RIVA 128ZX™

128-BIT 3D MULTIMEDIA ACCELERATOR

PRELIMINARY DATA

KEY FEATURES

- Fast 32-bit VGA/SVGA
- High performance 128-bit 2D/GUI/DirectDraw Acceleration
- Interactive, Photorealistic Direct3D Acceleration with advanced effects
- Pinout backwards compatible with RIVA 128
- Massive 1.6Gbytes/s, 100MHz 128-bit wide 8MByte SGRAM framebuffer interface
- Adds 16Mbit SDRAM support for cost sensitive 8MByte framebuffer applications
- Video Acceleration for DirectDraw/DirectVideo, MPEG-1/2 and Indeo®
  - Planar 4:2:0 and packed 4:2:2 Color Space Conversion
  - X and Y smooth up and down scaling
- 250MHz Palette-DAC supporting up to 1600x1200@85Hz
- NTSC and PAL output with flicker-filter
- Multi-function Video Port and serial interface
- Bus mastering DMA Accelerated Graphics Port (AGP) 1.0 Interface supporting 133MHz 2X data transfer mode
- Bus mastering DMA PCI 2.1 interface
- ACPI power management interface support
- 0.35 micron 5LM CMOS
- 300 PBGA

DESCRIPTION

The RIVA128ZX™ offers unparalleled 2D and 3D performance, meeting all the requirements of the mainstream PC graphics market and Microsoft's PC'97. RIVA128ZX combines all the features of RIVA 128 plus 8MByte SDRAM and SGRAM based framestore support and AGP 2X data transfer. It provides the most advanced Direct3D™ acceleration solution and delivers leadership VGA, 2D and Video performance, enabling a range of applications from 3D games through to DVD, Intercast™ and video conferencing.

BLOCK DIAGRAM

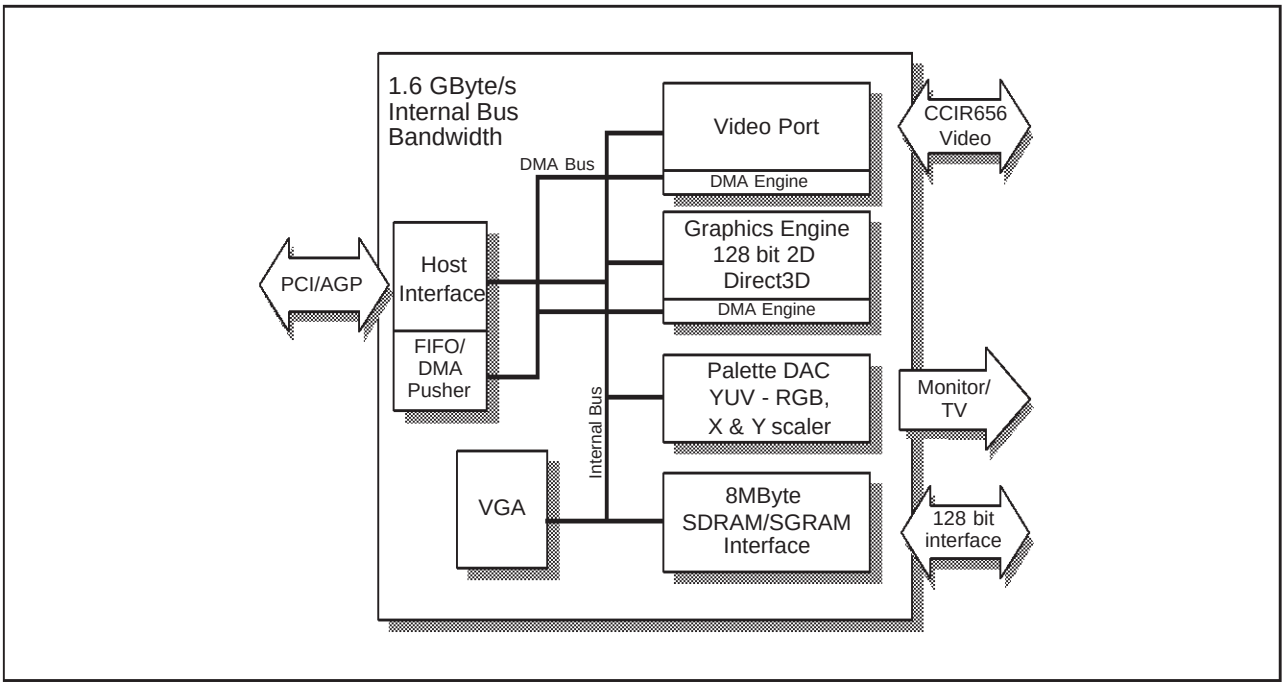
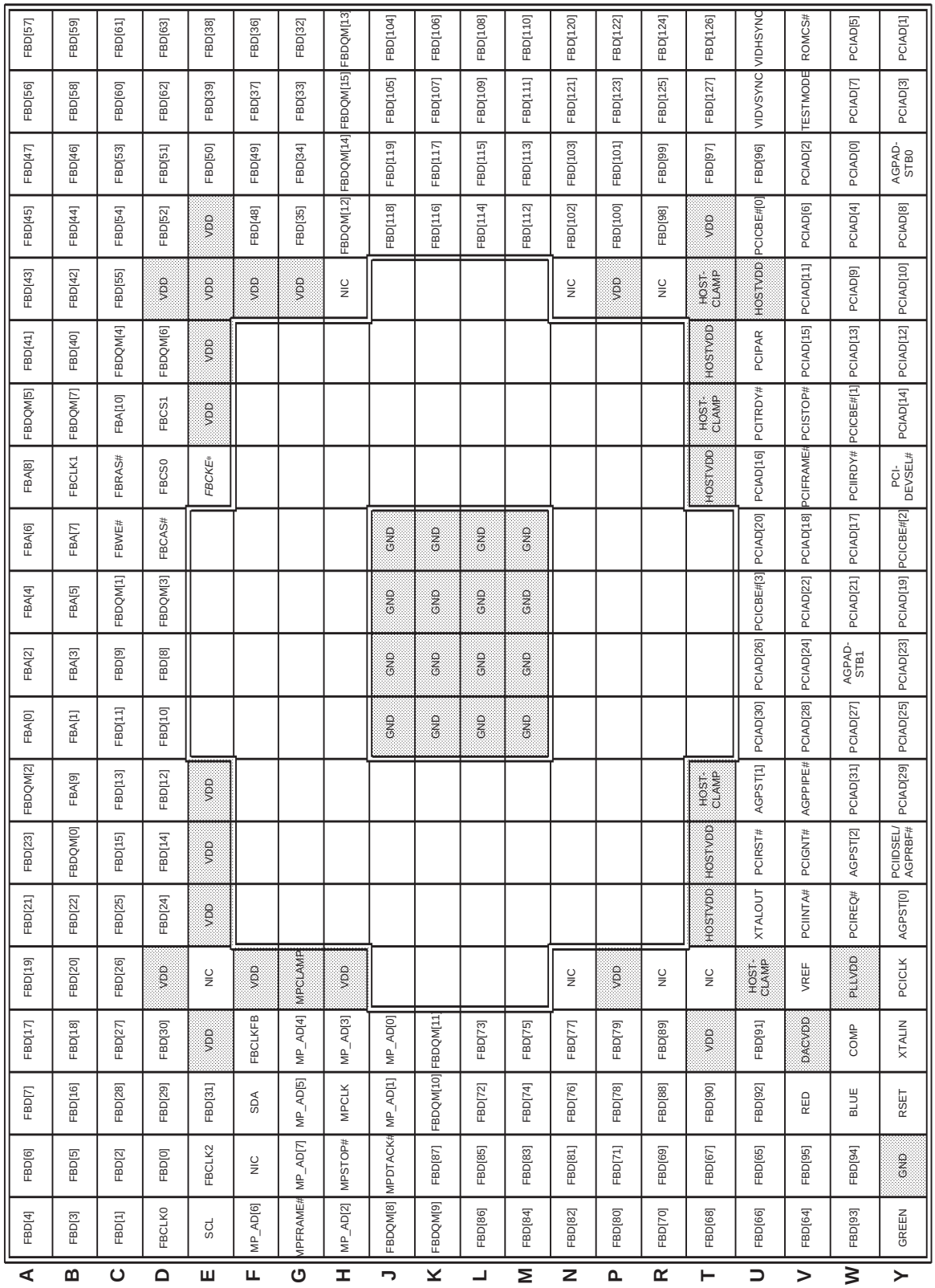


TABLE OF CONTENTS

1 RIVA128ZX 300PBGA DEVICE PINOUT



	1	2	3	4	5	6	7	8	9	10	11	12	13	15	14	16	17	18	19	20
A	FBD[4]	FBD[6]	FBD[7]	FBD[17]	FBD[19]	FBD[21]	FBD[23]	FBDQM[2]	FBA[0]	FBA[2]	FBA[4]	FBA[6]	FBA[8]	FBD[41]	FBDQM[5]	FBD[43]	FBD[45]	FBD[47]	FBD[56]	FBD[57]
B	FBD[3]	FBD[5]	FBD[16]	FBD[18]	FBD[20]	FBD[22]	FBDQM[0]	FBA[9]	FBA[1]	FBA[3]	FBA[5]	FBA[7]	FBCLK1	FBD[40]	FBDQM[7]	FBD[42]	FBD[44]	FBD[46]	FBD[58]	FBD[59]
C	FBD[1]	FBD[2]	FBD[28]	FBD[27]	FBD[26]	FBD[25]	FBD[15]	FBD[13]	FBD[11]	FBD[9]	FBDQM[1]	FBWE#	FBRAS#	FBDQM[4]	FBA[10]	FBD[55]	FBD[54]	FBD[53]	FBD[60]	FBD[61]
D	FBCLK0	FBD[0]	FBD[29]	FBD[30]	VDD	FBD[24]	FBD[14]	FBD[12]	FBD[10]	FBD[8]	FBDQM[3]	FBCAS#	FBCS0	FBDQM[6]	FBCS1	VDD	VDD	FBD[51]	FBD[62]	FBD[63]
E	SCL	FBCLK2	FBD[31]	VDD	NIC	VDD	VDD	VDD					*FBCKE**	VDD	VDD	VDD	VDD	FBD[50]	FBD[39]	FBD[38]
F	MP_AD[6]	NIC	SDA	FBCLKFB	VDD											VDD	FBD[48]	FBD[49]	FBD[37]	FBD[36]
G	MPFRAME#	MP_AD[7]	MP_AD[5]	MP_AD[4]	MPCLAMP											VDD	FBD[35]	FBD[34]	FBD[33]	FBD[32]
H	MP_AD[2]	MPSTOP#	MPCLK	MP_AD[3]	VDD											NIC	FBDQM[12]	FBDQM[14]	FBDQM[15]	FBDQM[13]
J	FBDQM[8]	MPDTACK#	MP_AD[1]	MP_AD[0]					GND	GND	GND	GND					FBD[118]	FBD[119]	FBD[105]	FBD[104]
K	FBDQM[9]	FBD[87]	FBDQM[10]	FBDQM[11]					GND	GND	GND	GND					FBD[116]	FBD[117]	FBD[107]	FBD[106]
L	FBD[86]	FBD[85]	FBD[72]	FBD[73]					GND	GND	GND	GND					FBD[114]	FBD[115]	FBD[109]	FBD[108]
M	FBD[84]	FBD[83]	FBD[74]	FBD[75]					GND	GND	GND	GND					FBD[112]	FBD[113]	FBD[111]	FBD[110]
N	FBD[82]	FBD[81]	FBD[76]	FBD[77]	NIC											NIC	FBD[102]	FBD[103]	FBD[121]	FBD[120]
P	FBD[80]	FBD[71]	FBD[78]	FBD[79]	VDD											VDD	FBD[100]	FBD[101]	FBD[123]	FBD[122]
R	FBD[70]	FBD[69]	FBD[88]	FBD[89]	NIC											NIC	FBD[98]	FBD[99]	FBD[125]	FBD[124]
T	FBD[68]	FBD[67]	FBD[90]	VDD	NIC	HOSTVDD	HOSTVDD	HOST-CLAMP					HOSTVDD	HOSTVDD	HOST-CLAMP	HOST-CLAMP	VDD	FBD[97]	FBD[127]	FBD[126]
U	FBD[66]	FBD[65]	FBD[92]	FBD[91]	HOST-CLAMP	XTALOUT	PCIRST#	AGPST[1]	PCIAD[30]	PCIAD[26]	PCICBE#[3]	PCIAD[20]	PCIAD[16]	PCIPAR	PCITRDY#	HOSTVDD	PCICBE#[0]	FBD[96]	VIDVSYNC	VIDHSYNC
V	FBD[64]	FBD[95]	RED	DACVDD	VREF	PCIINTA#	PCIGNT#	AGPPIPE#	PCIAD[28]	PCIAD[24]	PCIAD[22]	PCIAD[18]	PCIFRAME#	PCIAD[15]	PCISTOP#	PCIAD[11]	PCIAD[6]	PCIAD[2]	TESTMODE	ROMCS#
W	FBD[93]	FBD[94]	BLUE	COMP	PLLVDD	PCIREQ#	AGPST[2]	PCIAD[31]	PCIAD[27]	AGPAD-STB1	PCIAD[21]	PCIAD[17]	PCIIRDY#	PCIAD[13]	PCICBE#[1]	PCIAD[9]	PCIAD[4]	PCIAD[0]	PCIAD[7]	PCIAD[5]
Y	GREEN	GND	RSET	XTALIN	PCICLK	AGPST[0]	PCIIDSEL/ AGPRBF#	PCIAD[29]	PCIAD[25]	PCIAD[23]	PCIAD[19]	PCICBE#[2]	PCI-DEVSEL#	PCIAD[12]	PCIAD[14]	PCIAD[10]	PCIAD[8]	AGPAD-STB0	PCIAD[3]	PCIAD[1]

NOTES

1 NIC = No Internal Connection. Do not connect to these pins.

2 VDD=3.3V

* Signals denoted with an asterisk are defined for future expansion. See *Pin Descriptions*, Section 2, page 5 for details.

2 PIN DESCRIPTIONS

2.1 ACCELERATED GRAPHICS PORT (AGP) INTERFACE

Signal	I/O	Description
AGPST[2:0]	I	AGP status bus providing information from the arbiter to the RIVA128ZX on what it may do. **AGPST[2:0]** only have meaning to the RIVA128ZX when **PCIGNT#** is asserted. When **PCIGNT#** is de-asserted these signals have no meaning and must be ignored. 000 Indicates that previously requested low priority read or flush data is being returned to the RIVA128ZX. 001 Indicates that previously requested high priority read data is being returned to the RIVA128ZX. 010 Indicates that the RIVA128ZX is to provide low priority write data for a previous enqueued write command. 011 Indicates that the RIVA128ZX is to provide high priority write data for a previous enqueued write command. 100 Reserved 101 Reserved 110 Reserved 111 Indicates that the RIVA128ZX has been given permission to start a bus transaction. The RIVA128ZX may enqueue AGP requests by asserting **AGPPIPE#** or start a PCI transaction by asserting **PCIFRAME#**. **AGPST[2:0]** are always an output from the Core Logic (AGP chipset) and an input to the RIVA128ZX.
AGPRBF#	O	Read Buffer Full indicates when the RIVA128ZX is ready to accept previously requested low priority read data or not. When **AGPRBF#** is asserted the arbiter is not allowed to return (low priority) read data to the RIVA128ZX. This signal should be pulled up via a 4.7KΩ resistor (although it is supposed to be pulled up by the motherboard chipset).
AGPPIPE#	O	Pipelined Read is asserted by RIVA128ZX (when the current master) to indicate a full width read address is to be enqueued by the target. The RIVA128ZX enqueues one request each rising clock edge while **AGPPIPE#** is asserted. When **AGPPIPE#** is de-asserted no new requests are enqueued across **PCIAD[31:0]**. **AGPPIPE#** is a sustained tri-state signal from the RIVA128ZX and is an input to the target (the core logic).
AGPADSTB0, **AGPADSTB1**	I/O	Bus strobe signals providing timing for AGP 2X data transfer mode on **PCIAD[15:00]** and **PCIAD[31:16]** respectively. The agent that is supplying data drives these signals.

2.2 PCI 2.1 LOCAL BUS INTERFACE

Signal	I/O	Description
PCICLK	I	PCI clock. This signal provides timing for all transactions on the PCI bus, except for **PCIRST#** and **PCIINTA#**. All PCI signals are sampled on the rising edge of **PCICLK** and all timing parameters are defined with respect to this edge.
PCIRST#	I	PCI reset. This signal is used to bring registers, sequencers and signals to a consistent state. When **PCIRST#** is asserted all output signals are tristated.
PCIAD[31:0]	I/O	32-bit multiplexed address and data bus. A bus transaction consists of an address phase followed by one or more data phases.

Signal	I/O	Description
PCICBE[3:0]#	I/O	Multiplexed bus command and byte enable signals. During the address phase of a transaction **PCICBE[3:0]#** define the bus command, during the data phase **PCICBE[3:0]#** are used as byte enables. The byte enables are valid for the entire data phase and determine which byte lanes contain valid data. **PCICBE[0]#** applies to byte 0 (LSB) and **PCICBE[3]#** applies to byte 3 (MSB). When connected to AGP these signals carry different commands than PCI when requests are being enqueued using **AGPPIPE#**. Valid byte information is provided during AGP write transactions. **PCICBE[3:0]#** are not used during the return of AGP read data.
PCIPAR	I/O	Parity. This signal is the even parity bit generated across **PCIAD[31:0]** and **PCICBE[3:0]#**. **PCIPAR** is stable and valid one clock after the address phase. For data phases **PCIPAR** is stable and valid one clock after either **PCIIRDY#** is asserted on a write transaction or **PCITRDY#** is asserted on a read transaction. Once **PCIPAR** is valid, it remains valid until one clock after completion of the current data phase. The master drives **PCIPAR** for address and write data phases; the target drives **PCIPAR** for read data phases.
PCIFRAME#	I/O	Cycle frame. This signal is driven by the current master to indicate the beginning of an access and its duration. **PCIFRAME#** is asserted to indicate that a bus transaction is beginning. Data transfers continue while **PCIFRAME#** is asserted. When **PCIFRAME#** is deasserted, the transaction is in the final data phase.
PCIIRDY#	I/O	Initiator ready. This signal indicates the initiator's (bus master's) ability to complete the current data phase of the transaction. See extended description for **PCITRDY#**. When connected to AGP this signal indicates the initiator (AGP compliant master) is ready to provide all write data for the current transaction. Once **PCIIRDY#** is asserted for a write operation, the master is not allowed to insert wait states. The assertion of **PCIIRDY#** for reads, indicates that the master is ready to transfer a subsequent block of read data. The master is never allowed to insert a wait state during the initial block of a read transaction. However, it may insert wait states after each block transfers.
PCITRDY#	I/O	Target ready. This signal indicates the target's (selected device's) ability to complete the current data phase of the transaction. **PCITRDY#** is used in conjunction with **PCIIRDY#**. A data phase is completed on any clock when both **PCITRDY#** and **PCIIRDY#** are sampled as being asserted. During a read, **PCITRDY#** indicates that valid data is present on **PCIAD[31:0]**. During a write, it indicates the target is prepared to accept data. Wait cycles are inserted until both **PCIIRDY#** and **PCITRDY#** are asserted together. When connected to AGP this signal indicates the AGP compliant target is ready to provide read data for the entire transaction (when transaction can complete within four clocks) or is ready to transfer a (initial or subsequent) block of data, when the transfer requires more than four clocks to complete. The target is allowed to insert wait states after each block transfers on both read and write transactions.
PCISTOP#	I/O	**PCISTOP#** indicates that the current target is requesting the master to terminate the current transaction.
PCIIDSEL	I	Initialization device select. This signal is used as a chip select during configuration read and write transactions. For AGP applications note that IDSEL is not a pin on the AGP connector. The RIVA128ZX performs the device select decode internally within its host interface. It is not required to connect the AD16 signal to the IDSEL pin as suggested in the AGP specification.
PCIDEVSEL#	I/O	Device select. When acting as an output **PCIDEVSEL#** indicates that the RIVA128ZX has decoded the PCI address and is claiming the current access as the target. As an input **PCIDEVSEL#** indicates whether any other device on the bus has been selected.
PCIREQ#	O	Request. This signal is asserted by the RIVA128ZX to indicate to the arbiter that it desires to become master of the bus.

Signal	I/O	Description
PCIGNT#	I	Grant. This signal indicates to the RIVA128ZX that access to the bus has been granted and it can now become bus master. When connected to AGP additional information is provided on **AGPST[2:0]** indicating that the master is the recipient of previously requested read data (high or low priority), it is to provide write data (high or low priority), for a previously enqueued write command or has been given permission to start a bus transaction (AGP or PCI).
PCIINTA#	O	Interrupt request line. This open drain output is asserted and deasserted asynchronously to **PCICLK**.

2.3 FRAMEBUFFER INTERFACE

Signal	I/O	Description
FBD[127:0]	I/O	The 128-bit memory data bus. **FBD[31:0]** are also used to access up to 64KBytes of 8-bit ROM or Flash ROM, using **FBD[15:0]** as address ROMA[15:0], **FBD[31:24]** as ROMD[7:0], **FBD[17]** as ROMWE# and **FBD[16]** as ROMOE#.
FBA[10:0]	O	Memory Address bus. Configuration strapping options are also decoded on these signals during PCIRST# as described in Section 10, page 55.
FBRAS#	O	Memory Row Address Strobe for all memory devices.
FBCAS#	O	Memory Column Address Strobe for all memory devices.
FBCS[1:0]#	O	Memory Chip Select strobes. For SDRAM the **FBCS[1]** pin provides the memory's internal bank select bit (BA/A11).
FBWE#	O	Memory Write Enable strobe for all memory devices.
FBDQM[15:0]	O	Memory Data/Output Enable strobes.
FBCLK0, FBCLK1, FBCLK2	O	Memory Clock signals. Separate clock signals **FBCLK0** and **FBCLK1** are provided for each bank of memory for reduced clock skew and loading. Details of recommended memory clock layout are given in Section 6.4, page 37.
FBCLKFB	I	Framebuffer clock feedback. **FBCLK2** is fed back to **FBCLKFB**.
FBCKE	O	Framebuffer memory clock enable signal.

2.4 VIDEO PORT

Signal	I/O	Description
MP_AD[7:0]	I/O	Media Port 8-bit multiplexed address and data bus or ITU-R-656 video data bus when in 656 mode.
MPCLK	I	40MHz Media Port system clock or pixel clock when in 656 mode.
MPDTACK#	I	Media Port data transfer acknowledgment signal.
MPFRAME#	O	Initiates Media Port transfers when active, terminates transfers when inactive.
MPSTOP#	I	Media Port control signal used by the slave to terminate transfers.

2.5 DEVICE ENABLE SIGNALS

Signal	I/O	Description
ROMCS#	O	Enables reads from an external 64Kx 8 or 32Kx8 ROM or Flash ROM. This signal is used in conjunction with framebuffer data lines as described above in Section 2.3.

2.6 DISPLAY INTERFACE

Signal	I/O	Description
SDA	I/O	Used for DDC2B+ monitor communication and interface to video decoder devices.
SCL	I/O	Used for DDC2B+ monitor communication and interface to video decoder devices.
VIDVSYNC	O	Vertical sync supplied to the display monitor. No buffering is required. In TV mode this signal supplies composite sync to an external PAL/NTSC encoder.
VIDHSYNC	O	Horizontal sync supplied to the display monitor. No buffering is required.

2.7 VIDEO DAC AND PLL ANALOG SIGNALS

Signal	I/O	Description
RED, GREEN, BLUE	O	RGB display monitor outputs. These are software configurable to drive either a doubly terminated or singly terminated 75Ω load.
COMP	-	External compensation capacitor for the video DACs. This pin should be connected to **DACVDD** via the compensation capacitor, see Figure 66, page 60.
RSET	-	A precision resistor placed between this pin and GND sets the full-scale video DAC current, see Figure 66, page 60.
VREF	-	A capacitor should be placed between this pin and GND as shown in Figure 66, page 60.
XTALIN	I	A series resonant crystal is connected between these two points to provide the reference clock for the internal MCLK and VCLK clock synthesizers, see Figure 66 and Table 20, page 60. Alternately, an external LVTTL clock oscillator output may be driven into **XTALOUT**, connecting **XTALIN** to GND. For designs supporting TV-out, **XTALOUT** should be driven by a reference clock as described in Section 11.6, page 61.
XTALOUT	O	

2.8 POWER SUPPLY

Signal	I/O	Description
DACVDD	P	Analog power supply for the video DACs.
PLLVDD	P	Analog power supply for all clock synthesizers.
VDD	P	Digital power supply.
GND	P	Ground.
MPCLAMP	P	**MPCLAMP** is connected to +5V to protect the 3.3V RIVA128ZX from external devices which will potentially drive 5V signal levels onto the Video Port input pins.
HOSTVDD	P	**HOSTVDD** is connected to the Vddq 3.3 pins on the AGP connector. This is the supply voltage for the I/O buffers and is isolated from the core VDD. On AGP designs these pins are also connected to the **HOSTCLAMP** pins. On PCI designs they are connected to the 3.3V supply.
HOSTCLAMP	P	**HOSTCLAMP** is the supply signalling rail protection for the host interface. In AGP designs these signals are connected to Vddq 3.3. For PCI designs they are connected to the I/O power pins ($V_{(I/O)}$).

2.9 TEST

Signal	I/O	Description
TESTMODE	I	For designs which will be tested in-circuit, this pin should be connected to GND through a 10KΩ pull-down resistor, otherwise this pin should be connected directly to GND. When **TESTMODE** is asserted, **MP_AD[3:0]** are reassigned as **TESTCTL[3:0]** respectively. Information on in-circuit test is given in Section 12, page 63.

3 OVERVIEW OF THE RIVA128ZX

The RIVA128ZX is the first 128-bit 3D Multimedia Accelerator to offer unparalleled 2D and 3D performance, meeting all the requirements of the mainstream PC graphics market and Microsoft's PC'97. The RIVA128ZX introduces the most advanced Direct3D™ acceleration solution and also delivers leadership VGA, 2D and Video performance, enabling a range of applications from 3D games through to DVD, Intercast™ and video conferencing.

3.1 BALANCED PC SYSTEM

The RIVA128ZX is designed to leverage existing PC system resources such as system memory, high bandwidth internal buses and bus master capabilities. The synergy between the RIVA128ZX graphics pipeline architecture and that of the current generation PCI and next generation AGP platforms, defines ground breaking performance levels at the cost point currently required for mainstream PC graphics solutions.

