rtfSpriteController

Table of Contents

Overview

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

The core is parameterized to allow 1,2,4,6,8,14, or 32 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 $FFDADxxx. Note that the sprite registers are 8, 16, or 32 bit addressable. For instance the vertical position may be updated by writing a 16 bit value to register $02.

Unused bits in the registers should be set to zero.

Register	Bits	Function		
00	[11:0]	Horizontal position	Position	
	[27:16]	Vertical position		
04	[7:0]	Width of sprite in pixels	Size	
	[15:8]	Height of sprite in vertical pixels		
	[19:16]	Horizontal size of pixel in video clock cycles		
	[23:20]	Vertical size of pixels in scanlines		
08	[11:0]	Sprite image offset in image cache		
	[31:12]	Sprite image system memory address Bits 12 to 31	DMA address	
0C	[15:0]	Transparent color		
10-1FC		These are registers reserved for up to 31 more sprites same format as above four registers		
Global Registers				
3C0	[31:0]	Sprite enable		
3C4	[0]	Sprite-sprite collision interrupt enable	Interrupt Enable / Status	
	[1]	Sprite-background collision interrupt enable		
3C8	[31:0]	Sprite-sprite collision record		
3CC	[31:0]	Sprite-background collision record		
3D0	[31:0]	DMA trigger on		
3D4	[31:0]	DMA trigger off		
3E8	[23:0]	Background transparent color		
3EC	[23:0]	Background color		
3FC	[31:0]	DMA address bits 63 to 32	currently unimplemented	

Definitions

Image Cache

The image cache is a block of memory containing the sprite image data that is 4096 x 8 or 2048 x 16 bits in width. The sprite image cache may be loaded directly under program control (PIO) 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 $FFD8xxxx. Eight 4kB cache memories are combined into 32kB 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 an 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 4096 pixels in extent when the pixel size is increased to the maximum. Pixel size may be varied from one to sixteen 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 4KB. 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 4096 for eight bit color or 2048 for sixteen bit color. In this case the sprite image cache may hold multiple images. For example, if 16x16 sprites are used, sixteen separate images would be able to fit into a single image cache. Setting the sprite size to 8x8 would allow 64 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 256/32k 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”.

Color Representation

The core may be configured via a parameter to use either 8 bits or 16 bits per pixel to represent color. In the sixteen bits per pixel mode, 1 bit is reserved to indicate alpha blending. Colors are (3,3,2) for (R,G,B) in eight bit mode or (1,5,5,5) for (A,R,G,B) in sixteen bit color mode.

Alpha Blending

Color alpha blending functionality is available when the core is configured for 15 bit color representations. The alpha blending factor may be used to create a shadow effect under the sprite. The alpha blending is indicated by the most significant bit of the color. If the MSB is set to a one, then the lower eight bits of the color represent an alpha blending factor. The alpha blending blends towards black or white. A fixed point arithmetic multiply is used for blending.

The alpha is eight bits ranging between 0 and 1.999...

1 bit whole, 7 bits fraction

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 12 bits of the DMA address are automatically generated by the DMA controller. The image memory must be aligned on a 4kB boundary. Note that a 32 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. 1024, 32 bit memory accesses are required to load each sprite memory.

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 $FFD80000 to $FFD9FFFF. Each images cache is 4KB 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	(slave) 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	34	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. currently only normal cycles (000) are supported	
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	32	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	Blanking signal input to the core	
rgbIn	24	Background image input.	
rgbOut	24	Video output from core	
irq	1	Interrupt request line	

Parameters

pnSpr – controls the number of sprites, values 1,2,4,6,8, 14, or 32

pColorBits – controls the number of bits used for color representation, valid values are 8 or 16.

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.wd0,(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	WISHBONE Equiv. ACK_O ADR_I() CLK_I DAT_I() DAT_O() CYC_I

	S_stb_i S_we_i	STB_I WE_I
Special Requirements:		