el puntero del Blitter queda apuntando
    a la siguiente palabra que sera "bliteada"
;
;   Cuando esta rutina retorna, el último "blit" se ha iniciado y
;   podria no haber acabado, para estar seguro llamar a waitblit
;   antes de asumir que los datos estan borrados.
;
;   a0 = puntero a la primera palabra a borrar
;   d0 = número de bytes a borrar (debe ser par)
;
      xdef     clearmem
clearmem:
      lea      _custom,a1          ; Base de los custom chips...
      bsr      waitblit            ; Mira si el último "blit" ha finalizado
      move.l   a0,BLTDPT(a1)       ; El puntero D con la dirección adecuada
      clr.w    BLTDMOD(a1)         ; Borra el módulo D
      asr.l    #1,d0               ; Obtiene el número de palabras
      clr.w    BLTCON1(a1)         ; Ningún modo especial
      move.w   #DEST,BLTCON0(a1)   ; Sólo conecta el canal D

;
;   Primero se trataran los "blits" pequeños
;
      moveq    #$3f,d1             ; Se obtiene la mascara con 64
      and.w    d0,d1
      beq      dorest              ; ¿nada? bueno, se hace un "blit"
      sub.l    d1,d0               ; Si no se elimina el denominador
      or.l     #$40,d1             ; Se pone la altura a 1, el ancho a n
      move.w   d1,BLTSIZE(a1)      ; Dispara el "blit"
```

Ahora se hace el resto de las palabras como bloques de 128K

```
dorest:
      move.w   #$ffc0,d1          ; Busca los bits superiores.
      and.w    d0,d1              ; Extrae 10 bits
      beq      dorest2            ; algo para hacer?
      sub.l    d1,d0              ;
      bsr      waitblit           ; Espera que termine el "blit" anterior
      move.w   d0,BLTSIZE(a1)     ; Hacer otro "blit"
dorest2:
      swap     d0                 ; mas?
      beq      done               ; Nada mas.
      clr.w    d1                 ; Hace un "blit" de 1024*64 palabras (128K)
keepon:
      bsr      waitblit           ; Finalización de este "blit"
      move.w   d1,BLTSIZE(a1)     ; Y de nuevo, "blit"
      subq.w   #1,d0              ; queda mas?
      bne      keepon             ; se mantiene en el bucle

done:
      rts                         ; Fin. El Blit continua en funcionamiento.
      end
```

EJEMPLO: SIMPLE LINE (LINEA SIMPLE)

```
;
;  Este ejemplo usa el modo de dibujo de lineas del Blitter
;  para dibujar una linea. La linea se dibuja sin ningún patrón
;  y con un "blit" simple de "or" (se suma a lo que hay debajo)
;  sobre un bit-plane.
;
;  Entradas:  d0=x1  d1=y1  d2=x2  d3=y2  d4=ancho  a0=aptr
;
      include  'exec/types.i'
      include  'hardware/custom.i'
      include  'hardware/dmabits.i'
      include  'hardware/blit.i'
      include  'hardware/hw_examples.i'
;
      xref     _custom
;
      xref     _
;
;  El punto de entrada
;
simpleline:
      lea      _custom,a1         ; Base de los custom chips...
      sub.w    d0,d2              ; Calcula dx
      bmi      xneg               ; Si negativo, octante es uno de [3,4,5,6]
      sub.w    d1,d3              ; Calcula dy,  octante es uno de [1,2,7,8]
      bmi      yneg               ; Si negativo, octante es uno de [7,8]
      cmp.w    d3,d2              ; cmp [dx],[dy] octante es uno de [1,2]
      bmi      ygtx               ; Si y > x octante es el 2
      moveq.l  #OCTANTE1+LINEMODE,d5 ; Si no el octante es el 1
      bra      lineagain          ; Va a la sección común

ygtx:
      exg      d2,d3              ; La X debe ser mayor que la Y
      moveq.l  #OCTANT2+LINEMODE,d5 ; Estamos en el octante 2
      bra      lineagain          ; Va a la sección común
```

```
xneg:
     neg.w     d3                    ; Calcula el v. absoluto de dy
     cmp.w     d3,d2                 ; cmp [dx],[dy], octante es [7,8]
     bmi       ynygtx                ; Si y > x, octante es 7
     moveq.l   #OCTANT8+LINEMODE,d5  ; Si no octante es 8
     bra       lineagain

xnygtx:
     exg       d2,d3                 ; La X debe ser mayor que la Y
     moveq.l   #OCTANT7+LINEMODE,d5  ; Este es el octante 7
     bra       lineagain

xneg:
     neg.w     d2                    ; dx era negativo. Octante es [3,4,5,6]
     sub.w     d1,d3                 ; Calcula dy
     bmi       xyneg                 ; Si negativo, octante es uno de [5,6]
     cmp.w     d3,d2                 ; Si no es uno de [3,4]
     bmi       xnygtx                ; Si y > x octante es 3
     moveq.l   #OCTANT4+LINEMODE,d5  ; si no es el 4
     bra       lineagain

xnygtx:
     exg       d2,d3                 ; X debe ser mayor que Y
     moveq.l   #OCTANT3+LINEMODE,d5  ; Este es el octante 3
     bra       lineagain

xyneg:
     neg.w     d3                    ; Y era negativo, en uno de [5,6]
     cmp.w     d3,d2                 ; Es y > x ?
     bmi       xynygtx               ; Si es asi, octante es el 6
     moveq.l   #OCTANT5+LINEMODE,d5  ; Si no es el octante 5
     bra       lineagain

xynygtx:
     exg       d2,d3                 ; X debe ser mayor que Y
     moveq.l   #OCTANT6+LINEMODE,d5  ; Este es el octante 6

lineagain:
     mulu.w    d4,d1                 ; Calcula y1 * ancho
     ror.l     #4,d0                 ; Los 4 bits superiores a la palabra alta
     add.w     d0,d0                 ; Multiplica por 2
     add.l     d1,a0                 ; ptr += (x1 >> 3)
     add.w     d0,a0                 ; ptr += y1 * ancho
     swap      d0                    ; Coge los cuatro bits de x1
     or.w      #$BFA,d0              ; Usa canal A, C y D y función F=A+C
     lsl.w     #2,d3                 ; Y = 4 * Y
     add.w     d2,d2                 ; X = 2 * X
     move.w    d2,d1                 ; Prepara la palabra del tamaño
     lsl.w     #5,d1                 ; Desplaza cinco a la izquierda
     add.w     #$42,d1               ; Y suma 1 a la altura, 2 al ancho.
     btst      #DMAB_BLTDONE-8,DMACONR(a1)  ; Comprobación de seguridad

waitblit:
     btst      #DMAB_BLTDONE-8,DMACONR(a1)  ; Espera al Blitter
     bne       waitblit
     move.w    d3,BLTBMOD(a1)        ; B mod = 4 * Y
     sub.w     d2,d3
     ext.l     d3
     move.l    d3,BLTAPT(a1)         ; A ptr = 4 * Y - 2 * X
     bpl       lineover              ; Si negativo
     or.w      #SIGNFLAG,d5          ; Pone a 1 el bit de signo

lineover:
     move.w    d0,BLTCON0(a1)        ; Escribe los registros de control
     move.w    d0,BLTCON1(a1)
```

```
        move.w    d2,BLTCMOD(a1)      ; C mod = ancho del bitplane
        move.w    d0,BLTDMOD(a1)      ; D mod = ancho del bitplane
        sub.w     d2,d3
        move.w    d3,BLTAMOD(a1)      ; A mod = 4 * Y - 4 * X
        move.w    #$8000,BLTADAT(a1)  ; A data = 0x8000
        moveq.l   #-1,d5              ; Pone las mascaras a 1
        move.l    d5,BLTAFWM(a1)      ; Se escriben las dos mascaras a la vez
        move.l    a0,BLTCPT(a1)       ; Apunta al primer pixel a activar
        move.l    a0,BLTDPT(a1)
        move.w    d1,BLTSIZE(a1)      ; Inicia el "Blit"
        rts                           ; Y retorna, el "Blit" continua funcionando
        end
```

EJEMPLO: ROTATE BITS (ROTACION DE BITS)

```
;
;  Aqui se hacen rotar unos bits. Este programa coge una linea de un
;  bit-plane, y la "rota" en una matriz de palabras de 16 bits, poniendo
;  los bits especificos de cada palabra en la matriz de acuerdo con el
;  bit correspondiente de la linea. Se usa el modo de lineas en conjunción
;  con los patrones para hacer este efecto.
;
;  Entrada: d0 contiene el número de palabras de la linea.
;           d1 contiene el número del bit a activar (0..15)
;           a0 contiene la dirección de la linea en el bit-plane
;           a1 contiene la dirección de la matriz que se va a llenar;
;              la matriz debe ser como minimo (d0)*16 palabras de larga
;
        include   'exec/types.i'
        include   'hardware/custom.i'
        include   'hardware/dmabits.i'
        include   'hardware/blit.i'
        include   'hardware/hw_examples.i'
;
        xref      _custom
;
        xref      _rotatebits
;
; Punto de entrada
;
rotatebits:
        lea       _custom,a2          ; Base de los custom Chips...
        tst       d0                  ; Si no hay palabras, retorna
        beq       gone
        lea       DMACONR(a2),a3      ; Obtiene la dirección de DMACONR
        moveq.l   #DMAB_BLTDONE-8,d2  ; Obtiene el número de bit BLTDONE
        btst      d2,(a3)             ; Comprueba si el "Blit" ha terminado

wait1:
        btst      d2,(a3)             ; Comprueba de nuevo
        bne       wait1               ; No ha terminado?, sigue esperando
        moveq.l   #-30,d3             ; Modo de lineas: aptr = 4Y-2X, Y=0, X=15
        move.l    d3,BLTAPT(a2)
        move.w    #-60,BLTAMOD(a2)    ; A mod = 4Y-4X
        clr.w     BLTBMOD(a2)         ; B mod = 4Y
        move.w    #2,BLTCMOD(a2)      ; C mod = ancho del bit-map (2)
        move.w    #2,BLTDMOD(a2)      ; D mod = ancho del bit-map (2)
        ror.w     #4,d1               ; Coge los 4 bits del número de bit
        and.w     #$f000,d1           ; Los enmascara
        or.w      #$bca,d1            ; Usa canal A, C y D y función F=AB+AC
        move.w    d1,BLTCON0(a2)      ; Lo almacena
        move.w    #$f049,BLTCON1(a2)  ; BSH=15, SGN, LINE
        move.w    #$8000,BLTADAT(a2)  ; Inicializa A dat para la linea
```

```
      move.w    #$ffff,BLTAFWM(a2) ; Inicializa las mascaras
      move.w    #$ffff,BLTALWM(a2)
      move.l    a1,BLTCPT(a2)      ; Inicializa punteros
      move.l    a1,BLTDPT(a2)
      lea       BLTBDAT(a2),a4     ; Para un acceso rapido, se guardan
      lea       BLTSIZE(a2),a5     ; estas dos direcciones.
      move.w    #$402,d1           ; Almacena BLTSIZE; ancho=2, alto=16
      move.w    (a0)+,d3           ; Obtiene la siguiente palabra
      bra       inloop             ; Va al bucle

again:
      move.w    (a0)+,d3           ; Coge otra palabra
      btst      d2,(a3)            ; Comprueba el indicador BLTDONE

wait2:
      btst      d2,(a3)            ; Comprueba de nuevo
      bne       wait2

inloop:
      move.w    d3,(a4)            ; Guarda nueva palabra para hacer vertical
      move.w    d1,(a5)            ; Inicia el "blit"
      subq.w    #1,d0              ; Es esta la última palabra?
      bne       again              ; Repite si no lo es

gone:
      rts
      end
```

CAPITULO 7

EL HARDWARE DE CONTROL DEL SISTEMA

INTRODUCCION

Este capitulo cubre el Hardware de control del sistema del Amiga, incluyendo los siguientes aspectos:

- Cómo se pueden especificar las prioridades de los Playfields relativas a los Sprites.
- Cómo se detectan las colisiones entre los objetos.
- Cómo se controla el acceso directo a memoria del sistema (DMA).
- Cómo se controlan y detectan las interrupciones.
- Cómo se controla el RESET y el POWERUP (encendido).

PRIORIDADES DE VIDEO

Se pueden controlar las prioridades de varios objetos en la pantalla para producir la ilusión de tres dimensiones. La siguiente sección muestra cómo se puede cambiar la prioridad de los playfields relativa a los Sprites.

PRIORIDADES FIJAS DE LOS SPRITES

No se puede cambiar las prioridades relativas de los Sprites. Siempre aparecen en la pantalla con los Sprites de número inferior al frente (teniendo mayor prioridad) de los sprites de número superior. Esto se muestra en la Figura 7-1. Cada cuadrado representa la imagen del sprite cuyo número se encuentra en el interior del cuadrado.

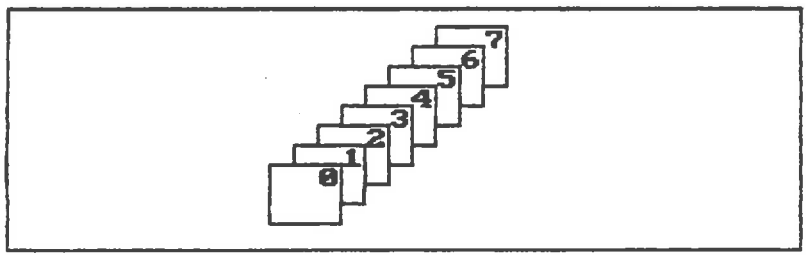


Figura 7-1: Prioridad de los Sprites

COMO SE AGRUPAN LOS SPRITES

Sólo para los propósitos de colisión y prioridad con los playfields, los Sprites se tratan como cuatro grupos de dos Sprites cada uno. Los grupos de Sprites son:

Sprites 0 y 1, Sprites 2 y 3, Sprites 4 y 5 y Sprites 6 y 7.

COMPRENDIENDO LAS PRIORIDADES DE VIDEO

El concepto de prioridades de video es facil de comprender si se imagina que cuatro dedos de una mano representan los cuatro pares de Sprites y dos de dos de la otra mano representan los dos Playfields. Igual que no se puede cambiar el orden de los cuatro dedos de una mano, tampoco se puede

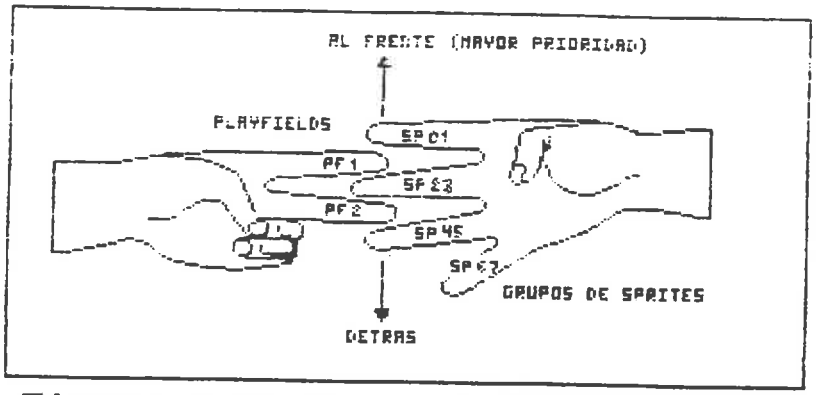


Figura 7-2: Prioridades de Video

cambiar la prioridad relativa de los Sprites. Sin embargo, igual que se pueden entrelazar los dedos de una mano de diferentes formas respecto a los dedos de la otra mano, así se pueden colocar los Playfields delante o detras de los Sprites.

Se pueden elegir cinco posiciones distintas para cada uno de los dos "dedos playfields." Por ejemplo, se puede poner el Playfield 1 sobre el grupo de sprites 1 (0), entre el grupo 1 y 2 (1), entre el grupo 2 y 3 (2), entre el grupo 3 y 4 (3) o debajo del grupo 4 (4). Hay las mismas posibilidades para el Playfield 2.

Los números del 0 al 4 encerrados entre parentesis en el parrafo anterior son los valores que se deben usar para seleccionar las posiciones de los playfields. Ver "Poniendo el Registro de Control de Prioridades" mas adelante.

Tambien se puede controlar la prioridad del Playfield 2 sobre el 1. Esto da mas posibilidades a la hora de diseñar las prioridades de la pantalla.

PONIENDO EL REGISTRO DE CONTROL DE PRIORIDADES

Este registro permite definir cómo unos objetos pasan por delante o detras de los otros objetos. Normalmente, el Playfield 1 aparece delante del Playfield 2. El bit PF2PRI invierte esta relación, haciendo al Playfield 2 mas importante. la prioridades de video se pueden controlar usando los bits del registro BPLCON2 (BitPlane Control Register number 2 - Registro de Control de Bit-planes número 2).

Tabla 7-1: Bits de BPLCON2

Bit	Nombre	Función
15-7		No se usan (Dejar a 0)
6	PF2PRI	Prioridad del Playfield 2
5-3	PF2P2 - PF2P0	(Prioridad del Playfield 2 sobre los Sprites)
2-0	PF1P2 - PF1P0	(Prioridad del Playfield 1 sobre los Sprites)

Los valores en binario que se dan a los bits PF1P2-PF1P0 determinan en que lugar de la cadena de prioridades (Tabla 7-2) se encuentra el Playfield 1. Esto coincide con la descripción dada en la sección anterior.

Nota: PF2P2 - PF2P0, bits 5-3, son los bits de prioridad para los playfields normales (no doble Playfield).

Tabla 7-2: Prioridad de los Playfields según los bits PF1P2-PF1P0

Bits	Prioridades (De mayor a menor)				
000	PF1	SP01	SP23	SP45	SP67
001	SP01	PF1	SP23	SP45	SP67
010	SP01	SP23	PF1	SP45	SP67
011	SP01	SP23	SP45	PF1	SP67
100	SP01	SP23	SP45	SP67	PF1

En esta tabla, PF1 es el Playfield 1, SP01 es el grupo de los Sprites 0 y 1, SP23 es el grupo de los Sprites 2 y 3, SP45 es el grupo de los Sprites 4 y 5, y SP67 es el grupo de los Sprites 6 y 7.

Los bits PF2P2-PF2P0 permiten colocar al Playfield entre las prioridades de los Sprites exactamente de la misma manera. Sin embargo, es el bit PF2PRI el que determina cual de los dos Playfields aparece delante del otro en la pantalla. Este es un ejemplo de contenidos posibles del registro BPLCON2 que originarian algo inusual:

BITS	15-7	PF2PRI	PF2P2-0	PF1P2-0
VALOR	0s	1	010	000

Esto daria como resultado la siguiente cadena de prioridades entre Playfields y Sprites:

PF1 SP01 SP23 PF2 SP45 SP67

En otras palabras, el Playfield 1 esta delante de los Sprites 0 y 1, y los Sprites 0 a 3 estan delante del Playfield 2. Sin embargo, el Playfield 2 esta delante del Playfield 1 en cualquier area donde se superpongan y donde el Playfield no este bloqueado por los Sprites del 0 al 3.

La Figura 7-3 muestra un uso de las prioridades Sprites/Playfields. El objeto Sprite mostrado en el diagrama es el Sprite 0. El Sprite puede desplazarse sobre el Playfield 2, pero cuando pase por el Playfield 1 el Sprite desaparecera detras del playfield. El resultado es un efecto de video inusual que provoca que el objeto desaparezca cuando cruza un limite invisible en la pantalla.

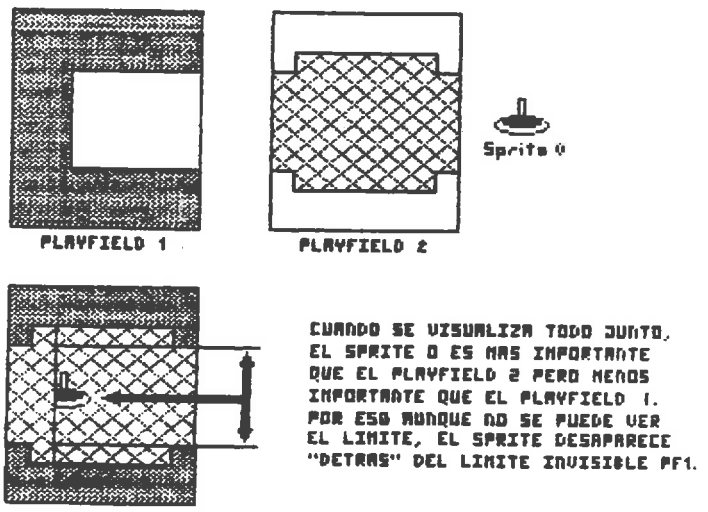


Figura 7-3: Prioridad Sprite/Playfield

DETECCION DE COLISIONES

Se puede usar el Hardware para detectar colisiones entre un grupo de Sprites y otro grupo de Sprites, cualquier grupo de Sprites y un Playfield, entre los dos Playfields, o cualquier combinación de estos elementos.

El primer tipo de colisión se usa tipicamente en juegos para determinar si un misil a chocado con un jugador móvil, por ejemplo. El segundo tipo de colisión se usa tipicamente para mover un objeto en unos límites especificos de la pantalla. El tercer tipo de colisión permite definir secciones de un Playfield como objetos individuales, que se pueden mover usando el Blitter. Esto se llama animación de Playfields. Si un Playfield esta definido como el escenario o area de juego y el otro Playfield se usa para definir los objetos (ademas de los Sprites), se pueden detectar las colisiones entre los objetos del Playfield y los Sprites o entre los objetos del Playfield y el otro Playfield.

COMO SE DETERMINAN LAS COLISIONES

La salida del video se forma cuando los datos procedentes de todos los bit-planes y los Sprites se combinan en una cadena común de datos para su visualización. Para cada uno de los pixels de la pantalla, se visualiza el color del objeto de mayor prioridad en esa posición. Las colisiones se detectan cuando dos o mas objetos intentan superponerse en la misma posición del pixel. Esto pone a 1 un bit en el registro de datos de las colisiones.

COMO INTERPRETAR LOS DATOS DE LAS COLISIONES

El registro de datos de las colisiones, CLXDAT, es de sólo-lectura, y sus contenidos se borran automaticamente (se ponen a 0) despues de leerlos.

Tabla 7-3: Bits de CLXDAT

Bit	Colisiones Registradas
15	sin usar
14	Sprite 4 (o 5) con Sprite 6 (o 7)
13	Sprite 2 (o 3) con Sprite 6 (o 7)
12	Sprite 2 (o 3) con Sprite 4 (o 5)
11	Sprite 0 (o 1) con Sprite 6 (o 7)
10	Sprite 0 (o 1) con Sprite 4 (o 5)
9	Sprite 0 (o 1) con Sprite 2 (o 3)
8	Bit-planes pares con Sprite 6 (o 7)
7	Bit-planes pares con Sprite 4 (o 5)
6	Bit-planes pares con Sprite 2 (o 3)
5	Bit-planes pares con Sprite 0 (o 1)
4	Bit-planes impares con Sprite 6 (o 7)
3	Bit-planes impares con Sprite 4 (o 5)
2	Bit-planes impares con Sprite 2 (o 3)
1	Bit-planes impares con Sprite 0 (o 1)
0	Bit-planes pares con Bit-planes impares

Nota: Los números entre parentesis de la Tabla 7-3 se refieren a las colisiones que se registraran sólo si se desea que se muestren. El registro de control de colisiones que se describe mas adelante permite ignorar o incluir los Sprites impares en la detección de colisión.

Nótese que en esta tabla, la detección de colisión NO cambia cuando se selecciona modo de doble o simple Playfield. La detección de colisión depende sólo de los bits presentes en los bit-planes pares e impares. El registro de control de colisión especifica cómo manejar los bit-planes durante la detección de colisión.

COMO SE CONTROLA LA DETECCION DE COLISION

El registro de control de colisión, CLXCON, contiene los bits que definen ciertas caracteristicas de la detección de colisión.

Tabla 7-4: Bits de CLXCON

Bit	Nombre	Función
15	ENSP7	Permite Sprite 7 (OR con el 6)
14	ENSP5	Permite Sprite 5 (OR con el 4)
13	ENSP3	Permite Sprite 3 (OR con el 2)
12	ENSP1	Permite Sprite 1 (OR con el 0)
11	ENBP6	Permite Bit-plane 6
10	ENBP5	Permite Bit-plane 5
9	ENBP4	Permite Bit-plane 4
8	ENBP3	Permite Bit-plane 3
7	ENBP2	Permite Bit-plane 2
6	ENBP1	Permite Bit-plane 1
5	MVBP6	Valor colisión de bit-plane 6
4	MVBP5	Valor colisión de bit-plane 5
3	MVBP4	Valor colisión de bit-plane 4
2	MVBP3	Valor colisión de bit-plane 3
1	MVBP2	Valor colisión de bit-plane 2
0	MVBP1	Valor colisión de bit-plane 1

Los bits 15-12 permiten especificar que en las colisiones con un par de Sprites se ha de incluir el Sprite impar. Los Sprites pares siempre estan incluidos en las detecciones de colisiones. Los bits 11-6 permiten especificar los bit-planes que se incluyen en la detección de colisión. Los bits 5-0 permiten especificar la polaridad (0 o 1) del bit que causara la colisión. Por ejemplo, se puede desear registrar las colisiones sólo cuando un objeto choque con "algo rojo" o "algo verde". Esta característica, junto con los bits de permisión de bit-plane, posibilitan especificar los bits exactos y su polaridad, para que se registre la colisión.

Nota: Este registro es de sólo escritura. Si se impiden todos los bit-planes (desconectan), se detectara siempre la colisión de bit-plane.

DETECCION DE COLISION DEL RAYO

A veces se necesita sincronizar el 68000 con el rayo de electrones que barre la pantalla. En algunos casos, es necesario actualizar una parte del bloque de memoria que contiene la imagen despues de que el sistema haya leido los datos de esa parte.

Las direcciones para el acceso al contador del rayo se han incluido para que se pueda averiguar el valor de los contadores del rayo y ejecutar ciertas operaciones basadas en la posición del rayo.

Nota: El Copper es capaz de seguir la posición del rayo y ejecutar operaciones basadas en los registros de los custom chips automaticamente. Consultar la sección "Interrupciones del Copper" y el Capitulo 2, "El Hardware del Coprocesador" para mas información.

Ademas, cuando se usa un lapiz óptico con el sistema, se usan las mismas direcciones para leer la posición del lapiz óptico en lugar de la del rayo. Esto se explica con mas detalles en el Capitulo 8, "El Hardware de Interface".

USANDO EL CONTADOR DE POSICION DEL RAYO

Hay cuatro direcciones para acceder a los contadores de posición del rayo.

Tabla 7-5: Contenidos del Contador de Posición del Rayo

VPOSR	Sólo-Lectura	Contiene el bit alto de la posición vertical (V8) y el bit de imagen larga.
	Bit 15	LOF (Long-Frame bit o bit de imagen larga). Se usa para inicializar imagenes del modo entrelazado.
	Bits 14-1	Sin usar
	Bit 0	Bit alto de la posición vertical (V8). Permite utilizar el tamaño del sistema PAL (313 lineas).
VHPOSR	Sólo-Lectura	Contiene el contador vertical y horizontal que representan la posición del rayo que genera la imagen (tambien sirve para el lapiz óptico).
	Bits 15-8	Bits inferiores de la posición vertical, bits V7-V0.
	bits 7-0	Posición horizontal, bits H8-H1. La resolución horizontal es de 1/160 del ancho de la pantalla (equivalente a dos puntos de baja resolución.)
VPOSW	Sólo-escritura	Bits igual que VPOSR.
VHPOSW	Sólo-escritura	Bits igual que VHPOSR. Usado para sincronización del contador con los patrones de prueba de los chips.

Como es normal, las direcciones de VPOSR VHPOSR y VPOSW, VHPOSW se pueden leer y escribir como palabras largas con las direcciones mas significativas siendo VPOSR y VPOSW.

INTERRUPCIONES

El sistema soporta el rango completo de interrupciones del 68000. Los distintos tipos de interrupciones que genera el hardware se introducen en el chip de perifericos y se traducen en seis de las siete interrupciones disponibles del 68000.

INTERRUPCION NO-ENMASCARABLE

El nivel de interrupciones 7 es la interrupción no-enmascarable y no se genera en ninguna parte del sistema actual. Las lineas directas de interrupción del 68000, IPL2 - IPL0, llegan hasta el conector de expansión y se pueden usar para generar esta interrupción de nivel 7 para aplicaciones de depuración de programas (aunque ocasiona un cortocircuito muy breve en dichas lineas.)

INTERRUPCIONES ENMASCARABLES

El sistema genera niveles de interrupciones del 1 a 6. Los registros de control del chip de periféricos permiten enmascarar algunas de estas fuentes de interrupción y evitar que generen una interrupción del 68000.

INTERFACE DEL USUARIO AL SISTEMA DE INTERRUPCIONES

El software del sistema ha sido diseñado para manejar correctamente todas las interrupciones del sistema Hardware de niveles 1-6. Un conjunto separado de líneas de entrada, designado como INT2 e INT6 (activas en estado bajo) se han enviado al conector de expansión para que el hardware externo pueda enviar interrupciones. Son conocidas como las interrupciones externas de niveles bajo (INT2) y alto (INT6).

Estas líneas de interrupción están conectadas al chip de periféricos y crean interrupciones de nivel 2 y 6 respectivamente. Es recomendable que se usen los handlers (manipuladores) de interrupciones del sistema operativo cuando se usan las interrupciones externas en lugar de generar las interrupciones directamente sobre las líneas del 68000.

REGISTROS DE CONTROL DE INTERRUPCIONES

Hay dos registros de interrupciones, conexión de interrupciones (mascara) y requerimiento de interrupciones (status). Cada registro tiene una dirección de lectura y otra de escritura.

INTENA — INTerrupt ENAble (conexión de interurpciones) - sólo escritura. Activa o desactiva bits específicos de INTENA.

INTENAR — INTerrupt ENAble Read - sólo lectura. Lee los contenidos de INTENA.

INTREQ — INTerrupt REQest (requerimiento de interrupciones) - sólo escritura. Lo usa el 68000 para obligar a que se procese un cierto tipo de interrupción (interrupción de software). Tambien se usa para borrar el indicador de requerimiento una vez se ha completado el procesamiento de la interrupción.

INTREQR — INTerrupt REQest Read - sólo lectura. Contiene los bits que definen las interrupciones que se estan procesando o estan en espera de ser procesadas.

Las posiciones de los bits de INTREQ son iguales a las de INTENA excepto en que el bit 15 no tiene significado en los registros de sólo lectura.

ACTIVANDO Y DESACTIVANDO BITS

Bit 15: SET/CLR

Los registros de interrupciones, igual que el registro de control de DMA, utilizan una forma especial de seleccionar los bits que se han de poner a 0 o a 1. El bit 15 de estos registros se llama bit de SET/CLR.

Cuando se quiere activar un bit (poner a 1 "SET"), se debe poner un 1 en la posición que se quiere activar y un 1 en la posición 15.

Cuando se quiere desactivar un bit (poner a 0 "CLR"), se debe poner un 1 en la posición que se quiere desactivar y un 0 en la posición 15.

Las posiciones 14-0 son selectores de bit. Se escribe un 1 en uno o mas bits para seleccionar ese bit. Y al mismo tiempo se escribe un 1 o un 0 en el bit 15 para determinar si los bits seleccionados han de activarse o desactivarse. Si se desea activar unos bits y desactivar otros, habra de escribirse este registro dos veces (una para activar unos bits y la otra para desactivar los otros bits.)

BIT 14: MASTER INTERRUPT ENABLE

El bit 14 de los registros de interrupciones (INTEN) actua sobre todas las interrupciones. Si este bit esta a 0, desconecta TODAS las interrupciones, si esta a 1 sólo conecta las que esten seleccionadas. Se suele usar para desconectar temporalmente las interrupciones durante un proceso critico.

Nota: Este bit sólo se usa en INTENA. En INTREQ no crea ningún requerimiento de interrupción.

BITS 13 y 3: INTERRUPCIONES EXTERNAS

Los bits 13 y 3 de los registros de interrupciones estan reservados para las interurpciones externas.

El Bit 13, EXTER, pasa a 1 cuando la linea del sistema INT6 pasa a 0 (se cierra a masa, -0 V). El Bit 13 genera una interrupción de nivel 6.

El Bit 3, PORTS, pasa a 1 cuando la linea del sistema INT2 pasa a 0. El bit 3 causa una interrupción de nivel 2.

BIT 5: INTERRUPCION DE VERTICAL BLANK

El Bit 5, VERTB, origina una interrupción en la linea 0 del FRAME (inicio del vertical blank). A veces el sistema necesita ejecutar diferentes tareas durante el intervalo de vertical blanking. Entre esas tareas estan la actualización de varios registros de direcciones (punteros), reescritura de listas del Copper (a veces), y otras operaciones de control del sistema.

El tiempo minimo del vertical blanking es de 20 lineas en NTSC y 25 en PAL. El rango es de la linea 0 a la 20 (NTSC) o a la 25 (PAL). Despues de este rango minimo ya se puede controlar dónde aparece la imagen usando el registro DIWSTRT (DIsplay Window STaRT) [illegible] na visualización) extender el tiempo efectivo del vertical [illegible] el Capitulo [illegible] Hardware de los Playfields" para mas inform[illegible] de DI[illegible]T.

Si resulta que se necesita mas tiempo durante el [illegible]cal blanking puede usar el Copper para crear una interrupció[illegible] ivel 3. E[illegible] interrupción del Copper se produciria justo despues de [illegible]ltima linea de la imagen (despues de la parada de la ventana de visualización usando el registro DIWSTOP.)

BIT 4: INTERRUPCION DEL COPPER

El Bit 4, COPER, lo usa el Copper para enviar una interrupción de nivel 3. El Copper puede cambiar el contenido de cualquiera de los bits de este registro, igual que puede escribir cualquier valor en la mayoria de los registros del sistema. Sin embargo, este bit se ha reservado especificamente para el Copper para asi identificar el origen de la interrupción.

Generalmente, se usa este bit cuando se quiere indicar que el rayo ha alcanzado una posición especifica en la pantalla, y se quiere cambiar algo en la memoria que este relacionado con este hecho.

BITS 10 - 7: INTERRUPCIONES DE AUDIO

Los Bits 10 - 7, AUD3-0, estan asignados a los canales de audio. Se llaman AUD3, AUD2, AUD1 y AUD0 y estan asignados a los canales 3, 2, 1 y 0, respectivamente.

Esta interrupción de nivel 4 indica "bloque de audio terminado." Cuando el DMA del audio esta es modo automatico, esta interrupción ocurre cuando se ha accedido a la última palabra de la cadena de datos del sonido. En el modo manual, ocurre cuando el registro de datos del audio esta preparado para aceptar otra palabra de datos.

Ver el Capitulo 5, "El Hardware del Audio" para mas información sobre la generación de interrupciones y la temporización.

BIT 6: INTERRUPCION DEL BLITTER

El Bit 6, BLIT, indica "Blit finalizado". Si este bit es un 1, indica que el Blitter ha completado la transferencia de datos que se le habia pedido. El Blitter esta entonces preparado para aceptar otra tarea. Este bit genera una interrupción de nivel 3.

BITS 12 y 1: INTERRUPCIONES DEL DISCO

El Bit 12, DSKSYN, indica que el registro de sincronia coincide con los datos del disco. Este bit genera una interrupción de nivel 5.

El Bit 1, DSKBLK, indica "bloque de disco finalizado." Se usa para indicar que la tarea de DMA que se habia especificado ya se ha completado. Este bit genera una interrupción de nivel 1.

BITS 11 y 0: INTERRUPCIONES DEL PORT SERIE

El Bit 11, RBF (Receive Buffer Full, buffer de recepción lleno) indica que el buffer de entrada del UART tiene datos listos para leer. Este bit genera una interrupción de nivel 5.

El Bit 0, TBE (Transmit Buffer Empty, buffer de transmisión vacio) indica que el buffer de salida del UART necesita mas datos que pueden ser escritos en ese momento en el buffer. Este bit genera una interurpción de nivel 1.

Prioridad de Hardware	Prioridad de Software en Exec	Descripción	Nombre
1	1	Interrupción de Software	SOFTINT
	2	Bloque de disco completo	DSKBLK
	3	Buffer de transmisión vacio	TBE
2	4	INT2 externo y CIA.A	PORTS
3	5	Coprocesador grafico	COPER
	6	Intervalo de Vertical Blank	VERTB
	7	Blitter finalizado	BLIT
4	8	Canal de Audio 2	AUD2
	9	Canal de Audio 0	AUD0
	10	Canal de Audio 3	AUD3
	11	Canal de Audio 1	AUD1
5	12	Buffer de recepción lleno	RBF
	13	Sincronia de disco hallada	DSKSYNC
6	14	INT6 externo y CIA.B	EXTER
	15	Especial (MASTER ENABLE)	INTEM
7	--	Interrupción no-enmascarable	NMI

Figura 7-4: Prioridades de Interrupción

CONTROL DEL DMA

Durante el funcionamiento del sistema se suceden muchos accesos directos a memoria diferentes (DMA). Hay una dirección de lectura al igual que una de escritura para acceder al registro del DMA para poder indicar que canales DMA deben estar conectados.

Los nombres de las direcciones del registro de DMA son los siguientes:

DMACONR - Direct Memory Access CONtrol Read - sólo lectura

DMACON - Direct Memory Access CONtrol - sólo escritura

Tabla 7-6: Contenido del registro de DMA

Bit	Nombre	Función
15	SET/CLR	Bit de control SET/CLR.
14	BBUSY	Stado de Blitter ocupado - sólo lectura.
13	BZERO	Estado ZERO del Blitter - sólo lectura. Permanece a 1 si durante una operación del Blitter, la salida de este SIEMPRE fue 0.
12-11		Sin asignar
10	BLTPRI	Prioridad del Blitter. Tambien llamado "Blitter desagradable." Cuando esta a 1 tiene prioridad total sobre el 68000. Cuando esta a 0 tiene prioridad parcial.
9	DMAEN	Conexión de DMA. Es el bit principal de DMA. Conecta el DMA para todos los canales seleccionados en los bits 8-0.
8	BPLEN	Conecta el DMA de los Bit-planes.
7	COPEN	Conecta el DMA del Copper.
6	BLTEN	Conecta el DMA del Blitter.
5	SPREN	Conecta el DMA de los Sprites.
4	DSKEN	Conecta el DMA del disco.
3-0	AUDxEN	Conecta el DMA de los canales de audio 3-0.

Para mas información sobre el uso del DMA, ver los siguientes capitulos:

Copper	Capitulo 2	"El Hardware del Coprocesador"
BitPlanes	Capitulo 3	"El Hardware de los Bit-planes"
Sprites	Capitulo 4	"El Hardware de los Sprites"
Audio	Capitulo 5	"El Hardware del Audio"
Blitter	Capitulo 6	"El Hardware del Blitter"
Disco	Capitulo 8	"El Hardware de Interface"

ACCESO DEL MICROPROCESADOR A MEMORIA CHIP

Los chips del Amiga acceden a la memoria CHIP directamente, en lugar de utilizar los mecanismos tradicionales de distribución de bus. Por lo tanto las características añadidas para sistemas de multiprocesadores, como la instrucción del 68000 TAS (comprobar y activar), no puede cumplir su propósito y no es soportada por la arquitectura del Amiga.

OPERACION DE RESET Y POSTERIOR ENCENDIDO

Cuando se enciende el Amiga o se "resetea" externamente, el mapa de memoria se encuentra en un estado especial. Se produce un overlay de la ROM sobre la posición $00000000 del mapa de memoria. Y por tanto no se puede acceder a la RAM del sistema que esta normalmente en esta posición. En algunos modelos del Amiga, algunas porciones de la RAM continúan funcionando. En los demas modelos no funciona ningún tipo de RAM. El software debe asumir que no hay ninguna RAM disponible. El bit OVL de uno de los chip 8520 desconecta el overlay (Ver en el Apendice F la posición del bit).

La ROM del Sistema Operativo del Amiga contiene un código ID como primera palabra (código identificativo). El valor del código ID puede cambiar en un futuro. La segunda palabra de la ROM contiene una instrucción JMP ($4ef9). Las siguientes dos palabras se usan como PC inicial para el 68000.

La instrucción "RESET" del 68000 trabaja de forma muy parecida al reset externo o el encendido del ordenador. Toda la memoria y las tarjetas AUTOCONFIG desaparecen, y la imagen de la ROM aparece en la posición $00000000. La diferencia es que el 68000 continúa en funcionamiento con la siguiente instrucción. Debido a que no hay RAM disponible, se necesita un cuidado especial al escribir el código de "reseteado" del ordenador, que debe ser capaz de hacer responder a todos los modelos de Amiga. Este es el listado de asembler del UNICO código de "reseteado" que soporta el Amiga.

