rfSpriteController

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 to 32 sprites. The size of the core depends on the number of sprites selected.

The core has two interfaces to the system, a 32-bit slave interface to update the registers, and a 128-bit DMA master interface for loading sprite image data.

Register Set

Config Space

A 256-byte config space is supported. Most of the config space is unused. The only configuration is for the I/O address of the register set.

Regno	Width	R/W	Moniker	Description		
000	32	RO	REG_ID	Vendor and device ID		
004	32	R/W				
008	32	RO				
00C	32	R/W				
010	32	R/W	REG_BAR0	Base Address Register		
014	32	R/W	REG_BAR1	Base Address Register		
018	32	R/W	REG_BAR2	Base Address Register		
01C	32	R/W	REG_BAR3	Base Address Register		
020	32	R/W	REG_BAR4	Base Address Register		
024	32	R/W	REG_BAR5	Base Address Register		
028	32	R/W				
02C	32	RO		Subsystem ID		
030	32	R/W		Expansion ROM address		
034	32	RO				
038	32	R/W		Reserved		
03C	32	R/W		Interrupt		
040 to 0FF	32	R/W		Capabilities area		

REG_BAR0 defaults to $FDDAD001 which is used to specify the address of the controller's registers in the I/O address space.

The controller will respond with a memory size request of 0MB (0xFFFFFFFF) when BAR0 is written with all ones. The controller contains its own dedicated memory and does not require memory allocated from the system.

Parameters

CFG_BUS defaults to zero
CFG_DEVICE defaults to two

CFG_FUNC defaults to zero

Config parameters must be set correctly. CFG device and vendors default to zero.

Control Registers

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			
	[27:24]	output plane			
	[31:30]	color depth	1=8 bits ARGB2222, 2=16-bit ARGB4444, 3=32-bit ARGB5999		
08	[11:0]	Sprite image offset in image cache		Address	
	[31:12]	Sprite image system memory address Bits 12 to 31		DMA address low	
0C	[23:0]	Transparent color			
10-1FC		These are registers reserved for up to 31 more sprites same format as above four registers			
200	[8:0]	Sprite #0 Burst start – address bits 3 to 11			
	[24:16]	Sprite #0 Burst end – address bits 3 to 11			
204 to 27C		Burst start/end for 31 more sprites			
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			
3D8	[31:0]	Vertical Sync DMA trigger			
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 bytes in size. Each sprite has its own image cache which may be used to store multiple images. The sprite image cache is loaded automatically under DMA control and is not visible to the system. The product of the horizontal and vertical resolution for the sprite combined with the sprite's color pixel depth should not exceed 4096 bytes.

Register Descriptions

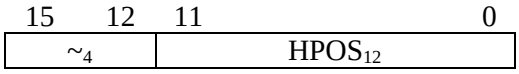
Position Registers

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.

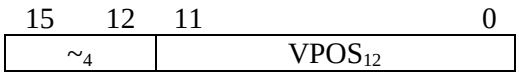
Horizontal Position (n0)

This register specifies the horizontal position of the sprite with respect to the horizontal sync signal. There are only 12 significant bits.



Vertical Position (n2)

This register specifies the vertical position of the sprite with respect to the vertical sync signal. There are only 12 significant bits.



Sprite Width

The width of the sprite is controlled by this register. The width may vary from 1 to 256 pixels. The default width is 56 pixels. The value in this register is one less than the width. Note that the product of width, height and color depth cannot exceed 4096 bytes.

Sprite Height

The height of the sprite is controlled by this register. The height may vary from 1 to 256 pixels. The default height is 36 pixels.

Output Plane

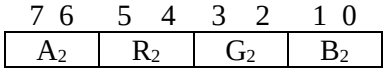
This register controls which plane the sprite is in relative to the external bitmap graphic input. The output plane and the external input plane work together to control the display. If the external input plane is numerically higher than the sprite's output plane then the external input will appear in front of the sprite, otherwise it will appear behind the sprite.