Execute versus DMA models

The RIVA128ZX is architected to optimize PC system resources in a manner consistent with the **AGP "Execute" model**. In this model texture map data for 3D applications is stored in system memory and individual texels are accessed as needed by the graphics pipeline. This is a significant enhancement over the DMA model where entire texture maps are transferred into off-screen framebuffer memory.

The advantages of the Execute versus the DMA model are:

- Improved system performance since only the required texels and not the entire texture map, cross the bus.
- Substantial cost savings since all the framebuffer is usable for the displayed screen and Z buffer and no part of it is required to be dedicated to texture storage or texture caching.
- There is no software overhead in the Direct3D driver to manage texture caching between application memory and the framebuffer.

To extend the advantages of the Execute model, the RIVA128ZX's proprietary texture cache and virtual DMA bus master design overcomes the bandwidth limitation of PCI, by sustaining a high texel throughput with minimum bus utilization. The host interface supports burst transactions up to 133MHz and provides over 400MBytes/s on AGP. AGP accesses offer other performance enhancements since they are from non-cacheable memory (no snoop) and can be low priority to prevent processor stalls, or high priority to prevent graphics engine stalls.

Building a balanced system

RIVA128ZX is architected to provide the level of 3D graphics performance and quality available in top arcade platforms. To provide comparable scene complexity in the 1997 time-frame, processors will have to achieve new levels of floating point performance. Profiles have shown that 1997 mainstream CPUs will be able to transform over 1 million lit, meshed triangles/s at 50% utilization using Direct3D. This represents an order of magnitude performance increase over anything attainable in 1996 PC games.

To build a balanced system the graphics pipeline must match the CPU's performance. It must be capable of rendering at least 1 million polygons/s in order to avoid CPU stalls. Factors affecting this system balance include:

- Direct3D compatibility. Minimizing the differences between the hardware interface and the Direct3D data structures.
- Triangle setup. Minimizing the number of format conversions and delta calculations done by the CPU.
- Display-list processing. Avoiding CPU stalls by allowing the graphics pipeline to execute independently of the CPU.
- Vertex caching. Avoids saturating the host interface with repeated vertices, lowering the traffic on the bus and reducing system memory collisions.
- Host interface performance.

3.2 HOST INTERFACE

The host interface boosts communication between the host CPU and the RIVA128ZX. The optimized interface performs burst DMA bus mastering for efficient and fast data transfer.

- 32-bit PCI version 2.1 or AGP version 1.0
- Burst DMA Master and target
- 33MHz PCI clock rate, 66MHz AGP clock rate and AGP 2X mode
- Supports over 100MBytes/s with 33MHz PCI to over 400MBytes/s on AGP 2X mode
- Implements read buffer posting on AGP

- Fully supports the “Execute” model on both PCI and AGP

3.3 2D ACCELERATION

The RIVA128ZX’s 2D rendering engine delivers industry-leading Windows acceleration performance:

- 100MHz 128-bit graphics engine optimized for single cycle operation into the 128-bit memory interface supporting up to 1.6GBytes/s
- Acceleration functions optimized for minimal software overhead on key GDI calls
- Extensive support for DirectDraw in Windows95 including optimized Direct Framebuffer (DFB) access with Write-combining
- Accelerated primitives including BLT, transparent BLT, stretchBLT, points, lins, lines, polylines, polygons, fills, patterns, arbitrary rectangular clipping and improved text rendering
- Pipeline optimized for multiple color depths including 8, 15, 24, and 30 bits per pixel
- DMA Pusher allows the 2D graphics pipeline to load rendering methods optimizing RIVA128ZX/host multi-tasking
- Execution of all 256 Raster Operations (as defined by Microsoft Windows) at 8, 15, 24 and 30-bit color depths
- 15-bit hardware color cursor
- Hardware color dithering
- Multi buffering (Double, Triple, Quad buffering) for smooth animation

3.4 3D ENGINE

Triangle setup engine

- Setup hardware optimized for Microsoft’s Direct3D API
- 5Gflop floating point geometry processor
- Slope and setup calculations
- Accepts IEEE Single Precision format used in Direct3D
- Efficient vertex caching

Rendering engine

The RIVA128ZX Multimedia Accelerator integrates an orthodox 3D rendering pipeline and triangle setup function which not only fully utilizes the capabilities of the Accelerated Graphics Port, but also supports advanced texture mapped 3D over the PCI bus. The RIVA128ZX 3D pipeline offers to Direct3D or similar APIs advanced triangle rendering capabilities:

- Rendering pipeline optimized for Microsoft’s **Direct3D** API
- Perspective correct true-color Gouraud lighting and texture mapping
- Full 32-bit RGBA texture filter and Gouraud lighting pixel data path
- Alpha blending for translucency and transparency
- Sub-pixel accurate texture mapping
- Internal pixel path: up to 24bits, alpha: up to 8 bits
- Texture magnification filtering with high quality bilinear filtering without performance degradation
- Texture minification filtering with MIP mapping without performance degradation
- LOD MIP-mapping: filter shape is dynamically adjusted based on surface orientation
- Texture sizes from 4 to 2048 texels in either U or V
- Textures can be looped and paged in real time for texture animation
- Perspective correct per-pixel fog for atmospheric effects
- Perspective correct specular highlights
- Multi buffering (Double, Triple, Quad buffering) for smooth 3D animation
- Multipass rendering for environmental mapping and advanced texturing

3.5 VIDEO PROCESSOR

The RIVA128ZX Palette-DAC pipeline accelerates full-motion video playback, sustaining 30 frames per second while retaining the highest quality color resolution, implementing true bilinear filtering for scaled video, and compensating for filtering losses using edge enhancement algorithms.

- Advanced support for DirectDraw (DirectVideo) in Windows 95
- Back-end hardware video scaling for video conferencing and playback
- Hardware color space conversion (YUV 4:2:2 and 4:2:0)
- Multi-tap X and Y filtering for superior image quality
- Optional edge enhancement to retain video sharpness

- Support for scaled field interframing for reduced motion artifacts and reduced storage
- Per-pixel color keying
- Multiple video windows with hardware color space conversion and filtering
- Planar YUV12 (4:2:0) to/from packed (4:2:2) conversion for software MPEG acceleration and H.261 video conferencing applications
- Accelerated playback of industry standard codecs including MPEG-1/2, Indeo, Cinepak

3.6 VIDEO PORT

The RIVA128ZX Multimedia Accelerator provides connectivity for video input devices such as Philips SAA7111A, ITT 3225 and Samsung KS0127 through an ITU-R-656 video input bus to DVD and MPEG2 decoders through bidirectional media port functionality.

- Supported through VPE extensions to DirectDraw
- Supports filtered down-scaling and decimation
- Supports real time video capture via Bus Mastering DMA
- Serial interface for decoder control

3.7 DIRECT RGB OUTPUT TO LOW COST PAL/NTSC ENCODER

The RIVA128ZX has also been designed to interface to a standard PAL or NTSC television via a low cost TV encoder chip. In PAL or NTSC display modes the interlaced output is internally flicker-filtered and CCIR/EIA compliant timing reference signals are generated.

3.8 SUPPORT FOR STANDARDS

- Multimedia support for MS-DOS, Windows 3.11, Windows 95, and Windows NT
- Acceleration for Windows 95 Direct APIs including Direct3D, DirectDraw and DirectVideo
- VGA and SVGA: The RIVA128ZX has an industry standard 32-bit VGA core and BIOS support. In PCI configuration space the VGA can be enabled and disabled independently of the GUI.
- Glue-less Accelerated Graphics Port (AGP 1.0) or PCI 2.1 bus interface
- ITU/CCIR-656 compatible video port
- VESA DDC2B+, DPMS, VBE 2.0 supported

3.9 RESOLUTIONS SUPPORTED

Resolution	BPP	2MByte	4MByte (128-bit)	8MByte (64-bit)	8MByte (128-bit)
640x480	4	120Hz	120Hz	120Hz	120Hz
	8	120Hz	120Hz	120Hz	120Hz
	16	120Hz	120Hz	120Hz	120Hz
	32	120Hz	120Hz	120Hz	120Hz
800x600	8	120Hz	120Hz	120Hz	120Hz
	16	120Hz	120Hz	120Hz	120Hz
	32	120Hz	120Hz	120Hz	120Hz
1024x768	8	120Hz	120Hz	120Hz	120Hz
	16	120Hz	120Hz	120Hz	120Hz
	32	-	120Hz	120Hz	120Hz
1152x864	8	120Hz	120Hz	120Hz	120Hz
	16	120Hz	120Hz	120Hz	120Hz
	32	-	100Hz	100Hz	100Hz
1280x1024	8	100Hz	100Hz	100Hz	100Hz
	16	-	100Hz	100Hz	100Hz
	32	-	-	t.b.d.	75Hz
1600x1200	8	85Hz	85Hz	85Hz	85Hz
	16	-	85Hz	85Hz	85Hz
	32	-	-	-	60Hz
1920x1080	8	-	85Hz	85Hz	85Hz
	16	-	-	85Hz	85Hz
1920x1200	8	-	75Hz	75Hz	75Hz
	16	-	-	75Hz	75Hz
1800x1440	16	-	-	60Hz	60Hz

3.10 CUSTOMER EVALUATION KIT

A Customer Evaluation Kit (CEK) is available for evaluating the RIVA128ZX. The CEK includes a PCI or AGP adapter card designed to support the RIVA128ZX feature set, an evaluation CD-ROM containing a fast-installation application, extensive device drivers and programs demonstrating the RIVA128ZX features and performance.

This CEK includes:

- RIVA128ZX evaluation board and CD-ROM
- QuickStart install/user guide
- OS drivers and files
  - Windows 3.11
  - Windows 95 Direct X/3D
  - Windows NT 3.5
  - Windows NT 4.0
- Demonstration files and Game demos
- Benchmark programs and files

3.11 TURNKEY MANUFACTURING PACKAGE

A Turnkey Manufacturing Package (TMP) is available to support OEM designs and development through to production. It delivers a complete manufacturable hardware and software solution that allows an OEM to rapidly design and bring to volume an RIVA128ZX-based product.

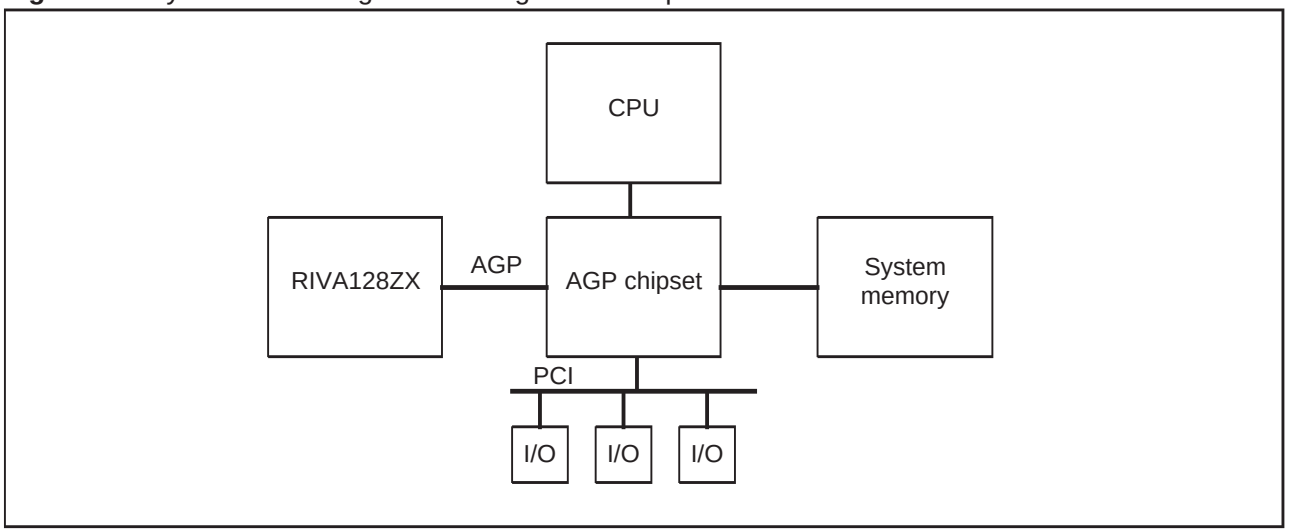
This TMP includes:

- CD-ROM
  - RIVA128ZX Datasheet and Application Notes
  - OrCAD™ schematic capture and PADS™ layout design information
  - Quick Start install/user guide/release notes
  - BIOS Modification program, BIOS binaries, utilities and BIOS Modification Guide documentation
  - Bring-up and OEM Production Diagnostics
  - Software and Utilities
- OS drivers and files
  - Windows 3.11
  - Windows 95 Direct X/3D
  - Windows NT 3.5
  - Windows NT 4.0
- Content developer and WWW information
- Partner solutions
- Access to our password-protected web site for upgrade files and release notes.

4 ACCELERATED GRAPHICS PORT (AGP) INTERFACE

The Accelerated Graphics Port (AGP) is a high performance, component level interconnect targeted at 3D graphical display applications and based on performance enhancements to the PCI local bus.

Figure 1. System block diagram showing relationship between AGP and PCI buses



Background to AGP

Although 3D graphics acceleration is becoming a standard feature of multimedia PC platforms, 3D rendering generally has a voracious appetite for memory bandwidth. Consequently there is upward pressure on the PC's memory requirement leading to higher bill of material costs. These trends will increase, requiring high speed access to larger amounts of memory. The primary motivation for AGP therefore was to contain these costs whilst enabling performance improvements.

By providing significant bandwidth improvement between the graphics accelerator and system memory, some of the 3D rendering data structures can be shifted into main memory, thus relieving the pressure to increase the cost of the local graphics memory.

Texture data are the first structures targeted for shifting to system memory for four reasons:

1 Textures are generally read only, and therefore do not have special access ordering or coherency problems.

2 Shifting textures balances the bandwidth load between system memory and local graphics memory, since a well cached host processor has much lower memory bandwidth requirements than a 3D rendering engine. Texture access comprises perhaps the largest single component of rendering memory bandwidth (compared with rendering, display and Z buffers), so avoiding loading or caching textures in graphics local memory saves not only this component of local memory bandwidth, but also the bandwidth necessary to load the texture store in the first place. Furthermore, this data must pass through main memory anyway as it is loaded from a mass store device.

3 Texture size is dependent upon application quality rather than on display resolution, and therefore subject to the greatest pressure for growth.

4 Texture data is not persistent; it resides in memory only for the duration of the application, so any system memory spent on texture storage can be returned to the free memory heap when the application finishes (unlike display buffers which remain in use).

Other data structures can be moved to main memory but the biggest gain results from moving texture data.

Relationship of AGP to PCI

AGP is a superset of the 66MHz PCI Specification (Revision 2.1) with performance enhancements optimized for high performance 3D graphics applications.

The PCI Specification is unmodified by AGP and 'reserved' PCI fields, encodings and pins, etc. are not used.

AGP does not replace the need for the PCI bus in the system and the two are physically, logically, and electrically independent. As shown in Figure 1

the AGP bridge chip and RIVA128ZX are the only devices on the AGP bus - all other I/O devices remain on the PCI bus.

The add-in slot defined for AGP uses a new connector body (for electrical signaling reasons) which is not compatible with the PCI connector; PCI and AGP boards are not mechanically interchangeable.

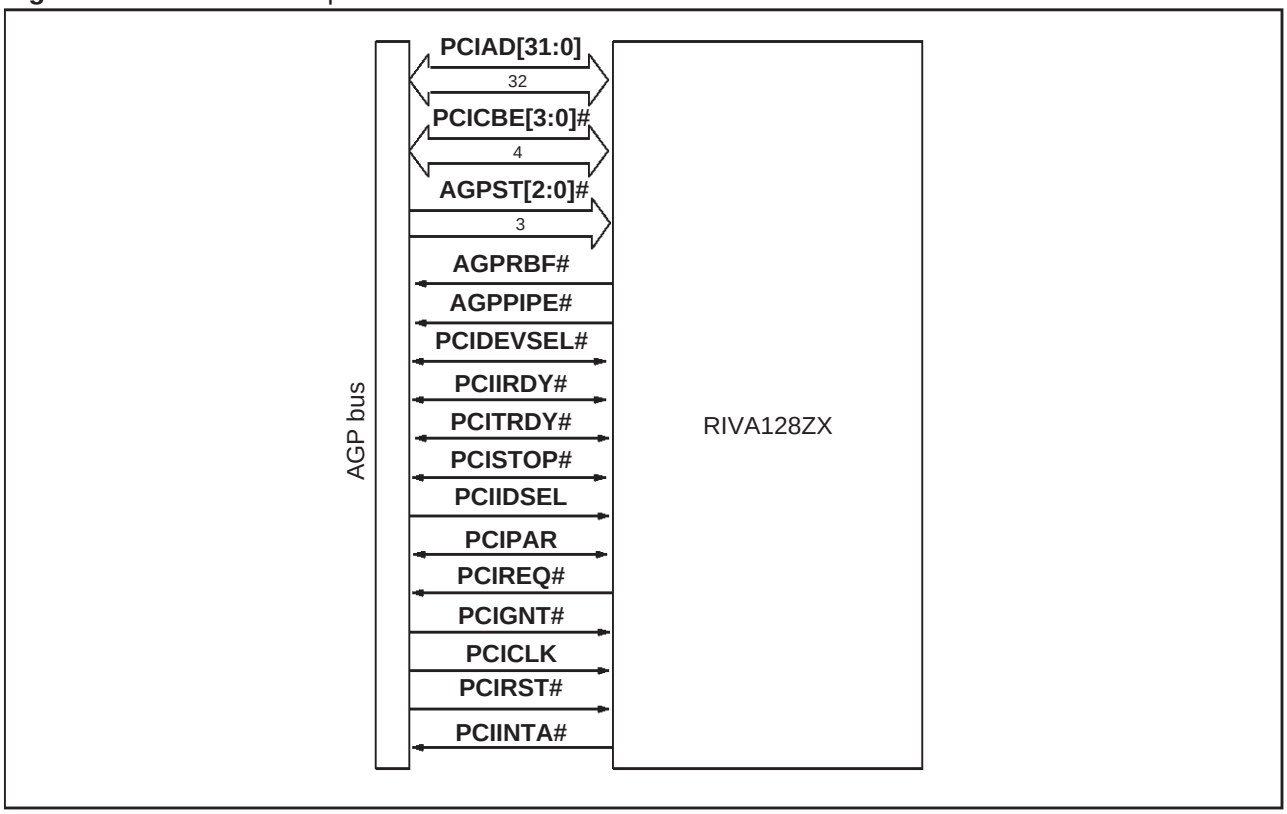
AGP accesses differ from PCI in that they are pipelined. This compares with serialized PCI transactions, where the address, wait and data phases need to complete before the next transaction starts. AGP transactions can only access system memory - not other PCI devices or CPU. Bus mastering accesses can be either PCI or AGP-style.

Full details of AGP are given in the *Accelerated Graphics Port Interface Specification* [3] published by Intel Corporation.

4.1 RIVA128ZX AGP INTERFACE

The RIVA128ZX glueless interface to AGP 1.0 is shown in Figure 2.

Figure 2. AGP interface pin connections



4.2 AGP BUS TRANSACTIONS

AGP bus commands supported

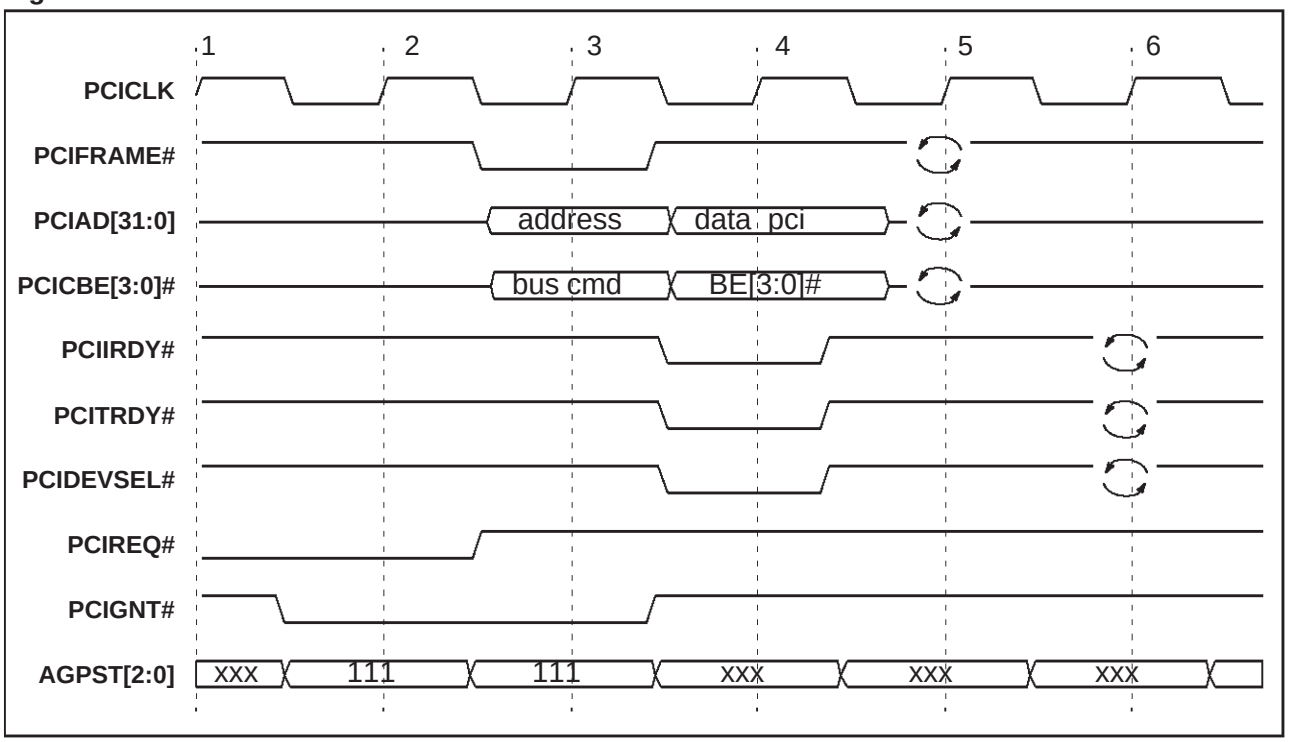
The following AGP bus commands are supported by the RIVA128ZX:

- Read
- Read (hi-priority)

PCI transactions on the AGP bus

PCI transactions can be interleaved with AGP transactions including between pipelined AGP data transfers. A basic PCI transaction on the AGP interface is shown in Figure 3. If the PCI target is a non AGP compliant master, it will not see **AGPST[2:0]** and the transaction appears to be on a PCI bus. For AGP aware bus masters, **AGPST[2:0]** indicate that permission to use the interface has been granted to initiate a request and not to move AGP data.

Figure 3. Basic PCI transaction on AGP



An example of a PCI transaction occurring between an AGP command cycle and return of data is shown in Figure 4. This shows the smallest number of cycles during which an AGP request can be enqueued, a PCI transaction performed and AGP read data returned.

Figure 4. PCI transaction occurring between AGP request and data

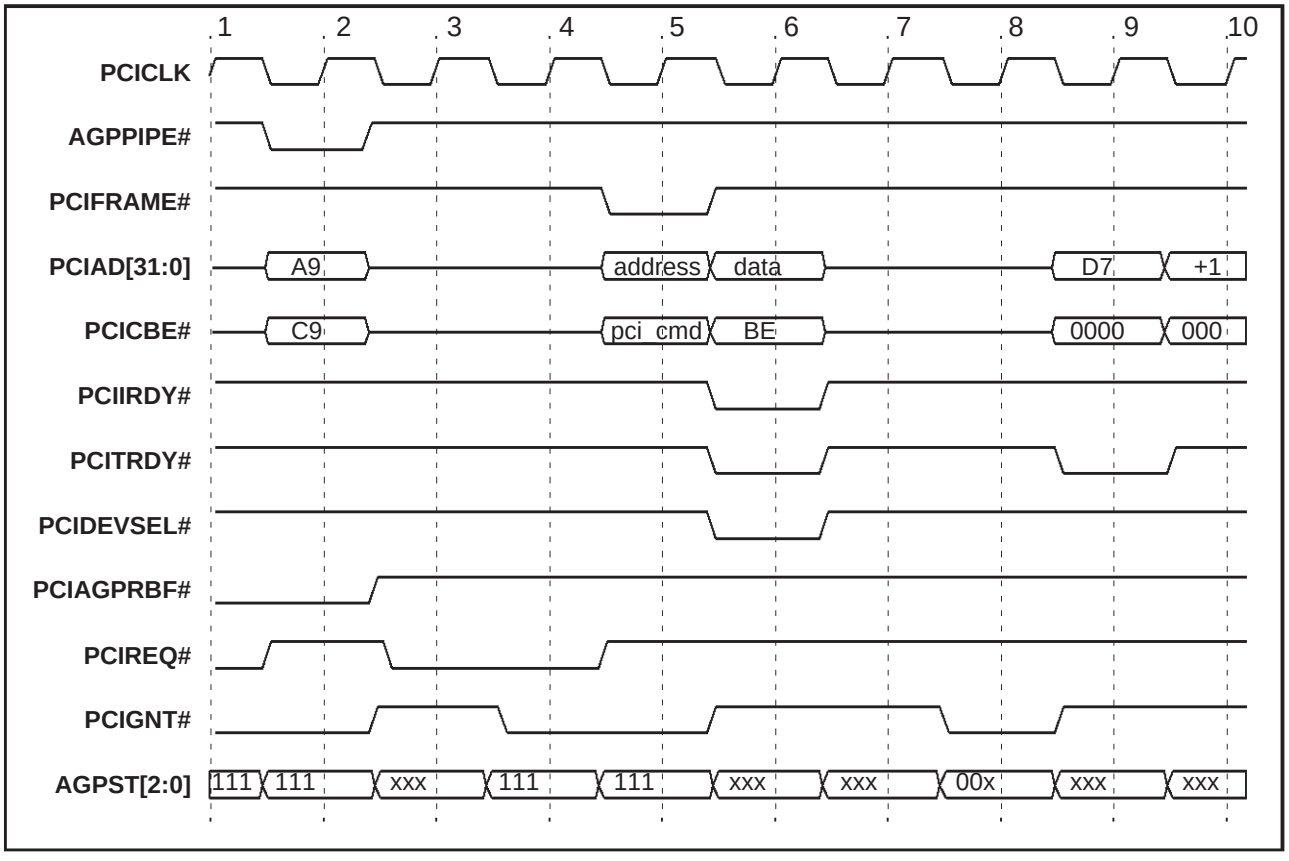
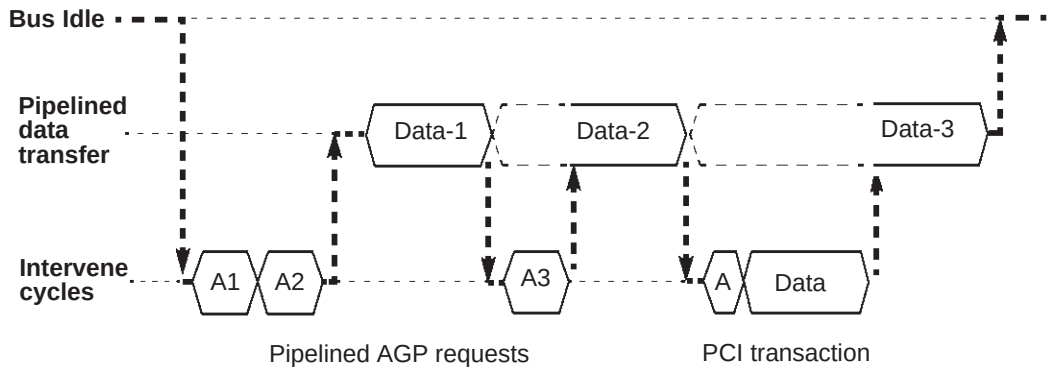


Figure 5. Basic AGP pipeline concept



Pipeline operation

Memory access pipelining provides the main performance enhancement of AGP over PCI. AGP pipelined bus transactions share most of the PCI signal set, and are interleaved with PCI transactions on the bus.