```
; ----- El *único* código de "reseteado"
      CNOP    0,4       ; IMPORTANTE: Deben ser posiciones de palabra larga
MagicResetCode:
      lea.l   2,a0      ; Apunta a la instrucción JMP del inicio de la ROM
      RESET             ; Toda la RAM desaparece ahora!
      jmp     (a0)      ; Se confia la ejecución de esta instrucción en el
                        prefetch (preobtención) que realiza el 68000.
```

La instrucción RESET es una instrucción privilegiada y por tanto se debe ejecutar en el nivel SUPERVISOR del 68000. Si se esta trabajando con el Sistema Operativo Exec, se debe usar el siguiente código:

```
_ColdReboot:
      move.l  4,a6                    ; Obtiene el puntero de ExecBase
      lea.l   MagicResetCode(pc),a5   ; Posición del código
      jsr     _LVOSupervisor(a6)      ; Inicia el código (debe ser JSR)
```

CAPITULO 8

EL HARDWARE DE INTERFACE

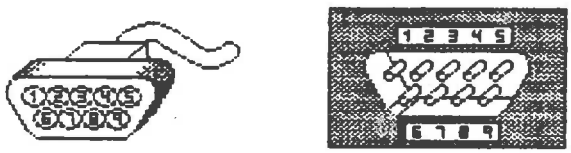
INTRODUCCION

Este capitulo cubre el Hardware de Interface, a traves del cual el Amiga se comunica con el mundo exterior. Incluye las siguientes caracteristicas:

- Interface con dos ports para controladores de propósito múltiple para ratón, joystick digital, lapiz óptico, trackballs joystick analógico, etc.
- Controlador de disco (para drives de floppy disk y otros dispositivos MFM y GCR)
- Teclado.
- Interface con port paralelo compatible Centronics para impresoras, digitalizadores, etc.
- Interface con port serie compatible RS232C para modems, impresoras y otros dispositivos de comunicación en serie.
- Conectores de salida de imagen para monitores RGB, monitores con Euroconector, monitores de video compuesto monocromo, modulador de RF (para televisores) y genlocks.

INTERFACE CON DOS PORTS PARA JOYSTICKS

Cada Amiga tiene dos conectores de nueve pines (patitas) que pueden usarse para salida o entrada de muchos tipos diferentes de controladores. La figura muestra uno de los dos conectores y la visión frontal correspondiente al enchufe tipico de joystick.



VISTA FRONTAL DEL ENCHUFE DEL JOYSTICK

VISTA FRONTAL DEL CONECTOR DEL AMIGA

Figura 8-1: Ports de 9 Pines

Tabla 8-1: Conexiones Tipicas de los Ports

Pin	Joystick	Ratón o Trackball	Joystick Proporcional	Joystick X-Y proporcional	Lapiz óptico
1	Arriba	Impulso V	---	Botón 3 *	---
2	Abajo	Impulso H	---	---	---
3	Izquierda	Impulso VQ	Botón izquierdo	Botón 1	---
4	Derecha	Impulso VH	Botón derecho	Botón 2	---
5$	---	Botón Medio *	POT derecho	POT X	Botón punta
6$	Botón 1	Botón izquierdo	---	---	Foto-diodo
7	---	+5V	+5V	+5V	+5V
8	GND	GND	GND	GND	GND
9$	Botón 2 *	Botón derecho	POT izquierdo	POT Y	Botón 2 *

$-Estos pines se pueden configurar tambien como salidas
* Estos botones son opcionales.

REGISTROS USADOS CON LOS PORTS

JOY0DAT	($DFF00A)	Contador para entrada digital (ratón) del port 1.
JOY1DAT	($DFF00C)	Contador para entrada digital (ratón) del port 2.
CIAAPRA	($BFE001)	I/O para el pin 6 (botón disparo de ambos ports)
POT0DAT	($DFF012)	Contador para entrada proporcional del port 1.
POT1DAT	($DFF014)	Contador para entrada proporcional del port 2.
POTGO	($DFF034)	Salida pines proporcionales, inicio de contadores.
POTGOR	($DFF016)	Entrada de los pines proporcionales.
BPLCON0	($DFF100)	El bit 3 conecta el modo de lapiz óptico.
VHPOSR	($DFF006)	Entrada posición lapiz óptico (bit superior).
VPOSR	($DFF004)	Entrada posición lapiz óptico (bits inferiores).

LEYENDO RATONES Y TRACKBALLS

Los impulsos que van desde el ratón al Amiga se convierten en dos contadores separados, uno vertical y el otro horizontal. Los registros de 8 bits horizontales y verticales pueden registrar el movimiento del ratón sin necesitad de que intervenga el 68000.

El ratón usa señales de cuadratura. Para cada dirección, una rueda mecanica que se encuentra dentro del ratón produce dos trenes de impulsos, estando uno desfasado 90 grados respecto al otro (ver Figura 8-2). Esta relación de fase es la que determina la dirección.

Los contadores se incrementan cuando el ratón se mueve hacia la derecha o hacia abajo, y se decrementan cuando el ratón se mueve hacia la izquierda o hacia arriba.

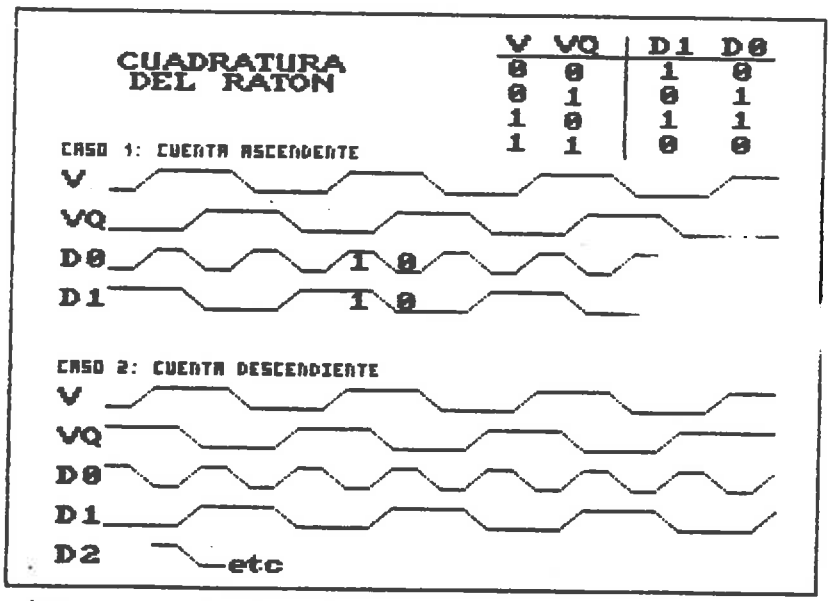


Figura 8-2: Cuadratura del Ratón

Leyendo los contadores.

Se puede acceder al contenido de los contadores de ratón/trackball leyendo las direcciones de los registros JOY0DAT y JOY1DAT. Estos contienen los contadores de los ports 1 y 2 respectivamente (bits 15-8 -> contador vertical, bits 7-0 -> contador horizontal).

Limitaciones de los contadores.

Estos contadores pueden "dar la vuelta" en ambas direcciones. Si se desea usar el ratón para controlar algo que esta sucediendo en la pantalla, se deberan leer los contadores como mínimo una vez en cada periodo de vertical blanking y guardar el contenido anterior de los registros. Entonces se puede restar de la lectura anterior para determinar la dirección y la velocidad del movimiento.

El ratón produce unos 200 impulsos en un movimiento de una pulgada (2,54 cm) en cualquier dirección. El vertical blank sucede cada 1/50 de segundo (2 centesimas). Si se lee el ratón una vez en cada vertical blank, la diferencia entre el valor actual de los contadores y el anterior sera menor a 127. Sólo si el usuario mueve el ratón a mas de 38 pulgadas por segundo (casi un metro por segundo) el resultado de la resta estara equivocado. Los juegos de acción rapida pueden necesitar leer el registro del ratón dos veces por FRAME para evitar estos errores.

Si se resta el valor actual del valor anterior, el valor absoluto de la diferencia representara la velocidad. El signo de la diferencia (positivo o negativo) permitira conocer la dirección del movimiento del ratón.

La forma mas sencilla de calcular la velocidad del ratón es con arimetica de 8 bits con signo. El nuevo valor de un contador menos el valor anterior representara el desplazamiento efectuado por el ratón desde la última comprobación. El ejemplo de la Tabla 8-2 presenta un metodo alternativo. Se tratan a ambos contadores como valores sin signo de rango 0-255. Se miden 100 pulsos en cada caso.

Tabla 8-2: Determinando la dirección del Ratón

Valor Anterior	Valor Actual	Dirección
200	100	Arriba (izquierda)
100	200	Abajo (derecha)
200	45	Abajo *
45	200	Arriba **

* Debido a que 200-45 = 155, que es mayor a 127, el resultado correcto seria 255 - (200 - 45) = 100; la dirección es hacia abajo.

* 45-200 = -155. Debido a que el valor absoluto de -155 supera a 127, el resultado correcto seria 255 + (-155) = 100; la dirección es hacia arriba.

Botones del Ratón.

Hay dos botones en el ratón standard del Amiga. Sin embargo, la circuiteria de control y el software soportan hasta tres botones.

- El botón izquierdo del ratón del Amiga esta conectado a, CIAAPRA ($BFE001). El botón del port 1 esta conectado al bit 6, y el del port 2 esta conectado al bit 7. Ver el Apendice de los 8520 para mas información. Un estado lógico 1 significa "interruptor abierto". Un estado lógico 0 significa "interruptor cerrado" (botón pulsado).
- El botón 2 (el derecho en el ratón del Amiga) esta conectado al pin 9 de los ports, uno de los pines proporcionañes. Ver los detalles en la sección "ENTRADA/SALIDA DIGITAL EN LOS PORTS".
- El botón 3, cuando se usa, esta conectado al pin 5, que es el otro pin proporcional.

LEYENDO JOYSTICKS DIGITALES

Los joysticks digitales (son los mas comunes) contienen cuatro interruptores direccionales. Cada interruptor se activa individualmente por la acción del palo de mando. Cuando se acciona el palo diagonalmente, se activan dos interruptores adyacentes. El número total de direcciones posibles en un joystick digital es de **8**. Todos los joysticks tienen como mínimo un botón de disparo.

Los interruptores de los joysticks digitales son del tipo "normalmente abierto" (normalmente desconectados). Cuando se pulsa el interruptor, la línea de entrada se cierra a masa, -0V. Al leer el estado de un interruptor abierto (sin pulsar) da un 1, y un interruptor cerrado (pulsado) da un 0.

Leer los estados lógicos de entrada del joystick no es tan sencillo, debido a que los registros de datos de los joysticks son los mismos que los contadores que se usan para el ratón/trackball. Los registros del joystick son JOY0DAT y JOY1DAT.

La Tabla 8-3 muestra como interpretar los datos una vez se han leido de los registros. El estado lógico verdadero de los datos del interruptor es, en estos registros, "1 -> interruptor pulsado y 0 -> interruptor sin pulsar".

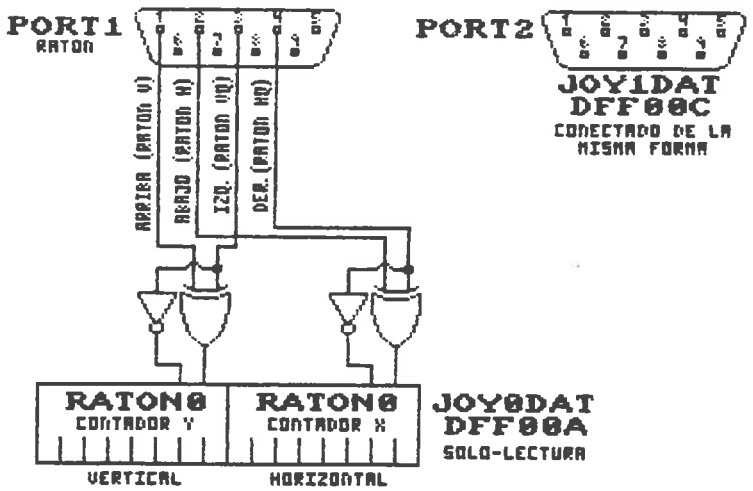


Figura 8-3: El joystick y los contadores

Tabla 8-3: Interpretando los Datos de JOY0DAT y JOY1DAT.

Bit de datos	Interpretación
1	Estado lógico verdadero del interruptor derecho.
9	Estado lógico verdadero del interruptor izquierdo.
1 (XOR) 0	Se debe calcular el OR-eXclusivo de los bits 1 y 0 para obtener el estado lógico del interruptor de abajo.
9 (XOR) 8	Se debe calcular el OR-eXclusivo de los bits 9 y 8 para obtener el estado lógico del interruptor de arriba.

Los botones de disparo de los ports 0 y 1 estan conectados a los bits 6 y 7 de CIAAPRA ($BFE001). Un 0 aquí indica que el interruptor esta pulsado.

Algunos joysticks, pero no todos, tienen un segundo botón. Sólo se aconseja usar este botón si la función que realiza esta duplicada por el teclado u otro mecanismo. este botón se puede leer de la misma forma que el botón derecho del ratón.

LEYENDO JOYSTICKS PROPORCIONALES

Cada uno de los ports puede manejar dos dispositivos de entrada de resistencia variable, tambien llamados dispositivos proporcionales de entrada. Esta sección describe cómo se puede averigar la posición de los dispositivos proporcionales. hay dos tipos comunes de controladores proporcionales: el par de control "paddle" y el joystick X-Y proporcional. Un par de control "paddle" consiste en dos compartimentos separados cada uno de ellos con un potenciomentro y un botón de disparo y conectados ambos al mismo port.

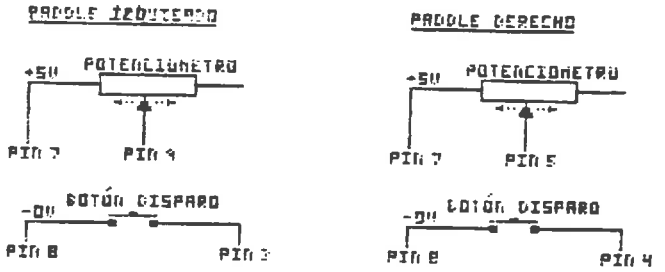


Figura 8-4: Esquema tipico de un Paddle

En un joystick X-Y proporcional, los potenciómentros estan conectados individualmente los ejes X e Y del mismo palo de control.

Leyendo los botones del joystick proporcional.

En los controladores de Paddle, las lineas normales de izquierda y derecha sirven como botones de disparo para los paddles izquierdo y derecho.

Interpretando la Posición del joystick proporcional.

Para interpretar la posición de un joystick proporcional se necesitan realizar unas operaciones en el periodo del vertical blanking.

Durante el vertical blank, se escribe un valor en la dirección llamada POTGO, En un joystick X-Y, este valor es $0001. Al escribir en este registro se pone en marcha un hardware especial que lee el valor del potenciometro y pone los valores de los registros POT a 0.

El circuito de lectura queda en un estado de reset durante siete u ocho lineas de video. Despues del intervalo de reset, el circuito deja que empiece a cargarse un condensador cuya velocidad de carga dependera de la posición del potenciometro externo (T = C * R). Durante cada linea horizontal posterior, el circuito compara la carga del condensador con un valor fijo. Si la carga es inferior a dicho valor, el contador POT se incrementa. Si la carga es superior al valor fijo, el contador se bloquea hasta que se vuelva a escribir en POTGO.

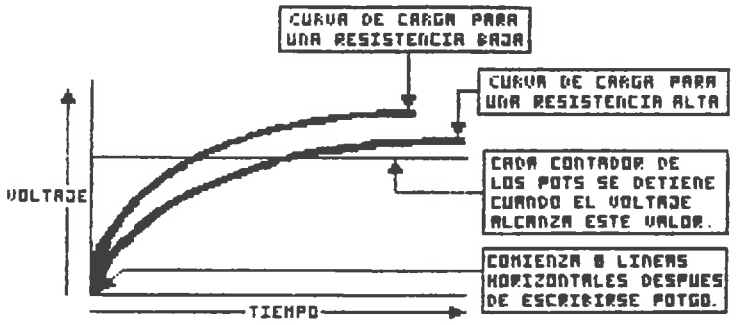


FIGURA 8-5: Efectos de la resistencia en la carga.

POTGO se escribe normalmente al principio del vertical blank, y despues se leen los valores de los registros POT en el siguiente vertical blank, justo antes de volver a escribir en POTGO.

No hay nada en el sistema que evite que los contadores lleguen al tope (despues de llegar a 255 volveran a empezar desde 0). Sin embargo el sistema esta diseñado para asegurar que el contador no llegara al tope en el tiempo que dura un FRAME. Esto permite saber si una el controlador indica una sobrecarga.

Registros de los Joysticks proporcionales.

Los joysticks proporcionales usan los siguientes registros:

POTØDAT - datos del port 1 (vertical/horizontal)
POT1DAT - datos del port 2 (vertical/horizontal)

Posiciones de los bits:

Bits 15-8 valor vertical, POT0Y o POT1Y
Bits 7-0 valor horizontal, POT0X o POT1X

Todos los contadores se resetean a 0 cuando se escribe un 1 en el bit 0 de POTGO. Los contadores se leen normalmente un FRAME despues de que se haya conectado el circuito proporcional.

Especificaciones de los potenciometros

La resistencia de los potenciometros debe ser de incremento lineal. Basandonos en el diseño del convertidor analógico-digital que se usa en el Amiga, la resistencia maxima no debe ser superior a 528 K (528000 ohmnios), se aconseja 470 K ±10% para cada uno de los pots X e Y. Este circuito se basa en la carga de un condensador de 0.047 µf ±10%, y un tiempo maximo de 16.6 ms para cargarse al maximo (+5V), es decir un FRAME de tiempo.

Todos los potenciomentros presentan una cierta cantidad de imprecisión. Para obtener resultados aceptables en todas las configuraciones, es necesario que se tomen varias muestras y se saque la media de todas ellas.

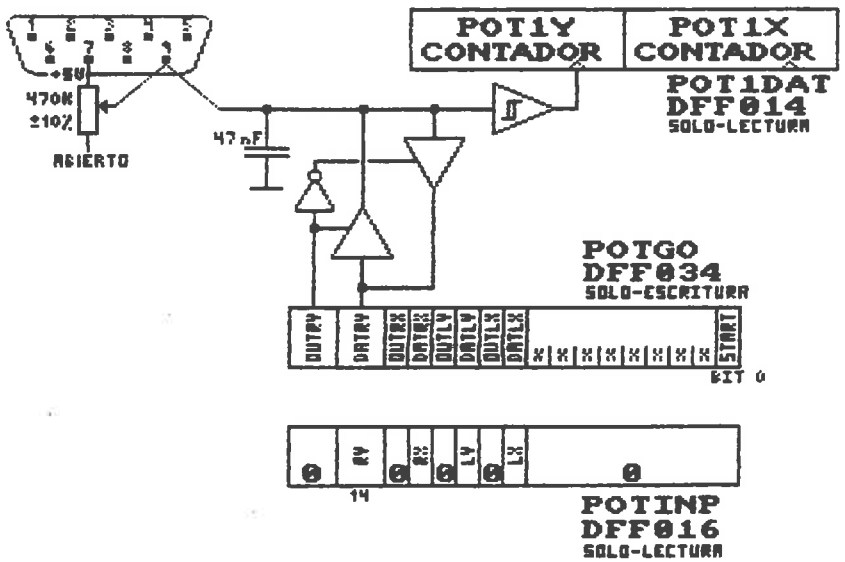


Figura 8-6: Circuito de carga del potenciometro.

LEYENDO UN LAPIZ OPTICO

Se puede conectar una lapiz óptico en el port 2 (en el Amiga 1000 se debe poner en el port 1). Aunque de todas formas se puede cambiar el port que alojara al lapiz óptico con unas modificaciones en el hardware interno (un pequeño "jumper" o puente) pero de todas formas, independientemente del port que se este usando, el diseño del lapiz óptico es el mismo.

La señal llamada "lapiz pulsa pantalla" se genera por un pequeño interruptor que se encuentra en la punta del lapiz. Si este interruptor se conecta a una de las entradas proporcionales se debera leer igual que los botones derecho y central de un ratón.

Los principios del funcionamiento del lapiz óptico son los siguientes:

1. Justo cuando el sistema sale del vertical blank, el circuito de captura para el lapiz óptico se conecta automaticamente.
2. El rayo de electrones comienza a crear la imagen, barriendo la pantalla en lineas horizontales de izquierda a derecha para generar la imagen de arriba a abajo.
3. Los sensores del lapiz óptico (foto-diodos o celulas foto-electricas) detectan un impulso de luz cuando el rayo pasa por delante de ellos. El lapiz óptico convierte el impulso de luz en un impulso electrico que envia al pin 6.
4. El impulso electrico hace que el circuito interno capture y guarde el contenido del registro de la posición del rayo, VPOSR. Esto permite saber dónde se encuentra el lapiz óptico leyendo los valores exactos (horizontal y vertical) del contador de posición del rayo en el instante en que este coincidió con el lapiz óptico.

Leyendo los registros del lapiz óptico

Los registros del lapiz óptico tienen las mismas direcciones que los contadores del rayo. Los bits son los siguientes.

VPOSR	Bit 15	Imagen larga o corta (0 = corta)
	Bits 14-1	Código ID del chip. No hacer caso de estos bits.
	Bit 0	V8 (bit superior de la posición vertical)
VHPOSR	Bits 15-8	V7-V0 (posición vertical)
	Bits 7-0	H8-H1 (posición horizontal)

El 68000 puede usar este registro como palabra larga de dirección VPOSR.

La resolución de posiciones en este registro es la siguiente:

Vertical 1 linea horizontal en modo no-entrelazado.
2 lineas horizontales en modo entrelazado (sin embargo si se conoce cual de las dos imagenes entrelazadas es la que ha originado el impulso en el lapiz óptico, se puede averiguar la posición exacta.)

Horizontal 2 pixels de baja resolución, 4 de alta resolución.

La calidad del lapiz óptico determinara la cantidad de imprecisión. En muchas aplicaciones sera necesario tomar varias lecturas de la misma posición y hacer la media arimetica.

Para permitir la entrada del lapiz óptico, se debe escribir un uno en el bit 3 de BPLCON0. Una vez esta permitida la entrada del lapiz optico y el lapiz envia un impulso del foto-diodo, el valor de VPOSR se bloquea. Si no se detecta ningún impulso, los contadores se detendran cuando lleguen al final de la imagen. Es imposible leer la posición real del rayo una vez se bloquean los contadores. Este bloqueo se elimina al final del vertical Blank (linea 20). No hay ningún bit en el sistema que indique que se ha recibido el impulso. Para determinar si ha ocurrido algún impulso, se usa uno de estos metodos:

1. Leer VPOSR dos veces (como palabra larga).

2. Si ambos valores son distintos, el lapiz óptico no ha enviado ningún impulso desde el último vertical blank.

3. Si ambos valores son iguales, han de eliminarse (AND) los 15 bits superiores de los 32 existentes y compararlo con $10500 (linea 261).

4. Si el valor de VPOSR es mayor que $10500, es que no se ha detectado ningún impulso en este FRAME. Si el valor es menor, es que el lapiz óptico ha enviado un impulso y el valor que se a leido es la posición donde ha ocurrido ese impulso en la pantalla.

Otro metodo mas simplificado de determinar la veracidad del valor del lapiz óptico es hacer que el sistem lea el registro VPOSR sólo durante el vertical blank (de la linea 0 a la 20):

1. Leer VPOSR durante el periodo del VBLANK (como palabra larga).

2. Enmascarar los 15 bits superiores y compararlo con $10500 (linea 261).

3. Si el valor de VPOSR es mayor que $10500, es que no se ha detectado ningún impulso en este FRAME. Si el valor es menor, corresponde a la posición donde esta el lapiz óptico.

Nótese que cuando el modo de lapiz óptico esta conectado, el registro VPOSR se puede bloquear en cualquier momento, y no se pueden reemplazar como contadores. Esto puede causar problemas con los programas que utilizen estos contadores para temporizar sus procesos internos.

ENTRADA/SALIDA DIGITAL EN LOS PORTS

El Amiga puede leer e interpretar muchos tipos de controladores. El registro POTGO puede redefinir las funciones de algunos pines de los ports. El registro de control de los pots controla un port bidireccional de cuatro bits que comparte los mismos pines con las entradas proporcionales.

Tabla 8-4: Registros POTGO ($DFF034) y POTINP ($DFF016)

Bit	Nombre	Función
15	OUTRY	Conexión de salida para bit 14 (1 = si)
14	DATRY	Dato para el port 2, pin 9.
13	OUTRX	Conexión de salida para bit 12
12	DATRX	Dato para el port 2, pin 5.
11	OUTLY	Conexión de salida para bit 10
10	DATLY	Dato para el port 1, pin 9. (botón derecho)
09	OUTLX	Conexión de salida para bit 8
08	DATLX	Dato para el port 1, pin 5. (botón central)
07-01	X	Código ID del chip.
00	START	Resetea pots (descarga condensadores, contadores a 0)

En lugar de usar los pines de los pots como entradas de potenciometros, se pueden usar como un port de entrada/salida de 4 bits. Este proporciona dos pines adicionales en cada uno de los ports de joystick de proposito general entrada/salida.

Si se pone a 1 la conexion de salida en algun pin, el Amiga desconecta el circuito de control de potenciometros de ese port, y configura el pin como salida. El estado del bit de datos controla el nivel logico en el pin de salida. Este registro se escribirá a través de la dirección POTGO y se leera a traves de POTINP. Como hay condensadores en estas lineas, pueden tardar hasta 300 µs en cambiar de un estado a otro.

Para usar todo el registro como entrada, detectando el estado actual de los pines, se ponen todos los bits de la dirección POTGO a Ø. Despues se podra leer el estado de los pines usando la dirección de sólo-lectura POTINP. Nótese que los bits usados como entradas estaran conectados a los contadores proporcionales (ver la descripción del bit START de POTGO).

Estas lineas tambien se pueden usar como botones de disparo. Un botón es un interruptor normalmente abierto que cierra a masa cuando se pulsa. El amiga debe tener una resistencia (polariza a +5V) en el pin sensitivo. Para hacer esto, se debe configurar el pin como salida, y conducir la linea al estado alto (poniendo OUTxx y DATxx a 1). Al leer POTINP se observara un Ø si el botón esta pulsado, y un 1 si no lo esta.

Los botones de disparo del joystick tambien se pueden configurar como salidas. CIAADDRA ($BFE201) contiene una mascara que corresponde uno por uno con los bits del registro de datos, CIAAPRA ($BFE001). Poniendo un 1 en la posición de la dirección se convertira al bit correspondiente en una salida. Ver el Apendice del 8520 para mas detalles.

CONTROLADOR DE FLOPPY DISK

El controlador de disco incluido en el sistema puede manipular hasta cuatro dispositivos MFM. Tipicamente son unidades de doble cara, doble densidad, 3.5" (90mm) o 5.25". Hay una unidad de 3.5" en la configuración basica.

El controlador es extremadamente flexible. Puede copiar a traves de DMA una pista completa de datos MFM en la memoria en una sola vuelta del disco. Los registros especiales permiten al 68000 sincronizarse con datos especificos, o leer un solo byte. El controlador puede leer y escribir cualquier disco codificado MFM de doble densidad, incluyendo el formato del Amiga V1.0, IBM PC (MS_DOS) 5.25", IBM PC (MS_DOS) 3.5", ATARI ST (TOS) 3.5", y muchos discosformateados en CP/M (con programas especiales). El controlador puede leer y escribir la mayoria de los discos usando el metodo GCR (Group Coded Recording), incluyendo discos de Apple II. Con trampas en la velocidad del motor, el controlador puede leer y escribir discos formateados de Commodore 1541/1571.

REGISTROS USADOS EN EL SUBSISTEMA DEL DISCO

El subsistema del disco usa dos ports de los chips 8520, y muchos registros del custom chip Paula:

CIAAPRA	($BFE001)	Entrada 4 bits de control del disco
CIABPRB	($BFD100)	Salida 8 bits de selección de disco, control, etc
ADKCON	($DFF09E)	Bits de control (sólo-escritura)
ADKCONR	($DFF010)	Bits de control (sólo-lectura)
DSKPTH	($DFF020)	Puntero DMA (32 bits)
DSKLEN	($DFF024)	Longitud del DMA
DSKBYTR	($DFF01A)	Byte del disco y lectura del status.
DSKSYNC	($DFF07E)	Buscador de sincronia, retiene una palabra

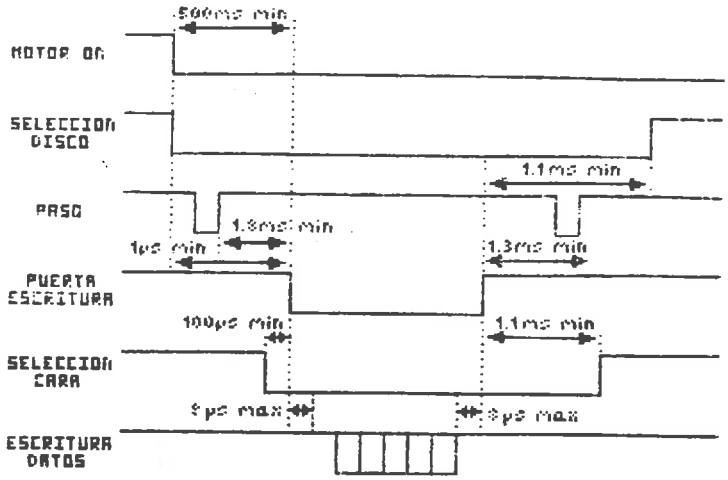


Figura 8-7: Temporización en el Disco.

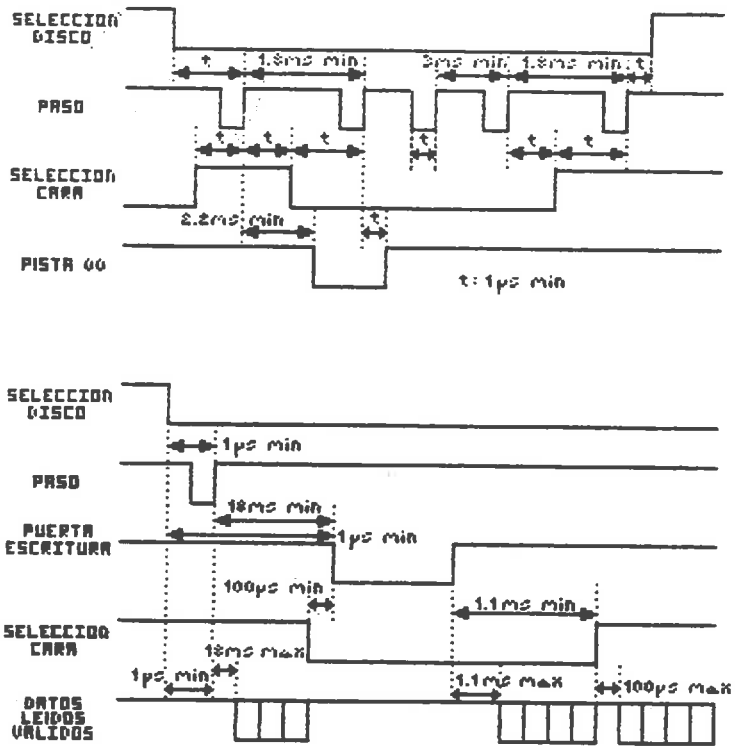


Figura 8-8: Temporización en el Disco (cont.)

CIAAPRA/CIABPRB - Selección de Disco, control y detección.

La siguiente tabla muestra cómo se usan algunos bits de los chips 8520 para el subsistema del disco. Los bits llamados "PA" son bits de entrada y se encuentran en CIAAPRA ($BFE001). Los bits llamados "PB" son bits de salida y estan en CIAPRB ($BFD100). Hay mas información sobre como funcionan los chips 8520 en el Apendice F.

Tabla 8-5: El Subsistema del Disco

Bit	Nombre	Función
PA5	DSKRDY*	Disco preparado. La unidad de disco pondra a 0 esta linea cuando detecte que el motor esta funcionando a máxima velocidad. La señal sólo es valida cuando el motor esta encendido, en otras ocasiones la información sobre esta entrada puede estar equivocada.
PA4	DSKTRACK0*	Detección de pista 0. La unidad de disco pondra esta linea a 0 cuando las cabezas del disco llegan a la pista 0. El Software no debe intentar mover la cabeza mas hacia afuera cuando se activa esta señal. Algunas unidades rechazaran el movimiento, otras lo intentaran realizar, posiblemente causando daños en el alineamiento del cabezal. Todas las nuevas unidades de disco deben rechazar el movimiento hacia afuera en esta posición.
PA3	DSKPROT*	El disco esta protegido contra escritura.
PA2	DSKCHANGE*	El disco ha sido EXTRAIDO de la unidad. La señal se hace 0 cuando se extrae el disco. Continua a 0 hasta que se inserta otro Y se recibe un impulso de movimiento del cabezal.

PB7	DSKMOTOR*	Control del motor del disco. Esta señal no es standard en el sistema del Amiga. Cada unidad de disco memoriza la señal del motor en el momento en que se conecta la señal de selección de unidad. El motor de la unidad continuara en este estado hasta que se vuelva a seleccionar el disco. DSKMOTOR* tambien controla el led que esta delante de la unidad de disco. En los programas que se seleccionan las unidades de disco se debe activar la señal del motor ANTES de seleccionar la unidad. La unidad recordara el estado de su motor cuando no este seleccionada. Todos los motores se desconectan despues de un reset del sistema. Despues de conectar el motor, el software debera esperar medio segundo (500 ms), o a la señal DSKRDY* antes de que el motor alcance la velocidad adecuada.
PB6	DSKSEL3*	Selecciona la unidad de discos 3.
PB5	DSKSEL2*	Selecciona la unidad de discos 2.
PB4	DSKSEL1*	Selecciona la unidad de discos 1.
PB3	DSKSEL0*	Selecciona la unidad de discos 0 (interna).
PB2	DSKSIDE	Especifica que cabezal se va a utilizar (0 -> cabezal superior). DSKSIDE debe ser estable 100 µs antes de escribir. Despues de escribir, deben pasar como minimo 1.3 ms antes de cambiar de cabezal.

CD1 DSKDIREC Especifica la dirección para desplazar los cabezales de la unidad (Ø → hacia el centro del disco). La pista Ø es la pista mas externa del disco. La linea se debe activar ANTES del impulso de paso (STEP), escribiendo por separado al registro.

CB0 DSKSTEP* Mueve los cabezales del disco un PASO. La señal se debe usar siempre para provocar un impulso rapido (nivel 1, momentaneamente a nivel Ø y rapidamente a nivel 1).

Las unidades de discos usadas en el Amiga garantizan llegar a la siguiente posición en 3 ms. Algunas unidades soportan velocidades mucho mayores, otras no. NO se pueden utilizar bucles de decremento de un registro para calcular los intervalos de tiempo. En el Apendice F hay una solución mejor.

Cuando se invierten las direcciones, se necesita esperar un mínimo de 18 ms antes de mover de nuevo el cabezal. El tiempo fijado para las unidades del Amiga son 15 ms.

FLAG DSKINDEX* Impulso del indice del disco (bit 4 de $BFDD00). Se puede usar para crear una interrupción de nivel 6. Ver el Apendice F para mas detalles.

* Esta señal esta activa cuando se pone a nivel lógico Ø (-ØV)

Control del canal DMA del Disco.

Los datos se transfieren normalmente al disco por acceso directo a memoria (DMA). El DMA del disco se controla a traves de 4 elementos:

- Un registro apuntando al area en la cual o desde la cual se van a mover los datos.
- Longitud del bloque de datos que movera el DMA.
- Dirección de la transferencia de datos (leer/escribir).
- Conexión del DMA.

DSKPTH - Puntero a los datos.

Se debe especificar la dirección de 32 bits en la cual o desde la cual se van a mover los datos. El bit menos significativo de la dirección debe ser Ø, y el buffer debe estar en memoria CHIP. El valor se debe escribir como una sola palabra larga en el registro DSKPTH ($DFF020).

DSKLEN - Longitud, Dirección, Conexión de DMA.

Todos estos bits de control relativos a este tema estan contenidos en un registro de sólo-escritura, llamado DSKLEN.

Tabla 8-6: Registro DSKLEN ($DFF024)

Bit	Nombre	Uso
15	DMAEN	Conexión secundaria del DMA del disco.
14	WRITE	Escritura al disco (RAM -> Disco si es 1)
13-Ø	LENGTH	Número de palabras a transferir.

El Hardware necesita una secuencia de orden especial para iniciar el DMA del disco. Esta secuencia evita escrituras accidentales al disco. De forma abreviada, el bit DMAEN del registro DSKLEN se debe poner a 1 dos veces para iniciar el DMA del disco. Esta es la secuencia que se debe seguir:

1. Conectar el DMA del disco en el registro DMACON (ver capitulo 7).

2. Poner DSKLEN a $4000, para desconectar el DMA del disco.

3. Poner el valor que se desee en el registro DSKLEN.

4. Despues de que se complete el DMA, poner de nuevo el registro DSKLEN a $4000, para evitar escrituras accidentales al disco.

Cada vez que se transfiere una palabra, se decrementa el valor de la longitud. Despues de que se produzca la transferencia, se incrementa el valor del puntero. El puntero apunta a la siguiente palabra que se debe escribir o leer. Cuando la cuenta atras de la longitud llega a 0, la transferencia se detiene.

El metodo recomendado para leer desde el disco es leer una pista entera en un buffer y despues buscar el sector(es) que se desea. Usando el registro DSKSYNC (se explicara mas adelante) se garantizara el alineamiento de las palabras. Con este procedimiento sólo se necesita leer una vez del disco para obtener la pista entera. En un cargador de alta velocidad, el paso a la siguiente pista puede suceder mientras los datos de la pista anterior se procesan y se calcula el ckecksum (suma de seguridad). Con este metodo no hay secciones de tiempo critico al leer los datos, con lo que se permite que funcionen otros subsistemas de alta prioridad (graficos, audio...)

Si se tiene muy poca memoria para el buffer de pistas (o por cualquier otra razón se decide no leer toda la pista de una vez), el hardware del disco soporta un conjunto limitado de facilidades para buscar sectores. Hay un registro que se puede utilizar la cadena de los datos que van entrando desde el disco al microprocesador del disco.

Hay un fallo en el software que hace que los últimos tres bits de los datos enviados al disco se pierdan. Tambien, la última palabra en una operación de lectura de disco por DMA puede no llegar (es decir, que se lee una palabra menos de las que se habian solicitado.)

DSKBYTR - Byte del Disco y lectura del status (sólo-lectura).

Este registro es el buffer de datos del microprocesador del disco. En el modo de lectura, los datos del disco se colocan en este registro un byte cada vez. Cada vez que se recibe un byte en el registro, el bit DSKBYT se pone a 1. DSKBYT se pone a 0 cuando se lee el registro DSKBYTR.

DSKBYTR se puede usar para sincronizar el microprocesador con la rotación del disco antes de enviar una lectura o escritura a traves de DMA.

Tabla 8-7: Registro DSKBYTR

Bit	Nombre	Función
15	DSKBYT	Cuando esta a 1, indica que el registro contiene un byte valido (pasa a 0 despues de leer el registro).
14	DMAON	Indica cuando esta en marcha el DMA del disco (cuando todos los bits de DMA estan a 1). Esto significa que DMAEN de DSKLEN debe ser distinto de 0, y los bits DSKEN y DMAEN de DMACON tambien.
13	DISKWRITE	El bit WRITE de DSKLEN esta conectado.
12	WORDEQUAL	Indica que el registro DSKSYNC es igual a la cadena entrante del disco. Este bit es valido mientras la cadena entrante coincide con el registro DSKSYNC (unos 2 µs).
11-8		Actualmente sin usar, no hacer caso del contenido.
7-0	DATA	Byte de datos del disco.

ADKCON y ADKCONR - Registro de Control del audio y del disco

ADKCON es la dirección de sólo-escritura y ADKCONR es la dirección de sólo-lectura de este registro. No todos los bits estan dedicados al disco. El bit 15 permite activar o desactivar a los demas bits de este registro. Si el bit 15 es 1 al escribir, cualquier 1 en las posiciones 0-14 pondra a 1 al bit correspondiente. Si el bit 15 es 0, cualquier 1 pondra a 0 al bit correspondiente.

Tabla 8-7B: Registro ADKCON y ADKCONR

Bit	Nombre	Función
15	SET/CLR	Bit de control que permite poner a 1 o a 0 a los otros bits seleccionados.
14	PRECOMP1	Bit mas significativo de la Precompensación.
13	PRECOMP0	Bit menos significativo de la Precompensación.
		00 -> nada, 01 -> 140 ns, 10 -> 280 ns, 11 -> 560 ns
12	MFMPREC	0 -> Precompensación GCR. 1 -> Precompensación MFM.
10	WORDSYNC	Si esta a 1 se permite la sicronización y el inicio del DMA para leer una palabra del disco. La palabra con la que debe sincronizar debe escribirse en el registro DSKSYNC ($DFF07E). Esto es altamente útil.
9	MSBSYNC	Si esta a 1 se permite la sincronización con el bit mas significativo de la entrada (se usa normalmente en GCR).
8	FAST	Si esta a 1 selecciona 2 µs por cada celda de bit (normalmente MFM). Los datos deben ser datos validos de MFM en bruto. 0 selecciona 4 µs por bit (GCR).

Los datos de MFM en bruto que se deben enviar al controlador del disco deben ser el doble de largos que los datos sin codificar:

1 -> 01 0 -> 10 (si sigue a un 0) 0 -> 00 (si sigue a un 1).

Con una manipulación inteligente, se puede usar el Blitter para codificar y decodificar el MFM.

En la forma común de grabación GCR, todos los bytes tienen el bit mas significativo puesto a 1. Cuando MSBSYNC esta a 1, se indica al controlador del disco que busque este bit de sincronia en cada byte del disco. Cuando se lee un disco formateado en GCR, el software debe usar una tabla de traducción llamada "nybble-izer" para asegurar que los datos escritos en el disco no tendran muchos unos o ceros consecutivos.

DSKSYNC - Sincronizador de la entrada del disco.

El registro DSKSYNC se usa para sincronizar la cadena de datos entrante. Esto es muy útil cuando se leen los discos. Si el bit WORDSYNC de ADKCON esta a 1, no se transferira ningún dato hasta que se encuentre una palabra en la cadena entrante que coincida con el registro DSKSYNC. Al leer, el DMA comenzara con la siguiente palabra del disco. Durante el DMA de lectura del disco, el controlador se resincronizara cada vez que encuentre la palabra de sincronia. Tipicamente DSKSYNC se pone con el valor "magico" de sincronia, que es $4489.

Ademas, el bit DSKSYNC de INTREQ se pone a 1 cuando la cadena de datos entrante coincide con el registro DSKSYNC. El bit DSKSYNC de INTREQ es independiente de la conecion del WORDSYNC de ADKCON.

INTERRUPCIONES DEL DISCO

El controlador del disco puede crear 3 tipos de interrupciones:

- DSKSYNC (nivel 5, bit 12 de INTREQ) - la cadena de datos entrante coincide con el registro DSKSYNC.
- DSKBLK (nivel 1, bit 1 de INTREQ) - se ha completado el DMA del disco.
- INDEX (nivel 6, pin Flag 8520) - sensor del indice del disco (cada vez que el disco da una vuelta completa).

Ver el Capitulo 7, "El Hardware de Control del Sistema" para mas información sobre las interrupciones. Ver el Apendice F para mas información sobre los 8520.

EL TECLADO

El teclado esta conectado al sistema a traves del registro serie de uno de los chips 8520 CIA. La linea de datos del teclado esta conectada al pin SP, el reloj del teclado del reloj esta conectado al pin CNT. El Apendice H contiene una descripción completa del interface.

COMO SE RECIBEN LOS DATOS DEL TECLADO

La linea CNT se usa como reloj para el teclado. En cada transición negativa de esta linea, se obtiene un bit de datos del teclado. El teclado envia esta señal de reloj cuando cada bit de datos es estable en la linea SP. El reloj es un impulso activo de nivel bajo. El borde descendiente de este impulso es el que indica la presencia del dato en SP.

Despues de recibir un byte del teclado, el 8520 envia una interrupción al 68000. El teclado espera a una señal de respuesta del sistema antes de transmitir mas pulsaciones de teclas. Esta respuesta la envia el 68000

generando un impulso negativo en la linea SP. Mientras que algunos teclados pueden detectar un impulso de 1 µs, el impulso debe ser como mínimo 85 µs para funcionar correctamente con todos los teclados.

Si se pulsa otras teclas antes de que el 68000 haya aceptado la anterior, el microprocesador del teclado las ira guardando en un buffer interno de 10 teclas.

TIPO DE DATOS RECIBIDOS

Los datos del teclado no se reciben en forma de caracteres ASCII. En cambio, para una maxima versatilidad, se reciben en forma de códigos de teclas. Estos códigos incluyen la pulsación y la liberación de la tecla. Esto permite al software usar ambos conjuntos de información para determinar exactamente lo que esta sucediendo en el teclado.

Mas adelante hay una lista de los valores hexadecimales asignados al teclado. Cuando se pulsa una tecla se trasmite el valor de la tabla. Cuando se libera la tecla se transmite el mismo valor mas $80. El dibujo del teclado al final de la sección muestra las posiciones que corresponden a la descripción de los parrafos superiores.

Nótese que los códigos proporcionan SOLO información sobre la posición de las teclas, el dibujo impreso en la parte superior de las teclas cambia de un pais a otro.

CODIGOS DEL $00 AL $3F

Estos son los códigos asignados a posiciones especificas en la parte principal del teclado. Las letras de estas teclas son diferentes en los distintos paises; no todos los paises usan la disposición QWERTY. Los teclados internacionales tienen dos teclas mas que son "porciones" de teclas mas grandes del teclado americano (SHIFT izquierdo, y RETURN.)

CODIGOS DEL $40 AL $5F

$40	Espacio
$41	Backspace
$42	Tab
$43	Enter (teclado numerico)
$44	Return
$45	Esc
$46	Delete
$4C	Cursor arriba
$4D	Cursor abajo
$4E	Cursor derecha
$4F	Cursor izquierda
$50-$59	Teclas de función F1-F10
$5F	Help

CODIGOS DEL $60 AL $67

$60	SHIFT izquierdo
$61	SHIFT derecho
$62	Caps lock
$63	Control
$64	ALT izquierdo
$65	ALT derecho
$66	Amiga izquierdo
$67	Amiga derecho

CODIGOS DEL $F0 AL $FF

Estos códigos los usa el teclado para comunicarse con el 68000, y no estan asociados con la pulsación de teclas. No tienen bit de indicador de tecla pulsada/liberada, y por tanto son códigos de 8 bits:

$78 Alerta de Reset. Se ha pulsado CTRL-AMIGA-AMIGA. El teclado esperara un maximo de 10 segundos antes de resetear al ordenador (no en todos los teclados.)

$F9 El último código estaba equivocado, el siguiente es el mismo código retransmitido.

$FA Buffer del teclado lleno.

$FC Fallo en el auto-test del teclado. El led de la tecla caps lock parpadeara para indicar el origen del error. Una vez para fallo de ROM, dos veces para fallo de RAM y tres veces si el temporizador "watchdog" falla en su funcionamiento.

$FD Inicia la secuencia de teclas del encendido (para las teclas pulsadas durante el encendido)

$FE Finaliza la secuencia de teclas del encendido.

Los programas que manipulan el teclado suelen filtrar estos códigos.

LIMITACIONES DEL TECLADO

El teclado del Amiga es una matriz de filas y columnas con un interruptor en cada intersección (Ver en el Apendice H el diagrama de la matriz). Debido a esto, el teclado esta sujeto a un fenómeno llamado "teclas fantasmas". Aunque esto no ocasiona ningún problema al escribir, si que puede ocasionarlo en los juegos que necesitan de muchas teclas pulsadas al mismo tiempo. Examinando la matriz, se puede determinar que teclas interferiran con otras, y que teclas nunca.

Las teclas fantasmas pueden aparecer cuando se pulsan ciertas combinaciones simultaneamente. Por ejemplo, se pulsan la "A" y la "S" simultaneamente y ambas se transmiten. Mientras se siguen pulsando, se pulsa la "Z". En el teclado original del Amiga 1000 se generarian la "Z" y un fantasma, "X". A partir del Amiga 500 el controlador del teclado se actualizó para que al detectar situaciones de posibles fantasmas como la anterior, simplemente deje de enviar códigos hasta que se despeje la matriz (soltando la "A" o la "S" se despejaria la matriz). Algunos de los últimos teclados del Amiga incorporan un sistema (N-key rollover) para detectar cualquier combinación de teclas simultaneas.

Todos los teclados estan diseñados para que los fantasmas no aparezcan durante la escritura normal, sólo durante las combinaciones de teclas inusuales como la descrita antes. Normalmente el teclado parece tener "N-key rollover", lo que quiere decir que te quedaras sin dedos antes de poder generar un fantasma.

NOTA: Hay siete teclas que no son parte de la matriz, y nunca contribuiran a generar una tecla fantasma. Estas teclas son: CTRL, los dos SHIFTs, los dos AMIGAs, y los dos ALTs.

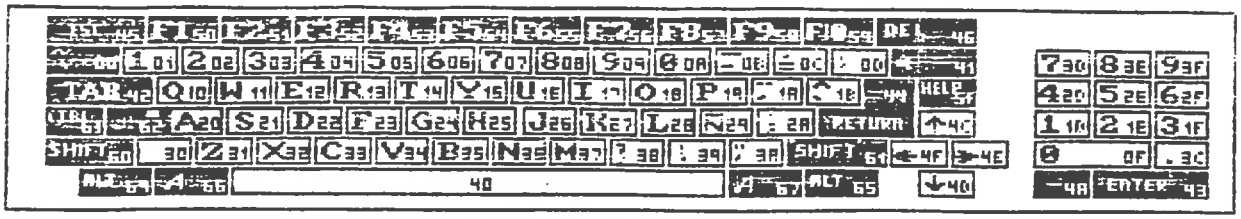


Figura 8-9: Teclado del Amiga 1000. Códigos en Hexadecimal.

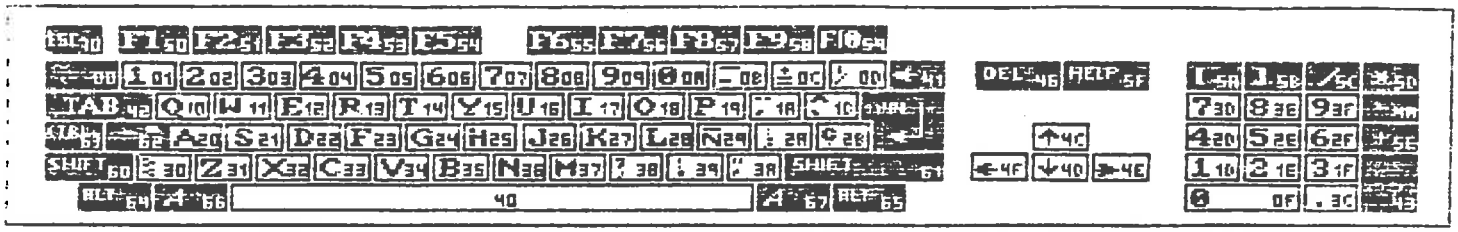


Figura 8-10: Teclado del Amiga 500 y 2000.

INTERFACE PARALELO DE ENTRADA/SALIDA

El interface paralelo bi-direccional de propósito general es un conector de 25 pines que se encuentra en la parte trasera del ordenador. Este conector se usa generalmente para impresoras Centronics.

Por cada byte escrito al registro del port paralelo, el hardware genera automaticamente un impulso en pin de "dato preparado" (STROBE). El impulso de "acknoweledge" del dispositivo paralelo se recibe como una interrupción. Para las conexiones de los pines y temporización ver los Apendices E y F.

INTERFACE SERIE

El interface serie de propósito general es un conector de 25 pines en la parte trasera del ordenador. Este conector puede soportar un amplio rango de diferentes periféricos, incluyendo un modem externo o una impresora en serie. Para las coneciones de los pines ver el Apendice E.

INTRODUCCION A LA CIRCUITERIA SERIE

El custom chip Paula contiene un "Universal Asynchronous Receiver-Transmiter" o UART (Receptor/Transmisor universal asincrono). Este UART se puede programar a una velocidad de 110 a 1000000 de bits por segundo. Puede recibir y enviar datos con una longitud programable de 8 o 9 bits.

El circuito del UART proporciona un alto grado de control a traves del software. El UART es capaz de detectar errores de "overrun", que ocurren cuando el otro sistema envia los datos mas rapido de lo que le cuesta al 68000 sacarlos del registro de recepción de datos. Hay tambien unos bits de status e interrupciones para las situaciones de "buffer de recepción lleno" o "buffer de transmisión vacio". Hay un bit adicional que indica "todos los bits han sido enviados fuera". Todos estos puntos se explican a continuación.

INDICANDO LA VELOCIDAD EN BAUDIOS

La velocidad de transmisión (baudios) se controla mediante el registro llamado SERPER. Los bits 14-0 de SERPER son los bits divisores de la velocidad en baudios.

Todas las temporizaciones se realizan sobre la base del "reloj de color", que es de 279.86 ns en NTSC y de 281.94 ns en PAL. Si el divisor SERPER se pone a un número N, entonces tendran que ocurrir N+1 ciclos del reloj de color entre cada muestreo del estado del pin de entrada (para recibir) o entre las transmisiones de los bits que se van enviando (para transmitir). Por ejemplo, el valor adecuado de SERPER para 9600 baudios en NTSC es (3579545/9600)-1 = 371 y en PAL es (2546895/9600)-1 = 368.

Con un cable de una longitud razonable, la maxima velocidad segura es de unos 150000 a 250000 bits por segundo. Las velocidades maximas pueden variar entre distintos ordenadores. Con estas velocidades tan altas no se puede manejar la sobrecarga de las interrupciones. El algoritmo de recepción tendria que ser un bucle de lectura muy ajustado. Mediante el uso de información de control de baja velocidad y "bursts" de alta velocidad, se puede construir una red local de comunicaciones por poco dinero.

SELECCIONANDO EL MODO DE RECEPCION

Se puede definir el número de bits que se recibiran antes de que el registro de recepción se llene como 8 o 9 (esto permite la transmisión de 8 bits con paridad). En cualquier caso, el circuito de recepción espera a encontrar un bit de inicio, ocho o nueve bits de datos, y como mínimo un bit de stop.

El modo de recepción se selecciona actuando sobre el bit 15 del registro de sólo-escritura SERPER. Si el bit 15 es un 1 se utilizan 9 bits para que el registro de recepción envie la señal de registro lleno, y si es un 0 se utilizan 8 bits. El estado normal de este bit es 0.

CONTENIDO DEL REGISTRO DE RECEPCION DE DATOS

El registro de recepción de datos es de 16 bits de ancho. Contiene los 8 o 9 bits de datos y los bits de status.

Los datos se reciben, un bit cada vez, en un registro interno desplazador de serie-a-paralelo. Cuando ha pasado el número de bits seleccionado, el contenido de este registro se transfiere al registro de lectura de datos (SERDAT) que se muestra en la Tabla 8-9, y se avisa al 68000 de que hay unos datos preparados en el registro SERDAT.

Inmediatamente despues de que haya ocurrido la transferencia de los datos, el registro de recepción vuelve a estar preparado para aceptar nuevos datos. Despues de recibir la interrupción de "registro lleno", se tiene el tiempo que tarda un caracter entero (8 a 10 bits) para aceptar los datos y poner a 0 la interrupción. Si la interrupción no se pone a 0 a tiempo, el bit OVERRUN se pondra a 1 indicando que la recepción es demasiado lenta.

Tabla 8-9: Registros SERDATR / ADKCON

Bit	Nombre	Función
15	OVRUN	Bit de OVERRUN (Espejo - Tambien aparece en INTREQR) Indica que se ha recibido otro byte de datos antes de que el 68000 haya aceptado el byte anterior. Para evitar esta situación, es necesario poner a 0 el bit INTF_RBF (bit 11 de INTREQR) cada vez que se recibe un byte.
14	RBF	READ BUFFER FULL (Buffer de lectura lleno) (Espejo - Tambien aparece en INTREQR) Cuando este bit es 1, indica que hay datos preparados para ser recogidos por el 68000. Despues de leer el contenido del registro de datos, se debe poner a 0 el bit INTF_RBF de INTREQR para evitar un OVERRUN.
13	TBE	TRANSMIT BUFFER EMPTY (Buffer de transmisión vacio) (No es un espejo - la interrupción ocurre cuando el buffer pasa a estar vacio). Cuando este bit es 1, los datos del registro del registro de salida de datos (SERDAT) han sido transferidos al registro desplazador de salida, por tanto SERDAT está preparado para aceptar otro datos. Este bit cambia a cero mientras el buffer este vacio.
		Este bit se usa normalmente en comunicaciones full-duplex.
12	TSRE	TRANSMIT SHIFT REGISTER EMPTY (Registro desplazador de transmión vacio) Cuando este bit es 1 indica que el registro desplazador de salida ha completado su tarea, todos los datos han sido transmitidos, y el registro ahora esta en reposo. Si se para de escribir datos en el registro de salida (SERDAT), este bit se pondra a 1 despues de que se haya transmitido la palabra del registro desplazador Y la que se encuentra en el registro SERDAT.
		Este bit se usa normalmente en comunicaciones half-duplex.
11	RXD	Lectura directa del pin RXD del chip Paula.
10		No se usa actualmente.
9	STP	Bit de stop si se ha especificado 9 bits para recibir.
8	STP/DB8	Bit de stop si se ha especificado 8 bits para recibir. Bit 9 de datos si se ha especificado 9 bits.
7-0	DB7-DB0	Bits inferiores de datos (8). Los datos son verdaderos (los datos que se leen tienen la polaridad adecuada, no estan invertidos).
AKCON		
15	SET/CLR	Bit SET/CLR. Si es 1 pone a 1 los bits seleccionados, si es 0, pone a 0 los bits seleccionados.
11	UARTBRK	Fuerza el pin de transmisión a 0.

COMO SE TRANSMITEN LOS DATOS

Los datos se envian a traves de las lineas de transmisión escribiendo al registro de salida de datos (SERDAT). Este registro es de sólo-escritura.

Los datos se envian a la misma velocidad que se habia establecido para la recepción. Inmediatamente despues de escribir los datos en este registro, el sistema inicia la transmisión a la velocidad en baudios establecida.

En el inicio de la operación, se transfieren los datos desde SERDAT a un registro interno desplazador. Cuando la transferencia concluye, SERDAT puede aceptar mas datos; la interrupción TBE indica esta situación.

Los datos se iran empujando al exterior del registro desplazador, un bit durante cada intervalo de tiempo, comenzando con el bit inferior. El desplazamiento continua hasta que todos los 1s se hayan extraido. Se puede usar cualquier combinación de datos y bits de stop.

SERDAT es un registro de 16 bits que permite controlar el formato (apariencia) de los datos transmitidos. Para formar una secuencia típica de datos, como un bit de inicio, ocho bits de datos, y un bit de stop, se debe escribir en SERDAT los contenidos mostrados en las Figuras 8-11 y 8-12.

Figura 8-11: Apariencia inicial de SERDAT y el registro desplazador.

Figura 8-12: Apariencia final del registro desplazador.

El registro para de desplazar e indica "registro desplazador vacio" (TSRE) cuando hay un bit a 1 en la posición "bit desplazado a fuera" y el contenido del registro es todo 0s. Cuando un nuevo contenido distinto a 0 se carga en este registro, se inicia de nuevo el desplazamiento.

ESPECIFICANDO EL CONTENIDO DEL REGISTRO

Los datos que se van a transmitir se colocan en el registro de salida (SERDAT). Se debe añadir un bit como bit de stop a los bits de datos. Normalmente se envian uno o dos bits de stop.

La transmisión del bit de inicio es independiente del contenido de este registro. Se genera un bit de inicio automaticamente antes de que se envie el primer bit de datos (bit 0).

Al escribir este registro se inicia la transmisión automaticamente. Si este registro se escribe con todo 0s, no se inicia la transmisión de datos.

CONECTORES DE SALIDA DE IMAGEN

Todos los Amigas tienen un conector de 23 pines en la parte trasera. Este conector contiene salidas de imagen y entradas para genlocks externos. Hay dos tipos de RGB distintos en el conector:

- Monitores RGB (RGB analógico). Proporciona cuatro salidas: Rojo (R), Verde (G), Azul (B) y Sincronía (S). Puede generar hasta 4096 colores en la pantalla simultaneamente utilizando la circuiteria del Amiga.

- Monitores Digitales RGB. Proporciona cinco salidas, distintas de las anteriores: Rojo (R), Verde (G), Azul (B), Media-Intensidad (I) y Sincronia (S). Los niveles de salida son los niveles lógicos (0 o 1). En algunos monitores estas salidas permiten hasta 15 combinaciones de color posibles, porque los valores 0000 y 0001 dan el mismo resultado (media intensidad de ningún color es lo mismo que intensidad completa de ningún color). Otros monitores crean 16 colores arbitrarios de las 16 combinaciones posibles.

Nótese que las señales de sincronia del Amiga estan sin amplificar. Para el uso con algún dispositivo que suponga una carga pesada para las salidas de sincronia, se necesitaran amplificadores externos.

Los Amigas 500 y 2000 tienen un conector de video compuesto monocromo de banda ancha para que se puedan usar monitores monocromo que son mas baratos. Los colores del Amiga se combinan en forma de intensidades basandose en la siguiente tabla:

Rojo	Verde	Azul
30%	60%	10%

El Amiga 1000 contiene un conector de RF. Hay un adaptador que permite usar el Amiga con una televisión, y aunque el sonido es stereo en el conector, las televisiones normales lo reproduciran como sonido mono.

El Amiga 1000 contiene un conector de video compuesto en color. Esto es adecuado para grabar en un magnetoscopio (video) directamente, pero la salida no es de calidad profesional. Si se usa con un monitor monocromo, la información del color puede tener efectos desagradables; una selección cuidadosa del color o una modificación de la circuiteria interna pueden mejorar los resultados. Hay adaptores compuestos de alta calidad para el A500, el A1000 y el A2000 que se conectan en el port RGB de 23 pines.

El Amiga 2000 tiene un slot especial para video que contiene muchas mas señales que el port de 23 pines: todas las señales del port de 23 pines, el video digital sin codificar, lapiz óptico, toma de corriente, sonido, burst de color, pixel switch (para genlock), sincronia, señales de reloj, etc.

APENDICE A

SUMARIO DE REGISTROS - EN ORDEN ALFABETICO

Este Apendice contiene el sumario definitivo, en orden alfabetico, del conjunto de registros y el uso de los bits individuales.

Las direcciones mostradas aqui son usadas por los custom chips ("Agnus", "Denise" y "Paula") para transferir datos entre ellos mismos. Tambien, el Copper usa estas direcciones para escribir en los registros de los custom chips. Para escribir estas direcciones con el 68000, se debe calcular la dirección en su espacio de direccionamiento:

dirección del 68000 = (direción de chip) + $DFF000

Por ejemplo para que el 68000 escribiera en el registro ADKCON ($9E), la dirección seria la $DFF09E. Ninguna otra dirección seria valida. No se debe acceder a los registros sin usar.

Todos los bits marcados como "sin usar" se deben escribir con ceros. El valor de cualquier bit sin usar leido no se debe usar. Los registros son de sólo-lectura (READ) o sólo-escritura (WRITE). Leer un registro de sólo-lectura destruira el contenido del registro. Escribir un registro de sólo-lectura causara resultados impredicibles.

Todos los registros tipo "puntero" estan organizados como 32 bits en forma de palabra larga. Estos registros se pueden escribir con la instrucción MOVE.L. El bit inferior de todos los punteros se debe escribir como 0. Los custom chips sólo pueden acceder a la memoria CHIP; si se usa una dirección de memoria no-CHIP no funcionara (ver la documentación de AllocMem() o el manual del ensamblador o compilador para mas información sobre la memoria CHIP). Los datos del disco, de los Sprites, de los bit-planes, del audio, las CopperLists y cualquier otra cosa que se deba "Blitear" o acceder por los custom chips a traves del DMA debe estar en memoria CHIP.

Cuando se actua sobre un registro del tipo "STROBE" que responde a una escritura o una lectura, (por ejemplo copjmp2) se debe usar un MOVE.W y no un CLR.W. La intrucción CLR causa una lectura y una escritura (dos accesos) en un 68000, pero sólo un acceso en un 68020. Esto daria diferentes resultados en diferentes microprocesadores.

Registro	Dirección	Read Write	Agnus Denise Paula	Función
ADKCON	$09E	W	Paula	Audio, Disk, control write.
ADKCONR	$010	R	Paula	Audio, Disk, control read. (Control del Audio y del disco)

BIT	NOMBRE	FUNCION
15	SET/CLR	Bit de control SET/CLR.
14	PRECOMP1	10 = 280 ns, 11 = 560 ns
13	PRECOMP0	00 = nada, 01 = 140 ns
12	MFMPREC	(0 = Precomp. GCR 1 = Precomp. MFM)
11	UARTBRK	Fuerza UART a 0. (pone TXD a 0)
10	WORDSYNC	Si esta a 1 se permite la sicronización con el registro DSKSYNC ($DFF07E).
09	MSBSYNC	Si esta a 1 se permite la sincronización con el bit mas significativo (se usa en GCR).
08	FAST	Control del reloj de datos del disco. 1=2 µs (MFM) 0 = 4 µs por bit (GCR).
07	USE3PN	Canal 3 modula nada.
06	USE2P3	Canal 2 modula periodo canal 3.
05	USE1P2	Canal 1 modula periodo canal 2.
04	USE0P1	canal 0 modula periodo canal 1.

ADKCONR(cont.) Ø3 USE3PN canal 3 modula nada.
Ø2 USE3PN canal 2 modula volumen canal 3.
Ø1 USE3PN canal 1 modula volumen canal 2.
ØØ USE3PN canal Ø modula volumen canal 1.

Nota: Si se modulan periodo y volumen en el mismo canal, el periodo y el volumen se alternaran. La primera palabra para el volumen, la segunda para el periodo, etc.

AUDxDAT $ØAA W Paula Audio Channel x data

Este registro es el buffer de datos de DMA para el canal x (Ø,1,2,3). Contiene 2 bytes de datos que estan en complemento a 2 y que son enviados secuencialmente al exterior a traves de un convertidor digital-a-analógico. (LSB= 3 mV) El controlador de DMA transfiere automaticamente los datos a este registro desde la RAM. El 68ØØØ tambien puede escribir directamente a este registro. Cuando el DMA ha finalizado (palabras transferidas=longitud) y los datos de este registro se han usado, se pone a 1 una petición de interrupción para este canal.

AUDxLCH $ØA0 W Agnus Audio channel x location (high 3 bits)
AUDxLCL $ØA2 W Agnus Audio channel x location (low 15 bits)

Este par de registro contiene la dirección de 18 bits inicial de los datos para el DMA del canal x de audio. No es un registro de puntero y por lo tanto sólo se necesita escribir a el cuando se desee cambiar su contenido.

AUDxLEN $ØA4 W Paula Audio channel x length

Este registro contiene la longitud de (número de palabras) de los datos para el DMA del canal x de audio.

AUDxPER $ØA6 W Paula Audio channel x Period

Este registro contiene el periodo (velocidad) de la transferencia de datos para el canal x de audio.
El periodo minimo (con DMA) es de 124 ciclos de reloj que corresponde a una frecuencia de 28.86 Khz.

AUDxVOL $ØA8 W Paula Audio channel x volume

Este registro contiene la selección del volumen para el canal x de audio. Los bits 5-Ø especifican 64 niveles lineales de volumen. (el bit 6 pone el volumen al maximo)

BLTAFWM $Ø44 W Agnus Blitter first-word mask for source A
BLTALWM $Ø46 W Agnus Blitter last-word mask for source A

Los patrones de estos registros se enmascaran (AND) con la primera y última palabra de cada linea de datos de la fuente A del Blitter. Un cero en cualquier bit ocultaria al bit correspondiente de la fuente A. Estos registros se deben poner todo a 1s en los modos de rellenado y lineas.

BLTCONØ $Ø4Ø W Agnus Blitter control register Ø.
BLTCON1 $Ø42 W Agnus Blitter control register 1.

Estos dos registros de control se usan juntos para controlar las operaciones del Blitter. Hay dos modos basicos, areas y lineas, que se seleccionan por el bit Ø de BLTCON1.

MODO DE AREAS ("normal")

BIT	BLTCON0	BLTCON1
15	ASH3	BSH3
14	ASH2	BSH2
13	ASH1	BSH1
12	ASH0	BSH0
11	USEA	X
10	USEB	X
09	USEC	X
08	USED	X
07	LF7	X
06	LF6	X
05	LF5	X
04	LF4	X
03	LF3	EFE
02	LF2	IFE
01	LF1	DESC
00	LF0	LINE(=0)

ASH3-0 Valor de desplazamiento para la fuente A.
BSH3-0 Valor de desplazamiento para la fuente B.
USEA Bit de control para usar la fuente A.
USEB Bit de control para usar la fuente B.
USEC Bit de control para usar la fuente C.
USED Bit de control para usar el destino D.
LF7-0 Selección de la función lógica de miniterms.
EFE Permite rellenado exclusivo.
IFE Permite rellenado inclusivo.
FCI Estado inicial del rellenado.
DESC Bit de control del modo descendiente
LINE Bit de control del modo de lineas (pusto a 0)

MODO DE LINEAS

BIT	BLTCON0	BLTCON1
15	START3	TEXTURE3
14	START2	TEXTURE2
13	START1	TEXTURE1
12	START0	TEXTURE0
11	1	0
10	0	0
09	1	0
08	1	0
07	LF7	0
06	LF6	SIGN
05	LF5	0 (reservado)
04	LF4	SUD
03	LF3	SUL
02	LF2	AUL
01	LF1	SING
00	LF0	LINE(=1)

START3-0 Punto inicial de la linea (0 a 15)

LF7-0 La selección de la función lógica por miniterms se debe poner a $4A para seleccionar la ecuación D=(AC+ABC). Debido a que A contiene un solo bit ($8000), la mayoria de los bits atravesaran a C sin ser modificados (no A y C), pero un bit invertira a C y se combinara con la textura (A y B y no C). El bit A se va desplazando automaticamente por la palabra por la acción del hardware.

LINE Control del modo de lineas (puesto a 1)
SIGN Indicador del signo.
0 Reservado para un modo nuevo.

SING Un solo bit por linea horizontal para el uso con el modo de rellenado de areas.
SUD A veces arriba o abajo (=AUD*)
SUL A veces abajo o izquierda
AUL Siempre arriba o izquierda.

Estos últimos tres bits seleccionan el octante de la linea

OCTANTE	SUD	SUL	AUL
0	1	1	0
1	0	0	1
2	0	1	1
3	1	1	1
4	1	0	1
5	0	1	0
6	0	0	0
7	1	0	0

La fuente B se usa para dar textura a la linea.

BLTDDAT ---- - --- Blitter destination data register.

Este registro retiene las palabras resultantes de cada operación del Blitter hasta que se envien a su destino en RAM. Esta es una dirección fantasma y no la puede leer el 68000. La transferencia es automatica durante las operaciones del Blitter.

BLTSIZE $058 W Agnus Blitter start and size.

Este registro contiene el ancho y el alto de la operación del Blitter (en modo de lineas el ancho debe ser 2 y el alto la longitud de la linea). Escribiendo a este registro se pone en marcha al Blitter, y esto se debe hacer en último lugar, despues de haber inicializado todos los punteros y registros de control.

BIT: 15,14,13,12,11,10,09,08,07,06,05,04,03,02,01,00
h9 h8 h7 h6 h5 h4 h3 h2 h1 h0 w5 w4 w3 w2 w1 w0

h=altura en lineas (10 bits = 1024 lineas max.)
w=anchura en palabras (6 bits = 64 palabras = 1024 pixels)

En el **modo de lineas** el campo h controla la longitud de la linea (10 bits = 1024 puntos por linea max.) El campo w se debe poner a 2 en todas las lineas.

BLTxDAT $074 W Agnus Blitter source x data register.

Este registro retiene los datos de la fuente x (A,B,C) para ser usados por el Blitter. Lo carga normalmente el canal DMA del Blitter; sin embargo, tambien lo puede precargar el 68000.

En el modo de lineas BLTADAT se usa como un registro indice que debe precargar el 68000. BLTBDAT se usa para la textura; debe ser precargado con $FFFF si no se desea textura (linea sólida).

BLTxMOD $064 W Agnus Blitter modulo x.

Este registro contiene el módulo para la fuente (x=A,B,C) o para el destino (x=D). Un módulo es un número que se suma automaticamente a la dirección al final de cada linea, para hacer que la dirección apunte al inicio de la siguiente

línea. Cada fuente o destino tiene su propio módulo, permitiendo a cada uno usar un tamaño diferente, mientras que se usa un area identica de cada uno en la operación del Blitter.

En el **modo de lineas** BLTAMOD y BLTBMOD se usan como registros de almacenamiento de la pendiente y se deben precargar con los valores (4Y-4X) y (4Y) respectivamente. Y/X es la pendiente de la linea. BLTCMOD y BLTDMOD se deben precargar con el ancho (en bytes) de la imagen dentro de la cual se va ha dibujar la linea (normalmente dos veces la anchura en palabras de la pantalla).

BLTxPTH	$050	W	Agnus	Blitter pointer to x (high 3 bits)
BLTxPTL	$052	W	Agnus	Blitter pointer to x (low 15 bits)

Este par de registros contienen la dirección de 18 bits de la fuente del Blitter (x=A,B,C) o el destino (x=D) de los datos del DMA. Este puntero se debe precargar con la dirección inicial de los datos a ser procesados por el Blitter. Despues de que el Blitter haya terminado, contendran la última dirección de los datos (mas el incremento y el módulo)

En el **modo de lineas** BLTAPTL se usa como registro acumulador y se debe precargar con el valor inicial de (2Y-X) donde Y/X es la pendiente de la linea. BLTCPT y BLTDPT (H y L) se deben precargar con la dirección inicial de la linea.

BPL1MOD	$108	W	Agnus	Bit plane modulo (odd planes)
BPL2MOD	$10A	W	Agnus	Bit plane modulo (even planes)

Estos registros contienen los módulos para los bit-planes impares y pares respectivamente. Un módulo es un número que se suma automaticamente a la dirección al final de cada linea, para hacer que la dirección apunte al inicio de la siguiente linea.
Debido a que hay dos módulos separados, los bit-planes pares e impares pueden tener tamaños diferentes, al igual que diferentes tamaños para la ventana de visualización.

BPLCON0	$100	W	Ag.Dn.	Bit plane control register (misc.)
BPLCON1	$102	W	Denise	Bit plane control register (scroll)
BPLCON2	$104	W	Denise	Bit plane control register (priority)

Estos registros controlan las operaciones de los bit-planes y varios aspectos de la imagen.

BIT	BPLCON0	BPLCON1	BPLCON2
15	HIRES	X	X
14	BPU2	X	X
13	BPU1	X	X
12	BPU0	X	X
11	HOMOD	X	X
10	DBLPF	X	X
09	COLOR	X	X
08	GAUD	X	X
07	X	PF2H3	X
06	X	PF2H2	PF2PRI
05	X	PF2H1	PF2P2
04	X	PF2H0	PF2P1
03	LPEN	PF1H3	PF2P0
02	LACE	PF1H2	PF1P2
01	ERSY	PF1H1	PF1P1
00	X	PF1H0	PF1P0

HIRES = Modo de alta resolución (640)
BPU = Número de bit-planes 000-110 (nada a 6 inclusive)
HOMOD = Modo HAM (Hold And Modify)
DBLPF = Doble Playfield (PF1 = planos impares, PF2 = pares)
COLOR = Conexión video compuesto en color.
GAUD = Conexión del audio del genlock (se envia al pin BKGND durante el vertical blank)
LPEN = Modo del lapiz óptico (normalmente a 0)
LACE = Modo de entrelazado (normalmente a 0)
ERSY = Resincronización externa (los pines HSYNC y VSYNC se convierten en entradas) (normalmente a 0)
PF2PRI = El Playfield 2 (planos pares) tiene prioridad sobre el Playfield 1 (planos impares).
PF2P = Código de prioridad del Playfield 2 (respepecto a los Sprites).
PF1P = Código de prioridad del Playfield 1 (respepecto a los Sprites).
PF2H = Scroll horizontal del Playfield 2.
PF1H = Scroll horizontal del Playfield 1.

BPLxDAT $110 W Denise Bit plane x data (parallel-to-serial convert)

Estos registros reciben los datos del DMA obtenidos de la RAM por los punteros de los bit-planes descritos anetes. Tambien los puede escribir el 68000. Actuan como buffer del convertidor de seis palabras paralelo-a-serie para seis bit-planes (x=1-6). La conversión paralelo-a-serie se inicia cuando se escribe el bit-plane 1, indicando que se encuentran todos los bit-planes para esta palabra (16 pixels). El byte mas significativo (MSB) se visualiza primero, y esta, por tanto, siempre a la izquierda.

BPLxPTH $0E0 W Agnus Bit plane x pointer (high 3 bits)
