rtfSpriteController

Table of Contents

[Overview 3](#_Toc347574591)

[Register Set 3](#_Toc347574592)

[Definitions 4](#_Toc347574593)

[Image Cache 4](#_Toc347574594)

[Register Descriptions 4](#_Toc347574595)

[Position Registers 4](#_Toc347574596)

[Horizontal Position 4](#_Toc347574597)

[Vertical Position 4](#_Toc347574598)

[Pixel Size 4](#_Toc347574599)

[Image Offset 4](#_Toc347574600)

[Transparent Color 5](#_Toc347574601)

[DMA Access 6](#_Toc347574602)

[DMA address 6](#_Toc347574603)

[DMA Trigger 6](#_Toc347574604)

[DMA Operation 6](#_Toc347574605)

[Programmed Access 6](#_Toc347574606)

[Global Registers 7](#_Toc347574607)

[Back Ground Color 7](#_Toc347574608)

[Sprite Enable 7](#_Toc347574609)

[Sprite Interrupt Enable 7](#_Toc347574610)

[Sprite Collisions 7](#_Toc347574611)

[Background Collision 7](#_Toc347574612)

[Sprite-Sprite Collision 7](#_Toc347574613)

[Clocks 8](#_Toc347574614)

[Ports 8](#_Toc347574615)

[Program Examples: 9](#_Toc347574616)

[WISHBONE Compatibility Datasheet 10](#_Toc347574617)

# Overview

This core provides support for moveable graphical images commonly known as sprites (or hardware cursors).

The core is parameterized to allow 1,2,4,6,8, or 14 sprites. The size of the core depends on the number of sprites selected.

# Register Set

The register set is located at the I/O address range of $DAD0xx.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Register | Bits | Function |  |  |
| 00 | [11:0] | Horizontal position | Position |  |
|  | [27:16] | Vertical position |  |  |
| 04 | [5:0] | Width of sprite in pixels | Size and Offset |  |
|  | [7:6] | Size of pixel in video clock cycles |  |  |
|  | [13:8] | Height of sprite in vertical pixels |  |  |
|  | [15:14] | Size of vertical pixels in scanlines |  |  |
|  | [26:16] | Sprite image offset in image cache |  |  |
| 08 | [20:0] | Sprite image system memory address  Bits 11 to 31 | DMA address |  |
| 0C | [7:0] | Transparent color |  |  |
| 10-DC |  | These are registers reserved for up to 13 more sprites same format as above four registers | | |
| Global Registers | | | | |
| E8 | [23:0] | Background transparent color |  |  |
| EC | [23:0] | Background color |  |  |
| F0 | [13:0] | Sprite enable |  |  |
|  | [26:16] | Sprite interrupt enable |  |  |
| F4 | [13:0] | Sprite-sprite collision |  |  |
|  | [29:16] | Sprite-background collision |  |  |
| F8 | [13:0] | DMA triggers |  |  |
| FC | [31:0] | DMA address bits 63 to 32 |  |  |

# Definitions

## Image Cache

The image cache is a block of memory containing the sprite image data that is 2048 x 8 bits in width. The sprite image cache may be loaded directly under program control like any other memory, or it may be loaded automatically under DMA control. The sprite image caches are exposed as a block of memory to the system at address $D8xxxx. Eight 2kB cache memories are combined into 16kB memory area.

# Register Descriptions

## Position Registers

The sprites position is relative to the positive edge of the externally supplied horizontal sync and vertical sync signals. The (zero, zero) point coincides with the horizontal sync and vertical sync signals and hence is not at the top left of the display. There is a offset from synchronization signals, required before the top left co-ordinate of the display. The top left of the visible display is approximately sprite co-ordinates (280, 50). Note that it is possible to position the sprite “off-screen” so that it doesn’t display.

The sprite extends to the right and downwards from the setting in the position register.

## Horizontal Position

This register specifies the horizontal position of the sprite with respect to the horizontal sync signal.

## Vertical Position

This register specifies the vertical position of the sprite with respect to the vertical sync signal.

## Pixel Size

The size of the pixels used to display the sprite may be controlled. Increasing the size of the pixels has the effect of increasing the size of the sprite. Sprites may be effectively 256 pixels in extent when the pixel size is increased to the maximum. Pixel size may be varied from one to four clock cycles or scan lines.

## Image Offset

The sprite uses a block RAM as an image cache. The amount of RAM available per sprite is 2Kb. Since the amount of RAM available is fairly large, multiple sprite images may be cached in a single buffer. The image offset is the offset into the cache buffer for the currently displayed sprite. Only one image at a time may be displayed from the image cache. Fortunately there is a separate image cache for each sprite.

Sprites may sized such that the product of the width and height is less than 2048. In this case the sprite image cache may hold multiple images. For example, if 16x16 sprites are used, eight separate images would be able to fit into a single image cache. Setting the sprite size to 8x8 would allow 32 different images to fit into the image cache. By cycling through the images different graphics effects can be created, For instance a rotating ball, or a flying bird.

## Transparent Color

The transparent color register defines which of 256 colors are transparent. If the color of the sprite pixel is equal to the transparent color, then the image underneath the sprite is visible. This has the effect of making portions of the sprite “transparent”.

# DMA Access

## DMA address

Sprite image caches may be loaded from memory using an internal DMA controller. The DMA address is formed from the global DMA address register coupled with the sprite DMA address register bits. The low order 11 bits of the DMA address are automatically generated by the DMA controller. The image memory must be aligned on a 2kB boundary. Note that a 64 bit address is supported. All sprites images must be within the same 4GB memory range.