The RIVA128ZX supports AGP pipelined reads with a 4-deep queue of outstanding read requests. Pipelined reads are primarily used by the RIVA128ZX for cache filling, the cache size being optimized for AGP bursts. Depending on the AGP bridge, a bandwidth of up to 248MByte/s is achievable for 128-byte pipelined reads. This compares with around 100MByte/s for 128-byte 33MHz PCI reads. Another feature of AGP is that for smaller sized reads the bandwidth is not significantly reduced. Whereas 16-byte reads on PCI transfer at around 33MByte/s, on AGP around 175MByte/s is achievable. The RIVA128ZX actually requests reads greater than 64 bytes in multiples of 32-byte transactions.

The pipe depth can be maintained by the AGP bus master (RIVA128ZX) intervening in a pipelined transfer to insert new requests between data replies. This bus sequencing is illustrated in Figure 5.

When the bus is in an idle condition, the pipe can be started by inserting one or more AGP access requests consecutively. Once the data reply to those accesses starts, that stream can be broken (or intervened) by the bus master (RIVA128ZX) inserting one or more additional AGP access requests or inserting a PCI transaction. This intervention is accomplished with the bus ownership signals, **PCIREQ#** and **PCIGNT#**.

The RIVA128ZX implements both high and low priority reads depending of the status of the rendering engine. If the pipeline is likely to stall due to system memory read latency, a high priority read request is posted.

Address Transactions

The RIVA128ZX requests permission from the bridge to use **PCIAD[31:0]** to initiate either an AGP request or a PCI transaction by asserting **PCIREQ#**. The arbiter grants permission by asserting **PCIGNT#** with **AGPST[2:0]** equal to ”111” (referred to as START). When the RIVA128ZX receives START it must start the bus operation within two clocks of the bus becoming available. For example, when the bus is in an idle condition when START is received, the RIVA128ZX must initiate the bus transaction on the next clock and the one following.

Figure 6 shows a single address being enqueued by the RIVA128ZX. Sometime before clock 1, the RIVA128ZX asserts **PCIREQ#** to gain permission to use **PCIAD[31:0]**. The arbiter grants permission by indicating START on clock 2. A new request (address, command and length) are enqueued on each clock in which **AGPPIPE#** is asserted. The address of the request to be enqueued is presented on **PCIAD[31:3]**, the length on **PCIAD[2:0]** and the command on **PCICBE[3:0]#**. In Figure 6 only a single address is enqueued since **AGPPIPE#** is just asserted for a single clock. The RIVA128ZX indicates that the current address is the last it intends to enqueue when **AGPPIPE#** is asserted and **PCIREQ#** is deasserted (occurring on clock 3). Once the arbiter detects the assertion of **AGPPIPE#** or **PCIFRAME#** it deasserts **PCIGNT#** on clock 4.

Figure 6. Single address - no delay by master

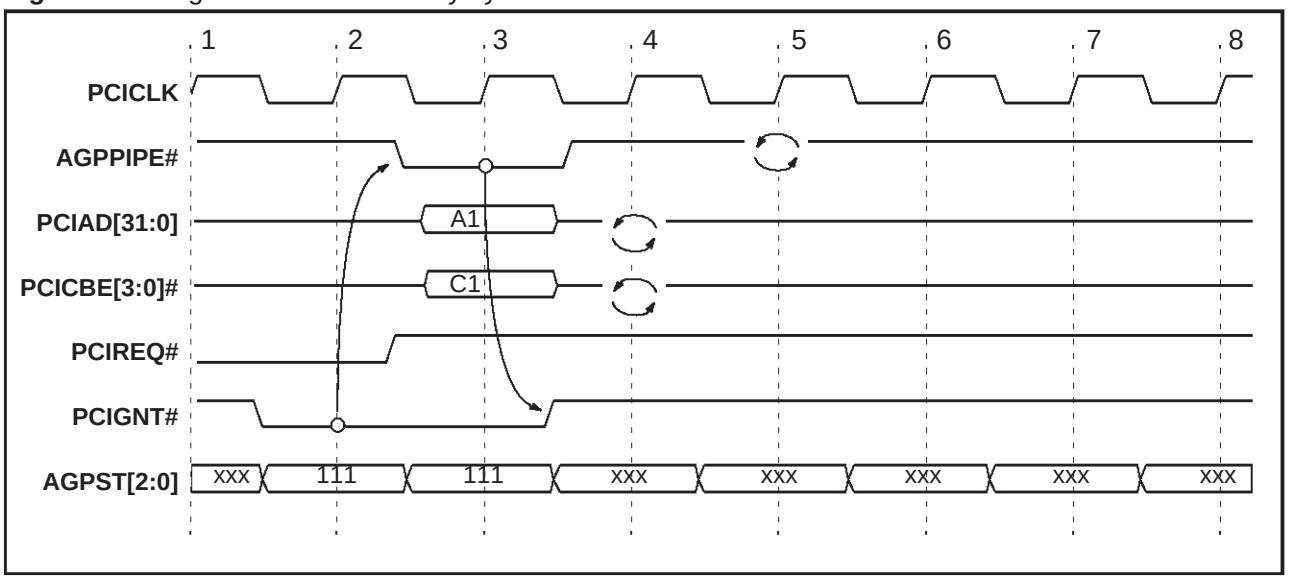
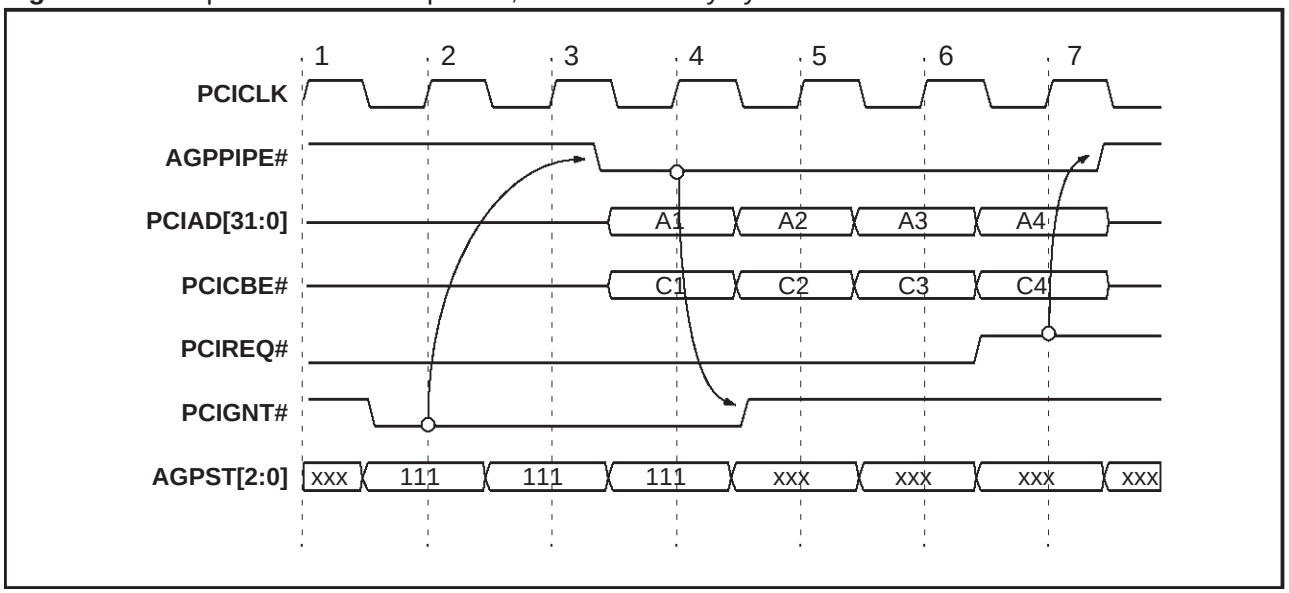


Figure 7 shows the RIVA128ZX enqueuing 4 requests, where the first request is delayed by the maximum 2 cycles allowed. START is indicated on clock 2, but the RIVA128ZX does not assert **AGPPIPE#** until clock 4. Note that **PCIREQ#** remains asserted on clock 6 to indicate that the current request is not the last one. When **PCIREQ#** is deasserted on clock 7 with **AGPPIPE#** still asserted this indicates that the current address is the last one to be enqueued during this transaction. **AGPPIPE#** must be deasserted on the next clock when **PCIREQ#** is sampled as deasserted. If the RIVA128ZX wants to enqueue more requests during this bus operation, it continues asserting **AGPPIPE#** until all of its requests are enqueued or until it has filled all the available request slots provided by the target.

Figure 7. Multiple addresses enqueued, maximum delay by RIVA128ZX



2X Data Transfers

2X data transfers are similar to 1X transfers except that an entire 8 bytes are transferred during a single **PCICLK** period. This requires that two 4 byte pieces of data are transferred across **PCIAD[31:0]** for each CLK period. A read data transfer is described followed by a write transfer.

Figure 8. 2X Read data, no delay

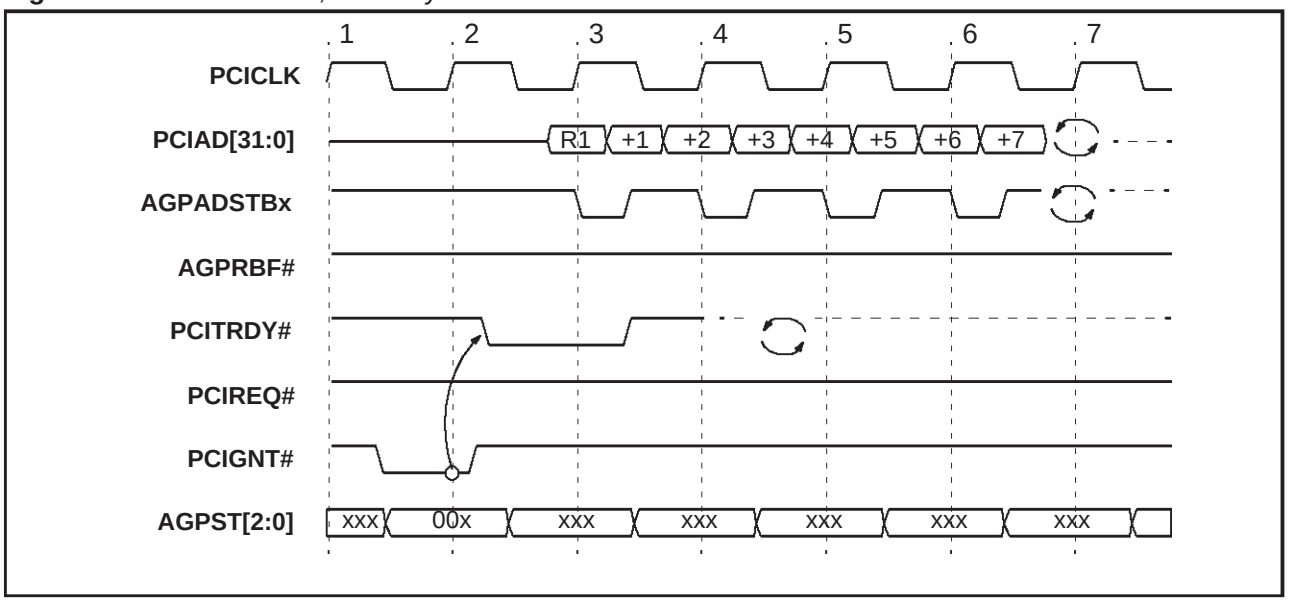


Figure 8 shows 32 bytes being transferred during 4 clocks (compared with 16 bytes in AGP 1x mode). The control signals are identical. The **AGPAD_STBx** signal has been added when data is transferred at 8 bytes per **PCICLK** period. **AGPAD_STBx** represents **AGPAD_STB0** and **AGPAD_STB1** and are used by the 2X interface logic to indicate when valid data is present on the AD bus. The control logic (**PCITRDY#** in this case) indicates when data can be used by the target.

Figure 9. 2X Back to back read data, no delay

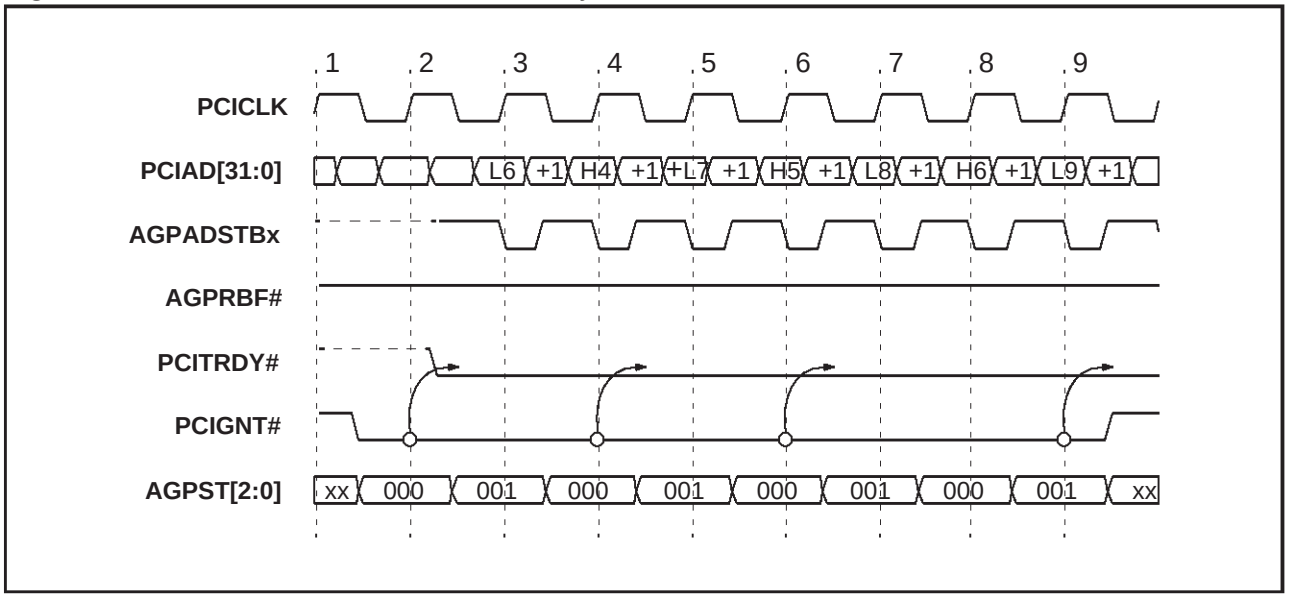


Figure 9 shows back to back 8 byte read transactions. **AGPST[2:0]** are shown toggling between “000”and “001” to illustrate that they are actually changing. However, they are not required to change between high and low priority to do back to back transactions. In this diagram, **PCITRDY#** is asserted on each clock since a new transaction starts on each clock.

Figure 10. 2X Basic write no delay

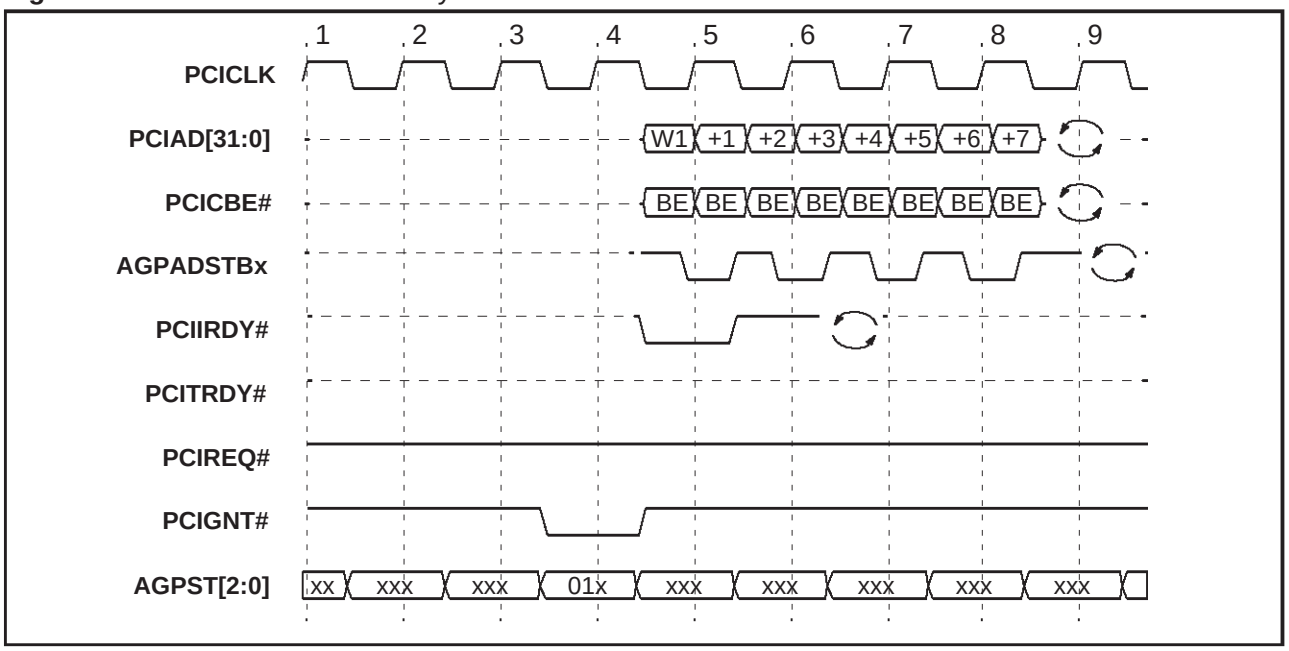


Figure 10 is a basic write transaction that transfers data at the 2X rate. There is no difference in the control signals from AGP 1x mode - only more data is moved. The normal control signals determine when data is valid.

Figure 11. QuadWord writes back to back - no delays

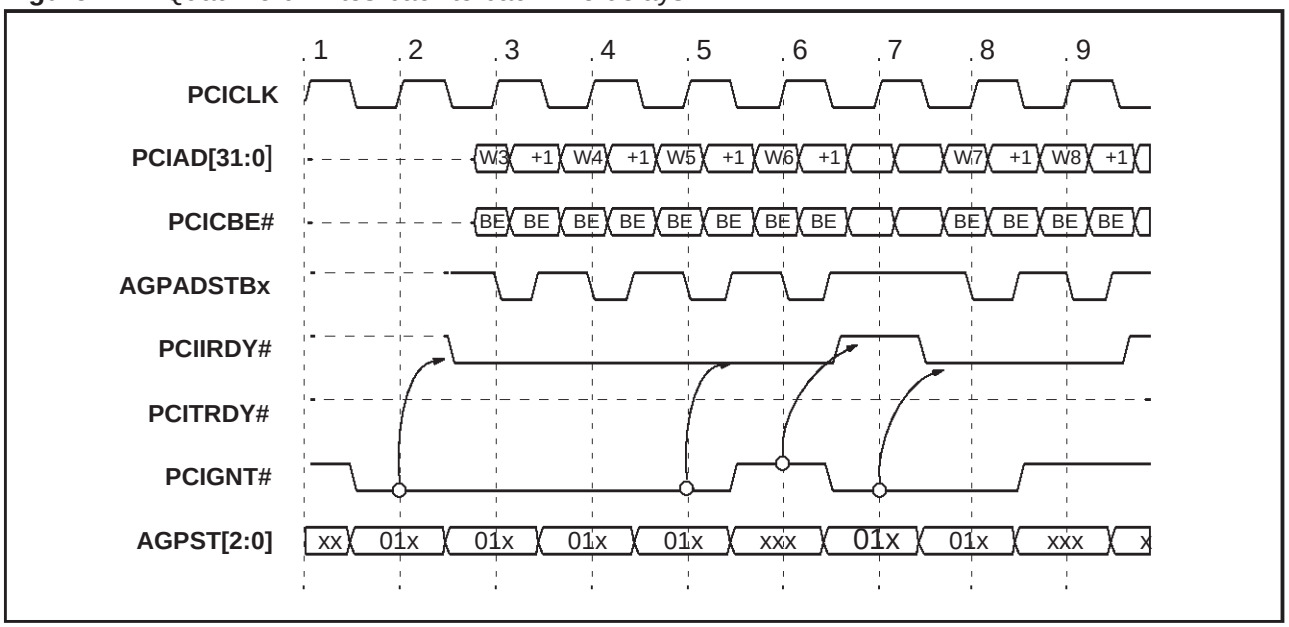


Figure 11 illustrates multiple 8 byte write operations compared with the single transfer shown in Figure 10. When the transactions are short, the arbiter is required to give grants on every clock or the AD bus will not be totally utilized. In this example a new write is started on each rising clock edge except clock 7, because the arbiter deasserted **PCIGNT#** on clock 6. Since a new transaction is started on each CLK, **PCIIRDY#** is only deasserted on clock 7.

AGP timing specification

Figure 12. AGP clock specification

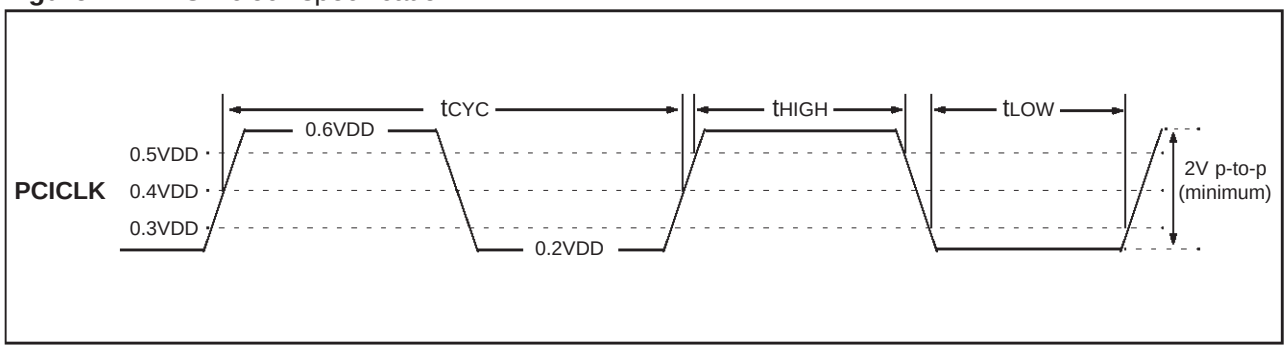


Table 1. AGP clock timing parameters

Symbol	Parameter	Min.	Max.	Unit	Notes
tCYC	**PCICLK** period	15	30	ns	
tHIGH	**PCICLK** high time	6		ns	
tLOW	**PCICLK** low time	6		ns	
	PCICLK slew rate	1.5	4	V/ns	1

NOTES

1 This rise and fall time is measured across the minimum peak-to-peak range as shown in Figure 12.

Figure 13. AGP timing diagram

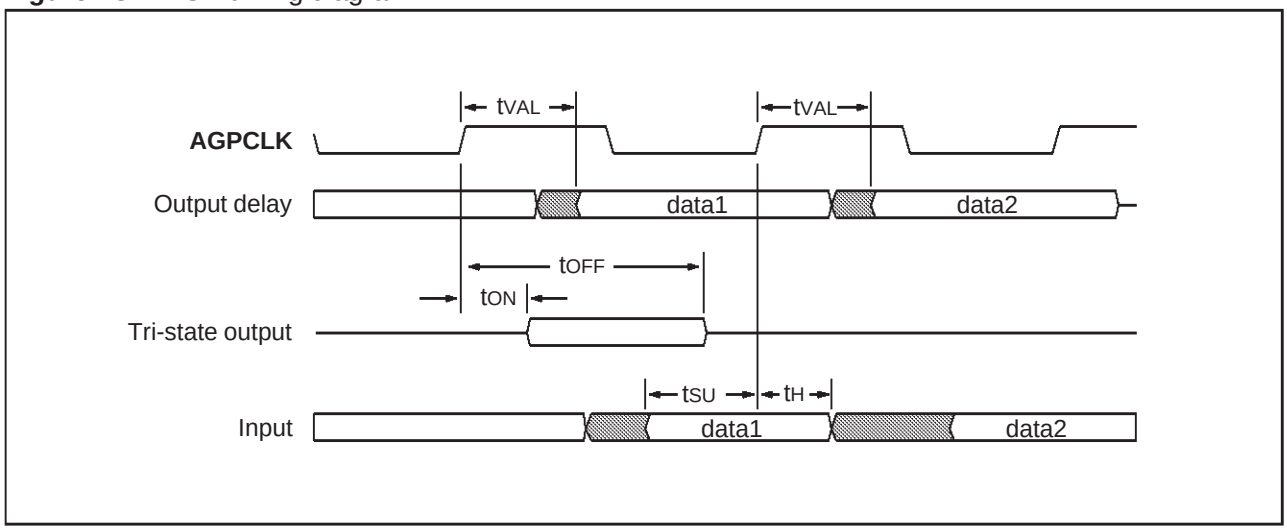


Table 2. AGP timing parameters

Symbol	Parameter	Min.	Max.	Unit	Notes
tVAL	**AGPCLK** to signal valid delay (data and control signals)	2	11	ns	
tON	Float to active delay	2		ns	
tOFF	Active to float delay		28	ns	
tSU	Input set up time to **AGPCLK** (data and control signals)	7		ns	
tH	Input hold time from **AGPCLK**	0		ns	