BPLxPTH $0E2 W Agnus Bit plane x pointer (low 15 bits)

Este par de registros contiene el puntero de 18 bits de la dirección de los datos DMA del bit-plane x (x=1,2,3,4,5,6). Este puntero se debe reinicializar al comienzo de los datos del bit-plane en cada vertical blank.

CLXCON $098 W Denise Collision control

Este registro controla los planos que se incluyen en la detección de colisión y el estado requerido si estan incluidos. Tambien controlan la inclusión individual de los sprites impares en la detección con los sprites pares.

Bit	Nombre	Función
15	ENSP7	Permite Sprite 7 (OR con el 6)
14	ENSP5	Permite Sprite 5 (OR con el 4)
13	ENSP3	Permite Sprite 3 (OR con el 2)
12	ENSP1	Permite Sprite 1 (OR con el 0)
11	ENBP6	Permite Bit-plane 6
10	ENBP5	Permite Bit-plane 5
9	ENBP4	Permite Bit-plane 4
8	ENBP3	Permite Bit-plane 3
7	ENBP2	Permite Bit-plane 2
6	ENBP1	Permite Bit-plane 1
5	MVBP6	Valor colisión de bit-plane 6
4	MVBP5	Valor colisión de bit-plane 5
3	MVBP4	Valor colisión de bit-plane 4
2	MVBP3	Valor colisión de bit-plane 3
1	MVBP2	Valor colisión de bit-plane 2
0	MVBP1	Valor colisión de bit-plane 1

Nota: Los bit-planes no conectados no pueden evitar las colisiones. Por lo tanto si se desconectan todos los bit-planes, las colisiones seran continuas, independientemente del valor para la colisión.

DLXDAT $00E R Denise Collision data register (read and clear)

Esta dirección lee (y borra) el registro de detección de colisión.

NOTA: El Playfield 1 son todos los bit-planes impares y el Playfield 2 son todos los bit-planes pares.

Bit	Colisiones Registradas
15	sin usar
14	Sprite 4 (o 5) con Sprite 6 (o 7)
13	Sprite 2 (o 3) con Sprite 6 (o 7)
12	Sprite 2 (o 3) con Sprite 4 (o 5)
11	Sprite 0 (o 1) con Sprite 6 (o 7)
10	Sprite 0 (o 1) con Sprite 4 (o 5)
9	Sprite 0 (o 1) con Sprite 2 (o 3)
8	Bit-planes pares con Sprite 6 (o 7)
7	Bit-planes pares con Sprite 4 (o 5)
6	Bit-planes pares con Sprite 2 (o 3)
5	Bit-planes pares con Sprite 0 (o 1)
4	Bit-planes impares con Sprite 6 (o 7)
3	Bit-planes impares con Sprite 4 (o 5)
2	Bit-planes impares con Sprite 2 (o 3)
1	Bit-planes impares con Sprite 0 (o 1)
0	Bit-planes pares con Bit-planes impares

COLORxx $180 W Denise Color table xx

Hay 32 de estos registros (x=00-31) y a veces son llamados "paleta de color". Contienen códigos de 12 bits que representan los colores rojo, verde y azul de los sistemas RGB. Se selecciona uno de estos registros cada vez (por el código de BPLxDAT serializado) para representar la salida del video en los pines de salida RGB.

BIT: 15,14,13,12,11,10,09,08,07,06,05,04,03,02,01,00
RGB: X X X X R3 R2 R1 R0 G3 G2 G1 G0 B3 B2 B1 B0

R = Rojo, G = Verde, B = Azul.

COP1LCH $080 W Agnus Copper first location register (high 3)
COP1LCL $082 W Agnus Copper first location register (low 15)
COP2LCH $084 W Agnus Copper second location register (high 3)
COP2LCL $086 W Agnus Copper second location register (low 15)

Estos registros contienen la primera y segunda direccion de salto para el Copper.

COPCON $02E W Agnus Copper control register

Este es un registro de 1 bit que cuando esta a 1, permite al Copper acceder al hardware Blitter. Este bit se pone a 0 en la operación de encendido del ordenador, por lo que el Copper no puede acceder al hardware del Blitter.

Bit	Nombre	Función
01	CDANG	Modo peligroso del Copper, permite al Copper acceder al hardware del Blitter.

COPINS $08C W Agnus Copper instruction fetch identify

Esta es una dirección fantasma que genera el Copper cuando esta cargando instrucciones en su propios registros. Esto ocurre en cada ciclo del Copper excepto en el segundo ciclo (IR2) de la instrucción MOVE.

MOVE Mueve un dato inmediato a la dirección del registro de hardware indicado.
WAIT Espera hasta que el contador del rayo sea igual, o mayor al dato inmediato. (mantiene al Copper fuera del bus mientras esta esperando)
SKIP Se salta la siguiente instrucción si el contador del rayo es igual o mayor al dato inmediato.

	MOVE		WAIT		SKIP	
Bit	IR1	IR2	IR1	IR2	IR1	IR2
15	X	RD15	VP7	BFD *	VP7	BFD *
14	X	RD14	VP6	VE6	VP6	VE6
13	X	RD13	VP5	VE5	VP5	VE5
12	X	RD12	VP4	VE4	VP4	VE4
11	X	RD11	VP3	VE3	VP3	VE3
10	X	RD10	VP2	VE2	VP2	VE2
9	X	RD09	VP1	VE1	VP1	VE1
8	DA8	RD08	VP0	VE0	VP0	VE0
7	DA7	RD07	HP8	HE8	HP8	HE8
6	DA6	RD06	HP7	HE7	HP7	HE7
5	DA5	RD05	HP6	HE6	HP6	HE6
4	DA4	RD04	HP5	HE5	HP5	HE5
3	DA3	RD03	HP4	HE4	HP4	HE4
2	DA2	RD02	HP3	HE3	HP3	HE3
1	DA1	RD01	HP2	HE2	HP2	HE2
0	0	RD00	1	0	1	1

IR1 Registro de la primera instrucción.
IR2 Registro de la segunda instrucción.
DA Dirección destino para la instrucción MOVE. Obtenida durante el tiempo de IR1, usada durante el tiempo IR2 en el bus RGA.
RD Datos de la RAM movidos por la instrucción MOVE en el tiempo de IR2 directamente desde la RAM a la dirección indicada en el campo DA.
VP Bit de comparación de la posición vertical del rayo.
HP Bit de comparación de la posición horizontal del rayo.
VE Bit de enmascaramiento de la comparación vertical.
HE Bit de enmascaramiento de la comparación horizontal.

* Nota: BFD = Blitter Finished Disable. Cuando este bit esta a 1, el indicador de Blitter finalizado no tendra efecto sobre el Copper. Cuando este bit esta a 0, el indicador del Blitter finalizado debera estar a 0 (ademas del resto de las comparaciones de bits) antes de que el Copper pueda salir de su estado de espera o saltarse la siguiente instrucción. Nótese que la comparación de V7 no se puede enmascarar.

El Copper es basicamente una maquina de dos ciclos que utiliza el bus sólo durante los ciclos de memoria impares (4 ciclos de memoria por instrucción). Esto evita colisiones con la visualización, el audio, el disco, el refresco de memoria, los sprites, y todo lo que use los ciclos pares. Por lo tanto necesita (y tiene) prioridad sólo sobre el Blitter y el 68000.

COPINS(cont.) Sólo hay tres tipos de instrucciones: MOVE inmediato, WAIT until (esperar hasta) y SKIP if (saltar si). Todas las instrucciones (excepto WAIT) necesitan dos ciclos del bus (y dos palabras como instrucción). Como sólo se puede acceder a los ciclos impares del bus, se necesitan cuatro ciclos de memoria por cada instrucción (los ciclos de memoria son de 280 ns)

Hay dos registros de salto indirectos, COP1LC y COP2LC. Estos son registros de 18 bits del tipo puntero cuyos contenidos se usan para modificar el contador de programa para inicialización o saltos. Se transfieren al contador de programa cuando se escriben las direcciones STROBE de COPJMP1 o COPJMP2. Ademas COP1LC se usa automaticamente al inicio del vertical blank.

Es importante que uno de los registros de salto se inicialice y se escriba a su dirección de STROBE despues del encendido del ordenador pero antes de que se inicialice el DMA del Copper. Esto asegura una dirección determinada y un status de puesta a punto.

COPJMP1	$088	S	Agnus	Copper restart at first location.
COPJMP2	$08A	S	Agnus	Copper restart at second location.

Estas direcciones son direcciones de tipo STROBE. Cuando se escriben, hacen que el Copper salte indirectamente usando las direcciones contenidas en los registros escritos anteriormente. El mismo Copper puede escribir estas direcciones, causando sus saltos indirectos.

DDFSTOP	$094	W	Agnus	Display data fetch stop (horiz.)
DDFSTRT	$092	W	Agnus	Display data fetch start (horiz.)

Estos registros controlan la temporización horizontal del inicio y final de la obtención de datos por el DMA de los bit-planes. La temporización vertical del DMA de los bit-planes es identico a las ventanas de visualización descritas mas adelante.
Los módulos de los bit-planes son dependientes del tamaño horizontal del bit-plane y de este tamaño de la ventana de obtención de datos.

BIT:	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
USO:	X	X	X	X	X	X	X	X	H8	H7	H6	H5	H4	H3	X	X

(Los bits X se deben mantener a 0 para posterior compatibilidad con futuros modelos)

Las tablas siguientes muestran los tiempos de inicio y final para los diferentes contenidos de estos registros.

DDFSTRT (borde izquierdo en la imagen)

PROPOSITO	H8	H7	H6	H5	H4
Extra ancho (max)*	0	0	1	0	1
Ancho	0	0	1	1	0
Normal	0	0	1	1	1
Estrecho	0	1	0	0	0

DDFSTOP (borde derecho en la imagen)

PROPOSITO	H8	H7	H6	H5	H4
Estrecho	1	1	0	0	1
Normal	1	1	0	1	0
Ancho (max)	1	1	0	1	1

DIWSTOP	$090	W	Agnus	Display window stop
DIWSTRT	$08E	W	Agnus	Display window start

Estos registros controlan el tamaño de la ventana de visualización y su posición definiendo sus esquinas superior izquierda e inferior derecha.

BIT: 15,14 13,12 11,10 09 08,07 06 05,04,03,02,01,00
USO: V7 V6 V5 V4 V3 V2 V1 V0 H7 H6 H5 H4 H3 H2 H1 H0

DIWSTRT esta restringido verticalmente a los 2/3 superiores de la imagen (V8=0) y horizontalmente a los 3/4 izquierdos de la imagen (H8=0).

DIWSTOP esta restringido horizontalmente a 1/2 inferior de la imagen (V8=/=V7) y horizontalmente a 1/4 de la imagen (H8=1).

DMACON	$096	W	A,D,P	DMA control write (clear o set)
DMACONR	$002	R	Ag, Pa	DMA control (and Blitter status) read

Este registro controla todos los canales DMA y contiene los bits de status del DMA del Blitter.

Bit	Nombre	Función
15	SET/CLR	Bit de control SET/CLR.
14	BBUSY	Estado de Blitter ocupado - sólo lectura.
13	BZERO	Estado ZERO del Blitter - sólo lectura.
12	X	
11	X	
10	BLTPRI	Prioridad del Blitter. Tambien llamado "Blitter desagradable." Cuando esta a 1 desconecta el pin /BLS, evitando que el 68000 robe ciclos de memoria mientras el DMA del Blitter esta en marcha.
9	DMAEN	Conexión todos los DMA siguientes.
8	BPLEN	Conecta el DMA de los Bit-planes.
7	COPEN	Conecta el DMA del Copper.
6	BLTEN	Conecta el DMA del Blitter.
5	SPREN	Conecta el DMA de los Sprites.
4	DSKEN	Conecta el DMA del disco.
3	AUD3EN	Conecta el DMA del canal de audio 3.
2	AUD2EN	Conecta el DMA del canal de audio 2.
1	AUD1EN	Conecta el DMA del canal de audio 1.
0	AUD0EN	Conecta el DMA del canal de audio 0.

DSKBYTR	$01A	R	Paula	Disk data byte and status read.

Este registro es el buffer de datos del microprocesador del disco. Los datos del disco (en modo de lectura) se cargan en este registro un byte cada vez, y el bit 15 (DSKBYT) se pone a 1.

Bit	Nombre	Función
15	DSKBYT	Byte preparado (pasa a 0 al leerlo).
14	DMAON	Espejo del bit 15 (DMAEN) de DSKLEN y (AND) el bit 9 (DMAEN) de DMACON.
13	DISKWRITE	Espejo del bit WRITE de DSKLEN.
12	WORDEQUAL	Este bit es sólo 1 sólo cuando el registro DSKSYNC es igual a los datos del disco.
11-8	X	Sin usar.
7-0	DATA	Byte de datos del disco.

DSKDAT	$026	W	Paula	Disk DMA data write
DSKDATR	$008	ER	Paula	Disk DMA data read

Este registro es el buffer de datos del DMA del disco. Contiene dos bytes de datos que son enviados (escritos) o recibidos (leidos) al/del disco.
El modo de escritura se conecta con el bit 14 del registro DSKLEN. El controlador de DMA transfiere automaticamente los datos a/o desde el disco y la RAM, y cuando los datos del DMA se terminan (longitud=0) provoca una interrupción de "bloque de disco finalizado". Ver las interrupciones.

DSKLEN	$024	W	Paula	Disk input.

Este registro contiene la longitud (número de palabras) del DMA del disco. Tambien contiene dos bits de control, un bit de conexión de DMA, y el bit de dirección del DMA (read/write) (lectura/escritura).

Bit	Nombre	Función
15	DMAEN	Conexión del DMA del disco.
14	WRITE	Escritura al disco (si es 1).
13-0	LENGTH	Longitud en palabras del DMA.

DSKPTH	$020	W	Agnus	Disk pointer (high 3 bits)
DSKPTL	$022	W	Agnus	Disk pointer (low 15 bits)

Este par de registros contiene la dirección de 18 bits de los datos del DMA del disco. Estos registros de direcciones los debe inicializar el 68000 o el Copper antes de que se conecte el DMA del disco.

DSKSYNC	$07E	W	Paula	Disk sync register.

Este registro retiene el código para la sincronización en las operaciones de lectura. Ver bit 10 de ADKCON.

INTENA	$09A	W	Paula	Interrupt enable bits (clear or set)
INTENAR	$01C	R	Paula	Interrupt enable bits (read)

Este registro contiene los bits que permiten las interrupciones. Las posiciones de los bits son las mismas para INTENA que para INTREQ.

Bit	Nombre	Nivel	Función
15	SET/CLR		Bit de control SET/CLR.
14	INTEN		Interrupción principal (sólo en INTENA)
13	EXTER	6	Interrpución externa.
12	DSKSYN	5	DSKSYNC coincide con datos del disco.
11	RBF	5	Buffer de recepción del RS232 lleno.
10	AUD3	4	Bloque del canal de audio 3 finalizado.
09	AUD2	4	Bloque del canal de audio 2 finalizado.
08	AUD1	4	Bloque del canal de audio 1 finalizado.
07	AUD0	4	Bloque del canal de audio 0 finalizado.
06	BLIT	3	Blitter finalizado.
05	VERTB	3	Inicio del vertical blank.
04	COPER	3	Copper.
03	PORTS	2	Ports de I/O y temporizadores.
02	SOFT	1	Para interrupciones de software.
01	DSKBLK	1	Bloque del disco finalizado.
00	TBE	1	Buffer de transmision del RS232 vacio.

INTREQ	$09C	W	Paula	Interrupt request bits (clear or set)
INTREQR	$01E	R	Paula	Interrupt request bits (read)

Este registro contiene los bits de petición de interrupción (o indicadores). Estos bits los puede modificar el 68000; si estan conectados por los bits de INTENA pueden causar las interrupciones del 68000. Se necesitan dos operaciones (1s y 0s) para poner datos arbitrarios en este registro. Estos bits de status no se ponen a 0 automaticamente cuando se sirve a la interrupción, y los debe poner el 68000 a 0 cuando se estime oportuno escribiendo a INTREQ. Las posiciones de los bits son las mismas que las de INTENA.

JOY0DAT	$00A	R	Denise	Joystick-mouse 0 data
JOY1DAT	$00C	R	Denise	Joystick-mouse 1 data

Cada una de estas direcciones acceden a dos contadores de 8 bits. 0 = port izquierdo, 1 = port derecho (cuatro contadores en total). Cada contador esta sincronizado con las señales de dos pines controladores. Los bits 1 y 0 de cada contador se pueden leer para determinar el estado de estos dos pines controladores. Esto permite usar estos pines como las entradas de los interruptores de un joystick.

(pines 1 y 3 = Contador Y, pines 2 y 4 = Contador X)

BIT:	15,14,13,12,11,10,09,08	07,06,05,04,03,02,01,00
0/1DAT:	Y7 Y6 Y5 Y4 Y3 Y2 Y1 Y0	X7 X6 X5 X4 X3 X2 X1 X0

La siguiente tabla muestra el uso de los pines del conector ratón/joystick. Los pines (y sus funciones) se muestrean (por multiplexado) en el chip DENISE durante los tiempos de reloj mostrados en la tabla. Esta tabla sólo es informativa y no es necesaria para programar. (Nótese que las funciones de los joysticks son "activas a nivel bajo" en los pines).

			Muestreo de DENISE	
Pin	Joystick	Ratón	Pin	Nombre Reloj
L1	ARRIBA*	Y	38	M0V en CCK
L3	IZQUI.*	YQ	38	M0V en CCK*
L2	ABAJO*	X	9	M0H en CCK
L4	DERECHA*	XQ	9	M0H en CCK*
R1	ARRIBA*	Y	39	M1V en CCK
R3	IZQUI.*	YQ	39	M1V en CCK*
R2	ABAJO*	X	8	M1H en CCK
R4	DERECHA*	XQ	8	M1H en CCK*

Despues de ser muestreados, estas señales de los pines se usan en cuadratura para sincronizar los contadores del ratón. Las funciones de IZQUIERDA y DERECHA (activas a nivel alto) del joystick estan disponibles directamente en los bits Y1 y X1 de cada contador. Sin embargo, para reconstruir las funciones del joystick ARRIBA y ABAJO, es necesario combinar con la función lógica XOR (EOR en asembler) los dos bits inferiores de cada contador.

Función	Bits que se deben leer
ARRIBA	Y1 xor Y0 (bit 9 xor bit 8)
IZQUIERDA	Y1
ABAJO	X1 xor X0 (bit 1 xor bit 0)
DERECHA	X1

JOYTEST $006 W Denise Write to all counters at once.

Este registro escribe a todos los contadores a la vez unos datos para realizar una prueba de los resultados:

BIT:	15,14,13,12,11,10,09,08	07,06,05,04,03,02,01,00
0/1DAT:	Y7 Y6 Y5 Y4 Y3 Y2 xx xx	X7 X6 X5 X4 X3 X2 xx xx

POT0DAT $012 R Paula Pot counter data left pair.
POT1DAT $014 R Paula Pot counter data right pair.

Cada uno de estas direcciones acceden a un par de contadores de 8 bits (cuatro POTS en total). Los contadores se detienen por las señales de los pines (derecho e izquierdo) de cada conector.

BIT:	15,14,13,12,11,10,09,08	07,06,05,04,03,02,01,00
DER/IZQ:	Y7 Y6 Y5 Y4 Y3 Y2 xx xx	X7 X6 X5 X4 X3 X2 xx xx

CONECTORES				PAULA	
Posición	Eje	Simb.	Pin	Pin	Nombre
DERECHA	Y	RY	9	36	POT1Y
DERECHA	X	RX	5	35	POT1X
IZQUIERDA	Y	LY	9	33	POT0Y
IZQUIERDA	X	LX	5	32	POT0X

POTGO $034 W PAULA Pot port data write and start
POTGOR $016 R Pot port data read (POTINP)

Este registro controla unport bidireccional de 4 bits que comparte los cuatro pines con los 4 contadores de los POTS.

Bit	Nombre	Función
15	OUTRY	Permite salida para el pin 36 de Paula.
14	DATRY	Dato I/O del pin 36 de Paula.
13	OUTRX	Permite salida para el pin 35 de Paula.
12	DATRX	Dato I/O del pin 35 de Paula.
11	OUTLY	Permite salida para el pin 33 de Paula.
10	DATLY	Dato I/O del pin 33 de Paula.
09	OUTLX	Permite salida para el pin 32 de Paula.
08	DATLX	Dato I/O del pin 32 de Paula.
07-01	0	Reservado para el código ID (0)
00	START	Inicia los POTS (descarga los condensadores, inicia los contadores.)

REFPTR $028 W Agnus Refresh pointer

Este registro se usa como un generador de direcciones para el refresco de las RAM dinamicas. Se puede escribir sólo para comprobaciones, y nunca lo debe escribir el 68000.

SERDAT $030 W Paula Serial port data and stop bits write.

Esta dirección escribe los datos a un buffer de transmisión de datos. Los datos de este buffer se mueven a un registro serie desplazador para realizar la transmisión de los datos cuando este se encuentra vacio. Esto activa la petición de interrupción TBE (buffer de transmisión vacio). Se debe añadir un bit de stop al byte de datos. La longitud de los datos la determina la posición del bit de stop.

BIT:	15,14,13,12,11,10,09,08,07,06,05,04,03,02,01,00
USO:	0 0 0 0 0 0 S D8 D7 D6 D5 D4 D3 D2 D1 D0

Nota: S = bit de stop, D = bits de datos.

SERDATR $018 R Paula Serial port data and status read

Esta dirección lee datos de un buffer de recepción de datos. Los datos de este buffer se cargan desde un registro desplazador de recepción cuando este se llena. Tambien se leen varios bits de status en esta dirección.

Bit	Nombre	Función
15	OVRUN	Indicador de OVERRUN (se ha recibido un byte, y el anterior aún no se habia procesado). Pasa a 0 poniendo a 0 el bit 11 de INTREQ.
14	RBF	Buffer de recepción del port serie lleno (espejo).
13	TBE	Buffer de transmisión del port serie vacio (no es un espejo)
12	TSRE	Registro desplazador de transmión vacio. Se pone a 0 cargando datos en el buffer, registro SERDAT.
11	RXD	Lectura directa del pin RXD del chip Paula.
10	0	No se usa actualmente.
9	STP	Bit de stop (en modo de 9 bits.)
8	STP/DB8	Bit de stop (en modo de 8 bits.) Bit 9 de datos (en modo de 9 bits.)
7-0	DB7-DB0	Bits inferiores de datos (8).

SERPER $032 W Paula Serial port period and control

Este registro contiene el bit de control LONG (bit 15 que define 8 o 9 bits para recepción), y un número de 15 bits (bits del 0 al 14) que define la velocidad en baudios del port serie. Si este número es N, entonces la velocidad en baudios es 1 bit cada (N+1)*.2794 µs.

SPRxCTL $142 W Ag, Dn Sprite x stop position and control data
SPRxPOS $140 W Ag, Dn Sprite x vert-horiz start position data.

Estos dos registros definen las caracteristicas del sprite: la posición, el tamaño y control del sprite. Normalmente son cargados por el DMA de los sprites durante el horizontal blank; sin embargo, tambien los pueden cargar el 68000 o el Copper en cualquier momento. Registro SPRxPOS:

Bit	Nombre	Función
15-8	SV7-SV0	Valor vertical inicial (SV8 en SPRxCTL).
7-0	SH8-SH1	Valor horizontal inicial (SH0 en SPRxCTL).

Registro SPRxCTL (escribiendo a esta dirección se desconecta el circuito de comparación horizontal).

Bit	Nombre	Función
15-8	EV7-EV0	Valor vertical final (8 bits inferiores).
7	ATT	Bit de control de conexión de sprites.
6-4	X	Sin usar.
2	SV8	Valor inicial vertical (bit superior)
1	EV8	Valor final vertical (bit superior)
0	SH0	Valor inicial horizontal (bit inferior)

SPRxDATA	$144	W	Denise	Sprite x image data register A
SPRxDATB	$144	W	Denise	Sprite x image data register B

Estos registros almacenan los datos de la imagen del sprite. Los cargan normalmente el DMA de los sprites pero tambien los pueden cargar el 68000 o el Copper. Cuando ocurre una comparación horizontal, estos registros se vuelcan en unos registros desplazadores y se van enviando en serie sus bits a la imagen, el bit mas significativo primero (izquierda).

Nota: Escribiendo al buffer A se conecta el sprite (se arma). Escribiendo al registro SPRxCTL se desconecta el sprite. Si esta conectado, los datos de los buffers A y B se visualizaran cuando el contador del rayo sea igual a la posición HORIZONTAL del registro SPRxPOS.

SPRxPOS (Ver SPRxCTL)

SPRPTH	$120	W	Agnus	Sprite x pointer (high 3 bits)
SPRPTL	$122	W	Agnus	Sprite x pointer (low 15 bits)

Este par de registros contiene la dirección de 18 bits de los datos para el DMA del sprite x (x=0,1,2,3,4,5,6,7). Estos registros de direcciones los debe inicializar el 68000 o el Copper en cada vertical blank.

STREQU	$038	S	Denise	Strobe horizontal sync with VB and EQU
STRHOR	$03C	S	Dn, Pl	Strobe horizontal sync
STRLONG	$03E	S	Denise	Strobe for long horizontal line

Una de las tres primeras direcciones de strobe mencionadas arriba se coloca en el bus de direcciones RGA durante el primer "slot" de refresco. El tercer strobe mostrado arriba se usa durante el segundo "slot" de refresco para identificar las lineas de mayor duración (228). Hay cuatro "slots" de refresco, y cualquiera que no se use para direcciones strobe dejara una dirección nula ($FF) en el registro de direcciones de los chips (RGA).

STRVBL	$03A	S	Denise	Strobe for horizontal sync with VB
VHPOSR	$006	R	Agnus	Read position of beam and light pen
VHPOSW	$02C	W	Agnus	Write position of beam and light pen

BIT: 15,14,13,12,11,10,09,08,07,06,05,04,03,02,01,00
USO: V7 V6 V5 V4 V3 V2 V1 V8 H8 H7 H6 H5 H4 H3 H2 H1

RESOLUCION = 1/160 del ancho de la pantalla (280 ns)

VPOSR	$004	R	Agnus	Read vertical most significant bit
VPOSW	$02A	W	Agnus	Write vertical most significant bit

BIT: 15,14,13,12,11,10,09,08,07,06,05,04,03,02,01,00
USO: LOF-- -- -- -- -- -- -- -- -- -- -- -- -- -- -- V8

LOF = Imagen larga

APENDICE B

SUMARIO DE REGISTROS - EN ORDEN DE DIRECCIONES

Este Apéndice contiene información sobre el conjunto de registros en orden de direcciones.

En este apéndice se usan los siguientes códigos (abreviaturas)

- & Registro usado sólo por canales DMA.
- % Registro usado por canales DMA normalmente, y por los procesadores algunas veces.
- + Par de registros de direcciones. Debe contener una dirección par apuntando a memoria CHIP.
- * Dirección no accesible por el Copper.
- - Dirección no accesible por el Copper excepto si el bit CDANO de COPCON se encuentra a 1.

W,R — W = sólo-escritura, R = sólo-lectura.

ER — Lectura temprana. Esto es una transferencia de datos de DMA a RAM, del disco o del Blitter. La temporización de la RAM necesita que los datos esten en el bus antes que los ciclos de lectura del 68000. Estas transferencias las inicia la temporización de Agnus, en lugar de leer una dirección en el bus RGA.

S — Strobe (dirección de escritura sin bits de registro). Escribiendo a esta dirección se causa el efecto.

PTL,PTH — Puntero a la memoria CHIP que direcciona los datos del DMA. Debe ser re-cargado por un procesador despues de su uso (en el vertical blank para los punteros de los bit-planes y lo sprites, y antes de poner en marcha al Blitter para los punteros del Blitter).

LCL,LCH — Localización en la memoria CHIP (dirección inicial) de los datos del DMA. Se usa para que los punteros se reinicialicen automaticamente, como el contador de programa del Copper (durante el vertical blank) y el contador de samples del audio (cuando la cuenta atras de la longitud termina).

MOD — Módulo de 15 bits. Es un número que se suma automaticamente a la dirección final de cada linea para generar la dirección inicial de la linea siguiente. Esto permite al Blitter (o a la ventana de visualización) operar en una ventana (o pantalla) de datos que es mas pequeña que la imagen de la memoria. Usa 15 bits, mas signo.

Registro	Dirección	R/W	CHIP	Función
BLTDDAT	& *070	ER	Agnus	Canal destino del Blitter.
DMACONR	*002	R	Ag, Pa	Control de DMA (y status del Blitter)
VPOSR	*004	R	Agnus	Bit superior posición vertical rayo.
VHPOSR	*006	R	Agnus	Posición del rayo y del lapiz óptico.
DSKDATR	& *008	ER	Paula	Datos del disco.
JOY0DAT	*00A	R	Denise	Datos del joystick/ratón 0.
JOY1DAT	*00C	R	Denise	Datos del joystick/ratón 1.
CLXDAT	*00E	R	Denise	Registro de datos de la colisión.
ADKCONR	*010	R	Paula	Registro de control de audio y disco.
POT0DAT	*012	R	Paula	Par de contadores 0 de los POTS.
POT1DAT	*014	R	Paula	Par de contadores 1 de los POTS.

POTGOR		*016	R	Paula	Datos del port de los POTS.
SERDATR		*018	R	Paula	Datos del port serie y status.
DSKBYTR		*01A	R	Paula	Byte del disco y status.
INTENAR		*01C	R	Paula	Bits de interrupciones permitidas.
INTREQR		*01E	R	Paula	Bits de interrupciones pedidas.
DSKPTH	+	*020	W	Agnus	Puntero del disco (3 bits superiores)
DSKPTL	+	*022	W	Agnus	Puntero del disco (15 bits inferiores)
DSKLEN		*024	W	Paula	Longitud del bloque del disco.
DSKDAT	&	*026	W	Paula	Dato DMA del disco.
REFPTR	&	*028	W	Agnus	Puntero de refresco.
VPOSW		*02A	W	Agnus	Bit superior posición vertical rayo.
VHPOSW		*02C	W	Agnus	Posición del rayo y del lapiz óptico.
COPCON		*02E	W	Agnus	Registro de control del Copper.
SERDAT		*030	W	Paula	Datos del port serie y bits de stop.
SERPER		*032	W	Paula	Periodo del port serie.
POTGO		*034	W	Paula	Datos del port de los POTS e inicio.
JOYTEST		*036	W	Denise	Escribe a los 4 contadores del ratón.
STREQU	&	*038	S	Denise	Strobe para sinc. horiz. con VB y EQU.
STRVBL	&	*03A	S	Denise	Strobe para sinc. horiz. con VB.
STRHOR	&	*03C	S	Dn, Pl	Strobe para sincronia horizontal.
STRLONG	&	*03E	S	Denise	Strobe para linea larga.
BLTCON0		-040	W	Agnus	Registro 0 de control del Blitter.
BLTCON1		-042	W	Agnus	Registro 1 de control del Blitter.
BLTAFWM		-044	W	Agnus	Primera palabra de mascara fuente A.
BLTALWN		-046	W	Agnus	Ultima palabra de mascara fuente A.
BLTCPTH	+	-048	W	Agnus	Puntero Blitter fuente C (3 bits sup.)
BLTCPTL	+	-04A	W	Agnus	Puntero Blitter fuente C (15 bits inf.)
BLTBPTH	+	-04C	W	Agnus	Puntero Blitter fuente B (3 bits sup.)
BLTBPTL	+	-04E	W	Agnus	Puntero Blitter fuente B (15 bits inf.)
BLTAPTH	+	-050	W	Agnus	Puntero Blitter fuente A (3 bits sup.)
BLTAPTL	+	-052	W	Agnus	Puntero Blitter fuente A (15 bits inf.)
BLTDPTH	+	-054	W	Agnus	Puntero Blitter destino D (3 bits sup.)
BLTDPTL	+	-056	W	Agnus	Puntero Blitter destino D (15 bits inf.)
BLTSIZE		-058	W	Agnus	Tamaño e inicio del Blitter.
		-05A			
		-05C			
		-05E			
BLTCMOD		-060	W	Agnus	Módulo para la fuente C del Blitter.
BLTBMOD		-062	W	Agnus	Módulo para la fuente B del Blitter.
BLTAMOD		-064	W	Agnus	Módulo para la fuente A del Blitter.
BLTDMOD		-066	W	Agnus	Módulo para el destino D del Blitter.
		-068			
		-06A			
		-06C			
		-06E			
BLTCDAT	%	-070	W	Agnus	Registro de datos fuente C del Blitter.
BLTBDAT	%	-072	W	Agnus	Registro de datos fuente B del Blitter.
BLTADAT	%	-074	W	Agnus	Registro de datos fuente A del Blitter.
		-076			
		-078			
		-07A			
		-07C			
DSKSYNC		-07E	W	Paula	Registro de sincronización del disco.
COP1LCH	+	080	W	Agnus	Registro 1 loc. Copper (3 bits sup.)
COP1LCL	+	082	W	Agnus	Registro 1 loc. Copper (15 bits inf.)
COP2LCH	+	084	W	Agnus	Registro 2 loc. Copper (3 bits sup.)
COP2LCL	+	086	W	Agnus	Registro 2 loc. Copper (15 bits inf.)
COPJMP1		088	S	Agnus	Reinicio del Copper en COP1LC.
COPJMP2		08A	S	Agnus	Reinicio del Copper en COP2LC.
COPINS		08C	W	Agnus	Instrucción obtenida por el DMA Copper.
DIWSTRT		08E	W	Agnus	Inicio ventana visualización.
DIWSTOP		090	W	Agnus	Parada ventana visualización.
DDFSTRT		092	W	Agnus	Inicio data-fetch (obtención datos).
DDFSTOP		094	W	Agnus	Fin data-fetch (obtención datos).
DMACON		096	W	A, D, P	Control del DMA.

CLXCON		098	W	Denise	Control de colisión.
INTENA		09A	W	Paula	Bits de interrupciones permitidas.
INTREQ		09C	W	Paula	Bits de interrupciones pedidas.
ADKCON		09E	W	Paula	Control del audio, disco y UART.
AUD0LCH	+	0A0	W	Agnus	Localización canal audio 0 (3 bits sup.)
AUD0LCL	+	0A2	W	Agnus	Localización canal audio 0 (15 bits inf)
AUD0LEN		0A4	W	Paula	Longitud bloque canal audio 0.
AUD0PER		0A6	W	Paula	Periodo del canal de audio 0.
AUD0VOL		0A8	W	Paula	Volumen del canal de audio 0.
AUD0DAT	&	0AA	W	Paula	Datos del canal de audio 0.
		0AC			
		0AE			
AUD1LCH	+	0B0	W	Agnus	Localización canal audio 1 (3 bits sup.)
AUD1LCL	+	0B2	W	Agnus	Localización canal audio 1 (15 bits inf)
AUD1LEN		0B4	W	Paula	Longitud bloque canal audio 1.
AUD1PER		0B6	W	Paula	Periodo del canal de audio 1.
AUD1VOL		0B8	W	Paula	Volumen del canal de audio 1.
AUD1DAT	&	0BA	W	Paula	Datos del canal de audio 1.
		0BC			
		0BE			
AUD2LCH	+	0C0	W	Agnus	Localización canal audio 2 (3 bits sup.)
AUD2LCL	+	0C2	W	Agnus	Localización canal audio 2 (15 bits inf)
AUD2LEN		0C4	W	Paula	Longitud bloque canal audio 2.
AUD2PER		0C6	W	Paula	Periodo del canal de audio 2.
AUD2VOL		0C8	W	Paula	Volumen del canal de audio 2.
AUD2DAT	&	0CA	W	Paula	Datos del canal de audio 2.
		0CC			
		0CE			
AUD3LCH	+	0D0	W	Agnus	Localización canal audio 3 (3 bits sup.)
AUD3LCL	+	0D2	W	Agnus	Localización canal audio 3 (15 bits inf)
AUD3LEN		0D4	W	Paula	Longitud bloque canal audio 3.
AUD3PER		0D6	W	Paula	Periodo del canal de audio 3.
AUD3VOL		0D8	W	Paula	Volumen del canal de audio 3.
AUD3DAT	&	0DA	W	Paula	Datos del canal de audio 3.
		0DC			
		0DE			
BPL1PTH	+	0E0	W	Agnus	Puntero bit-plane 1 (3 bits sup.)
BPL1PTL	+	0E2	W	Agnus	Puntero bit-plane 1 (15 bits inf.)
BPL2PTH	+	0E4	W	Agnus	Puntero bit-plane 2 (3 bits sup.)
BPL2PTL	+	0E6	W	Agnus	Puntero bit-plane 2 (15 bits inf.)
BPL3PTH	+	0E8	W	Agnus	Puntero bit-plane 3 (3 bits sup.)
BPL3PTL	+	0EA	W	Agnus	Puntero bit-plane 3 (15 bits inf.)
BPL4PTH	+	0EC	W	Agnus	Puntero bit-plane 4 (3 bits sup.)
BPL4PTL	+	0EE	W	Agnus	Puntero bit-plane 4 (15 bits inf.)
BPL5PTH	+	0F0	W	Agnus	Puntero bit-plane 5 (3 bits sup.)
BPL5PTL	+	0F2	W	Agnus	Puntero bit-plane 5 (15 bits inf.)
BPL6PTH	+	0F4	W	Agnus	Puntero bit-plane 6 (3 bits sup.)
BPL6PTL	+	0F6	W	Agnus	Puntero bit-plane 6 (15 bits inf.)
		0F8			
		0FA			
		0FC			
		0FE			
BPLCON0		100	W	Ag, Dn	Registro control bit-planes (misc.)
BPLCON1		102	W	Denise	Registro control bit-planes (scroll)
BPLCON2		104	W	Denise	Registro control bit-planes (prioridad)
		106			
BPL1MOD		108	W	Agnus	Módulo bit-planes (planos impares)
BPL2MOD		10A	W	Agnus	Módulo bit-planes (planos pares)
		10C			
		10E			
BPL1DAT	&	110	W	Denise	Datos bit-plane 1 (conv. paralelo-serie)
BPL2DAT	&	112	W	Denise	Datos bit-plane 2 (conv. paralelo-serie)
BPL3DAT	&	114	W	Denise	Datos bit-plane 3 (conv. paralelo-serie)
BPL4DAT	&	116	W	Denise	Datos bit-plane 4 (conv. paralelo-serie)
BPL5DAT	&	118	W	Denise	Datos bit-plane 5 (conv. paralelo-serie)

BPL6DAT	&	11A	W	Denise	Datos bit-plane 6 (conv. paralelo-serie)
		11C			
		11E			
SPR0PTH	+	120	W	Agnus	Puntero sprite 0 (3 bits inf.)
SPR0PTL	+	122	W	Agnus	Puntero sprite 0 (15 bits sup.)
SPR1PTH	+	124	W	Agnus	Puntero sprite 1 (3 bits inf.)
SPR1PTL	+	126	W	Agnus	Puntero sprite 1 (15 bits sup.)
SPR2PTH	+	128	W	Agnus	Puntero sprite 2 (3 bits inf.)
SPR2PTL	+	12A	W	Agnus	Puntero sprite 2 (15 bits sup.)
SPR3PTH	+	12C	W	Agnus	Puntero sprite 3 (3 bits inf.)
SPR3PTL	+	12E	W	Agnus	Puntero sprite 3 (15 bits sup.)
SPR4PTH	+	130	W	Agnus	Puntero sprite 4 (3 bits inf.)
SPR4PTL	+	132	W	Agnus	Puntero sprite 4 (15 bits sup.)
SPR5PTH	+	134	W	Agnus	Puntero sprite 5 (3 bits inf.)
SPR5PTL	+	136	W	Agnus	Puntero sprite 5 (15 bits sup.)
SPR6PTH	+	138	W	Agnus	Puntero sprite 6 (3 bits inf.)
SPR6PTL	+	13A	W	Agnus	Puntero sprite 6 (15 bits sup.)
SPR7PTH	+	13E	W	Agnus	Puntero sprite 7 (3 bits inf.)
SPR7PTL	+	13C	W	Agnus	Puntero sprite 7 (15 bits sup.)
SPR0POS	%	140	W	Ag, Dn	Posicion inicial sprite 0.
SPR0CTL	%	142	W	Ag, Dn	Posicion final sprite 0 y control.
SPR0DATA	%	144	W	Denise	Registro de datos A del sprite 0.