Color Depth

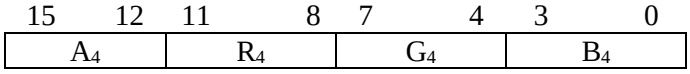
The sprite color depth register controls the number of bits used to represent color for the sprite. One of three depths are possible, eight-bit color, sixteen-bit color and thirty-two-bit color depths are available. In the sixteen bits per pixel mode, 4 bits are reserved to indicate alpha blending. Colors are (2,2,2,2) for (A,R,G,B) in eight-bit mode or (4,4,4,4) for (A,R,G,B) in sixteen-bit color mode. For thirty-two-bit color depth ARGB (5,9,9,9) is used.

Note that bigger color depths require the sprite dimensions to be smaller as the amount of memory for the sprite image is limited.

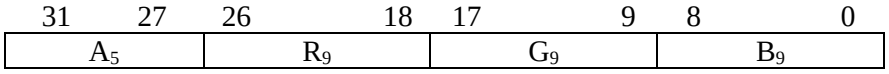
Eight-bit Color



Sixteen-bit Color



Thirty-two-bit Color



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 1024 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 64/4k 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”. The number of bits used to match the transparent color depends on the color depth for the sprite. For example, for eight-bit color depth only the low order eight bits of the transparent color register are tested.

Alpha Blending

The alpha blending factor may be used to create a shadow effect under the sprite. A fixed-point arithmetic multiply is used for blending.

The alpha bits range between 0 and 1.999...

1-bit whole, remaining bits are fractional

Thirty-two-bit colors have the capability of blending the input image color and the sprite image color according to an five-bit alpha value which is 1 bit whole, 4 bits fraction.

DMA Access

DMA address

Sprite image caches are 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 64-bit address is supported. However, all sprite images must be within the same 4GB memory range.

DMA Burst Start and End

The DMA burst start and end registers allow a subset of the total cache to be loaded. For instance, if images use only 1kB of memory then the burst start may be set to 0 and the end set to 127. This will load 128 consecutive 8-byte pieces of the sprite image into the cache.

DMA Trigger

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

DMA Vertical Sync Triggered

DMA operations may also be triggered by the vertical sync pulse.

DMA Operation

The DMA controller uses 64-bit memory accesses to load the sprite image caches. 512, 64-bit memory accesses are required to load each sprite memory.

Global Registers

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 that contains graphics on the same plane as the sprite's plane.

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

Bus master and slave ports use structured variable types that encapsulate the Wishbone bus. These are divided into request and response busses. The components of the request and response busses are outlined in the table below.

Port	Size	Description	
Rst_i	1	This signal reset the core	
Clk_i	1	(slave) Bus clock	
s_cs_reg_i	1	register set circuit select	
Wbs_req			
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	32	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	64	Data input to the core	
M_dat_o	64	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	
Zrgb_i	40	External image input.	
Zrgb_o	40	Video output from core	

irq	1	Interrupt request line	

Parameters

pnSpr – controls the maximum number of sprites, values 1 to 32

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.

```
RANDOM      EQU         0xFFDC0C00
SPRITERAM   EQU         0x00080000
        move.l  #0x80000,d1         ; set sprite #0 image data address
        move.l d1,SPRCTRL+8
        ; 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
        move.l  #1,SPRCTRL+0x3D0    ; trigger sprite DMA to load cache
```

WISHBONE Compatibility Datasheet

The rfSpriteController 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	WISHBONE Equiv. ACK_O ADR_I() CLK_I DAT_I() DAT_O() CYC_I STB_I

	S_we_i	WE_I
	M_ack_i	ACK_I
	M_adr_o(31:0)	ADR_O()
	M_clk_i	CLK_I
	m_dat_i(63:0)	DAT_I()
	m_dat_o(63:0)	DAT_O()
	m_cyc_o	CYC_O
	m_stb_o	STB_O
	m_we_o	WE_O
Special Requirements:		