Figure 14. AGP timing diagram (2X data transfer mode)

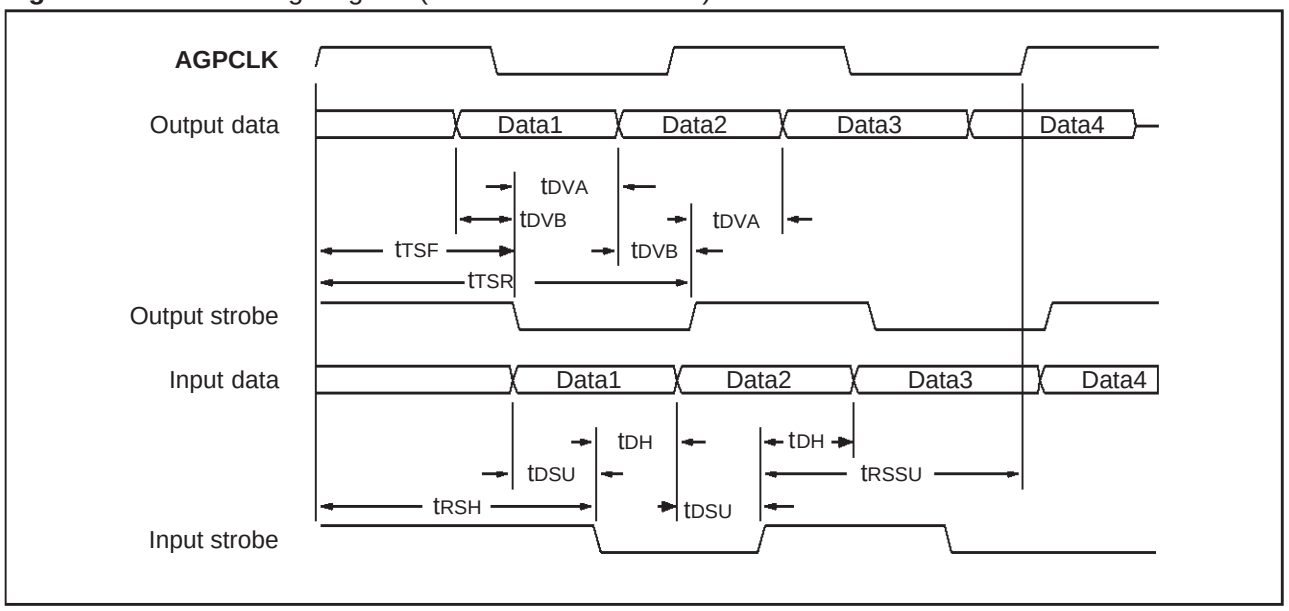


Table 3. AGP timing parameters (2X data transfer mode)

Symbol	Parameter	Min.	Max.	Unit	Notes
tTSF	**AGPCLK** to transmit strobe falling edge	2	12	ns	
tTSR	**AGPCLK** to transmit strobe rising edge		20	ns	
tDVB	Output data valid before strobe	1.7		ns	
tDVA	Output data valid after strobe	1.7		ns	
tRSSU	Receiver strobe setup time to **AGPCLK**	6		ns	
tRSH	Receiver strobe hold time from **AGPCLK**	1		ns	
tDSU	Input data to strobe setup time	1		ns	
tDH	Input data to strobe hold time	1		ns	

Figure 15. AGP Strobe/Data turnaround timing diagram (2X data transfer mode)

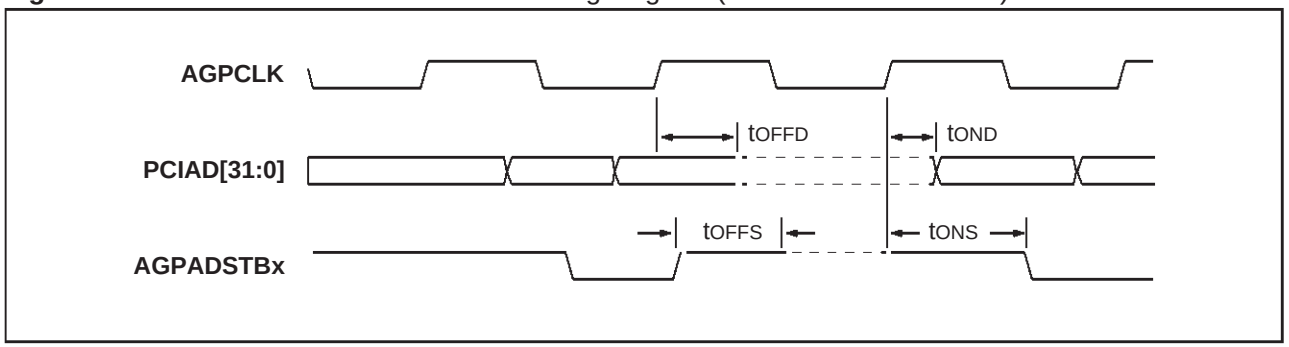


Table 4. AGP Strobe/Data turnaround timing parameters (2X data transfer mode)

Symbol	Parameter	Min.	Max.	Unit	Notes
tOND	Float to active delay	-1	9	ns	
tOFFD	Active to float delay	1	12	ns	
tONS	Strobe active to strobe falling edge setup	6	10	ns	
tOS	Strobe rising edge to strobe float delay	6	10	ns	

5 PCI 2.1 LOCAL BUS INTERFACE

5.1 RIVA128ZX PCI INTERFACE

The RIVA128ZX supports a glueless interface to PCI 2.1 with both master and slave capabilities. The host interface is fully compliant with the 32-bit PCI 2.1 specification.

The Multimedia Accelerator supports PCI bus operation up to 33MHz with zero-wait state capability and full bus mastering capability handling burst reads and burst writes.

Figure 16. PCI interface pin connections

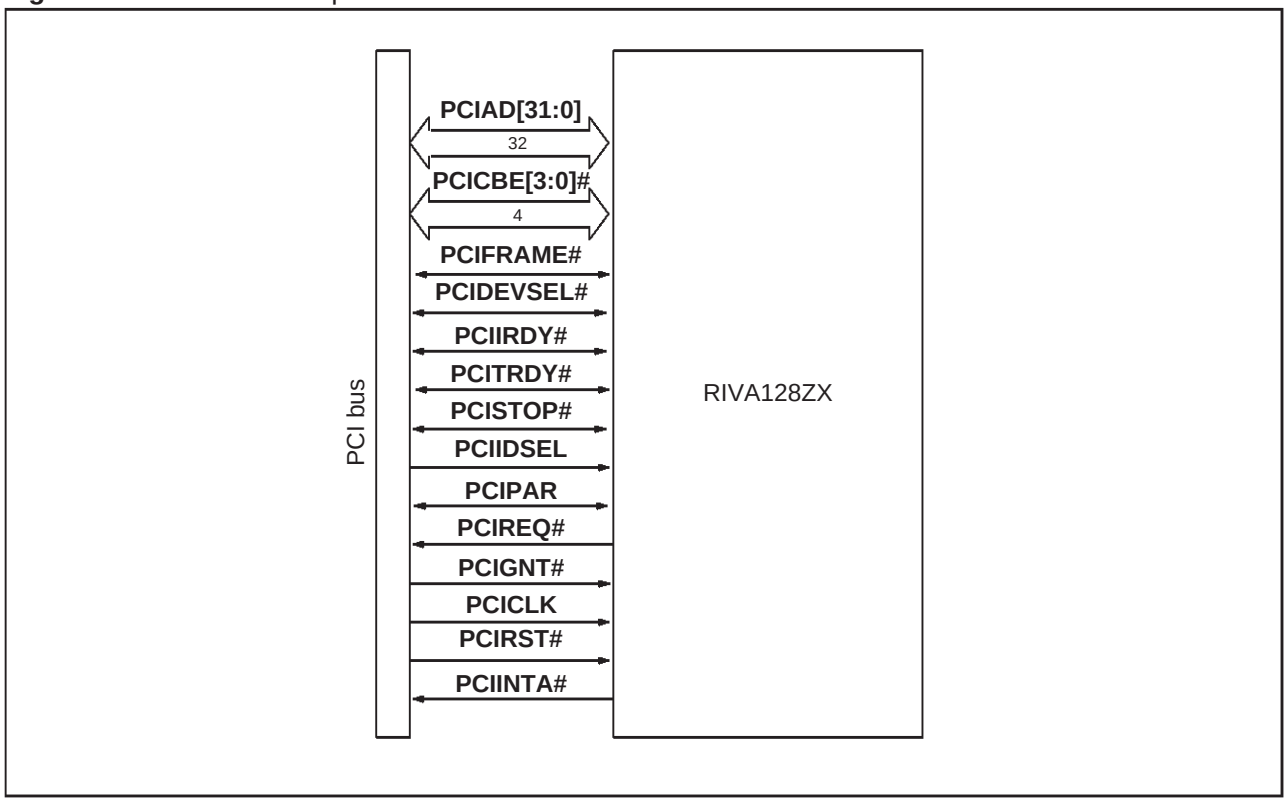


Table 5. PCI bus commands supported by the RIVA128ZX

Bus master	Bus slave
Memory read and write	Memory read and write
Memory read line	I/O read and write
Memory read multiple	Configuration read and write
	Memory read line
	Memory read multiple
	Memory write invalidate

5.2 PCI TIMING SPECIFICATION

The timing specification of the PCI interface takes the form of generic setup, hold and delay times of transitions to and from the rising edge of **PCICLK** as shown in Figure 17.

Figure 17. PCI timing parameters

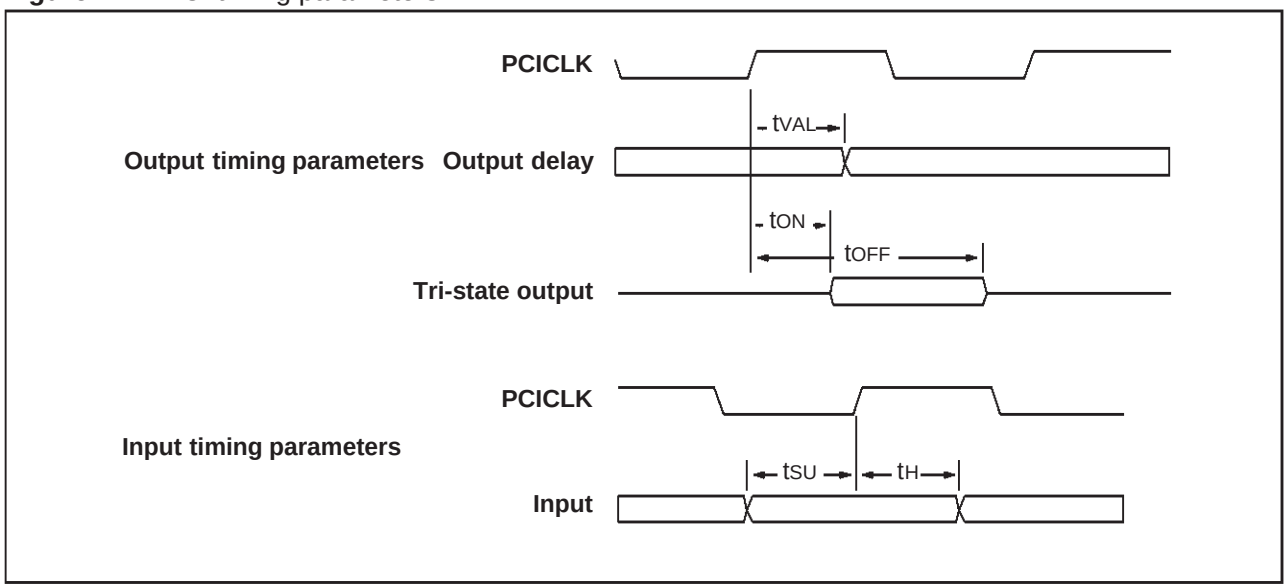


Table 6. PCI timing parameters

Symbol	Parameter	Min.	Max.	Unit	Notes
t_{VAL}	**PCICLK** to signal valid delay (bussed signals)	2	11	ns	1
$t_{VAL(PTP)}$	**PCICLK** to signal valid delay (point to point)	2	12	ns	1
t_{ON}	Float to active delay	2		ns	
t_{OFF}	Active to float delay		28	ns	
t_{SU}	Input set up time to **PCICLK** (bussed signals)	7		ns	1
$t_{SU(PTP)}$	Input set up time to **PCICLK** (**PCIGNT#**)	10		ns	1
$t_{SU(PTP)}$	Input set up time to **PCICLK** (**PCIREQ#**)	12		ns	
t_{H}	Input hold time from **PCICLK**	0		ns	

NOTE

1 **PCIREQ#** and **PCIGNT#** are point to point signals and have different valid delay and input setup times than bussed signals. All other signals are bussed.

Figure 18. PCI Target write - Slave Write (single 32-bit with 1-cycle **DEVSEL#** response)

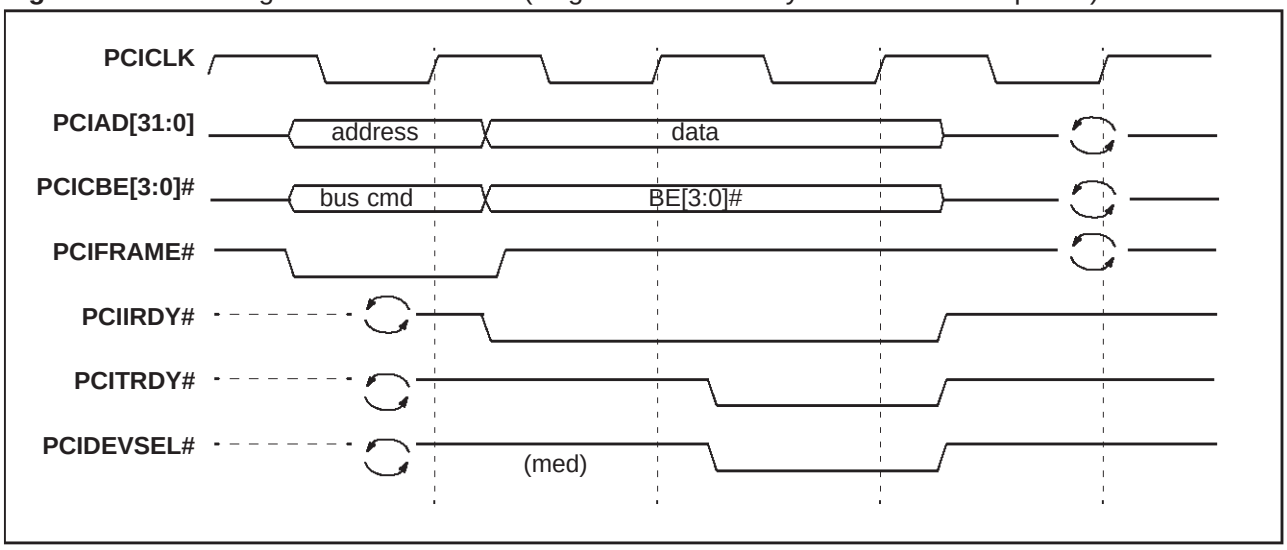


Figure 19. PCI Target write - Slave Write (multiple 32-bit with zero wait state **DEVSEL#** response)

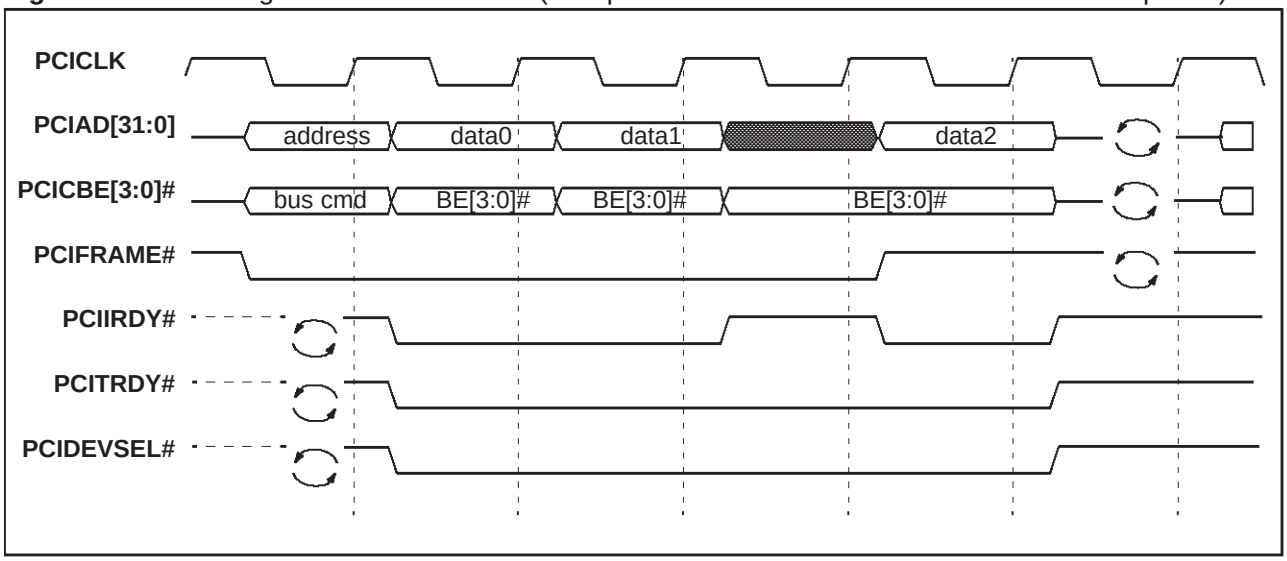


Figure 20. PCI Target read - Slave Read (1-cycle single word read)

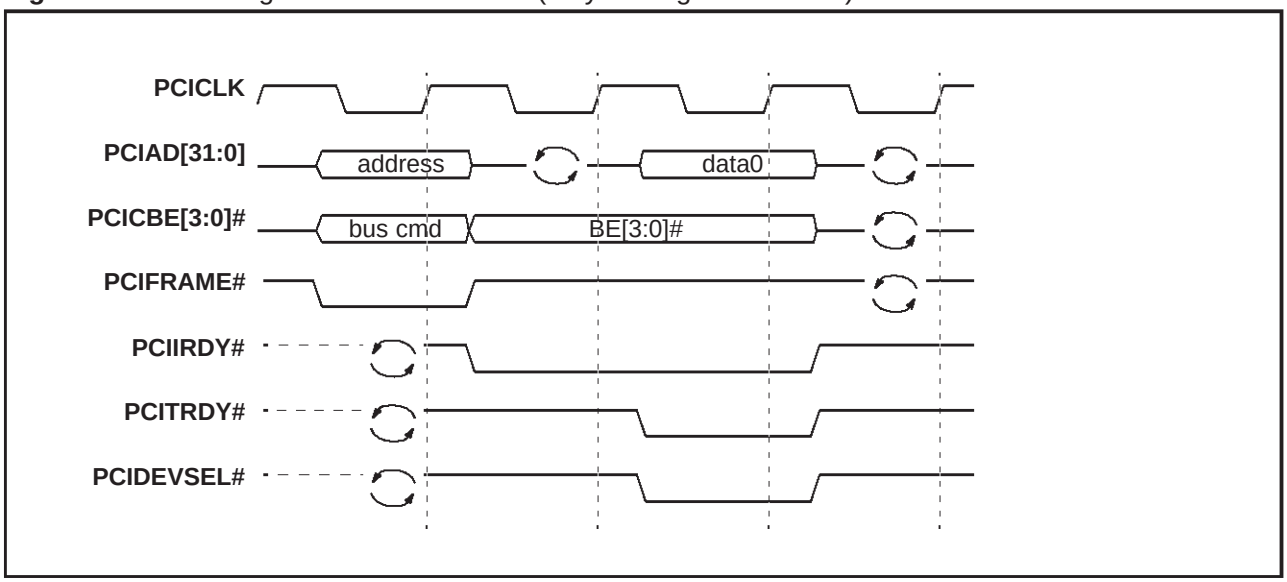


Figure 21. PCI Target read - Slave Read (slow single word read)

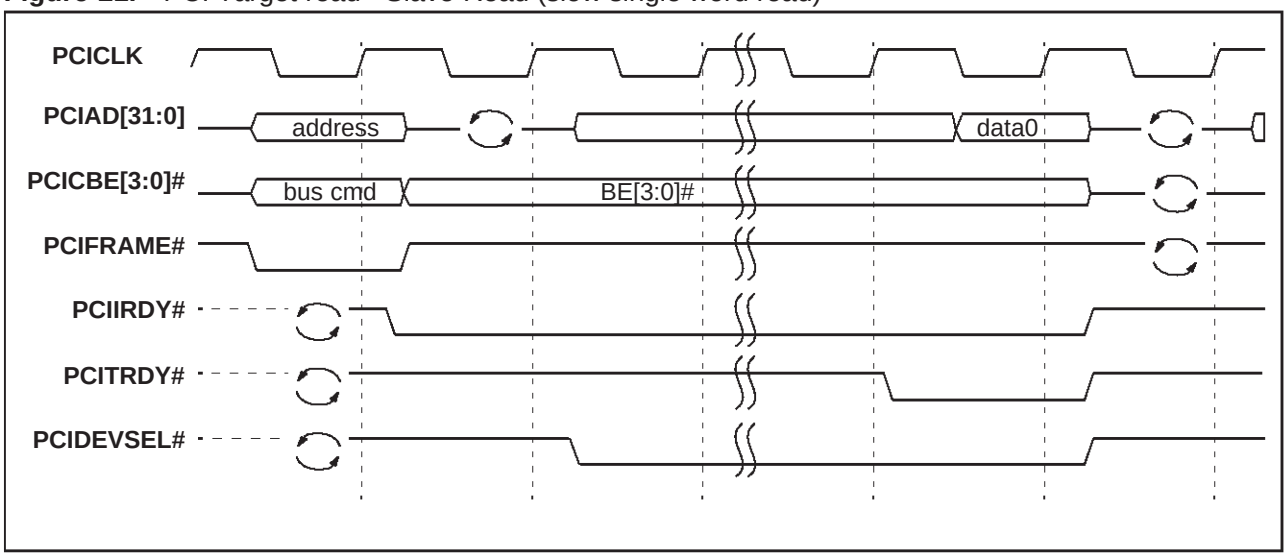


Figure 22. PCI Master write - multiple word

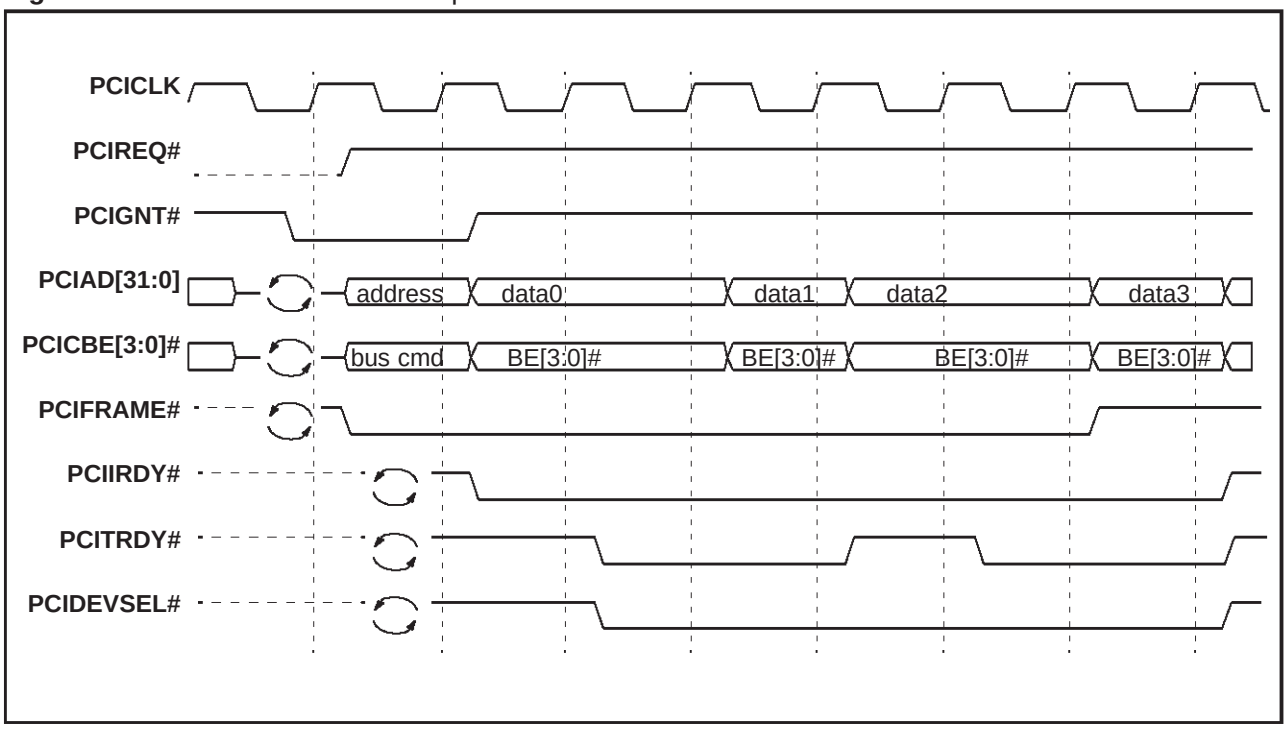
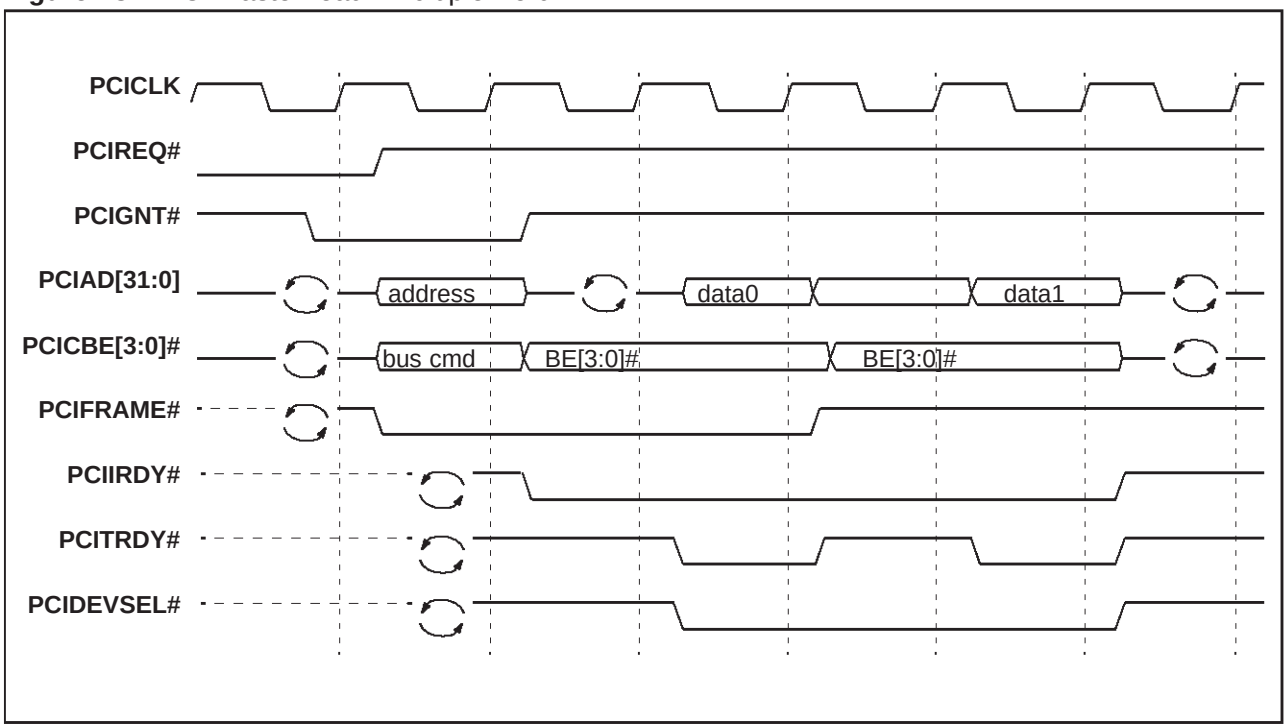