SPR0DATB	%	1	W	Denise	Registro de datos B del sprite 0.
SPR1POS	%	3	W	Ag, Dn	Posicion inicial sprite 1.
SPR1CTL	%	14A	W	Ag, Dn	Posicion final sprite 1 y control.
SPR1DATA	%	14C	W	Denise	Registro de datos A del sprite 1.
SPR1DATB	%	14E	W	Denise	Registro de datos B del sprite 1.
SPR2POS	%	150	W	Ag, Dn	Posicion inicial sprite 2.
SPR2CTL	%	152	W	Ag, Dn	Posicion final sprite 2 y control.
SPR2DATA	%	154	W	Denise	Registro de datos A del sprite 2.
SPR2DATB	%	156	W	Denise	Registro de datos B del sprite 2.
SPR3POS	%	158	W	Ag, Dn	Posicion inicial sprite 3.
SPR3CTL	%	15A	W	Ag, Dn	Posicion final sprite 3 y control.
SPR3DATA	%	15C	W	Denise	Registro de datos A del sprite 3.
SPR3DATB	%	15E	W	Denise	Registro de datos B del sprite 3.
SPR4POS	%	160	W	Ag, Dn	Posicion inicial sprite 4.
SPR4CTL	%	162	W	Ag, Dn	Posicion final sprite 4 y control.
SPR4DATA	%	164	W	Denise	Registro de datos A del sprite 4.
SPR4DATB	%	166	W	Denise	Registro de datos B del sprite 4.
SPR5POS	%	168	W	Ag, Dn	Posicion inicial sprite 5.
SPR5CTL	%	16A	W	Ag, Dn	Posicion final sprite 5 y control.
SPR5DATA	%	16C	W	Denise	Registro de datos A del sprite 5.
SPR5DATB	%	16E	W	Denise	Registro de datos B del sprite 5.
SPR6POS	%	170	W	Ag, Dn	Posicion inicial sprite 6.
SPR6CTL	%	172	W	Ag, Dn	Posicion final sprite 6 y control.
SPR6DATA	%	174	W	Denise	Registro de datos A del sprite 6.
SPR6DATB	%	176	W	Denise	Registro de datos B del sprite 6.
SPR7POS	%	178	W	Ag, Dn	Posicion inicial sprite 7.
SPR7CTL	%	17A	W	Ag, Dn	Posicion final sprite 7 y control.
SPR7DATA	%	17C	W	Denise	Registro de datos A del sprite 7.
SPR7DATB	%	17E	W	Denise	Registro de datos B del sprite 7.
COLOR00		180	W	Denise	Color 00 de la paleta.
COLOR01		182	W	Denise	Color 01 de la paleta.
COLOR02		184	W	Denise	Color 02 de la paleta.
COLOR03		186	W	Denise	Color 03 de la paleta.
COLOR04		188	W	Denise	Color 04 de la paleta.
COLOR05		18A	W	Denise	Color 05 de la paleta.
COLOR06		18C	W	Denise	Color 06 de la paleta.
COLOR07		18E	W	Denise	Color 07 de la paleta.
COLOR08		190	W	Denise	Color 08 de la paleta.
COLOR09		192	W	Denise	Color 09 de la paleta.
COLOR10		194	W	Denise	Color 10 de la paleta.
COLOR11		196	W	Denise	Color 11 de la paleta.
COLOR12		198	W	Denise	Color 12 de la paleta.
COLOR13		19A	W	Denise	Color 13 de la paleta.

COLOR14	19C	W	Denise	Color 14 de la paleta.
COLOR15	19E	W	Denise	Color 15 de la paleta.
COLOR16	1A0	W	Denise	Color 16 de la paleta.
COLOR17	1A2	W	Denise	Color 17 de la paleta.
COLOR18	1A4	W	Denise	Color 18 de la paleta.
COLOR19	1A6	W	Denise	Color 19 de la paleta.
COLOR20	1A8	W	Denise	Color 20 de la paleta.
COLOR21	1AA	W	Denise	Color 21 de la paleta.
COLOR22	1AC	W	Denise	Color 22 de la paleta.
COLOR23	1AE	W	Denise	Color 23 de la paleta.
COLOR24	1B0	W	Denise	Color 24 de la paleta.
COLOR25	1B2	W	Denise	Color 25 de la paleta.
COLOR26	1B4	W	Denise	Color 26 de la paleta.
COLOR27	1B6	W	Denise	Color 27 de la paleta.
COLOR28	1B8	W	Denise	Color 28 de la paleta.
COLOR29	1BA	W	Denise	Color 29 de la paleta.
COLOR30	1BC	W	Denise	Color 30 de la paleta.
COLOR31	1BE	W	Denise	Color 31 de la paleta.
RESERVADO	1110%			
RESERVADO	1111%			
NO-OP(NADA)	1FE			

APENDICE C

LISTA DE LAS POSICIONES DE LOS PINES EN LOS CUSTOM CHIPS

NOTAS: * = señal activa a nivel bajo.
I = pin de entrada al chip.
O = pin de salida del chip.

POSICIONES DE LOS PINES EN PAULA

PIN	NOMBRE	FUNCION	DIRECCION
1-7	D8-D2	Lineas del bus de datos 8 a 2	I/O
8	VSS	Masa (-0V)	I
9-10	D1-D0	Lineas del bus de datos 1 y 0	I/O
11	RES*	Reset del sistema	I
12	DMAL	Linea de petición de DMA	O
13-15	IPL0*-IPL2	Lineas de interrupciones 0-2	O
16	INT2*	Nivel de interrupción 2	I
17	INT3*	Nivel de interrupción 3	I
18	INT6*	Nivel de interrupción 6	I
19-26	RGA8-RGA1	Bus de direcciones de los registros	I
27	VCC	+5 Voltios	I
28	CCK	Reloj de color	I
29	CCKQ	Reloj de color retrasado (desfasado)	I
30	AUDB	Audio derecho	O
31	AUDA	Audio izquierdo	O
32	POY0X	POTenciometro 0X	I/O
33	POT0Y	POTenciometro 0Y	I/O
34	VSSANA	Masa analógica.	I
35	POT1X	POTenciometro 1X	I/O
36	POT1Y	POTenciometro 1Y	I/O
37	DKRD*	Datos leidos del disco.	I
38	DKWD*	Datos escritos al disco.	O
39	DKWE	Permiso de escritura al disco	O
40	TXD	Transmisión de datos al port serie	O
41	RXD	Recepción de datos del port serie	I
42-48	D15-D9	Lineas del bus de datos 15 a 9	I/O

POSICIONES DE LOS PINES EN DENISE

PIN	NOMBRE	FUNCION	DIRECCION
1-7	D6-D0	Lineas del bus de datos 6 a 0	I/O
8	M1H	Ratón 1 horizontal	I
9	M0H	Ratón 0 horizontal	I
10-17	RGA8-RGA1	Bus de direcciones de los registros	I
18	BURST*	Burst de color	O
19	VCC	+5 Voltios	I
20-23	R0-R3	Bits de video rojo 0 - 3	O
24-27	B0-B3	Bits de video azul 0 - 3	O
28-31	G0-G3	Bits de video verde 0 - 3	O
32	N/C	Sin conectar	N/C
33	ZD*	Indicador de decorado	O
34	N/C	Sin conectar	N/C
35	7M	7.15909 MHz	I
36	CCK	Reloj de color	I
37	VSS	Masa (-0V)	I
38	M0V	Ratón 0 vertical	I
39	M1V	Ratón 1 vertical	I
40-48	D15-D7	Lineas del bus de datos 15 a 7	I/O

POSICIONES DE LOS PINES EN AGNUS

PIN	NOMBRE	FUNCION	DIRECCION
1-9	D8-DØ	Lineas del bus de datos 8 a Ø	I/O
1Ø	VCC	+5 Voltios	I
11	RES*	Reset del sistema	I
12	INT3*	Nivel de interrupciones 3	O
13	DMAL	Linea de peticion de DMA	I
14	BLS*	Blitter slowdown	I
15	DBR*	Petición del bus de datos	O
16	ARW*	Escritura de Agnus a RAM	O
17-24	RGA8-RGA1	Bus de direcciones de los registros 8 - 1	I/O
25	CCK	Reloj de color	I
26	CCKQ	Reloj de color retardado	I
27	VSS	Masa (-ØV)	I
28-36	DRAØ-DRA8	Bus de direcciones DRAM Ø a 8	O
37	LP*	Entrada del lapiz óptico	I
38	VSY*	Sincronia vertical	I/O
39	CSY*	Sincronia compuesta	O
4Ø	HSY*	Sincronia horizontal	I/O
41	VSS	Masa (-ØV)	I
42-48	D15-D9	Lineas del bus de datos 15 a 9	I/O

POSICIONES DE LOS PINES EN FAT AGNUS

PIN	NOMBRE	FUNCION	DIRECCION
1-14	RD15-RD2	Bus de datos de los registros 15 a 2	I/O
15	VCC	+5 Voltios	I
16	RST*	Reset	I
17	INT3*	Interrupción nivel 3	O
18	DMAL	Petición de DMA audio/disco.	I
19	BLS*	Blitter slowdown	I
20	DBR*	Petición del bus de datos	O
21	RRW	DRAM write/read (escritura/lectura en DRAM)	O
22	PRW.	Proccessor write/read	I
23	RGEN*	RG enable	I
24	AS*	Adress Strobe	I
25	RAMEN*	RAM enable (permiso de RAM)	I
26-33	RGA8-RGA1	Bus de direcciones de los registros 8 - 1	O
34	28MHz	Reloj principal de 28 MHz	I
35	XCLK	Reloj alternativo al principal (externo)	I
36	XCLKEN*	Permiso de reloj principal	I
37	CDAC*	Reloj de 7MHz invertido y desplazado	O
38	7MHz	Reloj de 28 MHz dividido entre 4	O
39	CCKQ	Reloj de color retrasado (desfase)	O
40	CCK	Reloj de color	O
41	TEST	Test - acceso a los registros	I
43-51	MA0-MA8	Lineas de salida del bus 0 a 8	O
52	LDS*	Strobe de los datos inferiores	I
53	UDS*	Strobe de los datos superiores	I
54	CASL*	Column addr strobe lower	O
55	CASU*	Column addr strobe upper	O
56	RAS1*	Row address strobe one	O
57	RAS0*	Row address strobe zero	O
58	VSS	Masa (-ØV)	I
59-77	A19-A1	Bus de direcciones 19 a 1	I
78	LP*	Lapiz óptico	O
79	VSY*	Sincronia vertical	I/O
80	CSY*	Sincronia compuesta	O
81	HSY*	Sincronia horizontal	I/O
84	RD0	Bus de datos de los registro 0	I/O

APENDICE D

MAPA DE LA MEMORIA DEL SISTEMA

No se puede construir un mapa valido de memoria del software que muestre la utilización de la RAM por las diversas rutinas de la ROM. Toda la memoria la reserva dinamicamente el "distribuidor" de memoria, y las posiciones asctuales cambiaran con las distintas versiones de la ROM, los distintos modelos y los distintos discos de arranque. Para encontrar las posiciones de las estructuras de datos del sistema operativo se deben usar los procedimientos de acceso ya definidos, comenzando por obtener la dirección de la librería exec "exec.library" de la posición $4 de memoria; que es la única posición absoluta en el sistema operativo. Todo el software se escribe para que se pueda cargar y colocar en cualquier parte de la memoria. El esquema que sigue a continuación es una visión general de las distintas areas de memoria en el espacio de direccionamiento del 68000.

RANGO DE DIRECCIONES	NOTAS
$000000 - $03FFFF	256 K de CHIP RAM
$040000 - $07FFFF	256 K de CHIP RAM
$080000 - $0FFFFF	512 K de CHIP RAM extendida (hasta 1 MB)
$100000 - $1FFFFF	Reservado. No usar.
$200000 - $9FFFFF	Espacio de 8 MB de auto-configuración (FAST RAM).
$A00000 - $BEFFFF	Reservado. No usar.
$BFD000 - $BFDF00	Chip 8520_B (acceso en el la dirección par)
$BFE001 - $BFEF01	Chip 8520_A (acceso en el la dirección impar) (El digito subrayado inidica a cual de los 16 registros internos de los chips 8520 se esta accediendo con esa dirección. Ver Apendice F.)
$C00000 - $DFEFFF	Reservado. No usar.
! $C00000 - $D7FFFF	Memoria de expansión interna (PSEUDO-FAST RAM)
! $D80000 - $DBFFFF	Reservado. No usar.
! $DC0000 - $DCFFFF	Reloj de tiempo real.
! $DFF000 - $DFFFFF	Registros de los Custom Chips. Ver Apendices A y B
$E00000 - $E7FFFF	Reservado. No usar.
$E80000 - $E8FFFF	Espacio de auto-configuración. Las tarjetas aparecen aqui antes de que el sistema las desplace a sus posiciones finales.
$E90000 - $EFFFFF	Espacio secundario de auto-configuración (normalmente 64K de tarjetas de Entrada/Salida)
$F00000 - $FBFFFF	Reservado. No usar.
$FC0000 - $FFFFFF	ROM de 256K.

APENDICE E

INTERFACES

Este Apendice consta de cuatro partes distintas, relativas a la forma en que el Amiga se comunica con el mundo exterior.

La primera parte especifica las posiciones de los pines en los conectores que se pueden acceder desde fuera y la tensión disponible en cada conector. Pero no da información sobre la temporización o la carga.

La segunda parte describe brevemente las funciones de los pines que no tienen un propósito evidente.

La tercera parte contiene una lista de las conexiones de los conectores internos, como el disco.

La cuarta parte especifica cómo acceder a las señales de los ports de los chips 8520. Esta información permite al programador utilizar las direcciones de los ports directamente (o las señales internas).

Las partes tercera y cuarta son para el uso por programadores del sistema hardware y generalmente no las utilizan los programadores de aplicaciones.

Los sistemas de software estan configurados normalmente para manejar un conjunto de señales particulares, independientemente de que puedan cambiar las conexiones fisicas. En otras palabras, cuando se pregunta por el estado de un determinado bit, no hace falta preocuparse del port al que esta conectado. Por lo tanto, los programadores de aplicaciones deben confiar en las rutinas del sistema operativo en lugar de acceder directamente a los ports.

NOTA: En un sistema operativo multitarea, todas las tareas pueden acceder a los recursos del sistema. Los programadores de aplicaciones deben seguir las normas establecidas al acceder a los recursos para asegurar la compatibilidad del software con el sistema.

PARTE 1 - CONEXIONES CON EL MUNDO EXTERIOR

PORT PARALELO (CENTRONICS)

PIN	A1000	A500/2000	PCs de COMMODORE
1	DRDY*	STROBE*	STROBE*
2	Dato 0	Dato 0	Dato 0
3	Dato 1	Dato 1	Dato 1
4	Dato 2	Dato 2	Dato 2
5	Dato 3	Dato 3	Dato 3
6	Dato 4	Dato 4	Dato 4
7	Dato 5	Dato 5	Dato 5
8	Dato 6	Dato 6	Dato 6
9	Dato 7	Dato 7	Dato 7
10	ACK*	ACK*	ACK*
11	BUSY (data)	BUSY	BUSY
12	POUT (clk)	POUT	POUT
13	SEL	SEL	SEL
14	GND	+5V	AUTOFDXT
15	GND	NC	ERROR*
16	GND	RESET*	INIT*
17	GND	GND	SLCT IN*
18-22	GND	GND	GND
23	+5V	GND	GND
24	NC	GND	GND
25	Reset*	GND	GND

RS232 y PORT DE MIDI

PIN	RS232	A1000	A500/2000	PCs	HAYES	DESCRIPCION
1	GND	GND	GND	GND	GND	Masa de imagen
2	TXD	TXD	TXD	TXD	TXD	Datos transmitidos
3	RXD	RXD	RXD	RXD	RXD	Datos recividos
4	RTS	RTS	RTS	RTS	-	Petición para enviar
5	CTS	CTS	CTS	CTS	CTS	Preparado para enviar
6	DSR	DSR	DSR	DSR	DSR	
7	GND	GND	GND	GND	GND	Masa del sistema
8	CD	CD	CD	DCD	DCD	Detector de portadora
9	-	-	+12V	+12V	-	Tensión de +12V
10	-	-	-12V	-12V	-	Tensión de -12V
11	-	-	AUDO	-	-	Salida de audio
12	S.SD	-	-	-	SI	Indicador de velocidad
13	S.CTS	-	-	-	-	
14	S.TXD	-5Vdc	-	-	-	Tensión de -5V
15	TXC	AUDO	-	-	-	Salida de audio
16	S.RXD	AUDI	-	-	-	Entrada de audio
17	RXC	EB	-	-	-	Reloj 716 KHz amplific.
18	-	INT2*	AUDI	-	-	Interrupción al Amiga
19	S.RTS	-	-	-	-	
20	DTR	DTR	DTR	DTR	DTR	Terminal datos preparado
21	SQD	+5V		-	-	tensión de +5V
22	RI	-	RI	RI	RI	Indicador de llamada
23	SS	+12V	-	-	-	Tensión de +12V
24	TXC1	C2*	-	-	-	Reloj de 3.58 MHz
25	-	RESB*	-	-	-	Reset del sistema ampl.

DISCO EXTERNO ...DB23 HEMBRA

Para A500, A1000 y A2000 (En el A2000 SEL1B* no es la unidad 1, sino la primera unidad externa. No estan previstas todas las lineas).

1	RDY*	13	SIDEB*
2	DKRD*	14	WPRO*
3	GND	15	TK0*
4	GND	16	DKWEB*
5	GND	17	DKWDB*
6	GND	18	STEPB*
7	GND	19	DIRB
8	MTRXD*	20	SEL3B* (Sin usar, A2000)
9	SEL2B* (SEL3B, A2000)	21	SEL1B* (SEL2B, A2000)
10	DRESB*	22	INDEX*
11	CHNG*	23	+12V
12	+5V		

VIDEO ...DB23 MACHO

Para el A500, A1000 y A2000.

1	XCLK*	13	GNDRTN (Retorno para XCLKEN*)
2	XCLKEN*	14	ZD*
3	RED	15	C1*
4	GREEN	16	GND
5	BLUE	17	GND
6	DI	18	GND
7	DB	19	GND
8	DG	20	GND
9	DR	21	-5V (A1000/2000), -12V (A500)
10	CSYNC*	22	+12V
11	HSYNC*	23	+5V
12	VSYNC*		

EXPANSION ...86 PINES (.1) (P2)

1	GND	44	/IPL2
2	GND	45	A18
3	GND	46	BEER*
4	GND	47	A17
5	+5VDC	48	/VPA
6	+5VDC	49	GND
7	No Conectado	50	E Clock
8	-5VDC	51	/VMA
9	N.C. (28 MHz Clock &$)	52	A18
10	+12VDC	53	RST
11	N.C. (/COPCFG $)	54	A19
12	CONFIG IN, Grounded	55	/HLT
13	GND	56	A20
14	/C3 Clock	57	A22
15	Reloj CDAC	58	A21
16	/C1 Clock	59	A23
17	/OVR	60	/BR (/CBR $)
18	RDY	61	GND
19	/INT2	62	/BGACK
20	N.C (/PALOPE %) (/BOSS $)	63	D15
21	A5	64	/BG (/CBG $)
22	/INT6	65	D14
23	A6	66	/DTACK
24	A4	67	D13
25	GND	68	R/W
26	A3	69	D12
27	A2	70	/LDS
28	A7	71	D11
29	A1	72	/UDS
30	A8	73	GND
31	FC0	74	/AS
32	A9	75	D0
33	FC1	76	D10
34	A10	77	D1
35	FC2	78	D9
36	A11	79	D2
37	GND	80	D8
38	A12	81	D3
39	A13	82	D7
40	/IPL0	83	D4
41	A14	84	D6
42	/IPL1	85	GND
43	A15	86	D5

% - Sólo A1000. & - Sólo A2000. $ - Sólo A2000b.

TECLADO ..RJ11

	A1000	A2000	A500
1	+5V	KCLK	?
2	CLOCK	KDAT	?
3	DATA	NC	?
4	GND	GND	?
5		+5V	?

MONITOR RF ...DIN 8 PINES (J2) Sólo A1000

1	N.C.
2	GND
3	AUDIO IZQUIERDO
4	VIDEO COMPUESTO
5	GND
6	N.C.
7	+12V
8	AUDIO DERECHO

EXPANSION DE RAM ...60 PINES (.156) (P1) Sólo A1000

1	GND	16	D0	A	GND	T	D1
2	D15	17	GND	B	D14	U	GND
3	+5V	18	DRA4	C	+5V	V	DRA3
4	D12	19	DRA5	D	D13	W	DRA2
5	GND	20	DRA6	E	GND	X	DRA1
6	D11	21	DRA7	F	D10	Y	DRA0
7	+5V	22	GND	H	+5V	Z	GND
8	D8	23	RAS*	J	D9	AA	RRW*
9	GND	24	GND	K	GND	BB	GND
10	D7	25	GND	L	D6	CC	GND
11	+5V	26	CASU0*	M	+5V	DD	CASU1*
12	D4	27	GND	N	D5	EE	GND
13	GND	28	CASL*	P	GND	FF	CASL1*
14	D3	29	+5V	R	D2	HH	+5V
15	+5V	30	+5V	S	+5V	JJ	+5V

JOYSTICKS ...DB9 MACHO

PIN	JOYSTICK	RATON
1	ARRIBA*	V
2	ABAJO*	H
3	IZQUIERDA*	VQ
4	DERECHA*	HQ
5	POT X	Botón 3
6	DISPARO*	Botón 1
7	+5V	+5V
8	GND	GND
9	POT Y	Botón 2

PARTE 2 - MAS SOBRE EL MUNDO EXTERIOR

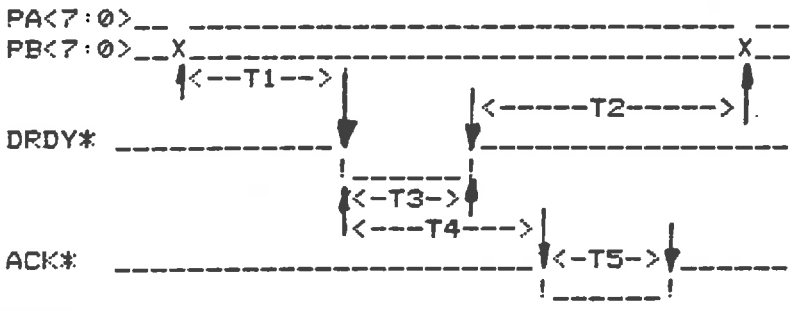
ESPECIFICACIONES SOBRE EL CONECTOR E INTERFACE PARALELO

El conector de 25 pines tipo D (DB25P - macho para el A1000, hembra para A500/2000 y PCs) de la parte trasera del Amiga se usa normalmente como interface para impresoras paralelas. En esta aplicación, los datos salen del Amiga hacia la impresora. Pero este interface tambien puede usarse como entrada o transmisiones de datos bidireccionales. Este interface es similar al Centronics, pero las funciones de los pines y las caracteristicas de funcionamiento varian significativamente respecto al Centronics.

FUNCIONES DE LOS PINES DEL PORT PARALELO

NOMBRE	DIR	NOTAS
DRDY*	O	Señal de "datos preparados" al dispositivo paralelo en modo de salida de datos, se usa junto con ACK* (pin 10) como un handshake asincrono de dos lineas. En modo de entrada de datos funciona como señal de "datos aceptados" (similar a ACK* en modo de salida).
D0	I/O	D0-D7 comprenden un bus bidireccional de ocho bits para comunicaciones con dispositivos paralelos: impresoras, digitalizadores, etc.
D1	I/O	
D2	I/O	
D3	I/O	
D4	I/O	
D5	I/O	
D6	I/O	
D7	I/O	
ACK*	I	En el modo de salida es la señal de "datos reconocidos" que envia el dispositivo paralelo al Amiga, se usa junto con DRDY como hanshake asincrono. En modo de entrada de datos funciona como señal de "datos preparados" del dispositivo paralelo al Amiga (similar a DRDY* en modo de salida)
BUSY	I/O	Este es un pin de propósito general I/O que tambien esta conectado al pin 12 del port serie.
POUT	I/O	Este es un pin de propósito general I/O que tambien esta conectado al pin 11 del port serie.
SEL	I/O	Este es un pin de propósito general I/O. Normalemnte sirve como selección de salida de datos para el dispositivo paralelo. En el A500/A2000 esta compartido con la señal del RS232 "indicador de llamada".
RESET*	O	Reset del sistema del Amiga.

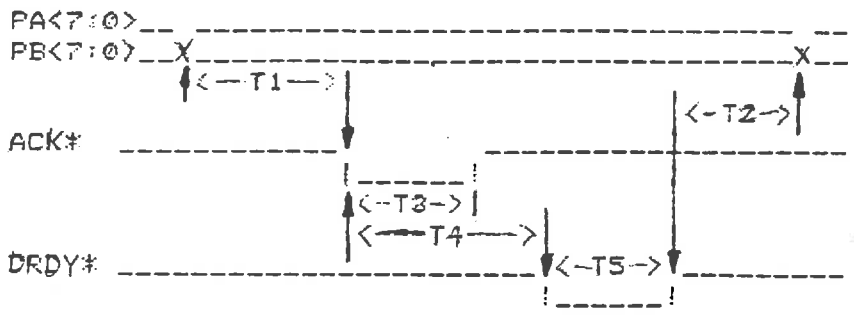
TEMPORIZACION DEL PORT PARALELO, EN MODO DE SALIDA



	Microsegundos Min	Tip	Max	
T1:	4.3	-x-	5.3	Tiempo de puesta a punto de los datos en la salida.
T2:	nsp	-x-	upc	Tiempo de retención de los datos.
T3:	nsp	1.4	nsp	Ancho del impulso de "datos preparados".
T4:	0	-x-	upc	Tiempo para recivir el impulso de "datos reconocidos".
T5:	nsp	-x-	upc	Ancho del impulso de "datos reconocidos".

nsp = no especificado. upc = bajo el control del programa.

TEMPORIZACION DEL PORT PARALELO, EN MODO DE ENTRADA



	Microsegundos Min	Tip	Max	
T1:	0	-x-	upc	Tiempo de puesta a punto de los datos en la entrada.
T2:	nsp	-x-	upc	Tiempo de retención de los datos.
T3:	nsp	-x-	upc	Ancho del impulso de "datos preparados".
T4:	upc	-x-	upc	Tiempo para emitir el impulso de "datos reconocidos".
T5:	nsp	1.4	nsp	Ancho del impulso de "datos reconocidos".

nsp = no especificado. upc = bajo el control del programa.

ESPECIFICACIONES DEL CONECTOR SERIE

Este conector de 25 pines (DB25S=hembra) se usa como interface para las señales standard del RS232C.

ATENCION: En los pines del conector RS232 hay señales standard, y otras diversas tensiones y señales no-standard. Cuando se fabrican cables para RS232, se deben conectar sólo los pines que se van a usar. No se deben usar los cables "straight-thru" (cables standard para otros ordenadores).

POSICIONES DE LOS PINES EN EL INTERFACE SERIE

NOMBRE	DIR	STD	NOTAS
FGND		si	Masa de la imagen - no unir a la masa de la señal
TXD	O	si	Transmisión de datos
RXD	I	si	Recepción de datos
RTS	O	si	Petición para enviar
CTS	I	si	Preparado para enviar
DSR	I	si	Data set ready (¿conjunto de datos preparado?)
GND		si	Masa de la señal - no unir a la masa de la imagen
CD	I	si	Detección de portadora
-5V		no*	-5V, 50ma maximo.
AUDO	O	no*	Salida de audio (canles 0 y 3), pensado para enviar sonido al modem.
AUDI	I	no*	Entrada de sonido (canales 1 y 2), pensado para oir el sonido del modem mezclado con el de los canales 1 y 2. El ordenador no usa este sonido, ni lo digitaliza de ninguna forma.
DTR	O	si	Terminal de datos preparado.
RI	I	si	Indicador de llamada (sólo A500 y A2000)
RESB*	O	no*	Reset del sistema del Amiga.

no* : ver la advertencia anterior.

TEMPORIZACION DEL INTERFACE SERIE

La frecuencia maxima de funcionamiento es de 19.2 KHz. Consultar el standard EIA del RS232C las caracteristicas del funcionamiento e instalación.

Las señales de control del modem (CTS, RTS, DTR, DSR, CD) estan completamente bajo el control del software. Las lineas de control del modem no tienen efecto sobre el hardware y son completamente asincronas a TxD y RXD.

CARACTERISTICAS ELECTRICAS DEL CONECTOR SERIE

SALIDAS	MIN	TIP	MAX		
Vo(-):	-13.2	-x-	-2.5	V	Rango de voltaje negativo de salida.
Vo(+):	8.0	-x-	13.2	V	Rango de voltaje positivo de salida.
Io:	-x-	-x-	10.0	ma	Corriente de salida.

ENTRADAS	MIN	TIP	MAX		
Vo(-):	-25.0	-x-	0.5	V	Rango de voltaje negativo de entrada.
Vo(+):	3.0	-x-	25.0	V	Rango de voltaje positivo de entrada.
Vhys:	-x-	1.0	-x-	V	Histeresis de voltage en la entrada.
Io:	0.3	-x-	10.0	ma	Corriente de entrada.

Las entradas sin conectar se interpretan igual que los voltajes positivos.

ESPECIFICACIONES DEL INTERFACE DE JOYSTICK

Los dos conectores de 9 pines (macho) se usan como interface para cuatro tipos de dispositivos:

1. Ratón o trackball, 3 botones max.
2. Joystick digital, 2 botones max. (3 teoricamente)
3. Joystick proporcional, 2 botones max. (5 teoricamente)
4. Lapiz óptico, incluyendo botón en la punta.

J11 es el conector derecho (JOY1DAT, POT1DAT)
J12 es el conector izquierdo (JOY0DAT, POT0DAT)

Nota: Aunque todo el hardware que se explica aqui se puede acceder directamente, se aconseja acceder a el a traves del software de la ROM para que los cambios futuros en el hardware sean transparentes al usuario.

USANDO RATON/TRACKBALL

Un ratón o un trackball es un dispositivo que traduce los movimientos sobre un plano en trenes de impulsos. Se usan las tecnicas de cuadratura para transmitir la dirección y magnitud del desplazamiento. Los registros JOY0DAT y JOY1DAT son registros contadores, con el desplazamiento del eje Y en el byte superior y el del eje X en el byte inferior. Los movimientos provocan las siguientes acciones:

Arriba:	La Y se decrementa.
Abajo:	La Y se incrementa.
Derecha:	La X se incrementa.
Izquierda:	La X se decrementa.

Para determinar el desplazamiento, se debe leer JOYxDAT dos veces restando sus correspondientes valores de X y de Y (arimetica de módulo 128). Nótese que si alguno de los contadores cambia en mas de 127 unidades, la dirección y la velocidad se hacen ambiguas. Hay una relación entre el intervalo de la toma de muestras y la velocidad maxima (cambio en la distancia):

Velocidad < Distancia / tiempo de muestreado

Velocidad < $\sqrt{(X^2 + Y^2)}$ / tiempo de muestreado

En un Amiga con un ratón de 200 impulsos por pulgada y toma de muestras en cada vertical blank, la velocidad maxima en cualquiera de los dos ejes es:

Velocidad < (128 impulsos * 1 pulgada / 200 impulsos) / .017 s = 38 pulg/s

que sera suficiente para la mayoria de los usuarios.

Nota: El software del Amiga esta hecho para leer los contadores del ratón durante el vertical blank, pero estos se pueden leer en cualquer momento.

USO DE LAS ENTRADAS DE CUADRATURA PARA RATON/TRACKBALL

PIN	NOMBRE	DESCRIPCION	REGISTROS DEL HARDWARE
1	V	Impulsos verticales	JOY0/1DAT <15:8>
2	H	Impulsos horizontales	JOY0/1DAT <7:0>
3	VQ	Cuadratura de V	JOY0/1DAT <15:8>
4	HQ	Cuadratura de H	JOY0/1DAT <7:0>
5	UBUT*	Botón sin usar	Ver entradas de los POTS
6	LBUT*	Botón izquierdo	Ver botón de disparo
7	+5V	+5V de poca intensidad	
8	GND	Masa	
9	RBUT*	Botón derecho	Ver entradas de los POTS

USANDO JOYSTICKS DIGITALES

Un joystick es un dispositivo que tiene cuatro interruptores "normalmente abiertos", uno en cada dirección. Los registros JOY0/1DAT se pueden decodificar para averiguar el estado de estos interruptores:

Arriba:	BIT 9 XOR BIT 8
Izquierda:	BIT 9
Abajo:	BIT 1 XOR BIT 0
Derecha:	BIT 1

Los datos estan codificados para facilitar el uso por el ratón/trackball.

Nota: Las entradas de izquierda y derecha estan diseñadas para ser tambien los botones derecho e izquierdo en los joysticks proporcionales. En este caso, las entradas de arriba y abajo no se usan.

Los registros JOY0/1 son siempre validos y se pueden leer en cualquier momento.

USO DE LAS ENTRADAS PARA JOYSTICK DIGITAL

PIN	NOMBRE	DESCRIPCION	REGISTROS DEL HARDWARE
1	ARRIBA*	Interruptor superior	JOY0/1DAT <9 xor 8>
2	ABAJO*	Interruptor inferior	JOY0/1DAT <1 xor 0>
3	IZQUIERDA*	Interruptor izquierdo	JOY0/1DAT <9>
4	DERECHA*	Interruptor derecho	JOY0/1DAT <1>
5	Sin usar		
6	DISPARO*	Botón izquierdo	Ver botón de disparo
7	+5V	125ma max. de 200ma	Total para ambos ports
8	GND	Masa	
9	Sin usar		

CONECTANDO LOS BOTONES DE DISPARO

Los botones de disparo son interruptores "normalmente abiertos" conectados al registro PRA0 del chip 8520.A. El bit 7 de PRA0 es el botón de disparo del port izquierdo y el bit 6 es lo mismo, del port derecho.

Antes de leer este registro, los bits correspondientes del registro de direccion se deben poner a Ø para definir esta linea como entrada.

DDRAØ<7:6> puestos a cero es lo apropiado.

Nota: No alterar el estado de los otros bits de DDRA (Se recomienda usar las rutinas de la ROM)

USO DE LAS ENTRADAS PARA LOS BOTONES DE DISPARO

PIN	NOMBRE	DESCRIPCION
1	-x-	
2	-x-	
3	-x-	
4	-x-	
5	-x-	
6	DISPARO*	Botón izquierdo del ratón/botón de disparo.
7	-x-	
8	GND	
9	-x-	

USANDO JOYSTICKS PROPORCIONALES

Se pueden usar potenciometros lineales de hasta 528 Kohm max. (se recomienda 47Ø K +/-1Ø%). Los registros JOYØ/1DAT contienen la traduccion directa a valores digitales para la Y (en el byte superior) y la X (en el byte inferior). Un valor alto indica que el potenciometro externo tiene una resistencia muy alta. El Amiga hace la conversion digital-a-analogica como sigue:

1. Durante las primeras 7 (NTSC) u 8 (PAL) lineas horizontales de la imagen los condensadores de la entrada se descargan y los contadores de los registros POTØ/1DAT se mantienen a Ø.

2. El voltaje va incrementandose gradualmente y se compara constantemente con un nivel de referencia interno mientras los contadores van contando el numero de lineas desde el final del intervalo de reset.

3. Cuando el voltaje entrante excede al umbral interno en un determinado POT, el valor de su contador se bloquea en el registro POTØ/1DAT correspondiente a ese POT.

4. Durante el vertical blank, el software examina el resultado del registro POTØ/1DAT e interpreta los valores como la posicion del joystick.

NOTA: Las entradas POTY y POTX estan diseñadas como "botón derecho del ratón" y "botón no usado del ratón" respectivamente. Un interruptor abierto equivale a una resistencia elevada, un interruptor cerrado equivale a una resistencia baja. Los botones tambien estan disponibles en los registros POTGO y POTINP. Se recominda que se usen las rutinas de la ROM por compatibilidad con futuras versiones del hardware.

Es importante entender que el joystick proporcional es un dispositivo de posiciones relativas mas que un dispositivo de posiciones absolutas. Es tarea del software efectuar la calibracion, la limitacion del rango util y la media de los resultados.

Los registros POTØ/1 se leen tipicamente durante el vertical blank, pero pueden estar preparados antes que éste.

USO DE LAS ENTRADAS PARA LOS JOYSTICKS PROPORCIONALES

PIN	NOMBRE	DESCRIPCION	
1	XBUT	Botón extra	
2	Sin usar		
3	LBUT*	Botón izquierdo	Ver joystick digital
4	RBUT*	Botón derecho	Ver joystick digital
5	POT X	Entrada X analogica	POTØ/JDAT<7:0>, POTGO, POTINP
6	Sin usar		
7	+5V	125 ma max	
8	GND		
9	POT Y	Entrada Y analogica	POTØ/JDAT<15:8>, POTGO, POTINP

USANDO UN LAPIZ OPTICO

Un lapiz óptico es un dispositivo foto-electrico que consta de un elemento sensible a la luz que esta colocado en proximidad a la pantalla. Cuando el rayo de electrones pasa por delante del lapiz óptico, se genera un impulso que bloquea los registros de posicion del rayo. No hay ningun bit en el hardware que indique esto, pero se puede averiguar con los dos metodos explicados en el capitulo 8, "El Hardware de Interface".

La posicion del lapiz óptico se lee normalmente durante el vertical blank, pero puede estar preparada antes que este.

USANDO LAS ENTRADAS PARA UN LAPIZ OPTICO

PIN	NOMBRE	DESCRIPCION	
1	Sin usar		
2	Sin usar		
3	Sin usar		
4	Sin usar		
5	LPENPR*	Lapiz óptico toca pantalla	Ver entradas proporcionales
6	LPENTG*	Impulsos del lapiz óptico.	VPOSR, VHPOSR
7	+5V	max 125 ma	
8	GND		
9	Sin usar		

Nota: dependiendo del fabricante, las entradas del lapiz óptico podran ser otras.

ESPECIFICACIONES DEL INTERFACE DE DISCO EXTERNO

Los conectores de 23 pines (DB23S) de la parte trasera del Amiga se usan como interface directo a los dispositivos MFM.

POSICIONES DE LOS PINES EN EL CONECTOR DE DISCO EXTERNO (J7)

PIN	NOMBRE	DIR	NOTAS
1	RDY*	I/O	Si el motor está encendido, indica que ha llegado a la velocidad adecuada. Si no lo esta, es el modo de identificación. Se explica mas adelante.
2	DKRD*	I	Entrada de datos del MFM al Amiga
3	GND		
4	GND		
5	GND		
6	GND		
7	GND		
8	MTRXD*	OC	Datos de motor encendido, se envían a los flip-flops del motor cuando se produce la transicion de SELxB*. El tiempo garantizado de puesta a punto es de 1.4 µs. El tiempo garantizado de bloqueo es de 1.4 µs.
9	SEL2B*	OC	Selecciona la unidad 2.*
10	DRESB*	OC	Reset del sistema del Amiga. Las unidades deben poner a 0 los flip-flops del motor y a 1 los flip-flops de porteccion de escritura.
11	CHNG*	I/O	Nota: Se usa nominalmente como entrada de colector abierto. El flip-flop de cambio se pone a 1 al encender o cuando no hay disco. El flip-flop se pone a 0 cuando la unidad esta seleccionada y se mueve el cabezal, pero solo si hay un disco introducido.
12	+5V		Max. 270 ma; de 410 ma. Cuando esta cerca de 3.75 V es necesario que las unidades de disco pongan a 0 sus flip-flops del motor, y pongan a 1 sus flip-flops de proteccion de escritura.
13	SIDEB*	O	Si esta activo selecciona la cara 1, si no la cara 0.
14	WPRO*	I/O	El disco de la unidad seleccionada esta protegido de escritura.
15	TK0*	I/O	La unidad seleccionada indica cuando el cabezal esta en la pista 0.
16	DKWEB*	OC	Puerta de escritura (se conecta) a la unidad.
17	DKWDB*	OC	Salida de datos del Amiga al MFM.
18	STEPB*	OC	El disco seleccionado mueve el cabezal una pista en la direccion indicada en DIRB.
19	DIRB	OC	Direccion para desplazar el cabezal. Cuando esta inactiva (0) el cabezal se mueve hacia el centro del disco (se aleja de la pista 0)
20	SEL3B*	OC	Selecciona la unidad 3. *
21	SEL1B*	OC	Selecciona la unidad 1. *
22	INDEX*	I/O	El indice es un impulso generado una vez por cada revolucion del disco entre el final y el inicio de los cilindros. El 8520 se puede programar para generar una interrupcion de nivel 6 cuando INDEX* se activa.
23	+12V		160 ma maximo; de 540 ma.

* Nota: Las lineas de seleccion del disco se van desplazando a medida que atraviesan las cadenas DAISY de las unidades de disco. Por lo tanto la señal SEL2B* en la primera unidad, en la segunda unidad sera SEL1B* y no llegara a la tercera.

MODO DE IDENTIFICACION DE LA UNIDAD EXTERNA

Se ha incluido un modo de identificacion para leer una cadena de 32 bits de la unidad externa. Para inicializar este modo, el motor se debe encender, y despues apagar. Ver en el pin 8, MTRXD* como encender y apagar el motor. La transicion de motor encendido a motor apagado reinicializa el registro serie de desplazamiento.

Despues de la inicializacion, la señal SELxB* se debe dejar en estado inactivo (1).

Ahora se entra en un bucle donde SELxB* se activa, se lee el dato serie de RDY* (pin 1), y se vuelve a desactivara SELxB*. Este bucle se repite 32 veces para leer los 32 bits de datos. El bit mas significativo se recibe primero.

IDENTIFICACIONES DEFINIDAS DE LA UNIDAD EXTERNA

$0000 0000 - No hay ninguna unidad en esta linea.
$FFFF FFFF - Unidad standard Amiga de 3.25".
$5555 5555 - 48 TPI doble cara, doble densidad.

Con otros códigos ID, los usuarios deben contactar con Commodore Amiga para mas información.

La entrada de datos serie es activa en estado bajo y por tanto se debe invertir para leer los datos reales.

LIMITACIONES DEL CONECTOR DE UNIDADES EXTERNAS

1. La longitud total del cable, incluyendo la cadena DAISY, no debe exceder a un metro.

2. El maximo de unidades es de 3, aunque otras configuraciones soportaran menos unidades.

3. Cada dispositivo debe tener una resistencia de 1000 ohm en las salidas de colector abierto del Amiga (pines 8-10, 16-21)

4. El sistema suministra energia solo para la primera unidad externa de la cadena DAISY, las otras dos deberan tener su propia fuente de alimentación.

PARTE 3 - CONECTORES INTERNOS

DISCO INTERNO ..34 PINES (J10)

1	GND	18	DIRB
2	CHNG*	19	GND
3	GND	20	STEPB*
4	MTR0D*(led)	21	GND
5	GND	22	DKWDB*
6	N.C.	23	GND
7	GND	24	DKWEB*
8	INDEX	25	GND
9	GND	26	TK0*
10	SEL0B*	27	GND
11	GND	28	WPRO*
12	N.C.	29	GND
13	GND	30	DKRD*
14	N.C.	31	GND
15	GND	32	SIDEB*
16	MTR0D*	33	GND
17	GND	34	RDY*

ENERGIA DEL DISCO INTERNO ...4 PINES (J13)

1	+12V (Algunas unidades solo tienen +5V)
2	GND
3	GND
4	+5V

PARTE 4 - POSICIONES DE LAS SEÑALES EN LOS 8520

DIRECCION $BFFR01 BITS DE DATOS 7-0 (A12*) (INT2)