## DMA Trigger

DMA begins when the DMA trigger register bit for a sprite is set.

## DMA Operation

The DMA controller uses 32 bit memory accesses to load the sprite image caches. Eight sets of sixty-four word burst accesses are required to load the cache. The DMA burst length is sixty-four words. If memory does not respond to a DMA request for more than 20 cycles, then the DMA controller will move onto the next address.

# Programmed Access

The sprite image caches may be loaded or manipulated directly by a processor. The image caches look like normal memory mapped into the I.O address range of $D8xxxx. Each images cache is 2Kb in size. All the image caches are contiguously mapped into the address range.

# Global Registers

## Back Ground Color

The background color register identifies which color of the background image is background. A sprite-background collision will NOT occur when the sprite obscures images of the background of the background color.

## Sprite Enable

The sprite enable register acts as on/off switches for the sprite display. Sprites will not display unless enabled.

## Sprite Interrupt Enable

This register controls which sprites are capable of causing interrupts due to a collision with another sprite or a background object.

## Sprite Collisions

If the display of two sprites overlap, a sprite-sprite collision is signalled and recorded in the sprite-collision register. Note that the transparent color does not cause a collision. Sprite regions may overlap without a collision as long as a transparent color is being displayed. The transparent color allows irregularly shaped collision regions.

## Background Collision

A sprite-background collision is signalled when the sprite is in a display region not defined as the background color. As long as the sprite intersects display areas defined as the background color, no sprite-background collision will be signalled.

## Sprite-Sprite Collision

This register indicates which sprites are colliding.

Clocks

The sprite controller uses separate system bus and video pixel clocks which do not have to be related

# Ports

|  |  |  |  |
| --- | --- | --- | --- |
| Port | Size | Description |  |
| Rst\_i | 1 | This signal reset the core |  |
| Clk\_i | 1 | Bus clock |  |
| S\_cyc\_i | 1 | Slave bus cycle is active |  |
| S\_stb\_i | 1 | Slave data transfer is taking place |  |
| S\_ack\_o | 1 | Data transfer acknowledge, generated by the controller |  |
| S\_we\_i | 1 | Indicates a write to the controller is taking place |  |
| S\_sel\_i | 4 | Byte lane select, only byte lanes identified by this signal will be written. |  |
| S\_adr\_i | 24 | Slave address input, used to address the sprite registers and image caches. |  |
| S\_dat\_i | 32 | Data input to the core |  |
| S\_dat\_o | 32 | Data output from the core |  |
|  |  |  |  |
| M\_bte\_o | 2 | This signal indicate the burst type, only type 0 is supported |  |
| M\_cti\_o | 3 | This signal indicates that burst access is taking place |  |
| M\_bl\_o | 6 | This signal indicates the burst length. It outputs 63 for a burst length of 64 words. |  |
| M\_cyc\_o | 1 | This signal indicates that a DMA burst cycle is active |  |
| M\_stb\_o | 1 | This signal indicates when a data transfer is taking place |  |
| M\_ack\_i | 1 | Data transfer acknowledge from memory |  |
| M\_we\_o | 1 | Not used, always zero |  |
| M\_sel\_o | 4 | Will be hF when a DMA is taking place |  |
| M\_adr\_o | 64 | System address for DMA transfer |  |
| M\_dat\_i | 32 | Data input to the core |  |
| M\_dat\_o | 32 | Not used. Always zero |  |
|  |  |  |  |
| vclk | 1 | Video pixel clock |  |
| hSync | 1 | Horizontal sync input to the core |  |
| vSync | 1 | Vertical sync input to the core |  |
| blank | 1 | Blaking signal input to the core |  |
| rgbIn | 24 | Background image input. |  |
| rgbOut | 24 | Video output from core |  |
|  |  |  |  |
| irq | 1 | Interrupt request line |  |
|  |  |  |  |

# Program Examples:

The following code written in 68000 assembler language randomizes the sprite memory. It causes the sprites to display as a block of randomly colored pixels. It shows that the image cache is available to the system.

RANDOM EQU 0xFFDC0C00

SPRITERAM EQU 0xFFD80000

; randomize sprite memory

move.l #32768,d1

lea SPRITERAM,a0

main6:

move.l RANDOM,d0 ; load from hardware random # generator

move.w d0,(a0)+

subi.l #1,d1

bne main6

# WISHBONE Compatibility Datasheet

The rtfSpriteController core may be directly interfaced to a WISHBONE compatible bus.

|  |  |  |
| --- | --- | --- |
| WISHBONE Datasheet  WISHBONE SoC Architecture Specification, Revision B.3 | | |
|  |  | |
| Description: | Specifications: | |
| General Description: | Hardware cursor / sprite controller | |
| Supported Cycles: | SLAVE, READ / WRITE  SLAVE, BLOCK READ / WRITE  SLAVE, RMW | |
| Data port, size:  Data port, granularity:  Data port, maximum operand size:  Data transfer ordering:  Data transfer sequencing | 32 bit  8 bit  32 bit  Little Endian  any (undefined) | |
| Clock frequency constraints: |  | |
| Supported signal list and cross reference to equivalent WISHBONE signals | Signal Name:  S\_ack\_o  S\_adr\_i(23:0)  S\_clk\_i  S\_dat\_i(31:0)  S\_dat\_o(31:0)  S\_cyc\_i  S\_stb\_i  S\_we\_i | WISHBONE Equiv.  ACK\_O  ADR\_I()  CLK\_I  DAT\_I()  DAT\_O()  CYC\_I  STB\_I  WE\_I |
| Special Requirements: |  | |