Figure 23. PCI Master read - multiple word



Note: The RIVA128ZX does not generate fast back to back cycles as a bus master

Figure 24. PCI Target configuration cycle - Slave Configuration Write

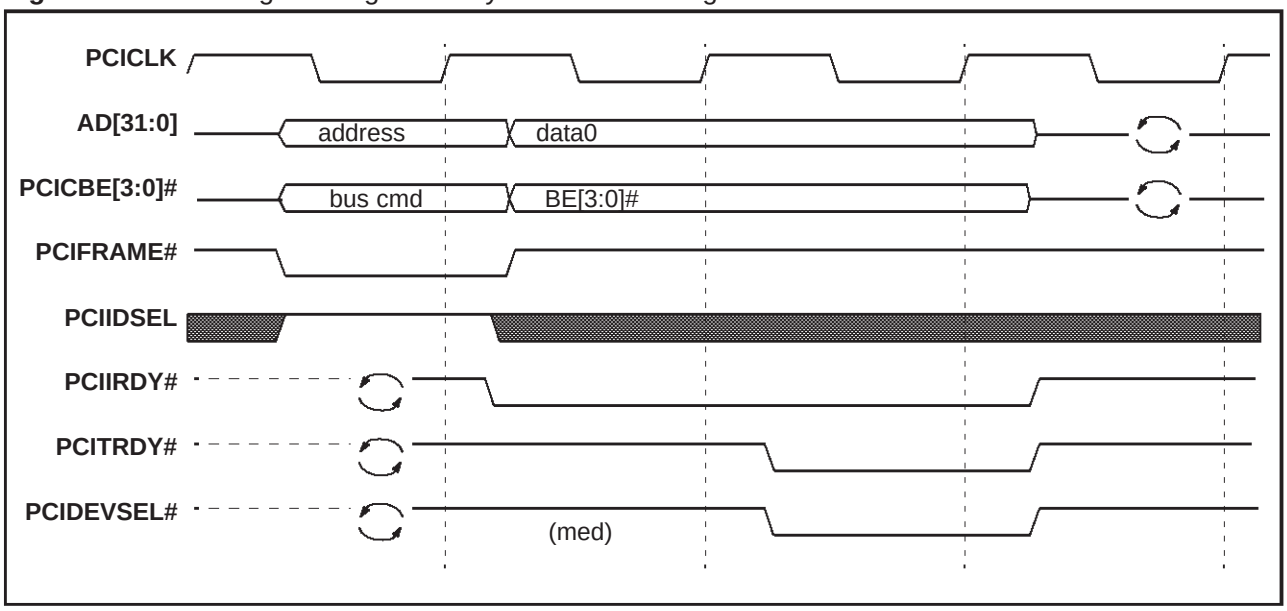


Figure 25. PCI Target configuration cycle - Slave Configuration Read

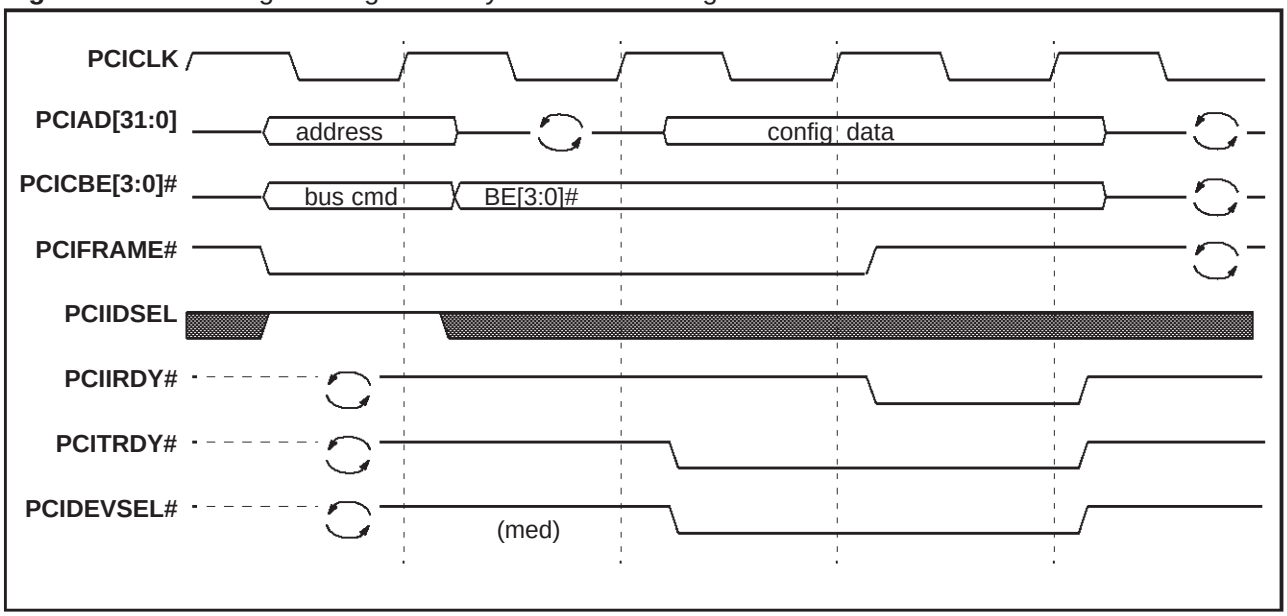


Figure 26. PCI basic arbitration cycle

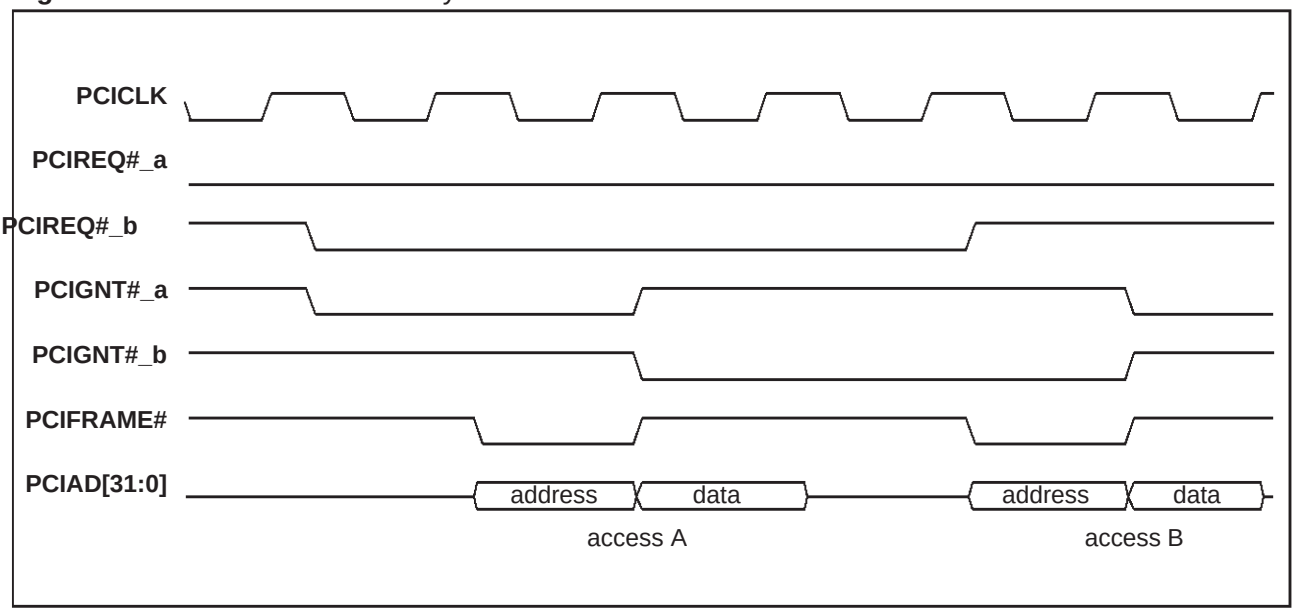
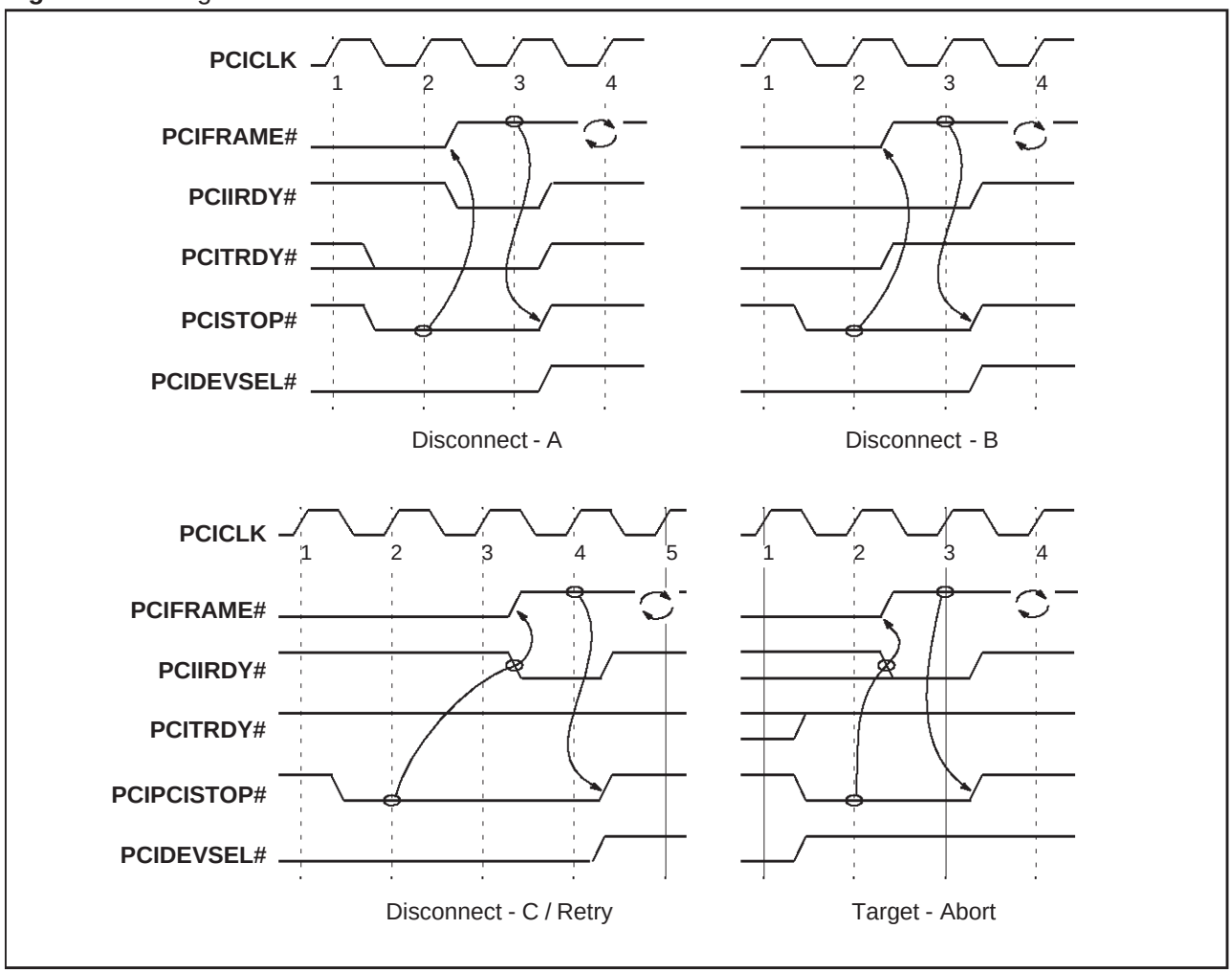


Figure 27. Target initiated termination



6 FRAMEBUFFER INTERFACE

The RIVA128ZX framebuffer interface supports SDRAM and SGRAM memory. Using SDRAM it can be configured with an 8MByte 64-bit data bus. With SGRAM it can be configured with a 2 or 4MByte 64-bit data bus or a 4 or 8MByte 128-bit data bus. The memory configurations supported by RIVA128ZX are shown in Table 7. All of the framebuffer signalling environment is 3.3V.

Table 7. RIVA128ZX memory configurations

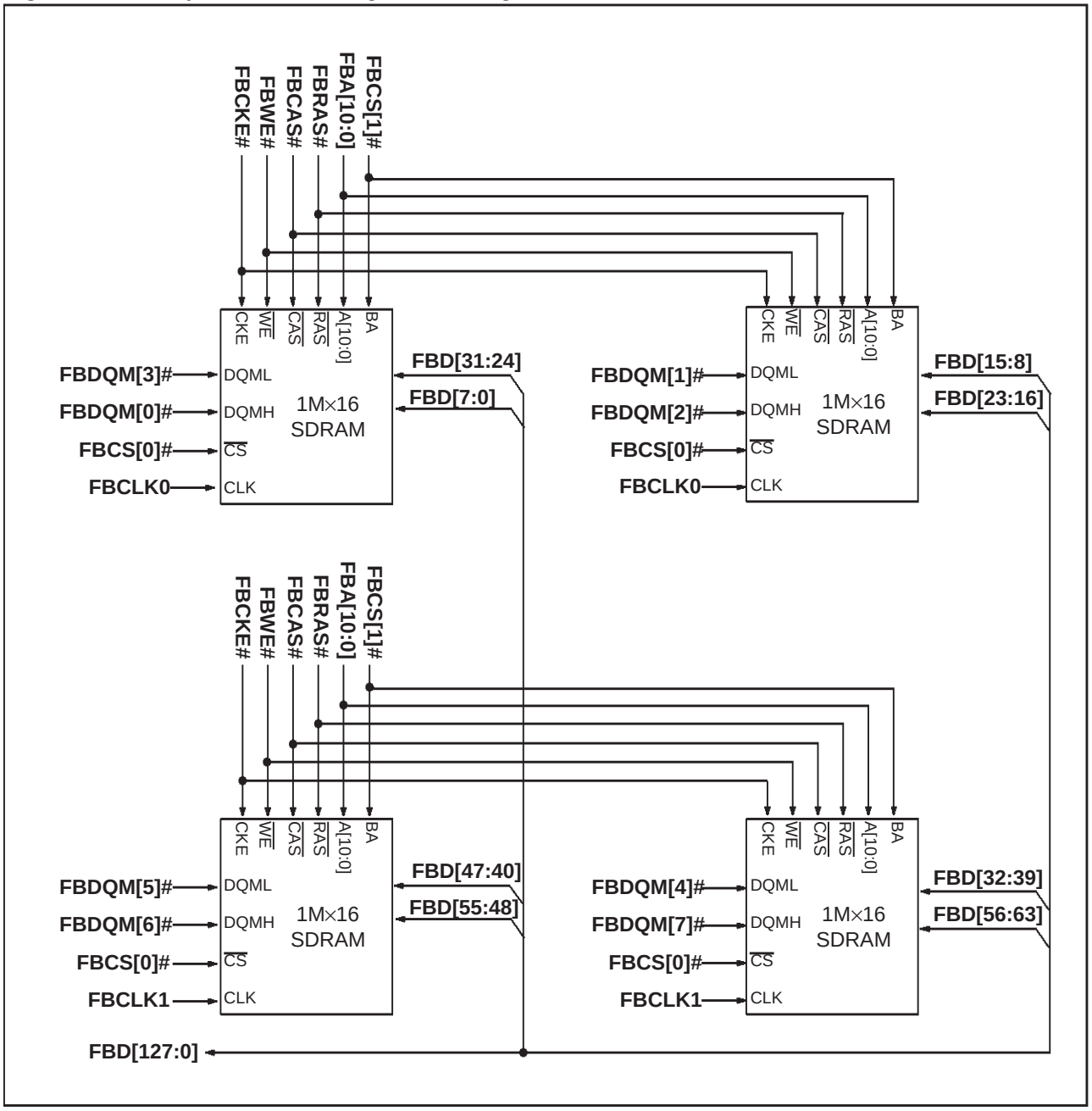
	8Mbit 2 internal bank SGRAM	**16Mbit 2 internal bank SGRAM**	**16Mbit 4 internal bank SGRAM**	**16Mbit 1M x 16 SDRAM**
2MByte	2 devices 64-bit	N/A	N/A	N/A
4MByte	4 devices 128-bit	2 devices 64-bit	2 devices 64-bit	N/A
8MByte	2 banks of 4 devices 128-bit	4 devices 128-bit	4 devices 128-bit	4 devices 64-bit

6.1 SDRAM INTERFACE

Two extra address lines are required to support 8MByte SDRAM compared with those needed for 4MByte SGRAM on RIVA 128. These are the A10 signal which was defined on the RIVA 128 pinout for future expansion and the SDRAM's internal bank select address bit (BA signal). To provide this extra signal the RIVA128ZX **FBCS[1]#** pin is internally redefined to be the SDRAM BA/A11 signal.

Since the RIVA128ZX supports a maximum addressable memory of 8MBytes, SDRAM support is only allowed with a 64-bit data bus. Figure 28 shows an example SDRAM memory configuration for RIVA128ZX. Note this figure attempts to scramble the bytes and data bits within bytes to simplify board layout, but this will depend on how the board components are placed in the layout.

Figure 28. 8MByte SDRAM configuration using 16Mbit devices



6.2 SGRAM INTERFACE

Signal changes between RIVA 128 and RIVA128ZX

The extra address signal (**FBA[10]**) required to address 16Mbit SGRAM devices was defined on RIVA 128 and was connected to pin 30 of the SGRAM in the RIVA 128 Reference Design schematics. This pin is a N/C on 8Mbit memory devices and it was also N/C on RIVA 128. For 8MByte designs using 8Mbit devices; **FBCS0#** drives the chip selects for the first external bank of memory and **FBCS1#** drives the second external bank.

The ROM BIOS implements code which automatically detects the configuration and memory type. If a mixture of 4-internal bank and 2-internal bank 16Mbit devices are used (e.g. 2 soldered down and 2 added by an end user as an SO-DIMM) then RIVA128ZX will program those devices and operate itself as 2-internal bank. There is no support for mixed 16Mbit and 8Mbit memories (i.e. there is no 6MByte mode) nor is there support for 16MByte using eight 16Mbit devices. The upgrade path from two devices to four devices has also been changed to better accommodate board layout for SO-DIMMs. As shown in Figure 29, the first two memories installed are on the left side of the chip and the upgrade is on the right hand side. This is different to RIVA 128 which populated the top two chips first and then populated the lower (far left and far right) memories as the upgrade. Both forms of 64-bit bus are supported in RIVA128ZX and the data paths populated are determined by the BIOS during its boot memory detection sequence.

Figure 29. Upgrade from 64-bit 4MByte to 128-bit 8MByte SGRAM via SO-DIMM

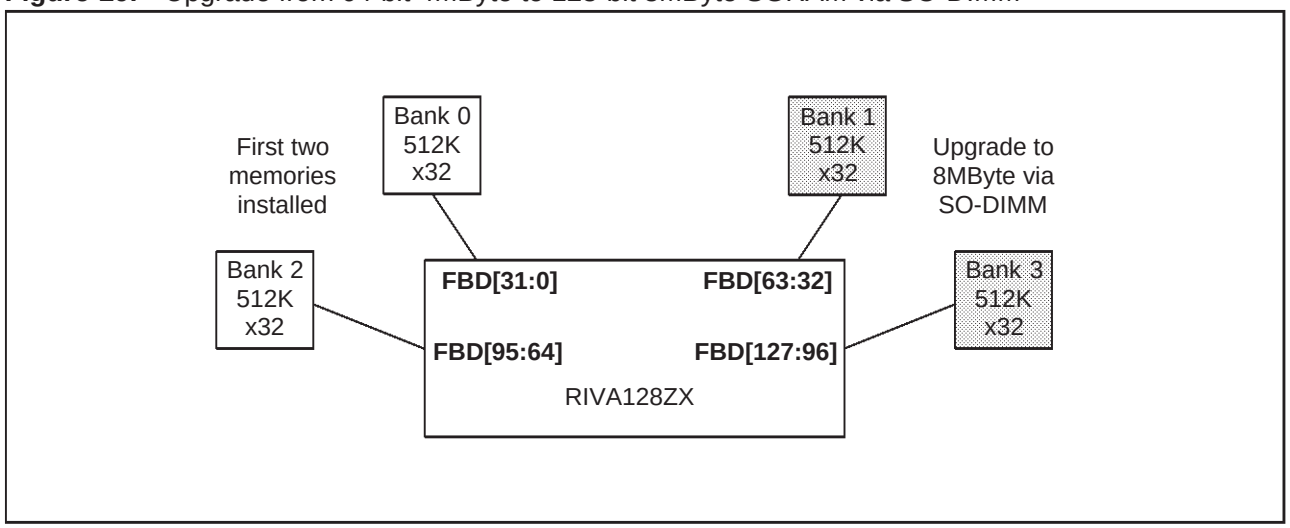
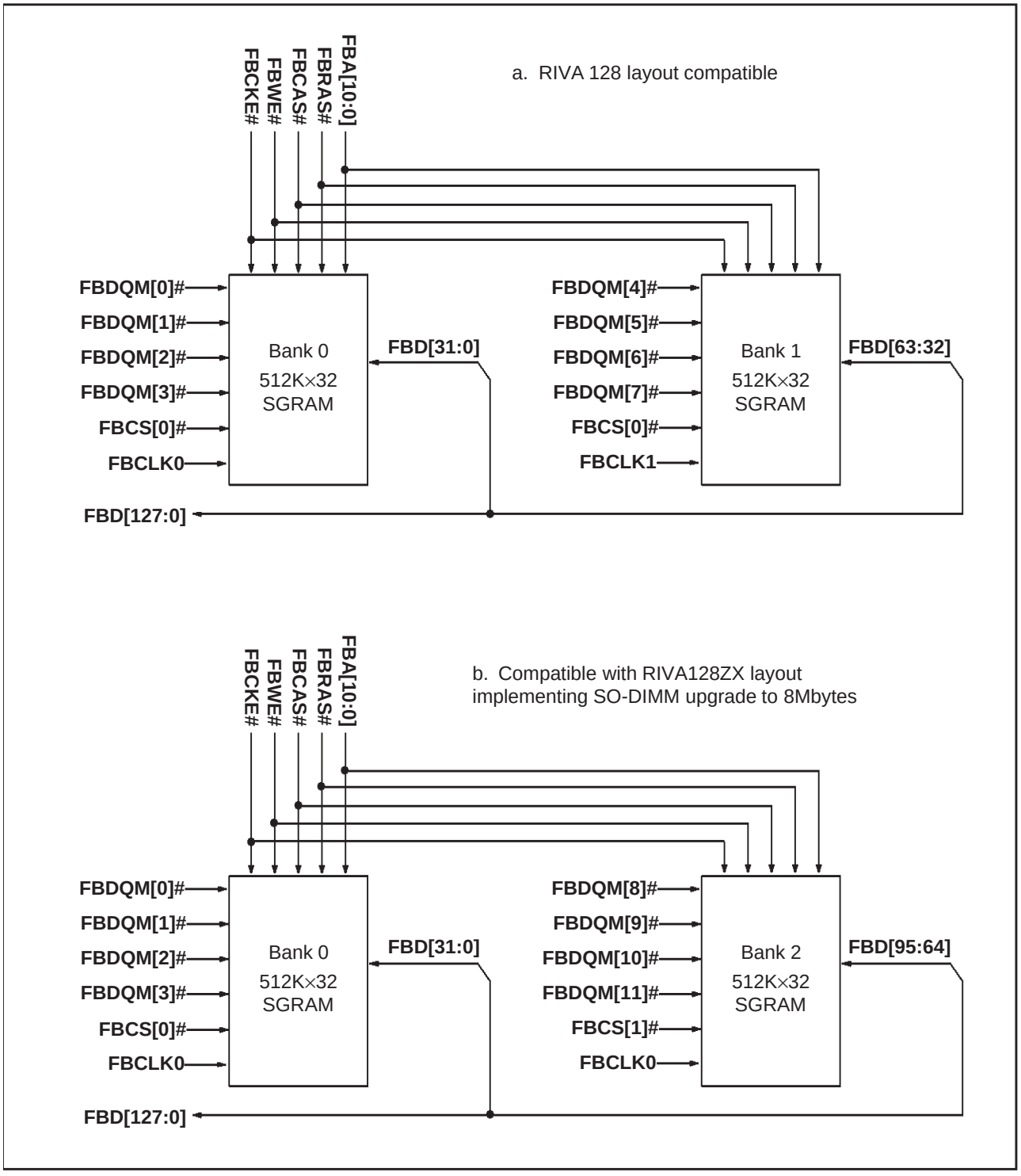