```
PA7..Pin 6, PORT 1, (Botón de disparo)
PA6..Pin 6, PORT 0, (Botón de disparo)
PA5..RDY*       Disco preparado*
PA4..TK0*       Pista 00*
PA3..WPRO*      Protección de escritura*
PA2..CHNG*      Cambio de disco*
PA1..LED*       Led (0=encendido), en A500/2000 conecta el filtro de audio
PA0..OVL*       Bit de overlay ROM/RAM

SP...KDAT       Datos del teclado
CNT..KCLK       Reloj del teclado

PB7..P7         Dato 7
PB6..P6         Dato 6
PB5..P5         Dato 5
PB4..P4         Dato 4     Datos del interface
PB3..P3         Dato 3     paralelo (centronics)
PB2..P2         Dato 2
PB1..P1         Dato 1
PB0..P0         Dato 0

PC...drdy*          Control del Centronics
F....ack*
```

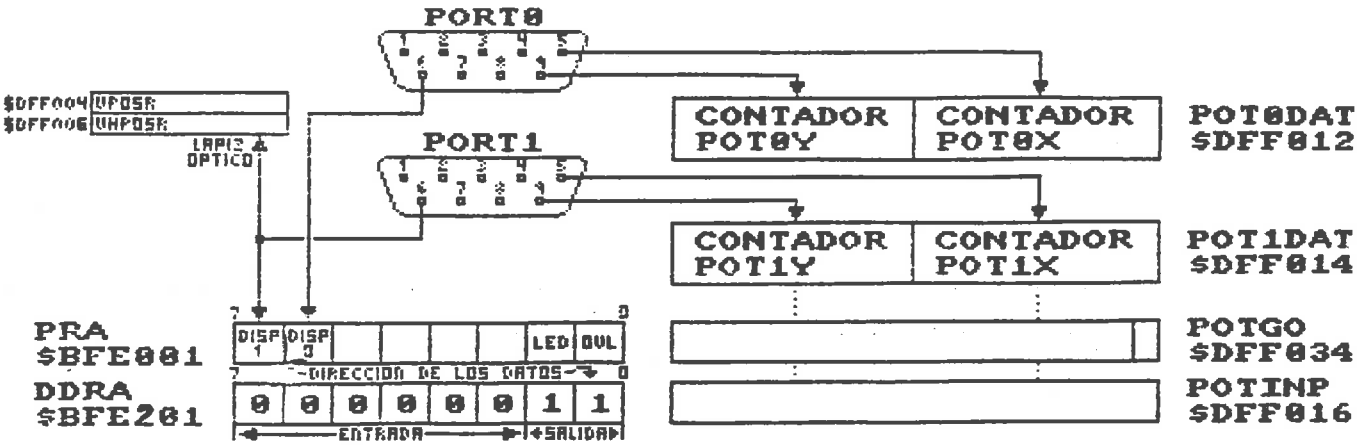
DIRECCION $BFDR00 BITS DE DATOS 15-8 (A13*) (INT6)

```
PA7..DTR*       Linea al RS232 (salida)
PA6..RTS*       Linea al RS232 (salida)
PA5..CD*        Linea al RS232 (entrada)
PA4..CTS*       Linea al RS232 (entrada)
PA3..DSR*       Linea al RS232 (entrada)
PA2..SEL        Control de Centronics
PA1..POUT       Impresora sin papel.
PA0..BUSY       Impresora ocupada.

SP...BUSY       Bus serie de Commodore. (sincrono)
CNT..POUT       Bus serie de Commodore.

PB7..MTR*       Motor
PB6..SEL3*      Selecciona el tercer disco externo.
PB5..SEL2*      Selecciona el segundo disco externo.
PB4..SEL1*      Selecciona el primer disco externo.
PB3..SEL0*      Selecciona el disco interno.
PB2..SIDE*      Selección de cabezal (cara del disco)*
PB1..DIR        Dirección.
PB0..STEP*      Movimiento del cabezal.