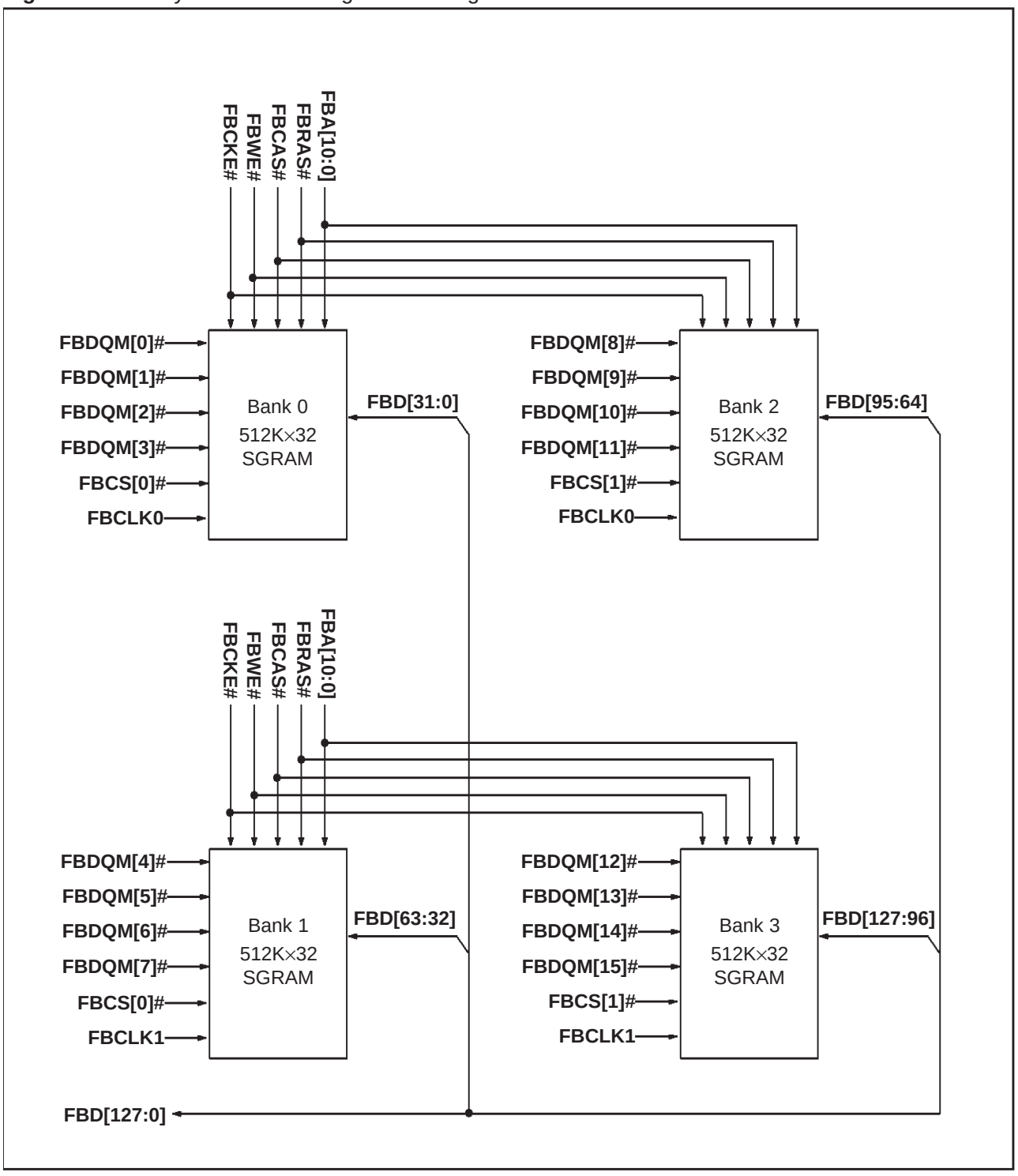
Figure 30. 4 MByte SGRAM configurations using 16Mbit devices



NOTE

1 The 64-bit bus data paths populated by RIVA128ZX are determined by the BIOS during its boot memory detection sequence.

Figure 31. 8MByte SGRAM configuration using 16Mbit devices



6.3 SDRAM/SGRAM ACCESSES AND COMMANDS

Read and write accesses to SDRAM/SGRAM are burst oriented. SDRAM/SGRAM commands supported by the RIVA128ZX are shown in Table 9. Initialization of the memory devices is performed in the standard SDRAM/SGRAM manner. Access sequences begin with an Active command followed by a Read or Write command. The address bits registered coincident with the Read or Write command are used to select the starting column location for the burst access. The RIVA128ZX always uses a burst length of one and can launch a new read or write on every cycle.

SDRAM/SGRAM has a fully synchronous interface with all signals registered on the positive edge of **FBCLKx.** Multiple clock outputs allow reductions in signal loading and more accuracy in data sampling at high frequency. The clock signals can be interspersed as shown in Figure 30, page 33 for optimal loading with either 4 or 8MBytes. The I/O timings relative to **FBCLKx** are shown in Figure 32, page 37.

Table 8. Truth table of supported SDRAM commands

Command[1]	**FBCS0#**	**FBRAS#**	**FBCAS#**	**FBWE#**	**FBDQM**	**FBCS[1]#, FBA[10:0]**	**FBD[63:0]**	**Notes**
Command inhibit (NOP)	H	x	x	x	x	x	x	
No operation (NOP)	L	H	H	H	x	x	x	
Active (select bank and activate row)	L	L	H	H	x	**FBCS[1]#**=bank **FBA[10:0]**=row	x	
Read (select bank and column and start read burst)	L	H	L	H	x	**FBCS[1]#**=bank **FBA[10]**=0 **FBA[7:0]**=col	x	
Write (select bank and column and start write burst)	L	H	L	L	x	**FBCS[1]#**=bank **FBA[10]**=0 **FBA[7:0]**=col	valid data	
Precharge (deactivate row in bank or banks)	L	L	H	L	x	**FBA[10]**=code	x	3
Load mode register	L	L	L	L	x	**FBCS[1]#, FBA[10:0] =** opcode		
Write enable/output enable	-	-	-	-	L	-	active	2
Write inhibit/output High-Z	-	-	-	-	H	-	high-Z	2

NOTES

1 **FBCKE** is high and DSF is low for all supported commands.

2 Activates or deactivates **FBD[63:0]** during writes (zero clock delay) and reads (two-clock delay).

3 For **FBA10** low, **FBCS[1]#** determines which bank is precharged; for **FBA10** high, all banks are precharged irrespective of the state of **FBCS[1]#**.

Table 9. Truth table of supported SGRAM commands

Command[1]	FBCS0#, FBCS1#	FBRAS#	FBCAS#	FBWE#	FBDQM	FBA[10:0]	FBD[127:0]	Notes
Command inhibit (NOP)	H	x	x	x	x	x	x	
No operation (NOP)	L	H	H	H	x	x	x	
Active (select bank and activate row)	L	L	H	H	x	**FBA[10]**=bank **FBA[9:0]**=row	x	
Read (select bank and column and start read burst)	L	H	L	H	x	**FBA[10]**=bank **FBA[9]**=0 **FBA[7:0]**=col	x	
Write (select bank and column and start write burst)	L	H	L	L	x	**FBA[10]**=bank **FBA[9]**=0 **FBA[7:0]**=col	valid data	
Precharge (deactivate row in bank or banks)	L	L	H	L	x	**FBA[10]**=code	x	3
Load mode register	L	L	L	L	x	**FBA[10:0] =** opcode		
Write enable/output enable	-	-	-	-	L	-	active	2
Write inhibit/output High-Z	-	-	-	-	H	-	high-Z	2

NOTES

1 **FBCKE** is high and DSF is low for all supported commands.

2 Activates or deactivates **FBD[127:0]** during writes (zero clock delay) and reads (two-clock delay).

3 For **FBA9** low, **FBA10** determines which bank is precharged; for **FBA9** high, all banks are precharged irrespective of the state of **FBA10**.

SDRAM/SGRAM Initialization

SDRAM/SGRAMs must be powered-up and initialized in a predefined manner. The first SDRAM/SGRAM command is registered on the first clock edge following **PCIRST#** inactive.

All internal SDRAM/SGRAM banks are precharged to bring the device(s) into the “all bank idle” state. The SDRAM/SGRAM mode registers are then programmed and loaded to bring them into a defined state before performing any operational command.

SDRAM/SGRAM Mode register

The Mode register defines the mode of operation of the SDRAM/SGRAM. This includes burst length, burst type, read latency and SDRAM/SGRAM operating mode. The Mode register is programmed via the Load Mode register and retains its state until reprogrammed or power-down.

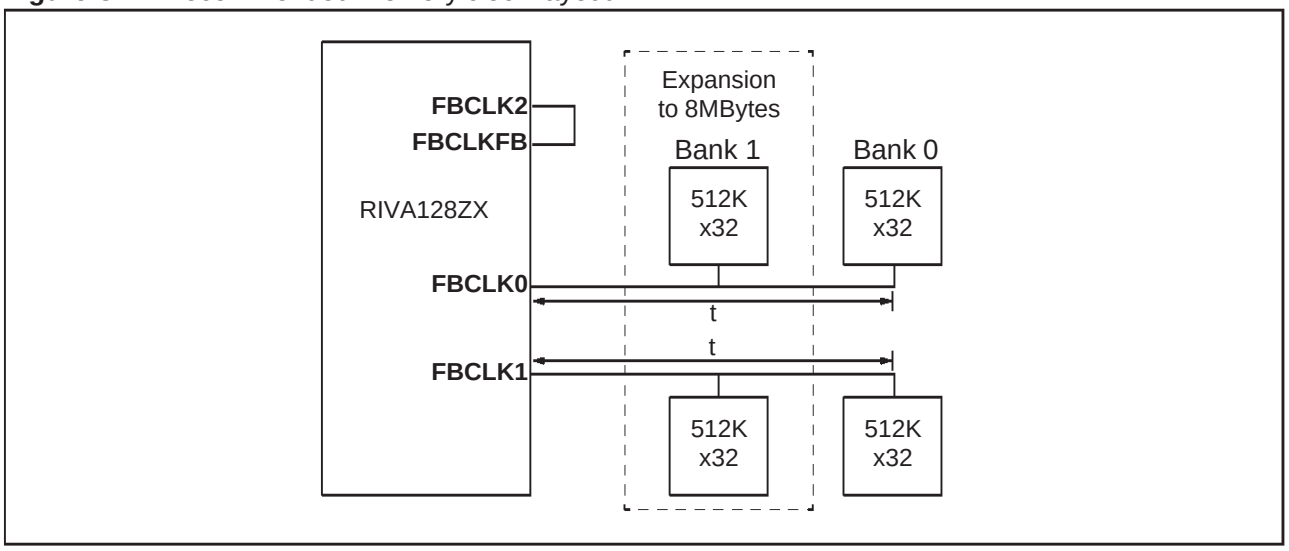
Mode register bits M[2:0] specify the burst length; for the RIVA128ZX SDRAM/SGRAM interface these bits are set to zero, selecting a burst length of one. In this case **FBA[7:0]** select the unique column to be accessed and Mode register bit M[3] is ignored. Mode register bits M[6:4] specify the read latency; for the RIVA128ZX SDRAM/SGRAM interface these bits are set to either 2 or 3, selecting a burst length of 2 or 3 respectively.

6.4 LAYOUT OF FRAMEBUFFER CLOCK SIGNALS

Separate clock signals **FBCLK0** and **FBCLK1** are provided for each bank of memory to give reduced clock skew and loading. Additionally there is a clock feedback loop between **FBCLK2** and **FBCLKFB**.

It is recommended that long traces are used without tunable components. If the layout includes provision for expansion to 8MBytes, the clock path to the 4MByte parts should be at the end of the trace, and the clock path to the 8MByte expansion located between the RIVA128ZX and the 4MByte parts as shown in Figure 32. **FBCLK2** and **FBCLKFB** should be shorted together as close to the package as possible.

Figure 32. Recommended memory clock layout



6.5 FRAMEBUFFER INTERFACE TIMING SPECIFICATION

Figure 33. SDRAM/SGRAM I/O timing diagram

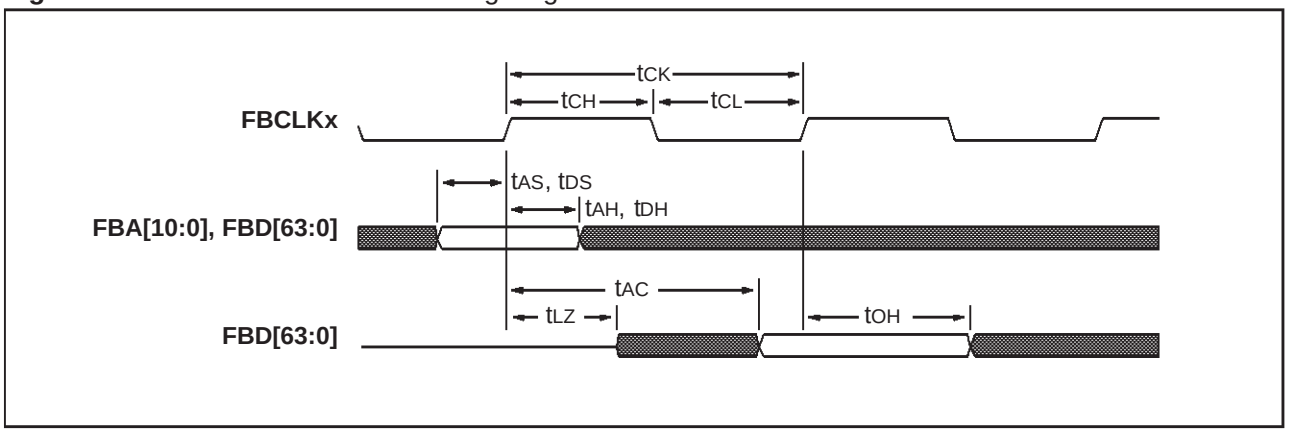
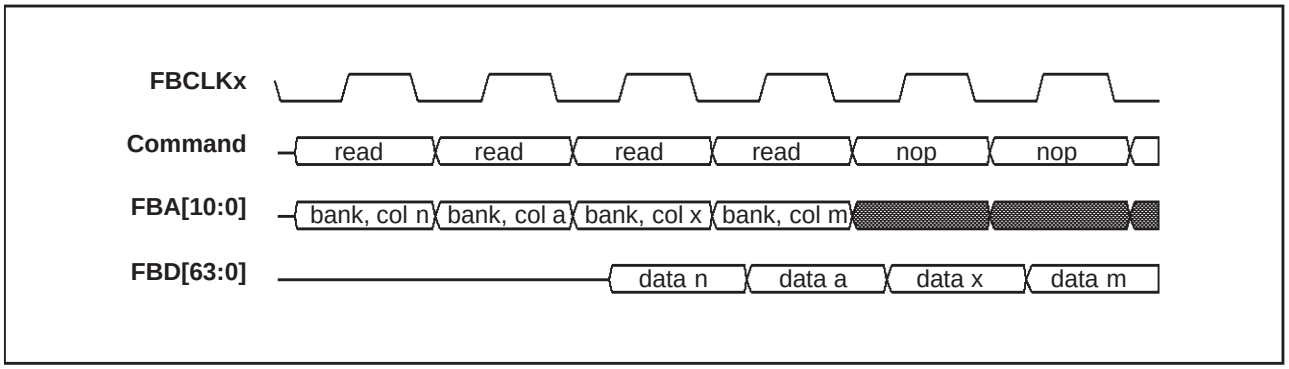


Table 10. SDRAM/SGRAM I/O timing parameters

Symbol	Parameter	Min.		Max.		Unit	Notes
		-10	-12	-10	-12		
t_{CK}	CLK period	10	12	-	-	ns	
t_{CH}	CLK high time	3.5	4.5	-	-	ns	
t_{CL}	CLK low time	3.5	4.5	-	-	ns	
t_{AS}	Address setup time	3	4	-		ns	
t_{AH}	Address hold time	1	1	-		ns	
t_{DS}	Write data setup time	3	4	-		ns	

Symbol	Parameter	Min.		Max.		Unit	Notes
		-10	-12	-10	-12		
t_{DH}	Write data hold time	1	1	-		ns	
t_{OH}	Read data hold time	3	3	-		ns	
t_{AC}	Read data access time	9	9	-		ns	
t_{LZ}	Data out low impedance time	0	0	-		ns	

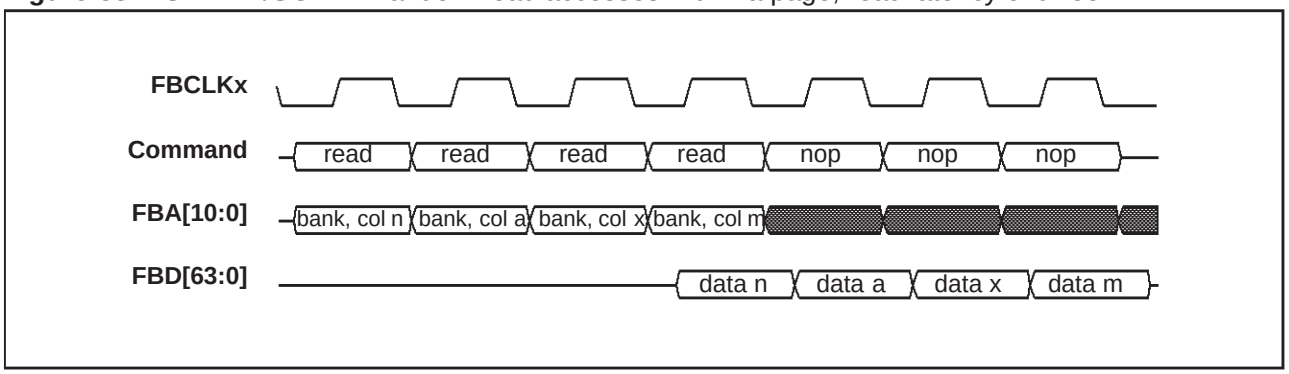
Figure 34. SDRAM/SGRAM random read accesses within a page, read latency of two[1]



NOTE

1 Covers either successive reads to the active row in a given bank, or to the active rows in different banks. DQMs are all active (LOW).

Figure 35. SDRAM/SGRAM random read accesses within a page, read latency of three[1]



NOTE

1 Covers either successive reads to the active row in a given bank, or to the active rows in different banks. **FBDQM** is all active (LOW).

Figure 36. SDRAM/SGRAM read to write, read latency of three

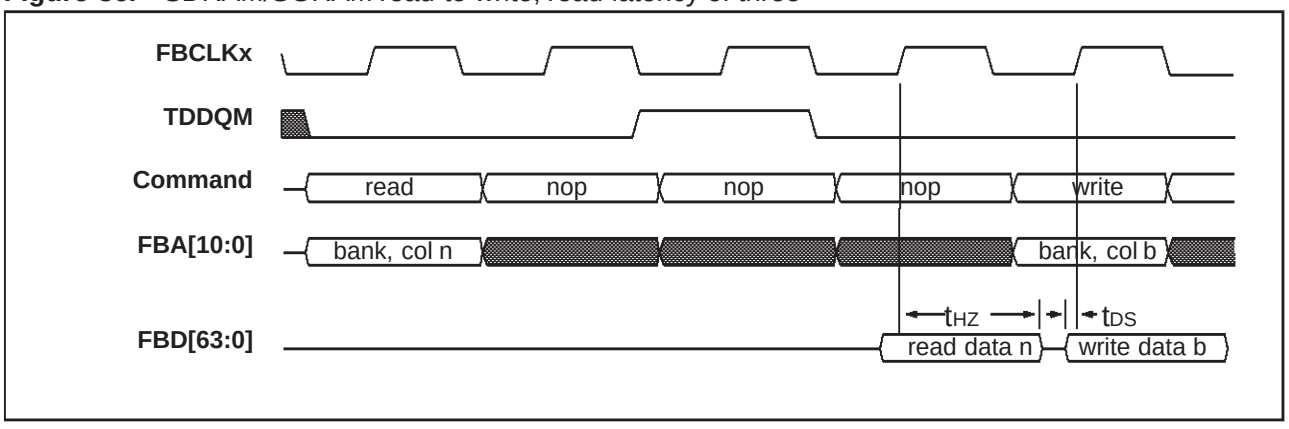
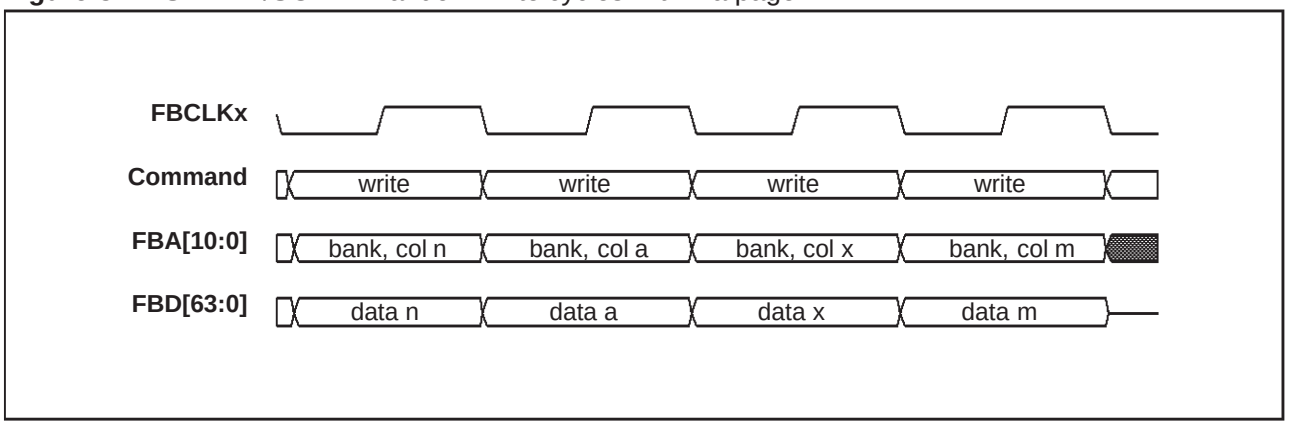


Table 11. SDRAM/SGRAM I/O timing parameters

Symbol	Parameter	Min.	Max.	Unit	Notes
t_{HZ}	Data out high impedance time	4	10	ns	
t_{DS}	Write data setup time	4		ns	

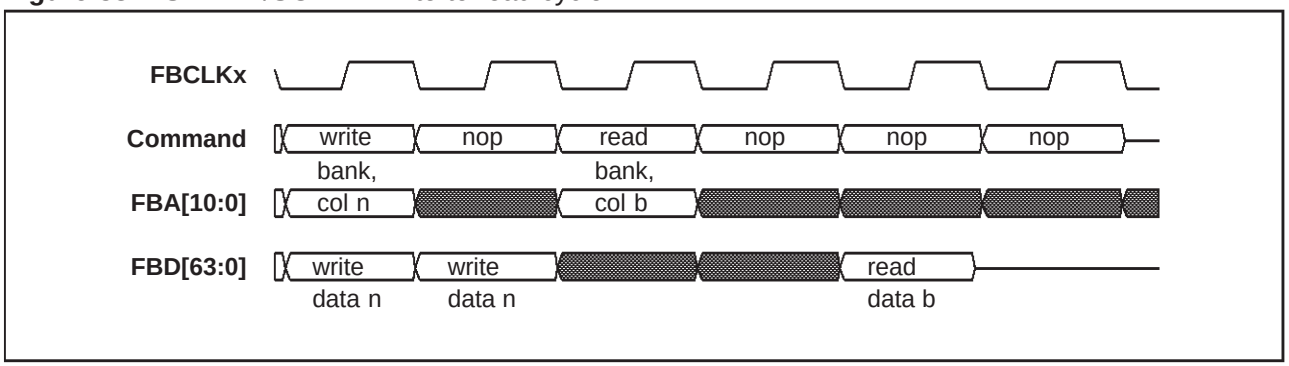
Figure 37. SDRAM/SGRAM random write cycles within a page



NOTE

1 Covers either successive writes to the active row in a given bank or to the active rows in different banks. **FBDQM** is active (low).

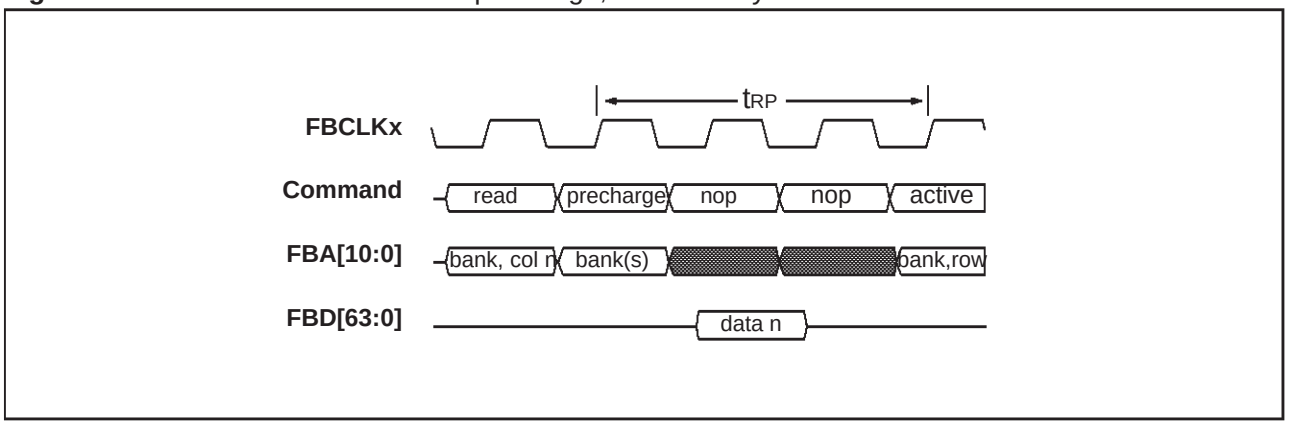
Figure 38. SDRAM/SGRAM write to read cycle



NOTE

1 A read latency of 2 is shown for illustration

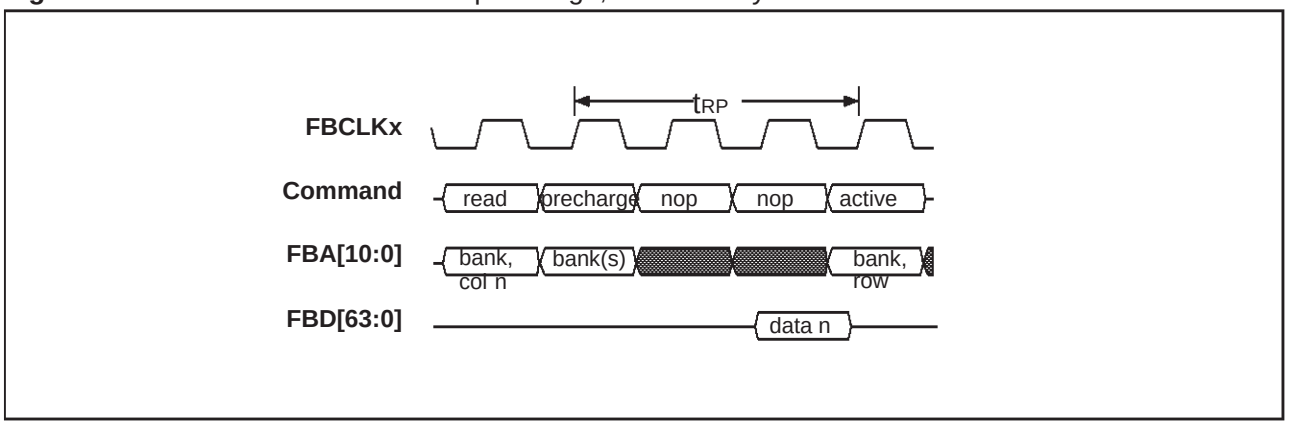
Figure 39. SDRAM/SGRAM read to precharge, read latency of two



NOTE

1 **FBDQM** is active (low)

Figure 40. SDRAM/SGRAM read to precharge, read latency of three



NOTE

1 **FBDQM** is active (low)

Figure 41. SDRAM/SGRAM Write to Precharge

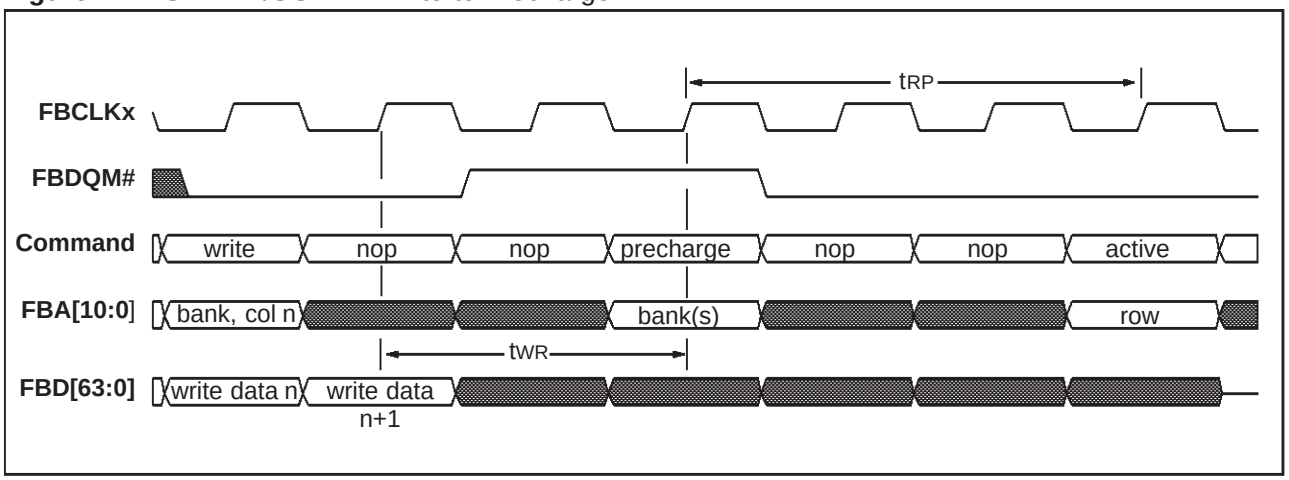


Figure 42. SDRAM/SGRAM Active to Read or Write

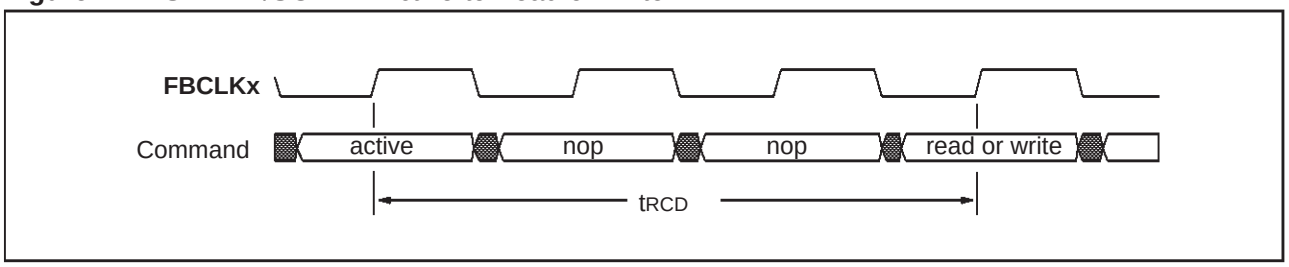


Table 12. SDRAM/SGRAM timing parameters

Symbol	Parameter	Min.	Max.	Unit	Notes
t_{CS}	**FBCSx**, **FBRAS#**, **FBCAS#**, **FBWE#**, **FBDQM** setup time	3		ns	
t_{CH}	**FBCSx**, **FBRAS#**, **FBCAS#**, **FBWE#**, **FBDQM** hold time	1		ns	
t_{MTC}	Load Mode register command to command	2		t_{CK}	
t_{RAS}	Active to Precharge command period	7		t_{CK}	
t_{RC}	Active to Active command period	10		t_{CK}	
t_{RCD}	Active to Read or Write delay	3		t_{CK}	

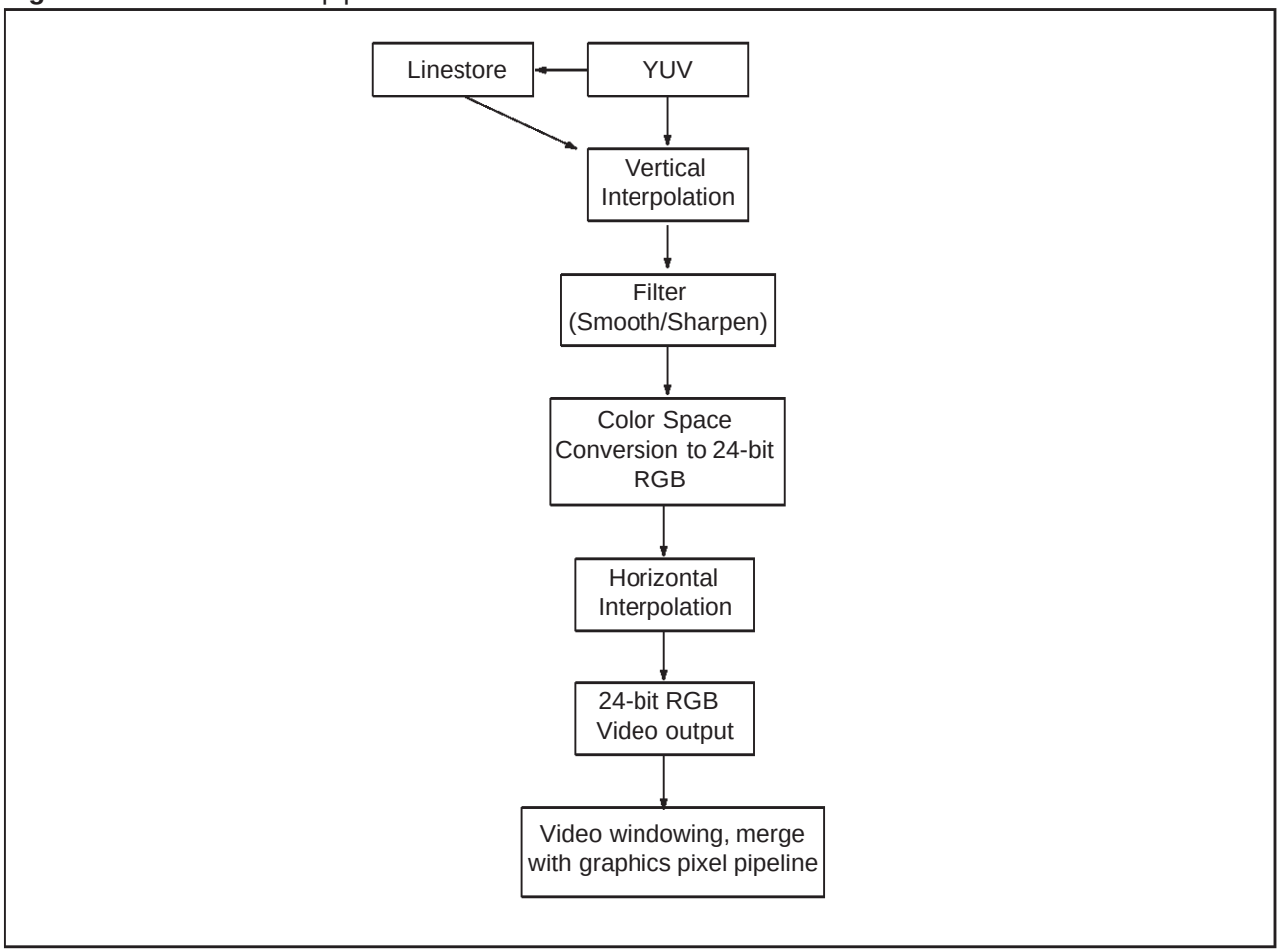
Symbol	Parameter	Min.	Max.	Unit	Notes
t_{REF}	Refresh period (1024 cycles)		16	ms	
t_{RP}	Precharge command period	4		t_{CK}	
t_{RRD}	Active bank A to Active bank B command period	3		t_{CK}	
t_T	Transition time		1	ns	
t_{WR}	Write recovery time	2		t_{CK}	

7 VIDEO PLAYBACK ARCHITECTURE

The RIVA128ZX video playback architecture is designed to allow playback of CCIR PAL or NTSC video formats with the highest quality while requiring the smallest video surface. The implementation is optimized around the Windows 95 Direct Video and ActiveX APIs, and supports the following features:

- Accepts interlaced video fields:
  - This allows the off-screen video surface to consume less memory since only one field (half of each frame) is stored. Double buffering between fields is done in hardware with 'temporal averaging' being applied based on intraframing.
- Linestore:
  - To support high quality video playback the RIVA128ZX memory controller and video overlay engine supports horizontal and vertical interpolation using a 3x2 multitap interpolating filter with image sharpening.
- YUV to RGB conversion:
  - YUV 4:2:2 format to 24-bit RGB true-color
  - Chrominance optimization/user control
- Color key video composition

Figure 43. Video scaler pipeline



7.1 VIDEO SCALER PIPELINE

The RIVA128ZX video scaler pipeline performs stretching of video images in any arbitrary factor in both horizontal and vertical directions. The video scaler pipeline consists of the following stages:

1 Vertical stretching
2 Filtering
3 Color space conversion
4 Horizontal stretching

Vertical stretching

Vertical stretching is performed on pixels prior to color conversion. The video scaler linearly interpolates the pixels in the vertical direction using an internal buffer which stores the previous line of pixel information.

Filtering

After vertical interpolation, the pixels are horizontally filtered using an edge-enhancement or a smoothing filter. The edge-enhancement filter enhances picture transition information to prevent loss of image clarity following the smoothing filtering stage. The smoothing filter is a low-pass filter that reduces the noise in the source image.

Color space conversion

The video overlay pipeline logic converts images from YUV 4:2:2 format to 24-bit RGB true-color. The default color conversion coefficients convert from YCrCb to gamma corrected RGB.

Saturation controls make sure that the conversion does not exceed the output range. Four control flags in the color converter provides 16 sets of color conversion coefficients to allow adjustment of the hue and saturation. The brightness of each R G B component can also be individually adjusted, similar to the brightness controls of the monitor.

Horizontal stretching

Horizontal stretching is done in 24-bit RGB space after color conversion. Each component is linearly interpolated using a triangle 2-tap filter.

Windowing and panning

Video images are clipped to a rectangular window by a pair of registers specifying the position and width.

By programming the video start address and the video pitch, the video overlay logic also supports a panning window that can zoom into a portion of the source image.

Video composition

With the color keying feature enabled, a programmable key in the graphics pixel stream allows selection of either the video or the graphics output on a pixel by pixel basis. Color keying allows any arbitrary portions of the video to overlay the graphics.

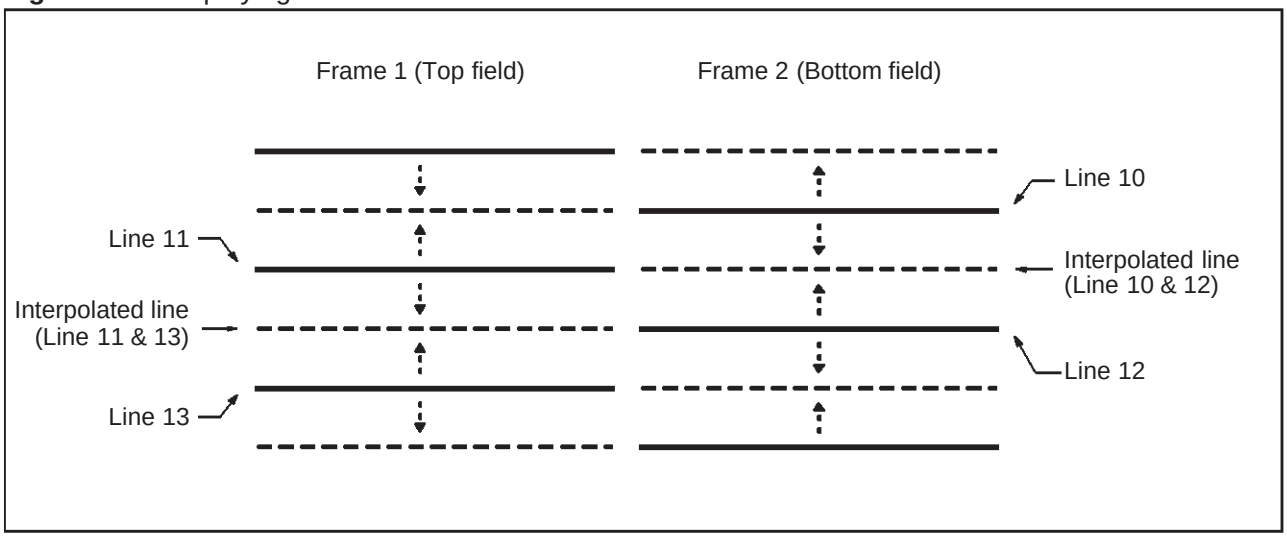
With color keying disabled and video overlay turned on, the video output overlays the graphics in the video window.

Interlaced video

The video overlay can display both non-interlaced and interlaced video.

Traditional video overlay hardware typically drops every other field of an interlaced video stream, resulting in a low frame rate. Some solutions have attempted to overcome the this problem by deinterlacing the fields into a single frame. This however introduces motion artifacts. Fast moving objects appearing in different positions in different fields, when deinterlaced, introduces visible artifacts which look like hair-like lines projecting out of the object.

Figure 44. Displaying 2 fields with 1:1 ratio



The RIVA128ZX video overlay handles interlaced video by displaying every field, at the original frame rate of the video (50Hz for PAL and 60Hz for NTSC). The video scaling logic upscales, in the vertical direction, the luma components in each field and linearly interpolates successive lines to produce the missing lines of each field. This interpolated scale is applied such that the full frame size of each field is stretched to the desired height.

The video scaler offsets the bottom field image by half a source image line to ensure that both frames when played back align vertically.

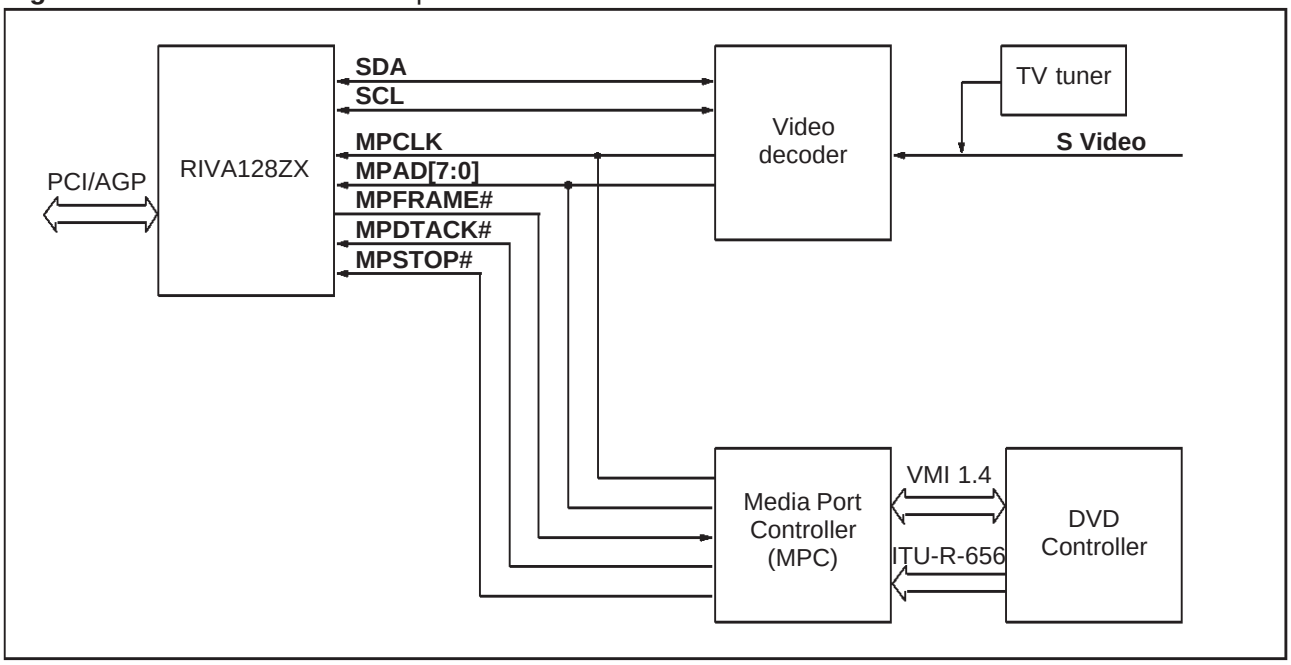
The vertical filtering results in a smooth high quality video playback. Also by displaying both fields one after another, any motion artifacts often found in deinterlaced video output are removed, because the pixels in each field are displayed in the order in which the original source was captured.

8 VIDEO PORT

The RIVA128ZX Multimedia Accelerator introduces a multi-function Video Port that has been designed to exploit the bus mastering functionality of the RIVA128ZX. The Video Port is compliant with a simplified ITU-R-656 video format with control of attached video devices performed through the RIVA128ZX serial interface. Video Port support includes:

- Windows 95 DirectMPEG API acceleration by providing:
  - Bus mastered compressed data transfer to attached DVD and MPEG-2 decoders
  - Local interrupt and pixel stream handling
  - Hardware buffer management of compressed data, decompressed video pixel data and decompressed audio streams
- Supports popular video decoders including the Philips SAA7111A, SAA7112, ITT 3225, and Samsung KS0127. The Video Port initiates transfers of video packets over the internal NV bus to either on or off screen surfaces as defined in the DirectDraw and DirectVideo APIs.
- Supports filtered down-scaling or decimation
- Allows additional devices to be added

Figure 45. Connections to multiple video modules



8.1 VIDEO INTERFACE PORT FEATURES

- Single 8-bit bus multiplexing among four transfer types: video, VBI, host and compressed data
- Synchronous 40MHz address/data multiplexed bus
- Hardware-based round-robin scheduler with predictable performance for all transfer types
- Supports multiple video modules and one ribbon cable board on the same bus
- ITU-R-656 Master Mode
- Video Port
  - Simplified ITU-R-656 Video Format -- supports HSYNC, VSYNC, ODD FIELD and EVEN FIELD
  - VBI data output from video decoder is captured as raw or sliced data

8.2 BI-DIRECTIONAL MEDIA PORT POLLING COMMANDS USING MPC

The Media Port transfers data using a Polling Protocol. The Media Port is enabled on the RIVA128ZX by the host system software. The first cycle after being enabled is a Poll Cycle. The MPC ASIC must respond to every poll cycle with valid data during DTACK active. If no transactions are needed, it responds with 00h. The Media Port will continue to Poll until a transaction is requested, or until there is a Host CPU access to an external register.

Polling Cycle

Media Port initiates a Polling Cycle whenever there is no pending transaction. This gives the MPC ASIC a mechanism to initiate a transaction. The valid Polling commands are listed in the Polling Command table. The priority for the polling requests should be to give the Display Data FIFO highest priority.

CPU Register Write

Initiated by the Host system software.

CPU Register Read Issue

Initiated by the Host system software. The read differs from the write in the fact that it must be done in two separate transfers. The Read Issue is just the initiation of the read cycle. The Media Port transfers the address of the register to be read during this cycle. After completion of the Read Issue cycle the media port goes back to polling for the next transaction. When it receives a Read Data ready command, it will start the next cycle in the read.

CPU Register Read Receive

Initiated by the MPC ASIC when it has read data ready to be transferred to the media port. The MPC ASIC waits for the next polling cycle and returns a Read Data Ready status. The media port will transfer the read data on the next Read Receive Cycle. The PCI bus will be held off and retry until the register read is complete.

Video Compressed Data DMA Write

Initiated by the MPC ASIC with the appropriate Polling Command. The media port manages the Video Compressed data buffer in system memory. Each request for data will return 32 bytes in a single burst.

Display Data DMA Read

Initiated by the MPC ASIC with the polling command. The MPC ASIC initiates this transfer when it wishes to transfer video data in ITU-R-656 format.

Table 13. Media Port Transactions

A0 Cycle	Transaction	Description
11xx0000	Poll_Cycle	Polling Cycle
00xx----	CPUWrite	CPU Register Write
01xx1111	CPURead_Issue	CPU Register Read Issue
11xx1111	CPURead_Receive	CPU Register Read Receive
01xx0001	VCD_DMA_Write	Video Compressed Data DMA Write
11xx1000	Display_Data_Read	Display Data DMA Read

Table 14. Polling Cycle Commands

BIT	Data		Description
0	000xxxx1	NV_PME_VMI_POLL_UNCD	Request DMA Read of Display Data
1	000xxx1x	NV_PME_VMI_POLL_VIDCD	Request DMA Write of Video Compressed Data
3	000x1xxx	NV_PME_VMI_POLL_INT	Request for Interrupt
4	0001xxxx	NV_PME_VMI_POLL_CPURDREC	Respond to Read Issue - Read Data Ready
	00000000	NULL	No Transactions requested