PC...no usado
F....INDEX*     Impulso del indice del disco.
```



Botones de disparo, POTs y Lapiz optico.

APENDICE F

CHIPS 8520 ADAPTADORES COMPLEJOS DE INTERFACES (CIA)

Cada sistema Amiga contiene dos chips 8520 Adaptadores complejos de interfaces (CIA). Cada chip tiene 16 pines de propósito general de entrada/salida, mas un registro desplazador serie, tres temporizadores, un pin de salida de impulsos y una entrada de detección de impulsos. En el sistema del Amiga hay varias tareas asignadas a las posibilidades de los chips.

Mapa de direcciones del CIA.A

Dirección del byte	Nombre registro	Bits de datos 7	6	5	4	3	2	1	0
$BFE001	pra	/FIR1	/FIR0	/RDY	/TK0	/WPRO	/CHNG	/LED	OVL
$BFE101	prb	Port paralelo							
$BFE201	ddra	Dirección del port A ($BFE001); 1=salida (puesto a $03)							
$BFE301	ddrb	Dirección del port B ($BFE101); 1=salida							
$BFE401	talo	CIA.A temporizador A byte inferior (.709379 Mhz en PAL)							
$BFE501	tahi	CIA.A temporizador A byte superior (.709379 Mhz en PAL)							
$BFE601	tblo	CIA.A temporizador B byte inferior (.709379 Mhz en PAL)							
$BFE701	tbhi	CIA.A temporizador B byte superior (.709379 Mhz en PAL)							
$BFE801	todlo	Contador de eventos 50/60 Hz, bits 7-0							
$BFE901	todmid	Contador de eventos 50/60 Hz, bits 15-8							
$BFEA01	todhi	Contador de eventos 50/60 Hz, bits 13-16							
$BFEB01		sin usar							
$BFEC01	acr	CIA.A registro serie de datos (conectado al teclado)							
$BFED01	icr	CIA.A registro de control de interrupciones							
$BFEE01	cra	CIA.A regitro de control A							
$BFEF01	crb	CIA.A regitro de control B							

Nota: CIA.A puede generar una interrupción INT2

Mapa de direcciones del CIA.B

Dirección del byte	Nombre registro	Bits de datos 7	6	5	4	3	2	1	0
$BFD000	pra	/DTR	/RTS	/CD	/CTS	/DSR	/SEL	POUT	BUSY
$BFD100	prb	/MTR	/SEL3	/SEL2	/SEL1	/SEL0	/SIDE	DIR	/STEP
$BFD200	ddra	Dirección del port A ($BFD000); 1=salida (puesto a $C0)							
$BFD300	ddrb	Dirección del port B ($BFD100); 1=salida (puesto a $FF)							
$BFD400	talo	CIA.B temporizador A byte inferior (.709379 Mhz en PAL)							
$BFD500	tahi	CIA.B temporizador A byte superior (.709379 Mhz en PAL)							
$BFD600	tblo	CIA.B temporizador B byte inferior (.709379 Mhz en PAL)							
$BFD700	tbhi	CIA.B temporizador B byte superior (.709379 Mhz en PAL)							
$BFD800	todlo	Contador de eventos sincronia horizontal, bits 7-0							
$BFD900	todmid	Contador de eventos sincronia horizontal, bits 15-8							
$BFDA00	todhi	Contador de eventos sincronia horizontal, bits 13-16							
$BFDB00		sin usar							
$BFDC00	acr	CIA.B registro serie de datos (sin usar)							
$BFDD00	icr	CIA.B registro de control de interrupciones							
$BFDE00	cra	CIA.B regitro de control A							
$BFDF00	crb	CIA.B regitro de control B							

Nota: CIA.B puede generar una interrupción INT6

MAPA DE LOS REGISTROS DE LOS CHIPS

Cada 8520 tiene 16 registros que se pueden leer o escribir. Esta es la lista de direcciones de cada uno dentro del especio de memoria dedicado a los 8520:

RS3	RS2	RS1	RS0	Registro	NOMBRE	SIGNIFICADO
0	0	0	0	0	pra	Registro A de datos de perifericos
0	0	0	1	1	prb	Registro B de datos de perifericos
0	0	1	0	2	ddra	Registro A de dirección
0	0	1	1	3	ddrb	Registro B de dirección
0	1	0	0	4	talo	Registro inferior temporizador A
0	1	0	1	5	tahi	Registro superior temporizador B
0	1	1	0	6	tblo	Registro inferior temporizador A
0	1	1	1	7	tbhi	Registro superior temporizador B
1	0	0	0	8	todlo	Eventos 7-0
1	0	0	1	9	todmed	Eventos 15-8
1	0	1	0	A	todhi	Eventos 23-16
1	0	1	1	B		Sin conectar
1	1	0	0	C	sdr	Registro de datos serie
1	1	0	1	D	ocr	Registro de control interrupciones
1	1	1	0	E	cra	Registro A de control
1	1	1	1	F	crb	Registro B de control

NOTA DE SOFTWARE:

El sistema operativo kernel utiliza la mayoria de los temporizadores de los chips 8520.

CIA.A, temporizador A — TECLADO (lo usa el hardware continuamente para recivir las pulsaciones de teclas) NO USAR.

CIA.A, temporizador B — Dispositivo virtual de temporización (usado continuamente cuando Exec esta funcionando; se usa para el cambio de tareas en el entorno multitarea, las interrupciones y temporizaciones)

CIA.A, TOD — Temporizador de 50/60 Hz usado por timer.device. El A1000 usa los ciclos de la red. El A500 usa la sincronia vertical. En el A2000 se selecciona con un puente en la circuiteria.

CIA.B, temporizador A — sin usar

CIA.B, temporizador B — sin usar

CIA.B, TOD — Seguidor del rayo para graphics.library. Este temporizador cuenta a la velocidad de la sincronia horizontal (15625 Hz en PAL) y se usa para sincronizar los eventos graficos con el rayo de electrones.

Las anteriores ediciones de esta tabla eran incorrectas.

DESCRIPCION DEL FUNCIONAMIENTO DE LOS REGISTROS

PORTS DE I/O (PRA, PRB, DDRA, DDRB)

Los Ports A y B constan cada uno de un registro de datos de 8 bits (PR) y un registro de dirección de 8 bits (DDR). Si un bit de DDR esta a 1, la posición correspondiente del registro PR se convierte en salida. Si un bit de DDR esta a 0, el bit correspondiente de PR se convierte en entrada.

Cuando se lee un registro PR, se lee el estado actual de los pines I/O (PA0-PA7,PB0-PB7), independientemente de que sean salidas o entradas.

Los ports A y B tienen dispositivos pasivos y activos de polarización, proporcionando compatibilidad con TTL y CMOS. Ambos ports tienen dos drives de carga TTL.

Ademas de las operaciones normales de I/O, los ports PB6 y PB7 tambien pueden tener la salida de los temporizadores.

HANDSHAKING

El Handshaking ocurre en las transferencias de datos usando el pin PC de salida y el pin FLAG de entrada (por ejemplo cuando se mandan datos a la impresora). PC pasara a estado bajo en el tercer ciclo despues de un acceso al port B. Esta señal se puede usar para indicar "datos preparados" en el port B o "datos aceptados" desde el port B. En las transmisiones de 16 bits (usando los dos ports) tambien es posible el handshaking siempre que se escriba o lea el port A primero. La entrada FLAG es sensible a los bordes negativos de la señal y se puede usar para recibir la salida PC de otro 8520 o como entrada de interrupción de propósito general. Cualquier transición negativa en FLAG pondra a 1 el bit FLAG de interrupción.

REG	NOMBRE	D7	D6	D5	D4	D3	D2	D1	D0
0	PRA	PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0
1	PRB	PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0
2	DDRA	DPA7	DPA6	DPA5	DPA4	DPA3	DPA2	DPA1	DPA0
3	DDRB	DPB7	DPB6	DPB5	DPB4	DPB3	DPB2	DPB1	DPB0

TEMPORIZADORES DE INTERVALOS (A Y B)

Cada temporizador de intervalos consiste en un contador de 16 bits de sólo-lectura y un latch de 16 bits de sólo-escritura. Los datos escritos al temporizador se bloquean en el latch, mientras que los datos leidos del temporizador son el contenido presente en el contador del temporizador.

El latch tambien es llamado preescalar porque representa la cuenta atras que se debe realizar antes de llegar al instante de tiempo requerido. El latch (preescalar) es un divisor de la frecuencia de entrada. Los temporizadores se pueden usar independientemente o unidos para operaciones extendidas. Los diversos modos de funcionamiento de los temporizadores permiten la generación de largos intervalos de tiempo, impulsos de ancho variable, trenes de impulsos, y ondas de frecuencia variable. Utilizando la entrada CNT, los temporizadores pueden contar impulsos externos o medir frecuencias, ancho de los impulsos y tiempos de retardo de señales externas.

Cada temporizador tiene un registro de control asociado, proporcionando control independiente para cada una de las siguientes funciones:

Start/stop

Es un bit de control que permite al 68000 parar y poner en marcha al temporizador en cualquier momemto.

PB on/off

Es un bit de control que permite que la salida del temporizador aparezca en una linea de salida del port B (PB6 para el temporizador A y PB7 para el temporizador B). Esta función ignora los bits de DDRB y hace que la linea correspondiente de PB sea una salida.

Toggle/pulse

Es un bit de control que selecciona la salida aplicada al port B cuando PB on/off esta conectado. Cada vez que la cuenta atras del temporizador llega a 0, la salida puede cambiar de estado o generar un impulso de un ciclo de duración. La salida esta en estado alto cuando se inicia la temporización.

One-shot/continuous

Es un bit de control que selecciona el modo del temporizador. En el modo de "one-shot", el temporizador realiza la cuenta atras, genera una interrupción, recarga el valor bloqueado en el latch, y para. En el modo "continuous", el temporizador realiza la cuenta atras, genera una interrupción, recarga el valor del latch y repite el proceso continuamente.

En el modo "one-shot", si se escribe al temporizador superior (registro 5 del temporizador A, o registro 7 del temporizador B) se transferira el valor del latch al contador y se iniciara la cuenta atras independientemente del bit "start/stop".

Force load

Es un bit de strobe que permite al latch del temporizador cargarse en el contador en cualquier momento, independientemente de que este en marcha o no.

MODOS DE ENTRADA

Los bits de control permiten la seleccion del reloj usado en la cuenta atras del temporizador. El temporizador A puede contar Φ2 impulsos del reloj o los impulsos aplicados al pin CNT. El temporizador B puede contar dos impulsos del reloj, impulsos externos del pin CNT, impulsos de "cuenta atras finalizada" del temporizador A, o impulsos de "cuenta atras finalizada" del temporizador A mientras el pin CNT se mantiene en estado alto.

El latch se carga en en el temporizador cuando la cuenta atras se finaliza, cuando se fuerza la carga (force load), o despues de escribir al byte superior del procesador mientras el temporizador esta parado. Si el temporizador esta funcionando y se escribe al byte superior, se cargara el latch pero no el contador.

← NOMBRES

NOMBRES DE LOS BITS EN EL REGISTRO DE SOLO-LECTURA

REG	NOMBRE	D7	D6	D5	D4	D3	D2	D1	D0
4	TALO	TAL7	TAL6	TAL5	TAL4	TAL3	TAL2	TAL1	TAL0
5	TAHI	TAH7	TAH6	TAH5	TAH4	TAH3	TAH2	TAH1	TAH0
6	TBLO	TBL7	TBL6	TBL5	TBL4	TBL3	TBL2	TBL1	TBL0
7	TBHI	TBH7	TBH6	TBH5	TBH4	TBH3	TBH2	TBH1	TBH0

NOMBRES DE LOS BITS EN EL REGISTRO DE SOLO-ESCRITURA

REG	NOMBRE	D7	D6	D5	D4	D3	D2	D1	D0
4	PRA	PAL7	PAL6	PAL5	PAL4	PAL3	PAL2	PAL1	PAL0
5	PRB	PAH7	PAH6	PAH5	PAH4	PAH3	PAH2	PAH1	PAH0
6	DDRA	PBL7	PBL6	PBL5	PBL4	PBL3	PBL2	PBL1	PBL0
7	DDRB	PBH7	PBH6	PBH5	PBH4	PBH3	PBH2	PBH1	PBH0

TIEMPO DEL RELOJ DEL DIA

El TOD consiste en un contador binario de 24 bits. Las transiciones positivas en este pin hacen que el contador binario se incremente. El pin TOD tiene un pull-up pasivo (para polarizar).

Hay una alarma programable para generar una interrupción en el momento programado. Los registros de la alarma estan colocados en la misma dirección que los del TOD. El acceso de la alarma se controla mediante un bit de control. La alarma es de sólo-escritura: una lectura a la dirección del TOD dara el tiempo independientemente del estado del bit ALARM.

Se debe seguir una secuencia especifica de acceso para leer y escribir adecuadamente el TOD. TOD se detiene automaticamente cuando se escribe al registro. El reloj no se pondra en marcha hasta que se escriba al byte inferior. Esto asegura que TOD siempre se iniciara en el momento adecuado.

Debido a que puede ocurrir una transición de un estado a otro mientras se lee el registro, se ha incluido una función de bloqueo que mantendra toda la información de TOD constante durante la secuencia de lectura. Todos los registros del TOD se bloquean cuando se lee el byte superior y continuan bloqueados hasta que se lee el byte inferior. El reloj TOD continua contando aunque los registros esten bloqueados. Si sólo se desea leer un registro, no hay problema de transición y el registro se puede leer "en el aire" teniendo en cuenta que si se lee el byte superior, se debera leer despues el inferior para desconectar el bloqueo.

BITS DE WRITE TIME/ALARM o READ TIME

REG	NOMBRE							
8	TODLO	E7	E6	E5	E4	E3	E2	E1	E0
9	TODMED	E15	E14	E13	E12	E11	E10	E9	E8
A	TODHI	E23	E22	E21	E20	E19	E18	E17	E16

CRB7 = 0 Para escribir el TOD, CRB7 = 1 para escribir la alarma

REGISTRO DE DATOS SERIE (SDR)

El port serie es un registro de 8 bits sincrono. Un bit de control selecciona el modo de entrada o de salida. En el sistema del Amiga un registro desplazador se usa para el teclado, y el otro esta sin usar. Nótese que el port serie compatible con RS232 lo controla el chip Paula; ver el Capitulo 8 para mas detalles.

MODO DE ENTRADA

En el modo de entrada, los datos del pin SP se introducen en el registro desplazador en la transición negativa de la señal aplicada al pin CNT. Despues de ocho impulsos de CNT, los datos del registro desplazador se vuelcan en el registro de datos serie y se genera una interrupción.

MODO DE SALIDA

En el modo de salida, se usa el temporizador A como generador del tren de impulsos para la comunicación sincrona. Los datos se desplazan fuera del registro desplazador sobre el pin SP a la mitad de la velocidad del temporizador A. La maxima velocidad posible es 02 ciclos dividido entre 4, pero la velocidad maxima usable la determinara la carga de la linea y la velocidad a la que el receptor responde a los datos entrantes.

Para iniciar la transmisión, se debe preparar el temporizador A en modo continuo, e iniciar el temporizador. La transmisión se iniciara despues de escribir al registro de datos serie. La señal de reloj derivada del temporizador A aparece como salida en el pin CNT. Los datos del registro de datos serie se cargaran en el registro desplazador, y entonces se desplazaran por el pin SP junto con los impulsos CNT. Los datos desplazados se hacen validos en la transición negativa de CNT y continuan validos hasta la siguiente transición negativa.

Despues de ocho impulsos de CNT, se genera una interrupción para indicar que se pueden enviar mas datos. Si el registro serie de datos se vuelve a cargar con mas información a continuación de esta interrupción, los nuevos datos se cargaran automaticamente en el registro desplazador y la transmisión continuaran.

Si no se transmiten mas datos despues del octavo impulso de CNT, CNT volvera al estado alto y SP quedara en el nivel del último bit transmitido.

Los datos de SDR se desplazan empezando con el bit mas significativo. Los datos de la entrada serie aparecen en el mismo formato.

CARACTERISTICA BIDIRECCIONAL

La capacidad de bidireccionalidad del registro desplazador y los impulsos de CNT permite que se conecten varios 8520s a un bus común de comunicaciones donde un 8520 actua como principal, proporcionando los datos y los impulsos de sincronismo. mientras que los demas 8520 actuan como esclavos. Ambas salidas CNT y SP son de drenador abierto (open drain) permitiendo un bus común. El protocolo para elegir al chip 8520 principal se puede transmitir por el propio bus o por lineas dedicadas a estos fines.

REG	NOMBRE	D7	D6	D5	D4	D3	D2	D1	D0
C	SDR	S7	S6	S5	S4	S3	S2	S1	S0

REGISTRO DE CONTROL DE INTERRUPCIONES

Hay cinco fuentes de interrupciones en un 8520:

- Cuenta atras del temporizador A (cuando llega a 0).
- Cuenta atras del temporizador B.
- Alarma del TOD.
- Port serie vacio/lleno.
- Flag.

Un registro de control contiene las mascaras y la información de las interrupciones. El registro de control de interrupciones consisste de un registro MASK de sólo-lectura y otro registro DATA de sólo-lectura. Cualquier interrupción pondra a 1 el bit correspondiente del registro DATA, y si en esa posición corresponde a un 1 en el registro MASK, se activara el bit IR del registro DATA y el pin IRQ pasara a estado bajo. En un sistema multichip, el bit IR se puede utilizar para averiguar que chip ha generado la interrupción.

Cuando se lee el registro DATA, se borra su contenido (se pone a 0), y la linea IRQ vuelve al estado alto. Como al leer el registro se borra su contenido, hay que asegurarse de que el programa que maneja la interrupción responda a todos los bits que estaban activados en el registro DATA en el momento en que se leia. Con la adecuada preservación de los datos y la respuesta a todos ellos, es posible entremezclar facilmente los metodos de servicio de interrupción directo y de lista.

Se puede activar o desactivar uno o mas bits del registro MASK sin afectar el estado de los demas bits del registro. Esto se hace poniendo el bit mas significativo de MASK al estado adecuado. En los bits 6-0 se forma una mascara que especifica a que bits se quiere afectar. Entonces, usando el bit 7, se especifica cómo se quiere afectar a las posiciones de la mascara.

- Si el bit 7 es un 1, entonces los bits 6-0 de la mascara que estan a 1, haran que los bits correspondientes de MASK tambien se pongan a 1. Cualquier bit que este a 0 hara que el bit correspondiente de MASK continue en su estado.

- Si el bit 7 es un 0, entonces los bits 6-0 de la mascara que estan a 1, haran que los bits correspondientes de MASK tambien se pongan a 0. Cualquier bit que este a 0 hara que el bit correspondiente de MASK continue en su estado.

Si debe ocurrir una interrupción basada en una condición particular, entonces el bit correspondiente de MASK debe ser un 1.

Ejemplo: Supóngase que se desea activar el bit de la interrupción del temporizador A (permitir la interrupción del temporizador), pero asegurandose de que todas las demas interrupciones estan desactivadas.

```
INCLUDE    "hardware/cia.i"
XREF       _ciaa
LEA        _ciaa,a0
MOVE.B     #%01111110,ciaicr(a0)
MOVE.B     #$10000001,ciaicr(a0)
```

Como el bit mas significativo es 0 en el primer MOVE, hace que todos los demas bits que estan a 1 desactiven a los bits correspondientes del registro MASK.

En el segundo MOVE, el bit mas significativo es 1, y por tanto los demas bits que estan a 1 activan a los bits correspondientes del registro MASK, dejando a los demas intactos.

REGISTRO DE CONTROL DE INTERRUPCION, LECTURA

REG	NOMBRE	D7	D6	D5	D4	D3	D2	D1	D0
D	ICR	IR	0	0	FLG	SP	ALRM	TB	TA

MASCARA DE CONTROL DE INTERRUPCION, ESCRITURA

REG	NOMBRE	D7	D6	D5	D4	D3	D2	D1	D0
D	ICR	S/C	X	X	FLG	SP	ALRM	TB	TA

REGISTROS DE CONTROL

Hay dos registros de control en el 8520, CRA y CRB. CRA esta asociado con el temporizado A y CRB con el temporizador B. El formato es el siguiente:

REGISTRO DE CONTROL A

BIT	NOMBRE	FUNCION
0	START	1 = inicia temporizador A, 0 = para temporizador A. Este bit se pone a 0 cuando termina la cuenta atras en el modo "one-shot".
1	PBON	1 = Temporizador A en PB6, 0 = modo normal.
2	OUTMODE	1 = cambio de estado, 0 = impulsos.
3	RUNMODE	1 = modo "one-shot", 0 = modo continuo.
4	LOAD	1 = force load (esta es una entrada strobe, no se guardan los datos; al leer el bit 4 se obtendra un 0 y si se escribe un 0 no se obtendrá ningún efecto)
5	INMODE	1 = Temporizador A cuenta transiciones positivas de CNT. 0 = Temporizador A cuenta 02 pulsos.
6	SPMODE	1 = Modo de salida (CNT es la fuente de impulsos) 0 = Modo de entrada (necesita una fuente de impulsos externa)
7	UNUSED	Sin usar

REGISTRO DE CONTROL B

BIT	NOMBRE	FUNCION
0	START	1 = inicia temporizador B, 0 = para temporizador B. Este bit se pone a 0 cuando termina la cuenta atras en el modo "one-shot".
1	PBON	1 = Temporizador B en PB7, 0 = modo normal.
2	OUTMODE	1 = cambio de estado, 0 = impulsos.
3	RUNMODE	1 = modo "one-shot", 0 = modo continuo.
4	LOAD	1 = force load (esta es una entrada strobe, no se guardan los datos; al leer el bit 4 se obtendra un 0 y si se escribe un 0 no se producira ningún efecto)
5,6	INMODE	1 = Temporizador cuenta transiciones positivas de CNT. 0 = Temporizador cuenta 02 pulsos. Si Bit 6=1 : 0=cuenta impulsos de A, 1= idem si CNT está a nivel alt
7	ALARMA	1 = Escribiendo a los registros TOD se escribe a la Alarma. 0 = Escribiendo a los registros TOD se escribe al reloj TOD. Leyendo los registros TOD siempre se lee el registro TOD, independientemente del estado del bit de Alarma.

EJEMPLO

```
; Un ejemplo completo de temporización con los chips 8520. Este hace
; parpadear al LED de POWER a (exactamente) intervalos de 3 milisegundos.
;
; Las frecuencias del cristal base del Amiga son:
; 28.37516 MHz en PAL  y  28.63636 MHz en NTSC
```

```
; Los dos temporizadores de los chips 8520 realizan la cuenta atras a la
; decima parte de la velocidad del 68000 (1.4096806 µs en PAL).
;
; Para esperar 3 milisegundos (3000 µs) habra que poner el registro con
; el número 2128    (3000 / 1.4096836 = 2129)
;
     INCLUDE  "hardware/cia.i"
     INCLUDE  "hardware/custom.i"
;
     XREF     _ciaa
     XREF     _ciab
     XREF     _custom
;
     LEA      _custom,a3
     LEA      _ciaa,a4
;
     MOVE.W    $7fff,dmacon(a3)       ; Desconecta todas las interrupciones
;
;----puesta a punto, sólo se hace una vez
;----Se preparan los bits para el modo "one-shot" del temporizador A.
     MOVE.W   ciacra(a4),d0           ; Registro de control A
     AND.B     %11000000,d0           ; No se alteran los demas bits
     OR.B      %00001000,d0
     MOVE.B   d0,ciacra(a4)
     MOVE.B    %01111111,ciaicr(a4)  ; Desconecta interrupciones del 8520

;
;----Coloca el tiempo (byte inferior, despues byte superior)
;----Saca el byte inferior con la mascara $FF
;----y desplaza el byte superior ocho bits.
;
TIME: equ 2128
     MOVE.B    (TIME&$FF),ciatalo(a4)
     MOVE.B    (TIME>>8),ciatahi(a4)

;
;----Espera la cuenta atras del temporizador
busy_wait:
     BTST.B    0,ciaicr(a4)           ; Espera que el temporizador termine
     BEQ.S    busy_wait
     BCHG      CIAB_LED,ciapra(a4)    ; Hace parpacear al led
     BSET.B    0,ciacra(a4)           ; Reinicia el temporizador
     BRA.S    busy_wait               ; Repite indefinidamente

     END
```

DETALLES DE LAS CONEXIONES DEL HARDWARE

El sistema hardware selecciona los CIAs cuando los tres bits superiores de direcciones son %101. CIA.A se selecciona cuando A12 es 0 y A13 es 1; CIAB se selecciona cuando A12 es 1 y A13 es 0. CIA.A se comunica con los bits de datos 7-0, CIA.B se comunica con los bits de datos 15-8.

Los bits de direcciones A11, A10, A9 y A8 se usan para especificar a cual de los 16 registros internos se desea acceder. Esto se indica con una "r" en la dirección. Los demas bits no se tienen en cuenta. Por lo que CIA.A se selecciona con la dirección %101x xxxx xx01 rrrr xxx0. Y CIA.B con la dirección: 101x xxxx xx10 rrrr xxxx xxx1.

Con la idea de una expansión futura, hemos decidido la siguientes direcciones: CIA.A = $BFEr01 y CIA.B = $BFDr00. El software debe usar estas direcciones y no otras.

SEÑALES DEL INTERFACE

Clock input

El reloj 02 es una entrada compatible TTL que se usa para el funcionamiento de los dispositivos internos ; como referencia de tiempo para la comunicación con el bus de datos del sistema. En el Amiga, esta conectado el reloj "E" del 68000 que funciona a la decima parte de la velocidad del 68000 0.715903 MHz en NTSC y 0.709379 MHz en PAL.

CS - chip select input

La entrada CS controla la actividad del 8520. Un nivel bajo en CS mientras 02 este alto hace que el dispositivo responda a las señales de R/W y las lineas de direcciones (RS). Un estado alto en CS evita que estas lineas controlen al 8520. La linea CS se activa normalmente con la dirección adecuada ($BFEn01 para el CIA.A y $BFDn00 para el CIA.B)

R/W - read/write input

La señal R/W la proporciona el 68000 y controla la dirección de la transferencia de datos del 8520. Un estado alto en R/W indica una lectura (los datos se transfieren fuera del 8520), mientras que un estado bajo indica una escritura (los datos se transfieren dentro del 8520).

RS3-RS0 - adress inputs

Estas entradas de direcciones seleccionan los registros internos descritos en el mapa de registros.

DB7-DB0 - data bus inputs/outputs

Los ocho pines de salida del bus de datos transfieren información entre el 8520 y el bus de datos del sistema. Estos pines son entradas de alta impedancia excepto cuando CS esta en estado bajo y R/W y 02 esta en estado alto, para leer el disposivo. Durante esta lectura, los buffers de salida del bus de datos estan conectados, llevando los datos del registro seleccionado al bus de datos del sistema.

IRQ - interrup request output

IRQ es una salida de drenador abierto (open drain) normalmente conectada la entrada de interrupción del procesador. Una resistencia pull-up externa mantiene la señal en estado alto, permitiendo que se conecten muchas IRQ juntas. La salida IRQ normalmente esta desconectada (alta impedancia) y se activa provocando un nivel bajo cuando el 8520 hace una petición de interrupción.

RES - reset input

Un estado bajo en el pin RES provoca un reset en todos los registros internos. Los pines de los ports se configuran como entradas y los registros de los ports se ponen a 0 (aunque al producirse la lectura de los ports se observara un estado alto debido a los pull-ups pasivos). Los registros de control de los temporizadores se ponen a 0 y los latchs de los temporizadores a 1s ($FFFF). Todos los demas registros se resetean a 0.

APENDICE G

AUTOCONFIG™

El protocolo AUTOCONFIG esta diseñado para permitir conectar las tarjetas de expansión en los "slots" de direcciones disponibles, eliminando la necesidad para el usuario de la configuración mediante puentes (jumpers). Cuando se produce un reset, cada tarjeta aparece en torno a la dirección $E80000 con su información de identificación, la mayor parte de la cual esta en formato de complemento a 1, almacenada en los nibbles altos de las primeras $40 palabras ($80 bytes) de la tarjeta. La información de identificación incluye el tamaño de la tarjeta, sus preferencias sobre el espacio de direccionamiento, tipo de la tarjeta (memoria u otro), y un número unico de de Fabricante de Hardware asignado por Commodore Amiga Technical Support, West Chester, Pennsylvania.

Cada tarjeta contiene hardware de configuración incluyendo un latch de direcciones que aparece en el nibble de offset $0048 y el nibble de offset $004a. Cuando se escriben los bits A23-A16 de la dirección base reservada para esta tarjeta en el registro, la tarjeta retiene la dirección y aparece en ella, entonces envia una señal llamada CONFIG-OUT que hace que la siguiente tarjeta aparezca en $E80000. Para hacer hacer ciertos tipos de tarjetas menos caros, el registro de la tarjeta de expansión puede estar organizado como registro de ancho de byte o dos registros de ancho de nibble. Si el registro es de ancho de nibble debera retener el nibble inferior de la dirección asignada (en $4A) hasta que se escriba el nibble superior (en $48). Esto permite que el siguiente algoritmo funcione con cualquier tipo de tarjeta:

```
Escribir el nibble inferior al offset $4A.
Escribir el byte de la dirección completa al offset $48.
```

A algunas tarjetas se les puede pedir "shut-up" (mandar la señal CONFIG-OUT y deja de funcionar) escribiendo a la dirección $004c de la tarjeta. Un bot en el nibble de offset $0008 indica si la tarjeta soporta "shut-up".

Todas las tarjetas de expansión comerciales para el Amiga deben incorporar el protocolo AUTOCONFIG. Si se desea una información mas profunda sobre el diseño y las caracteristicas del las tarjetas AUTOCONFIG contactar con CATS (Commodore Amiga Technical Support).

El sistema operativo del Amiga soporta unidades de discos construidas en tarjetas AUTOCONFIG. A partir de la versión 1.3, el OS (sistema operativo) tambien soporta la inicialización mediante el software de ROMs montadas en las tarjetas. Como regla general, las aplicaciones no deben intentar configurar los perifericos de expansión, dejando al software del Amiga manipular la configuración automatica. Algunas tarjetas contienen registros que una vez activados pueden causar daños irreparables, por ejemplo, los datos del disco duro se pueden perder si se configura inadecuadamente.

Sin embargo, algunos tipos de aplicaciones podrian necesitar configurar el hardware, como las tarjetas de RAM sin usar el OS del Amiga. Estas aplicaciones sólo deben configurar las tarjetas de RAM (tarjetas que piden ser añadidas a la lista de memoria libre) y las tarjetas dedicadas especificamente a estas aplicaciones. A todas las demas tarjetas se les debe pedir "shut-up" (si la tarjeta lo admite), o configurarlas e ignoralas si no admiten "shut-up" (hay muchas tarjetas que no admiten "shut-up"). La configuración de las tarjetas sólo la deben llevar a cabo las aplicaciones que toman todo el control sobre la maquina en el reset. La presencia de una tarjeta AUTOCONFIG se determina comparando los nibbles que aparecen en la dirección inicial de AUTOCONFIG con los nibbles de las especificaciones.

Las tarjetas de protocolo AUTOCONFIG se deben configurar en unos determinados limites de acuerdo con sus necesidades de espacio. Por ejemplo, una tarjeta de 1 MB de RAM se deberia configurar en unos limites de 1 MB. Hay dos excepciones a esta regla: las tarjetas con 4 MB de espacio de direccionamiento se pueden poner en $200000 y $600000 estando en ambos casos en unos limites de 4MB; las tarjetas de 8 MB se pueden poner en $200000. Estas excepciones son necesarias porque el espacio de 8 MB reservado para la expansión se inicia en la dirección $200000.

DEPURANDO LAS TARJETAS DE AUTOCONFIG

Si hay algún defecto en la información de configuración, la tarjeta podria ser ignorada, podria hacer "shut-up" o no funcionar correctamente con resultados impredidibles. Hay un sistema simple para comprobar la imformación de configuración. Se corta la linea CONFIGIN* de la tarjeta y se suelda un interrupctor en la linea. El interruptor se debe soldar de manera que cuando este en una posición deje pasar la linea CONFIGIN* del bus a la tarjeta. Esto permite que la tarjeta responda al proceso AUTOCONFIG. Cuando el interruptor este en la otra posición, debera estar soldado de manera que la entrada de la tarjeta se fuerza a estado alto (+5V). Esto desconectara el AUTOCONFIG de la tarjeta.

Poner el interruptor de manera que la linea CONFIGIN* pase a estado alto, despues encender el ordenador. La tarjeta sera invisible al OS. Cargar un depurador ("debbuger", o algun programa parecido) y activar el interruptor. Ahora la tarjeta respondera a la dirección normal $E80000. Los datos que se obtienen ahora de la tarjeta son identicos a los que obtiene el OS cuando esta en el proceso AUTOCONFIG. Ahora se pueden comparar los bits con los valores que deberian estar ahi.

NOTA: La tarjeta a depurar debe ser la última del sistema (cercana a los slots de PC y alejada de la fuente de alimentación). Porque las tarjetas que esten detras de esta no seran configuradas por el sistema.

TABLA DE DIRECCIONES

La siguiente tabla describe la información de identificación de la tarjeta y los registros de AUTOCONFIG que aparecen en los primeros $80 bytes de una tarjeta AUTOCONFIG en el momento de la configuración.

NOTAS:

- La información de identificación esta almacenada en los nibbles superiores de las direcciones pares (palabras) del inicio de la tarjeta AUTOCONFIG. Por ejemplo, las primeras dos palabras de una tarjeta podrian contener $Cxxx 1xxx. La información útil de estas dos palabras seria $C (nibble superior de la palabra de offset $00), y $1 (nibble superior de la palabra de offset $02). Gran parte de la información se interpreta combinando varios nibbles, con los nibbles de las direcciones inferior a superior conteniendo las partes de orden superior a inferior del resultado.

- Todos los nibbles, excepto los de offsets $00/02 y $40/42, estan invertidos (complemento a 1) y se les debe hacer un OR exclusivo (XOR) con $F para obtener los datos correctos. Los nibbles sin usar (los otros tres nibbles de cada palabra) no contendran ningún valor en particular. Todos los valores escritos en el area AUTOCONFIG, incluyendo la dirección asignada, se escriben sin invertir.

- Todas las direcciones se muestran como offsets de la base $E80000.

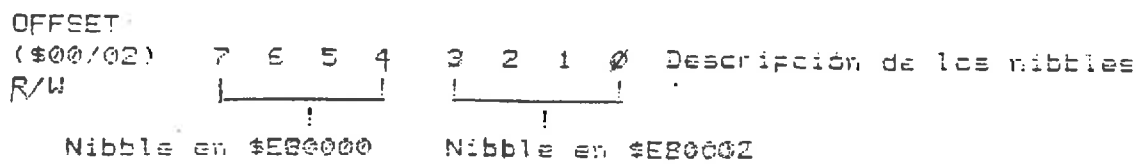


Figura G-1: Cómo leer la Tabla de direcciones.

NOTA: La numeración de los bits (7 6 5 4 3 2 1 0) se usa cuando se interpretan dos nibbles como un byte. Fisicamente, cada nibble es el nibble alto de la palabra (es decir los bits 15 14 13 12)

Tabla G-1: Tabla de las dirccciones

OFFSET	Dirección 1	Dirección 2	Descripción
($00/02) Lectura No invert.	7 6 5 4	3 2 1 0	Tamaño de la tarjeta 000 = 8 MB 100 = 512 KB 001 = 64 KB 101 = 1 MB 010 = 128 KB 110 = 2 MB 011 = 256 KB 111 = 4 MB
	7		1 = La siguiente tarjeta esta en esta.
	6		1 = Vector de ROM opcional valido.
	5		1 = Añadir a lista de memoria libre.
	4 3		Tipo de tarjeta 00 = Reservado 01 = Reservado 10 = Reservado 11 = Tipo normal
($04/06) Lectura Invertido	7 6 5 4	3 2 1 0	Número del fabricante para producto. Nibble alto Nibble bajo
($08/0A) Lectura Invertido	7 6 5 4 3 2 1 0		Reservado - debe ser 0.
	7		0 = esta tarjeta admite "shut-up" 1 = esta tarjeta ignora "shut-up"
	6 5 4 3 2 1 0		0 = cualquier espacio OK. 1 = preferiblemente el area de 8 MB.
($0C/0E) Lectura Invertido	7 6 5 4 3 2 1 0		Reservado - debe ser 0.
($10/12) Lectura Invertido	7 6 5 4	3 2 1 0	Byte superior del número único asignado al fabricante de hardware. (no el número del investigador) Nibble alto Nibble bajo
($14/16) Lectura Invertido	7 6 5 4	3 2 1 0	Byte inferior del número único asignado al fabricante de hardware. (no el número del investigador) Nibble alto Nibble bajo

NOTA: El número del fabricante es asignado por Commodore Amiga Technical Suport in West Chester, Pennsylvania (CATS). Contactar con CATS para mas información.

(\$18/1A)	7 6 5 4 3 2 1 0	Número de serie opcional, byte 1
(\$1C/1E)	7 6 5 4 3 2 1 0	Número de serie opcional, byte 2
(\$20/22)	7 6 5 4 3 2 1 0	Número de serie opcional, byte 3
(\$24/26)	7 6 5 4 3 2 1 0	Número de serie opcional, byte 4
Lectura Invertido		

(\$28/2A)	7 6 5 4 3 2 1 0	Byte superior del vector de ROM opcional.
Lectura Invertido	Nibble alto Nibble bajo	

(\$2C/2E)	7 6 5 4 3 2 1 0	Byte inferior del vector de ROM opcional. Si el bit de "vector de ROM valido" esta a 1, entonces este vector de ROM opcional es el offset sobre la base de la tarjeta hacia las estructuras de la ROM.
Lectura Invertido	Nibble alto Nibble bajo	

(\$30/32)	7 6 5 4 3 2 1 0	Lectura reservado - debe ser 0.
R/W Invertido	/	Escritura - reset opcional del registro de la base de la tarjeta a la dirección de pre-configuración.

(\$34/36)	7 6 5 4 3 2 1 0	Reservado - debe ser 00
(\$38/3A)	7 6 5 4 3 2 1 0	Reservado - debe ser 00
(\$3C/3E)	7 6 5 4 3 2 1 0	Reservado - debe ser 00
Invertido		

(\$40/42)	7 6 5 4 3 2 1 0	Esctritura	Lectura
R/W			
No invert.	---	INT conectada	INT conectada
	------	Definible usuario	Sin definir
	---------	Reset local	Debe ser 0
	----------	Definible usuario	Sin definir
	----------	Definible usuario	INT2 pendiente
	----------	Definible usuario	INT6 pendiente
	----------	Definible usuario	INT7 pendiente
	----------	Definible usuario	Tarjeta esperando INT.

NOTA: El uso de los registros \$40/42 es una característica opcional que se puede incorporar a las tarjetas para generar interrupciones. Esto hace posible a los programas de manejo de interrupciones averiguar que tarjeta ha generado la interrupción en una misma linea donde hay varias conectadas.

(\$44/46)	7 6 5 4 3 2 1 0	Reservado, la lectura debe dar 0
R/W Invertido		Escritura indefinida.

(\$48/4A)	7 6 5 4 3 2 1 0	Registro de la dirección base, sólo escritura. Estos bits se comparan con los bits de direcciones A23 - A16 (o *MENOS*) para determinar la dirección base de la tarjeta.
Escritura No invert.	Nibble alto Nibble bajo	

```
($4C/4E)   7  6  5  4   3  2  1  Ø   Registro opcional de "shut-up".
Escritura  |_____|                   Cualquier escritura a $4C hara que
                 |                   la tarjeta mande la señal CONFIG-OUT
                 ------------------- y no vuelva a responder de nuevo a
                                     la direccion, hasta el RESET. Un bit
                                     del nibble $ØB indica si la tarjeta
                                     admite "shut-up".

($5Ø hasta $7E)                      Reservados; deben ser ØØ.
Invertidos
```

Recuerda que todos los nibbles excepto $ØØ/Ø2 y $4Ø/42 muestran los valores de la tabla invertidos. Por ejemplo, un nibble "debe ser Ø" aparecera como $5. los indicadores y los valores hexadecimales tambien estan invertidos (por ejemplo, un valor de $1 se leera como $E, etc)

```
/*
 *  Este listado en C examina todas las tarjetas AUTOCONFIG del sistema
 */
 include "exec/types.h"
 include "libraries/configvars.h"

struct Library   *OpenLibrary();
struct ConfigDev *FindConfigDev;
struct Library   *ExpansionBase;

void main()
{
struct ConfigDev *myCD=0;

ExpansionBase=OpenLibrary("expansion.library",0L);

while(myCD=FindConfigDev(myCD,-1L,-1L)) /* busca cualquier ConfigDev */
      {
      printf("\n---Encontrada estructura ConfigDev en $%lx---\n",myCD);

      /* Estos valores se leen directamente de la tarjeta */
      printf(er_Manufacturer    =");
           printf(%d,",myCD->cd_Rom.er_Manufacturer);
           printf($%x,",myCD->cd_Rom.er_Manufacturer);
           printf("(~$%4x)\n",~myCD->cd_Rom.er_Manufacturer);

      printf(er_Product         =");
           printf(%d,",myCD->cd_Rom.er_Product);
           printf($%x,",myCD->cd_Rom.er_Product);
           printf("(~$%x)\n",~myCD->cd_Rom.er_Product);

      printf("er_Type           =$%x\n",myCD->cd_Rom.er_Type);

      printf(er_Flags           =");
           printf("$%x\n",myCD->cd_Rom.er_Flags);

      /* Estos valores se generan cuando el software AUTOCONFIG
       * relocaliza la tarjeta (la cambia de posición)
       */
      printf("cd_BoardAddr      =$%1x\n",myCD->cd_BoardAddr);
      printf("cd_BoardSize      =$%1x (%1dK)\n",
           myCD->cd_BoardSize,((ULONG)myCD->cd_BoardSize)/1024);

      printf(cd_Flags           =$%x\n",myCD->cd_Flags);
      }
CloseLibrary(ExpansionBase);
}
```

FABRICANTE

PRODUCTO

TIPO

INDICADORES

DIRECCIÓN

TAMAÑO

INDICADORES

APENDICE H

EL TECLADO

Este Apendice contiene las especificaciones sobre el interface del teclado del A1000, A500 y A2000.

El teclado esta conectado al Amiga mediante un cable de cuatro conexiones esenciales. Los cuatro hilos proporcionan tensión de 5 voltios, masa y unas lineas llamadas KCLK (reloj del teclado) y KDAT (datos del teclado). KCLK es unidireccional y siempre la controla el teclado. KDAT la controlan el ordenador y el teclado. Ambas lineas son de colector abierto (open collector), hay resistencias de pull-up en el teclado (dentro de su microprocesador) y en el ordenador.

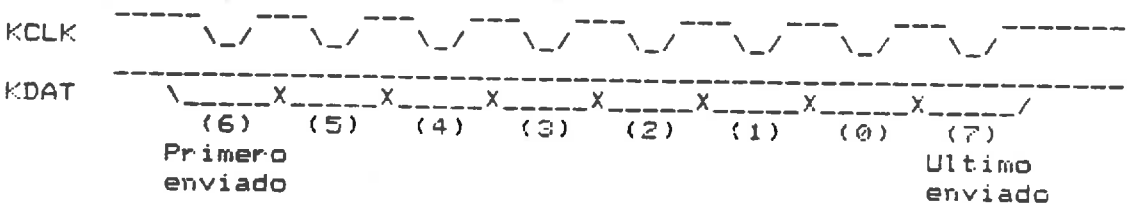
LAS COMUNICACIONES DEL TECLADO

El teclado transmite palabras de 8 bits en serie hacia el ordenador. Antes de comenzar la transmisión, KCLK y KDAT estan en estado alto. El teclado inicia la transmisión enviando el primer bit de datos (por KDAT), seguido de un impulso en KCLK (primero estado bajo y despues estado alto); despues envia el segundo bit y genera un impulso KCLK hasta que se hayan enviado los ocho bits. Despues del último impulso KCLK, el teclado deja a KDAT en estado alto.

Cuando el ordenador ha recibido el octavo bit, debe genrar un impulso en KDAT de como minimo 1 µs, como hanshake al teclado. La detección del handshake en el teclado usa un latch de Hardware. El teclado debe ser capaz de detectar impulsos mayores o iguales a 1us. El software DEBE generar un impulso de 85 µs para asegurar la compatibilidad con todos los modelos de teclado.

Todos los códigos transmitidos al ordenador se rotan (giran) un bit antes de la transmisión. El orden de la transmisión es por tanto 6-5-4-3-2-1-0-7. La razón de esto es para transmitir el indicador de tecla arriba/abajo el último, en orden a provocar un código de "tecla levantada" en el caso de que se fuuerce al teclado a restaurar la sincronia perdida (mas adelante se explicara con mas detalles).

La linea KDAT es activa en estado bajo; es decir, un nivel alto (+5V) se interpreta como 0, y un nivel bajo (-0V) se interpreta como 1.



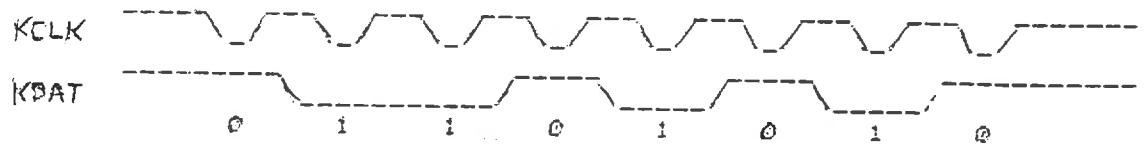
El microprocesador del teclado activa la linea KDAT 20 us antes de poner KCLK en estado bajo. KCLK permanece 20 us en estado bajo, y despues vuelve al estado alto. El teclado espera otros 20 us antes de cambiar KDAT.

Por lo tanto, la velocidad de transmisión de los bits es de unos 60 us por bit, o 17 Kbit/s (17000 baudios).

CODIGOS DE LAS TECLAS

Cada tecla tiene un codigo asociado. Los codigos de las teclas siempre son de 8 bits. El octavo bit es un indicador de 'tecla pulsada/soltada' un Ø (nivel alto porque está invertido) significa que se ha pulsado la tecla, y un 1 (nivel bajo) significa que se ha soltado la tecla (CAPS LOCK es diferente).

Por ejemplo este es el diagrama de la tecla "B" pulsada. El codigo de "B" es $35 = %ØØ110101 debido a la rotación de bits. los bits transmitidos son $Ø11Ø1Ø1Ø.



En el siguiente ejemplo, se suelta la tecla "B". El código de la tecla continuara siendo $35, excepto el bit 7 que se pone a 1 para indicar "tecla soltada". Resultando un código de $B5 = %10110101. Despues de la rotación, la transmisión sera $Ø11Ø1Ø11:

LA TECLA "CAPS LOCK"

Esta tecla se diferencia de todas las demas en que sólo genera un código cuando se pulsa, nunca cuando se suelta. Sin embargo, el bit arriba/abajo se sigue usando. Cuando se pulsa la tecla CAPS LOCK el led de la misma se enciende, y el bit arriba/abajo se pone a 0; cuando se pulsa de nuevo CAPS LOCK el led se apaga, y el bit arriba/abajo se pone a 1.

SITUACION DE "OUT-OF-SYNC"

El ruido en la linea y otras circunstancias pueden hacer que el teclado pierda la sincronia con el ordenador. Esto significa que el teclado ha terminado de transmitir un código, pero el ordenador esta en alguno de los bits intermedios.

Si esto sucede, el teclado no recibira el impulso de handshake al final de la transmisión. Si el hanshake no llega en 143 ms desde la última transmisión, el teclado asumira que el ordenador esta esperando al resto de la transmisión y por tanto esta en "out-of-sync" (fuera de sincronia). El teclado intentara restaurar la sincronia entrando en el modo de resincronia. En este modo, el teclado envia un 1 y espera al impulso de handshake. Si no llega nada en 143 ms, enviara otro 1 y volvera a esperar. Este proceso continua hasta que llega el handshake.

Una vez se ha restaurado la sincronia, en el ordenador se habra recibido un código del teclado erroneo. Por esta razón se transmite el indicador arriba/abajo el último, porque como el teclado utiliza 1s para resincronizarse con el ordenador, el código erroneo aparecera como una tecla soltada que producira menos errores en el programa que una tecla pulsada.

Cuando el teclado detecta que ha perdido la sincronia, asume que el ordenador no recibira el código que se estaba intentando transmitir. Como el ordenador no es capaz de detectar que ha perdido la sincronia, es

responsabilidad del teclado informarle de este hecho. Esto lo hace transmitiendo un código de "sincronía perdida" ($F9 = %11111001) al ordenador. Depues transmitirá el código que no había llegado correctamente.

NOTA: La única razón para transmitir el código de "sincronía perdida" es para avisar al programa del ordenador de que algo ha fallado. El código de "sincronía perdida" no ayudará al proceso de recuperación, porque el código erroneo no se puede detectar, , el código correcto se podría retransmitir sin necesidad de indicar al ordenador que el anterior estaba equivocado.

SECUENCIA DE ENCENDIDO

Hay dos formas posibles de que se conecte el teclado en circunstancias normales: <1> Se enciende el ordenador con el teclado conectado a el, o <2> El teclado se puede conectar a un ordenador que ya estaba encendido. El teclado y el ordenador deben controlar ambos casos sin originar ningun imprevisto.

En primer lugar el teclado produce un auto-test. Esto incluye el chequeo del checksum de la ROM, y el chequeo del temporizador watchdog. Cuando se enciende el teclado (o se pone en marcha), no debe transmitir nada hasta que no haya conseguido sincronizarse con el ordenador. La forma en que hace esto es enviando 1s, como se describia antes, hasta que se recibe el impulso de handshake.

Si el teclado se conecta antes de encender, el teclado continuará con este proceso varios minutos hasta que el ordenador haya arrancado y este en funcionamiento. El teclado continuara enviando 1s durante todo el tiempo que sea necesario, hasta que reciba el handshake.

Si el teclado se conecta despues de encender, no se necesitaran mas de ocho 1s para entrar en sincronia. En este caso, sin embargo, el ordenador podria estar en cualquier *ESTADO* imaginable pero no le afectaria perjudicialmente el código erroneo recibido. De nuevo, debido a que se recibe un código de tecla soltada, las "molestias" seran minimas. El programa que controla el teclado se debera anticipar a este hecho y controlarlo, como sucede en cualquier aplicación que usa los códigos directos.

NOTAS: El teclado no transmite el código de "sincronia perdida" hasta despues de resincronizarse.

Una vez el teclado y el ordenador estan sincronizados, el teclado debe informar al ordenador de los resultados del auto-test. Si el auto-test falla por cualquier razón, se enviara un código de "test fallado" ($FC = %11111100) (el teclado no esperara el hanshake). Despues de esto, el microprocesador del teclado entrara en un bucle para hacer parpadear el led de la tecla CAPS LOCK para informar al usuario de la existencia de un fallo. Los parpadeos se codifican como uno, dos, tres o cuatro parpadeos, aproximadamente a uno por segundo.

Un parpadeo	Fallo del checksum de la ROM
Dos parpadeos	Prueba de RAM fallada
Tres parpadeos	Prueba del temporizador watchdog fallada
Cuatro parpadeos	Hay un cortocircuito entre dos pistas distintas o una de las siete teclas especiales (no incorporado).

Si el auto-test se supera, el teclado comenzara a transmitir las teclas que esten pulsadas en ese momento. Primero, transmitira un código de "inicio de la cadena de teclas del encendido" ($FD = %11111101), seguido de los

codigos de todas las teclas pulsadas (el bit de arriba/abajo estará en la posición "abajo"). Despues de transmitir todas las teclas (o si no hay ninguna) se envia un codigo de "transmisión de la cadena de teclas" ($FE = %11111110). Finalmente se apaga el led de la tecla CAPS LOCK. Esto indica el final de la secuencia de inicio y el proceso normal comienza.

La secuencia normal sera por lo tanto encendido, sincronización, transmisión del codigo $FD (teclas del encendido), transmisión de las teclas pulsadas (si las hay) transmisión del código $FE (final de las teclas del encendido).

ALERTA DE RESET

NOTA: Está disponible en algunos teclados de los A1000 y A2000. No se puede esperar esta característica de todos los teclados.

El teclado tiene la tarea adicional de resetear el ordenador cuando el usuario pulsa determinadas teclas. El usuario inicia la alerta de reset pulsando simultaneamente la tecla CTRL, y las dos teclas "AMIGA"

El teclado responde abandonando las operaciones de transmisión que se esten ejecutando y enviando una alerta de reset al Amiga. Esto avisa a los programas del Amiga que terminen todas las operaciones pendientes (como el DMA del disco) y se preparen para el reset.

Una secuencia especifica de operaciones aseguran que el Amiga esta en un estado cuando puede responder a la alerta de reset. El teclado envia dos códigos de "alerta de reset". El Amiga respondera con un handshake al código como si fuera una tecla normal, si no el teclado pasara directamente al Hard Reset. En el segundo código de "alerta de reset" el Amiga debe poner la linea KDAT en estado bajo durante 250 ms, si no el teclado pasara directamente al Hard Reset. Si se pasan todas las comprobaciones, el Amiga tendra 10 segundos para ejecutar los procesos de emergencia. Cuando el Amiga devuelve la linea KDAT al estado alto, el teclado finalmente produce el Hard Reset.

Si el Amiga no pone la linea en estado alto durante los 10 segundos, se producira el Hard Reset de todas formas.

HARD RESET

NOTA: Esto sucede despues de la Alerta de Reset. Valido para todos los teclados excepto los del Amiga 500.

El teclado Resetea al Amiga poniendo la linea KCLK en estado bajo durante 500 ms. Cuando una o mas teclas se sueltan Y han pasado los 500 ms, el teclado dejara de actuar sobre KCLK. 500 ms es el tiempo minimo que se debe mantener a KCLK en estado bajo. El tiempo maximo dependera del tiempo que mantenga el usuario las tres teclas pulsada.

NOTA: La circuiteria del Amiga detecta este impulso de 500 ms en KCLK.

Despues de dejar de actuar sobre KCLK, el teclado salta a su programa de inicio (reset interno). Esto iniciara al teclado como si fuera un encendido en frio (con el ordenador apagado).

NOTA: El teclado volvera a enviar la "cadena de teclas del encendido".

CODIGOS ESPECIALES

Los códigos especiales que las el teclado para la Amiga es el. A todos primeros son los siguientes

Código	Nombre y significado
$78	Alerta de Reset. Se han pulsado CTRL-AMIGA-AMIGA - el ordenador será reseteado en 10 segundos (ver el texto anterior).
$F9	Ultimo código erroneo. o. siguiente código es el anterior retransmitido (se usa cuando el teclado y el ordenador pierden la sincronía).
$FA	Buffer de salida del teclado lleno.
$FB	Sin usar (si se recibe es por algún fallo)
$FC	Auto-test del teclado fallado.
$FD	Inicio de la cadena de teclas del encendido (las teclas que estaban pulsadas cuando se encendió el teclado).
$FE	Final de la cadena de teclas.
$FF	Sin usar.

NOTA: Los códigos especiales son números de 8 bits: no hay ningún indicador de tecla arriba/abajo junto a estos códigos. Sin embargo, el orden de transmisión de los bits es el mismo que el descrito anteriormente.

TABLA DE LA MATRIZ

COLUMNA	FILAS 5 (Bit 7)	FILA 4 (Bit 6)	FILA 3 (Bit 5)	FILA 2 (Bit 4)	FILA 1 (Bit 3)	FILA 0 (Bit 2)
15 (PD.7)	(Libre) (0E)	(Libre) (1C)	(Libre) (2C)	(Libre) (47)	(Libre) (46)	(Libre) (49)
14 (PD.6)	* nota 1 (5D)	<SHIFT> nota 2 (30)	CAPS LOCK (62)	TAB (42)	- (00)	ESC (45)
13 (PD.5)	+ (5E)	Z (31)	A (21)	Q (10)	! 1 (01)	(nota 1 (5A)
12 (PD.4)	9 (3F)	X (32)	S (21)	W (11)	@ 2 (02)	F1 (50)
11 (PD.3)	6 nota 3 (2F)	C (33)	D (22)	E (12)	# 3 (03)	F2 (51)
10 (PD.2)	3 nota 3 (1F)	V (34)	F (23)	R (13)	$ 4 (04)	F3 (52)
9 (PD.1)	. nota 3 (3C)	B (35)	G (24)	T (14)	% 5 (05)	F4 (53)
8 (PD.0)	8 nota 3 (3E)	N (36)	H (25)	Y (15)	^ 6 (06)	F5 - (54)

COLUMNA	FILAS 5 (Bit 7)	FILA 4 (Bit 6)	FILA 3 (Bit 5)	FILA 2 (Bit 4)	FILA 1 (Bit 3)	FILA 0 (Bit 2)
7 (PD.7)	5 nota 3 (2E)	M (37)	J (25)	U (16)	& 7 (07)	/ nota 1 (5B)
6 (PD.6)	2 nota 3 (1E)	< , (38)	K (27)	I (17)	* 8 (08)	FE (55)
5 (PD.5)	ENTER nota 3 (43)	> . (39)	L (28)	O (18)	(9 (09)	/ nota 1 (5C)
4 (PD.4)	7 nota 3 (3D)	? / (3A)	¡ (espáñola) ¡ (funboy cond) (29)	P (19)	) 0 (0A)	F7 (56)
3 (PD.3)	4 nota 3 (2D)	(Libre) (3B)	" ' (2A)	{ [(1A)	_ - (0B)	F8 (57)
2 (PD.2)	1 nota 3 (1D)	ESPACIO (40)	<RET> nota 2 (2B)	}] (1B)	+ = (0C)	F9 (58)
1 (PD.1)	0 nota 3 (0F)	BACK SPACE (41)	DEL (46)	RETURN (44)	| \ (0D)	F10 (59)
0 (PD.0)	- nota 3 (4A)	CURSOR ABAJO (4D)	CURSOR DERECHA (4E)	CURSOR IZQUIER. (4F)	CURSOR ARRIBA (4C)	HELP (5F)

nota 1: Teclado numerico sólo en los A500 y A2000.
nota 2: Sólo en los teclados internacionales (estas teclas son parte de otras teclas de los teclados americanos). La tecla que genera el código $30 es un "trozo" de la tecla izquierda de SHIFT (en España es la tecla < >). La tecla $2B es un "trozo" de la tecla return (en España es la tecla Ç).
nota 3: Teclado numerico.

La siguiente tabla muestra las teclas que se pueden leer independientemente. Estas teclas nunca generaran "fantasmas".

(Bit 7)	(Bit 6)	(Bit 5)	(Bit 4)	(Bit 3)	(Bit 2)
AMIGA IZQUIER. (0E)	ALT IZQUIER. (64)	SHIFT IZQUIER. (60)	CTRL (63)	ALT DERECHO (65)	SHIFT DERECHO (61)

APENDICE I

ESPECIFICACIONES DEL CONECTOR DE DISCO EXTERNO

GENERAL

El conector hembra de 23 pines de la parte trasera del ordenador se usa como interface para controlar dispositivos que generan y reciben datos MFM. Este interface se puede controlar como un recurso o mediante un programa driver. Las siguientes paginas describen el interface para ambos casos.

TABLA SUMARIO

Pin	Nombre	Notas	
1	RDY-	I/O	Código ID y señal de preparado.
2	DKRD-	I	Entrada del MFM.
3	GND	G	Masa.
4	GND	G	Masa.
5	GND	G	Masa.
6	GND	G	Masa.
7	GND	G	Masa.
8	MTRXD	O	Control del Motor.
9	SEL2B	O*	Selecciona la unidad 2.
10	DRESB	O	Reset.
11	CHNG-	I/O	Disco cambiado.
12	+5V	PWR	540 ma de una fuente de 870 ma.
13	SIDEB-	O	Cara 1 si esta en estado bajo.
14	WRPRO-	I/O	Protección de escritura.
15	TK00-	I/O	Pista 00.
16	DKWEB-	O	Puerta de escritura.
17	DKWDB-	O	Escritura de los datos.
18	STEPB-	O	Paso.
19	DIRB	O	Dirección (hacia afuera si es alto)
20	SEL3B-	O*	Selecciona la unidad 3.
21	SEL1B-	O*	Selecciona la unidad 1.
22	INDEX-	I/O	Indice del disco.
23	+12V	PWR	120 ma de una fuente de 370 ma.

I - Entrada polarizada a +5V con 1000 ohm.
I/O - Entrada (aunque bidireccional) a +5V con 1000 ohm.
O - Salida polarizada a +5V con 1000 ohm.
O* - Salida, para unidades distintas.
PWR - voltage para la unidad externa.

SEÑALES CUANDO SE MANEJA UN DISCO

SEL1B-, SEL2B-, SEL3B- Lineas de selección de las tres unidades de disco externas, se activan a nivel bajo.

TK0- La unidad seleccionada pone esta linea a nivel bajo cuando el cabezal de lectura/escritura esta en la pista 00.

RDY- Cuando el motor del disco esta encendido, esta linea indica que el motor ha llegado a su maxima velocidad. El controlador ignora esta señal. Cuando el motor esta apagado se usa como linea de datos para el código ID (mas adelante).

WPRO- (Pin 14) La unidad seleccionada pone esta linea a nivel bajo cuando el disco del interior de la misma esta protegido de escritura.

INDEX- (Pin 22) La unidad seleccionada manda un impulso [illegible] por cada vuelta completa del motor.

SIDEB- (Pin 13) El sistema envía esta señal a todas las unidades de disco (el estado bajo indica la cara 1, y el estado alto la cara Ø.

STEPB- (Pin 18) Se utiliza para mandar impulsos y mover el cabezal.

DIRB (Pin 19) El sistema pone esta línea a estado alto o bajo para indicar a la unidad seleccionada la dirección del movimiento indicado con STEPB.

DKRD- (Pin 2) La unidad seleccionada envia los datos por cada línea.

DKWDB- (Pin 17) El sistema envia los datos a todos los discos por esta línea. Los datos sólo se graban en los discos seleccionados y cuando DKWEB- esta activa (nivel bajo).

DKWEB- (Pin 16) Esta señal hace que los discos seleccionados comiencen a grabar datos (de la linea DKWEB-) en el disco.

CHNG- (Pin 11) La unidad seleccionada pondra esta linea a estado bajo cuando su latch interno de "cambio de disco" este activado. Este latch se activa cuando se enciende la unidad, o cuando no hay ningún disco en la misma. Para desconectar el latch, el sistema debe seleccionar la unidad y mover el cabezal. Por supuesto, el latch no se desactivara si no hay un disco en la unidad.

MTRXD- (Pin 8) Esta es la linea de control del motor para todas las unidades de disco. Cuando el sistema quiere conectar un motor de una unidad, primero deselcciona la unidad (si estaba seleccionada) pone MTRXD- a estado bajo, y selecciona la unidad. Para desconectar el motor, el sistema deselecciona la unidad, pone MTRXD- a estado alto, y selecciona la unidad. El sistema debe utilizar MTRXD- como minimo 1.4 µs antes de seleccionar la unidad. Todos los discos externos deben tener una lógica equivalente a un flip-flop D, cuya entrada D es la señal MTRXD-, y cuya entrada de reloj (C) se activa por la transición negativa (alto a bajo) de la linea SELxB-. Como se indicaba antes, los tiempos de puesta a punto y bloqueo de MTRxD- con respecto a SELxB- deben ser como minimo de 1.4 µs. La salida de este flip-flop controla el motor del disco. Por lo tanto, el sistema puede controlar los cuatro motores usando uno solo de los cables (MTRXD-).

DRESB- (Pin 10) Esta señal es una "versión" [illegible] la señal de reset del sistema. Tres cosas la pueden [illegible] estado bajo):

- Que se encienda el ordenador (DRESB- estara en estado bajo durante 1 segundo aproximadamente).
- Que el 68000 ejecute una instrucción RESET (DRESB- estara en estado bajo durante 17 µs aproximadamente).
- Un Hard Reset del teclado (durara el tiempo que se mantengan pulsadas las tres teclas del reset)

Si la linea de alimentación de +5V llega a un nivel de 3.75V o menor sera necesario que se protejan los discos externos y se apague el motor de la linea.

CODIGO I.D. DEL DISPOSITIVO

Esta interface soporta ... ratios para ... el tipo de unidades de disco conectadas. La secuencia del I.D. es la siguiente:

1. Poner MTEXB- a nivel bajo.
2. Poner SELxB- a nivel bajo.
3. Poner SELxB a nivel alto.
4. Poner MTEXB a nivel alto.
5. Poner SEL3B- a nivel bajo.
6. Poner SEL3B- a nivel alto.
7. Poner SELxB- a nivel bajo.
8. Leer y guardar el estado de RDY.
9. Poner SELxB- a nivel alto.

Repetir los pasos del 6 al 9 quince veces (con un bucle).

Introducir los 16 valores de RDY- en una palabra de 16 bits. El bit mas significativo sera el primer valor obtenido.

0000 0000 0000 0000	Reservado
1111 1111 1111 1111	Standard del Amiga de 3.25"
1010 1010 1010 1010	Reservado
0101 0101 0101 0101	48 TPI doble densidad, doble cara
1000 0000 0000 0000	Reservado
0111 1111 1111 1111	Reservado
0000 1111 xxxx xxxx	Disponibles para los usuarios
1111 0000 xxxx xxxx	Extensión reservada
xxxx 0000 0000 0000	Reservado
xxxx 1111 1111 1111	Reservado
0011 0011 0011 0011	Reservado
1100 1100 1100 1100	Reservado

APENDICE J

FICHERO INCLUDE DE LOS EJEMPLOS DEL HARDWARE

Este Apendice contiene [illegible] de los registros del hardware. Todos en el Apendice J [illegible] comentarios en los nombres de los [illegible] en [illegible] todos los ejemplos de este libro se refiere a los nombres en este fichero.

```
                                IFND    HARDWARE_HW_EXAMPLES_I
                                IFND    [illegible]
HARDWARE_HW_EXAMPLES_I          SET     1
**
**      Nombre del fichero  hardware/ha_[illegible]plos.i
**      $Version 1.2 [illegible]
**
**
**      (C) Copyright 1985, 1986, 1987, 1988, 1989 Commodore-Amiga, Inc.
**          Todos los derechos reservados.
**
*****************************************************************************

        IFND    HARDWARE_CUSTOM_I
        INCLUDE "hardware/custom.i"
        ENDC

*****************************************************************************
*
*       Este fichero include esta diseñado para usarse junto con los
*       ejemplos del hardware de este libro. Este fichero define los nombres
*       de los registros basandose en el fichero hardware/custom.i. No hay
*       versión en C de este fichero.
*
*****************************************************************************
*
*       Esta instrucción del Copper provoca que este espere indefinidamente
*       como ya se explicó en el Capitulo 2.
*
COPPER_HALT     equ  $FFFFFFFE
*
*****************************************************************************
*
*       Esta es la base de los registros de los custom chips para el espacio
*       de direccinamiento del 68000. Es lo mismo que _custom cuando se hacen
*       links con AMIGA.lib.
*
CUSTOM          equ  $DFF000
*
*       Varios registros de control
*
        DMACONR         equ     dmaconr
        VPOSR           equ     vposr
        VHPOSR          equ     vhposr
        JOY0DAT         equ     joy0dat
        JOY1DAT         equ     joy1dat
        CLXDAT          equ     clxdat
        ADKCONR         equ     adkconr
        POT0DAT         equ     pot0dat
        POT1DAT         equ     pot1dat
        POTINP          equ     potinp
        SERDATR         equ     serdatr
        INTENAR         equ     intenar
        INTREQR         equ     intreqr
        REFPTR          equ     refptr
        VPOSW           equ     vposw
```

```
VHPOSW          equ     vhposw
SERDAT          equ     serdat
SERPER          equ     serper
POTGO           equ     potgo
JOYTEST         equ     joytest
STREQU          equ     strequ
STRVBL          equ     strvbl
STRHOR          equ     strhor
STRLONG         equ     strlong
DIWSTRT         equ     diwstrt
DIWSTOP         equ     diwstop
DDFSTRT         equ     ddfstrt
DDFSTOP         equ     ddfstop
DMACON          equ     dmacon
INTENA          equ     intena
INTREQ          equ     intreq

;   Registros de control del disco

DSKBYTR         equ     dskbytr
DSKPT           equ     dskpt
DSKPTH          equ     dskpt
DSKPTL          equ     dskpt+$02
DSKLEN          equ     dsklen
DSKDAT          equ     dskdat
DSKSYNC         equ     dsksync

;   Registros del Blitter

BLTCON0         equ     bltcon0
BLTCON1         equ     bltcon1
BLTAFWM         equ     bltafwm
BLTALWM         equ     bltalwm
BLTCPT          equ     bltcpt
BLTCPTH         equ     bltcpth
BLTCPTL         equ     bltcptl
BLTBPT          equ     bltbpt
BLTBPTH         equ     bltbpth
BLTBPTL         equ     bltbptl
BLTAPT          equ     bltapt
BLTAPTH         equ     bltapth
BLTAPTL         equ     bltaptl
BLTDPT          equ     bltdpt
BLTDPTH         equ     bltdpth
BLTDPTL         equ     bltdptl
BLTSIZE         equ     bltsize
BLTCMOD         equ     bltcmod
BLTBMOD         equ     bltbmod
BLTAMOD         equ     bltamod
BLTDMOD         equ     bltdmod
BLTCDAT         equ     bltcdat
BLTBDAT         equ     bltbdat
BLTADAT         equ     bltadat
BLTDDAT         equ     bltddat

;   Registros de control del Copper

COPCON          equ     copcon
COPINS          equ     copins
COPJMP1         equ     copjmp1
COPJMP2         equ     copjmp2
COP1LC          equ     cop1lc
COP1LCH         equ     cop1lc
COP1LCL         equ     cop1lc+$02
COP2LC          equ     cop2lc
```

```
BPL1DATA        equ     bpldat+$00
BPL2DATA        equ     bpldat+$02
BPL3DATA        equ     bpldat+$04
BPL4DATA        equ     bpldat+$06
BPL5DATA        equ     bpldat+$08
BPL6DATA        equ     bpldat+$0A

;
; Registros de control de los Sprites
;
SPR0PT          equ     sprpt+$00
SPR0PTH         equ     sprpt+$00
SPR0PTL         equ     sprpt+$02
SPR1PT          equ     sprpt+$04
SPR1PTH         equ     sprpt+$04
SPR1PTL         equ     sprpt+$06
SPR2PT          equ     sprpt+$08
SPR2PTH         equ     sprpt+$08
SPR2PTL         equ     sprpt+$0A
SPR3PT          equ     sprpt+$0C
SPR3PTH         equ     sprpt+$0C
SPR3PTL         equ     sprpt+$0E
SPR4PT          equ     sprpt+$10
SPR4PTH         equ     sprpt+$10
SPR4PTL         equ     sprpt+$12
SPR5PT          equ     sprpt+$14
SPR5PTH         equ     sprpt+$14
SPR5PTL         equ     sprpt+$16
SPR6PT          equ     sprpt+$18
SPR6PTH         equ     sprpt+$18
SPR6PTL         equ     sprpt+$1A
SPR7PT          equ     sprpt+$1C
SPR7PTH         equ     sprpt+$1C
SPR7PTL         equ     sprpt+$1E
;
; Nota: SPRxDATAB se define sumando $06 a SPRxPOS
; sd_datab se debe definir como +$06, sin embargo, en el
; fichero include 1.3 hardware/custom.i esta definido
; incorrectamente como $08.
;
SPR0POS         equ     spr+$00
SPR0CTL         equ     SPR0POS+sd_ctl
SPR0DATA        equ     SPR0POS+sd_dataa
SPR0DATB        equ     SPR0POS+$06

SPR1POS         equ     spr+$08
SPR1CTL         equ     SPR1POS+sd_ctl
SPR1DATA        equ     SPR1POS+sd_dataa
SPR1DATB        equ     SPR1POS+$06

SPR2POS         equ     spr+$10
SPR2CTL         equ     SPR2POS+sd_ctl
SPR2DATA        equ     SPR2POS+sd_dataa
SPR2DATB        equ     SPR2POS+$06

SPR3POS         equ     spr+$18
SPR3CTL         equ     SPR3POS+sd_ctl
SPR3DATA        equ     SPR3POS+sd_dataa
SPR3DATB        equ     SPR3POS+$06

SPR4POS         equ     spr+$20
SPR4CTL         equ     SPR4POS+sd_ctl
SPR4DATA        equ     SPR4POS+sd_dataa
SPR4DATB        equ     SPR4POS+$06
```

```
COP2LCH         equ     cop2lc
COP2LCL         equ     cop2lc+$02
*
*       Registros de los canales de Audio
*
ADKCON          equ     adkcon

AUD0LC          equ     aud0
AUD0LCH         equ     aud0
AUD0LCL         equ     aud0+$02
AUD0LEN         equ     aud0+$04
AUD0PER         equ     aud0+$06
AUD0VOL         equ     aud0+$08
AUD0DAT         equ     aud0+$0A

AUD1LC          equ     aud1
AUD1LCH         equ     aud1
AUD1LCL         equ     aud1+$02
AUD1LEN         equ     aud1+$04
AUD1PER         equ     aud1+$06
AUD1VOL         equ     aud1+$08
AUD1DAT         equ     aud1+$0A

AUD2LC          equ     aud2
AUD2LCH         equ     aud2
AUD2LCL         equ     aud2+$02
AUD2LEN         equ     aud2+$04
AUD2PER         equ     aud2+$06
AUD2VOL         equ     aud2+$08
AUD2DAT         equ     aud2+$0A

AUD3LC          equ     aud3
AUD3LCH         equ     aud3
AUD3LCL         equ     aud3+$02
AUD3LEN         equ     aud3+$04
AUD3PER         equ     aud3+$06
AUD3VOL         equ     aud3+$08
AUD3DAT         equ     aud3+$0A
*
*       Los registros de los Bit-plane
*
BPL1PT          equ     bplpt+$00
BPL1PTH         equ     bplpt+$00
BPL1PTL         equ     bplpt+$02
BPL2PT          equ     bplpt+$04
BPL2PTH         equ     bplpt+$04
BPL2PTL         equ     bplpt+$06
BPL3PT          equ     bplpt+$08
BPL3PTH         equ     bplpt+$08
BPL3PTL         equ     bplpt+$0A
BPL4PT          equ     bplpt+$0C
BPL4PTH         equ     bplpt+$0C
BPL4PTL         equ     bplpt+$0E
BPL5PT          equ     bplpt+$10
BPL5PTH         equ     bplpt+$10
BPL5PTL         equ     bplpt+$12
BPL6PT          equ     bplpt+$14
BPL6PTH         equ     bplpt+$14
BPL6PTL         equ     bplpt+$16

BPLCON0         equ     bplmod0
BPLCON1         equ     bplmod1
BPLCON2         equ     bplcon2
BPL1MOD         equ     bpl1mod
BPL2MOD         equ     bpl2mod
```

```
SPR5POS         equ     spr+$28
SPR5CTL         equ     SPR5POS+sd_ctl
SPR5DATA        equ     SPR5POS+sd_data
SPR5DATB        equ     SPR5POS+$06

SPR6POS         equ     spr+$30
SPR6CTL         equ     SPR6POS+sd_ctl
SPR6DATA        equ     SPR6POS+sd_data
SPR6DATB        equ     SPR6POS+$06

SPR7POS         equ     spr+$38
SPR7CTL         equ     SPR7POS+sd_ctl
SPR7DATA        equ     SPR7POS+sd_data
SPR7DATB        equ     SPR7POS+$06
*
*       Registros del color
*
COLOR00         equ     color+$00
COLOR01         equ     color+$02
COLOR02         equ     color+$04
COLOR03         equ     color+$06
COLOR04         equ     color+$08
COLOR05         equ     color+$0A
COLOR06         equ     color+$0C
COLOR07         equ     color+$0E
COLOR08         equ     color+$10
COLOR09         equ     color+$12
COLOR10         equ     color+$14
COLOR11         equ     color+$16
COLOR12         equ     color+$18
COLOR13         equ     color+$1A
COLOR14         equ     color+$1C
COLOR15         equ     color+$1E
COLOR16         equ     color+$20
COLOR17         equ     color+$22
COLOR18         equ     color+$24
COLOR19         equ     color+$26
COLOR20         equ     color+$28
COLOR21         equ     color+$2A
COLOR22         equ     color+$2C
COLOR23         equ     color+$2E
COLOR24         equ     color+$30
COLOR25         equ     color+$32
COLOR26         equ     color+$34
COLOR27         equ     color+$36
COLOR28         equ     color+$38
COLOR29         equ     color+$3A
COLOR30         equ     color+$3C
COLOR31         equ     color+$3E

**********************************************************************
**
**
                END   ; HARDWARE_HW_EXAMPLES_I
```