8.3 TIMING DIAGRAMS

Figure 46. Poll cycle

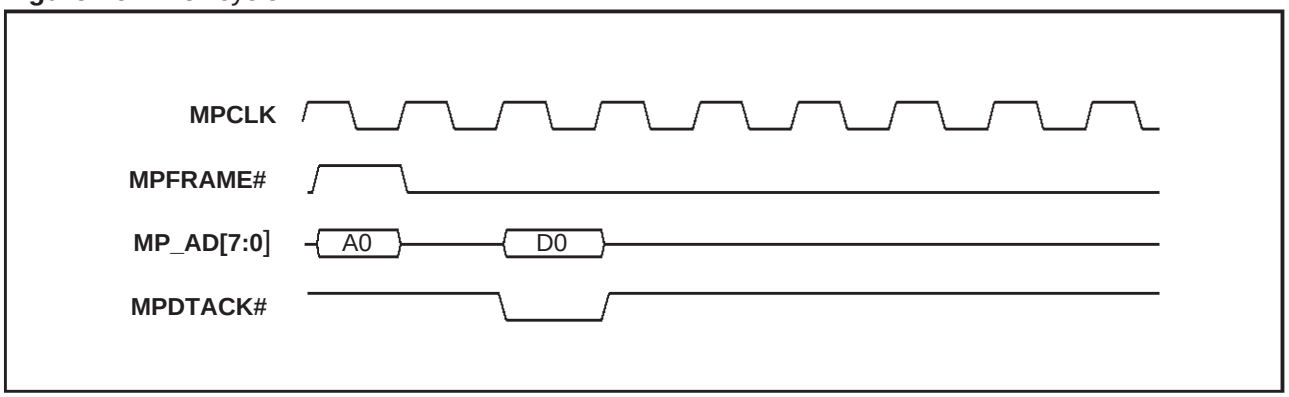


Figure 47. Poll cycle throttled by slave

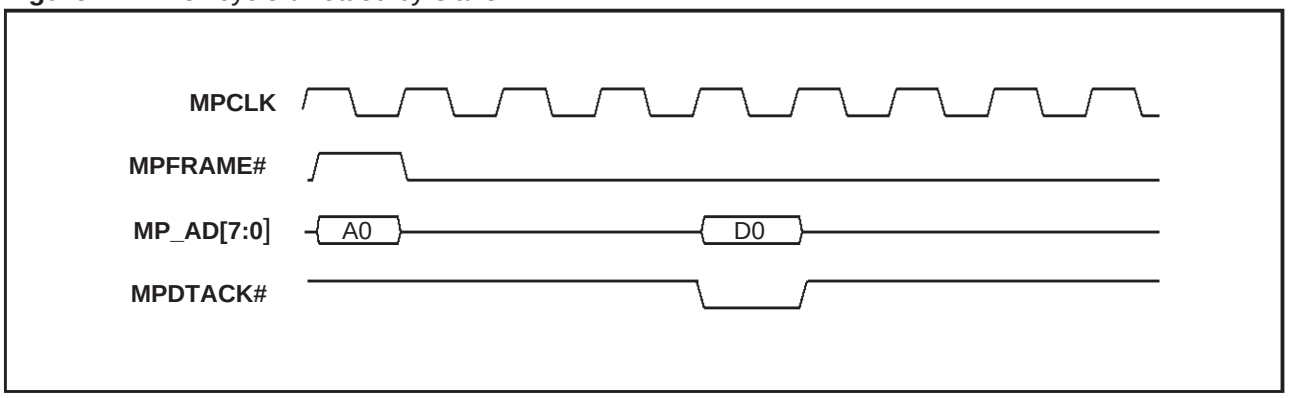


Figure 48. CPU write cycle

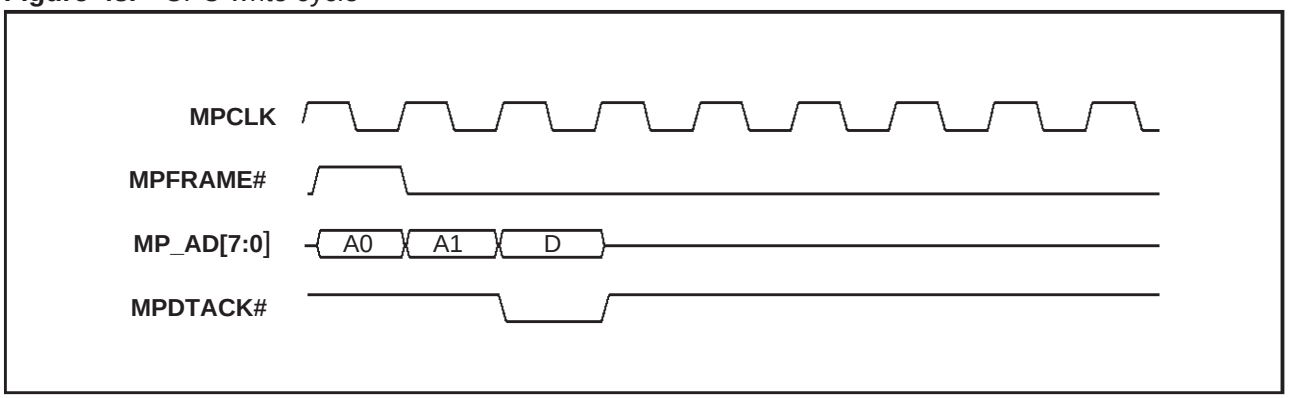


Figure 49. CPU write cycle throttled by slave

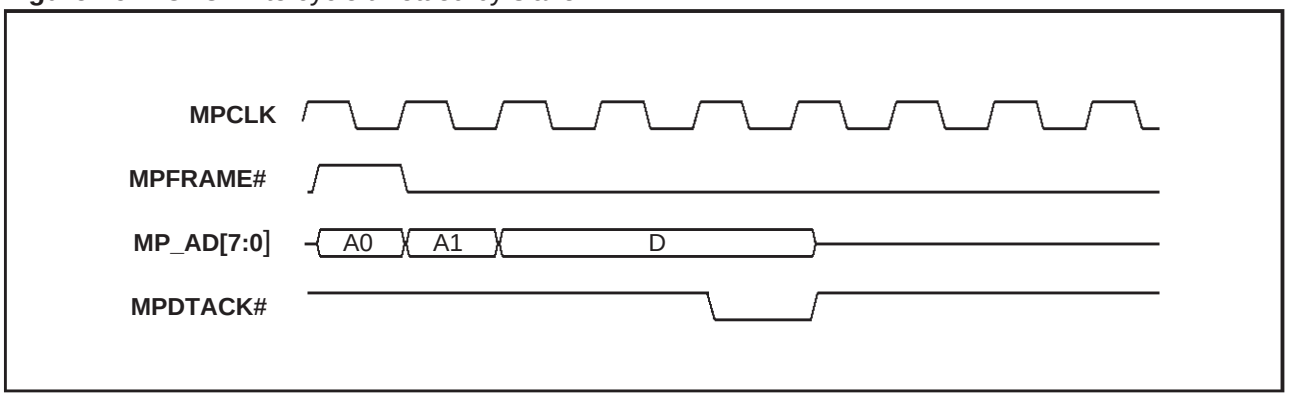


Figure 50. CPU read issue cycle - cannot be throttled by slave

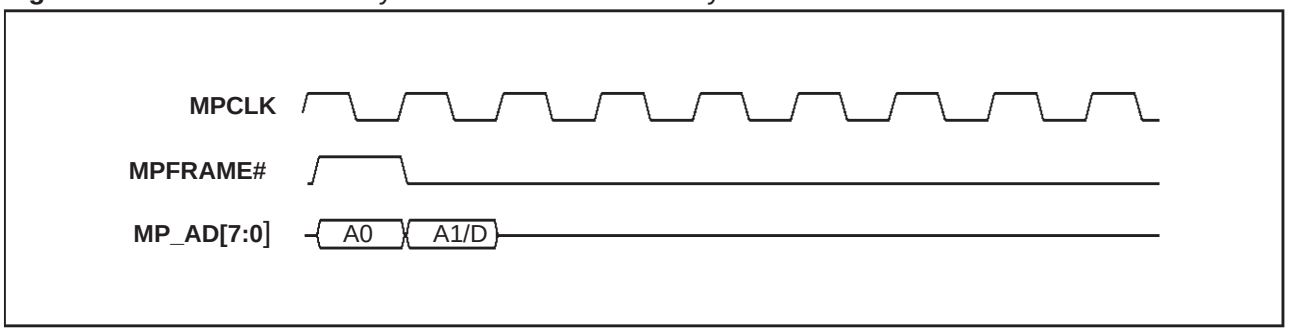


Figure 51. CPU read_receive cycle

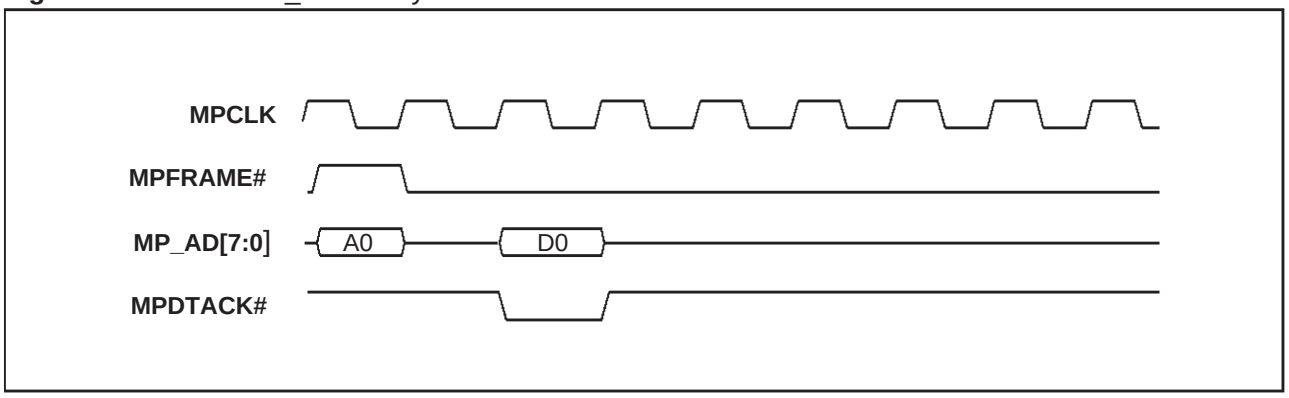


Figure 52. CPU read_receive cycle - throttled by slave

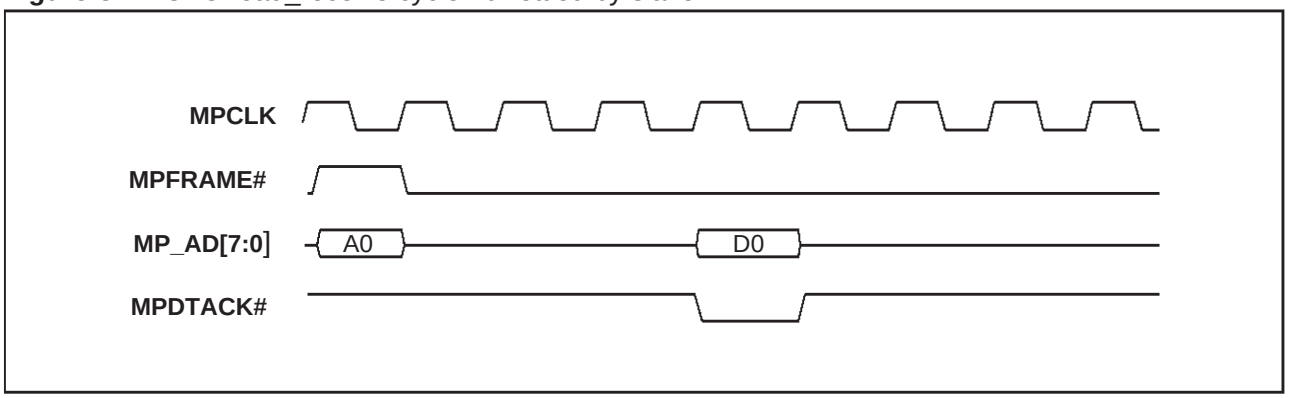


Figure 53. CD write cycle - terminated by master

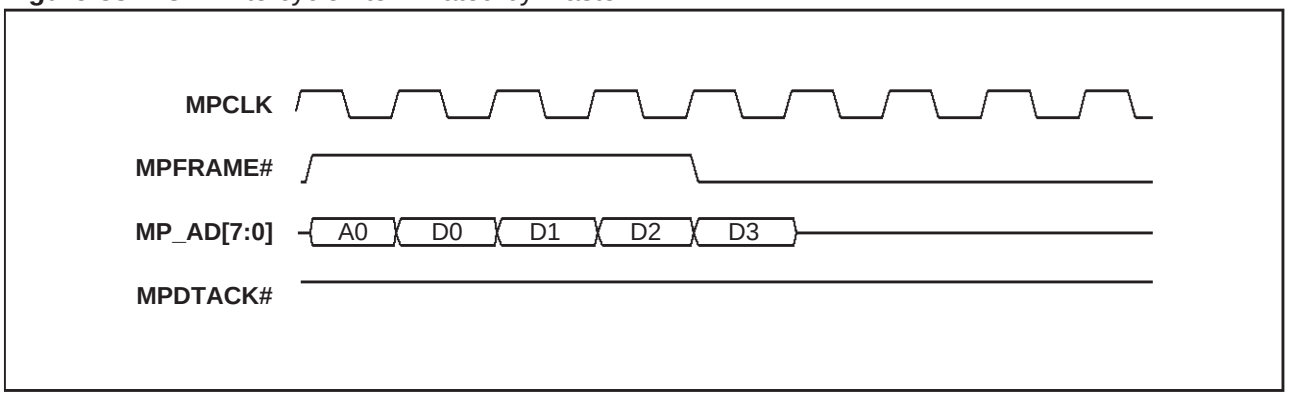


Figure 54. CD write cycle - terminated by slave in middle of transfer

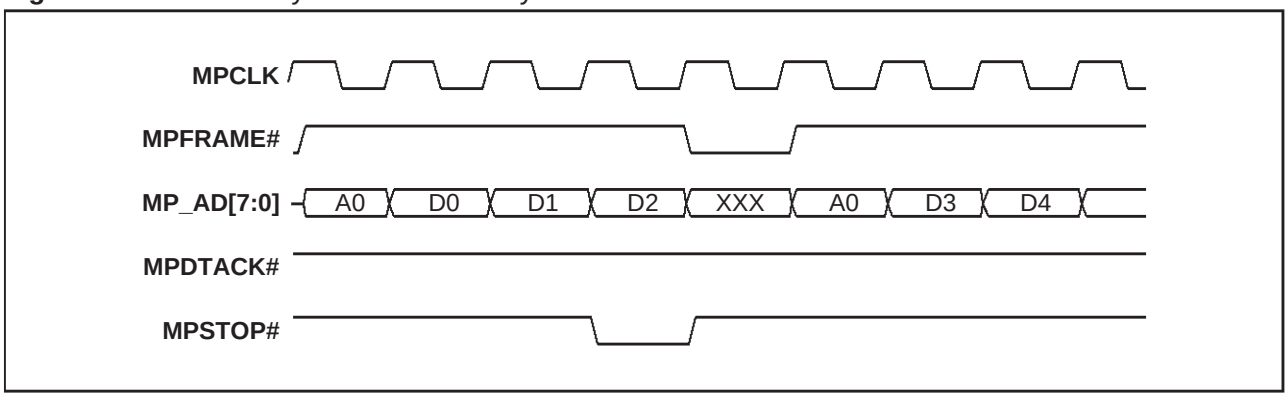


Figure 55. CD write cycle - terminated by slave on byte 31

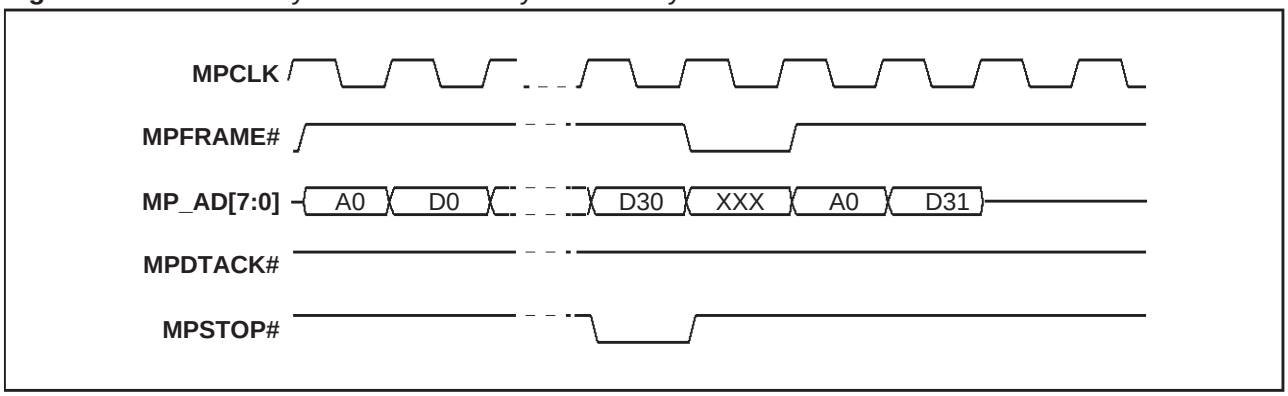


Figure 56. CD write cycle - terminated by slave on byte 32, no effect

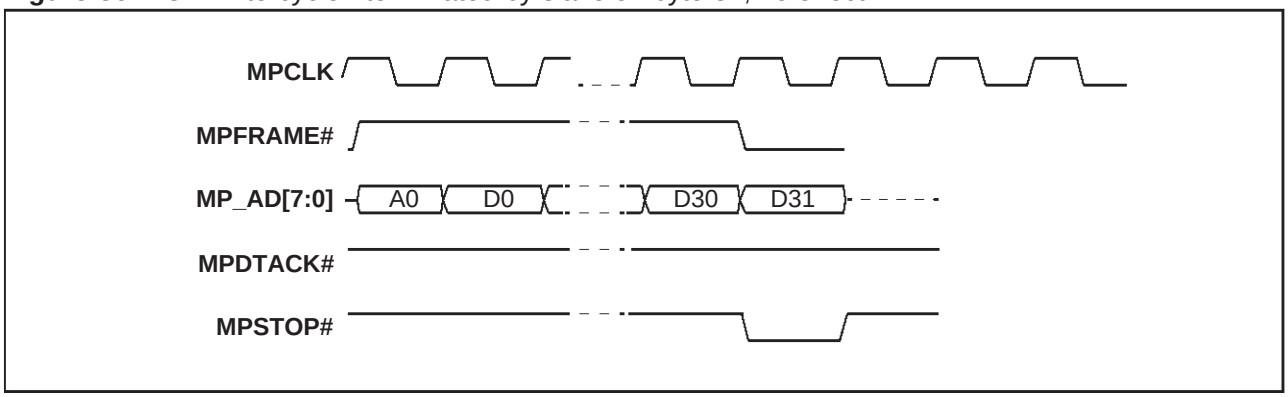


Figure 57. UCD read cycle, terminated by master, throttled by slave

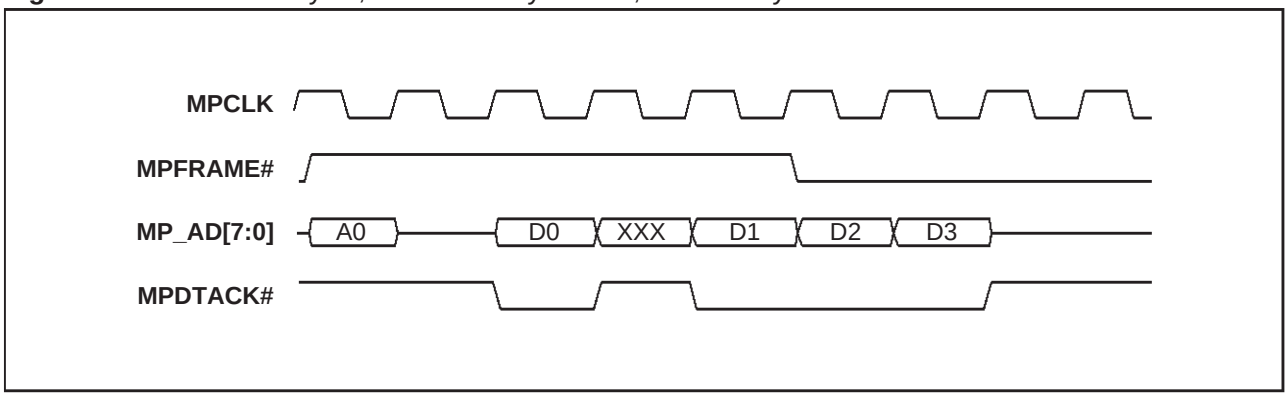


Figure 58. UCD read cycle, terminated by slave, throttled by slave

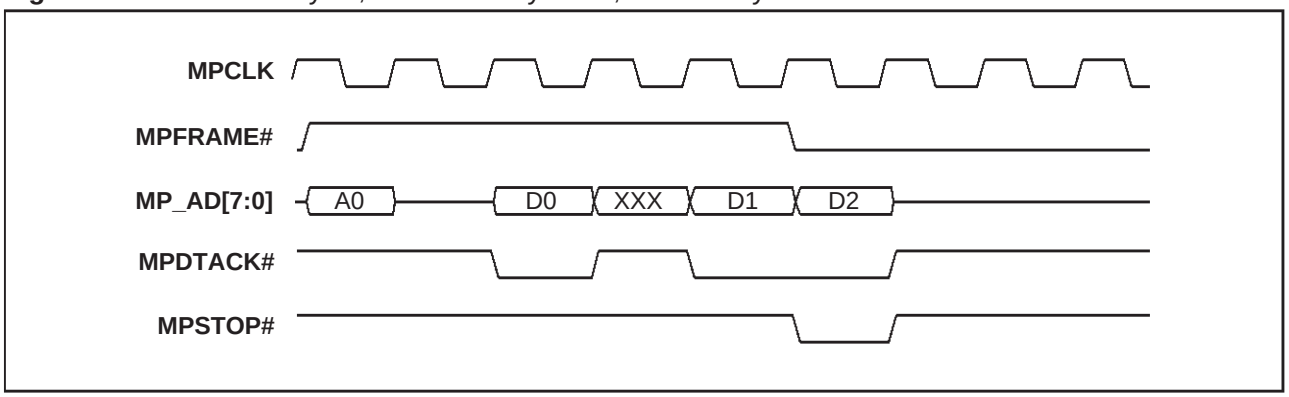


Figure 59. UCD read cycle, slave termination after MPFRAME# deasserted, data taken

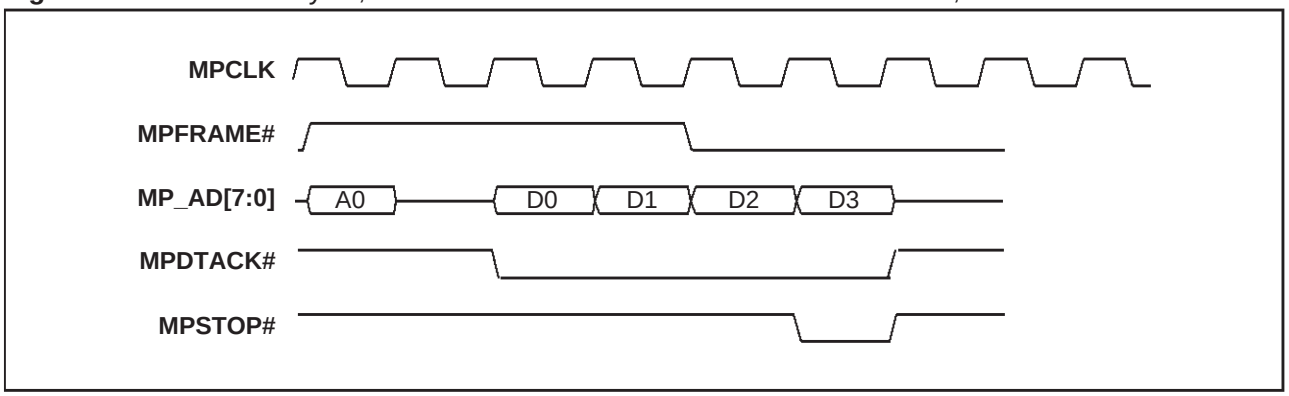


Figure 60. UCD read cycle, slave termination after MPFRAME# deasserted, data not taken

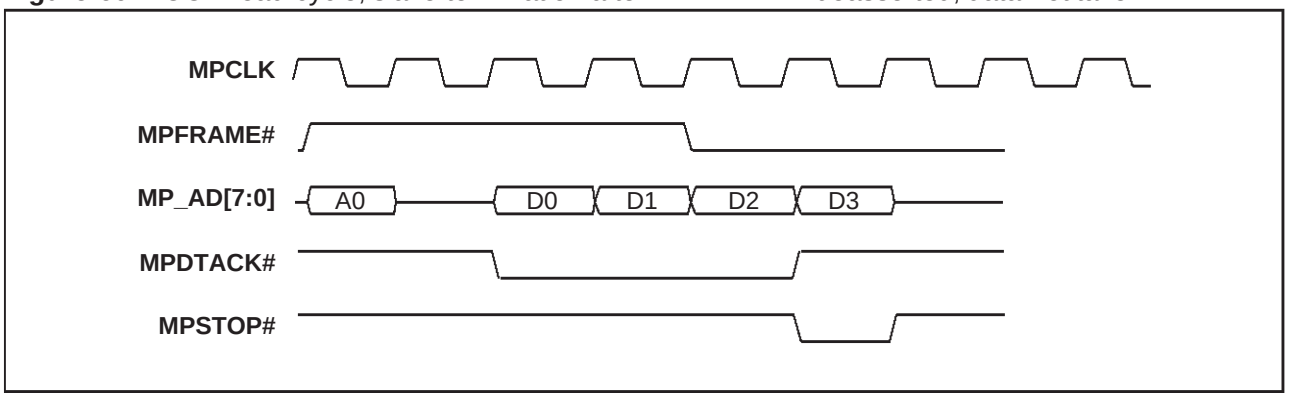
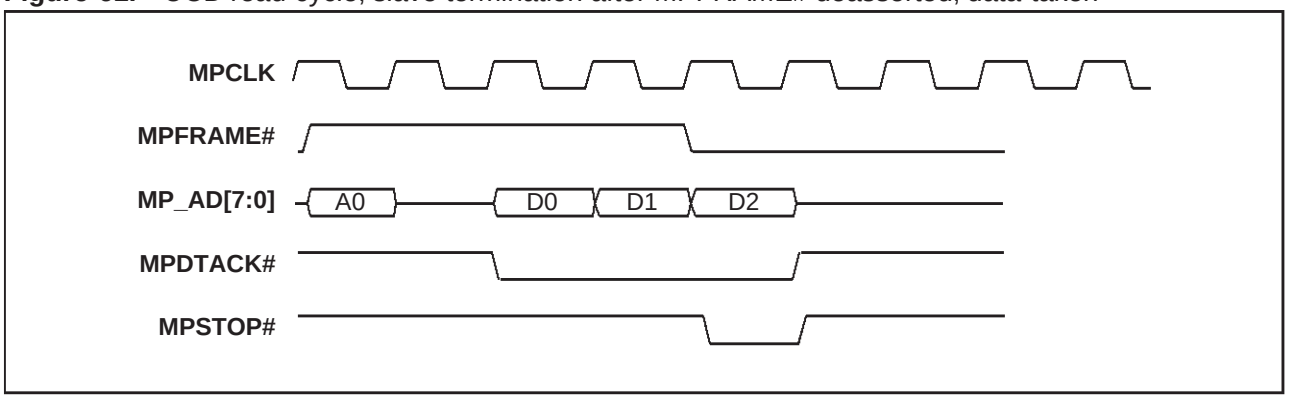


Figure 61. UCD read cycle, slave termination after MPFRAME# deasserted, data taken



8.4 656 MASTER MODE

Table 15 shows the Video Port pin definition when the RIVA128ZX is configured in ITU-R-656 Master Mode. Before entering this mode, RIVA128ZX disables all Video Port devices so that the bus is tri-stated. The RIVA128ZX will then enable the video 656 master device through the serial bus. In this mode, the video device outputs the video data continuously at the PIXCLK rate.

Table 15. 656 master mode pin definition

Normal Mode	656 Master Mode
MPCLK	**PIXCLK**
MPAD[7:0]	**VID[7:0]**
MPFRAME#	Not used
MPDTACK#	Not used
MPSTOP#	Not used

The 656 Master Mode assumes that **VID[7:0]** and **PIXCLK** can be tri-stated when the slave is inactive. If a slave cannot tri-state all its signals, an external tri-state buffer is needed.

Video data capture

Video Port pixel data is clocked into the port by the external pixel clock and then passed to the RIVA128ZX's video capture FIFO.

Pixel data capture is controlled by the ITU-R-656 codes embedded in the data stream; each active line beginning with SAV (start active video) and ending with EAV (end active video).

In normal operation, when SAV = x00, capture of video data begins, and when EAV = xx1, capture of video data ends for that line. When VBI (Vertical Blanking Interval) capture is active, these rules are modified.

656 master mode timing specification

Figure 62. 656 Master Mode timing diagram

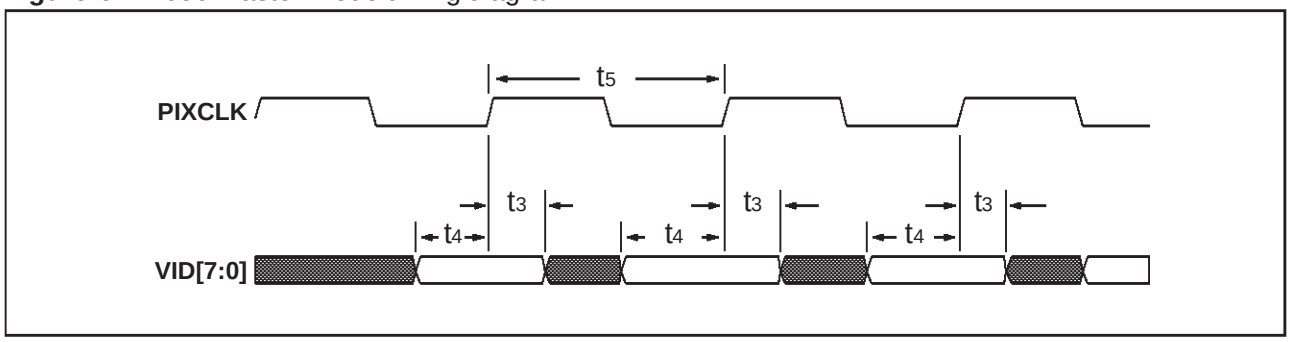


Table 16. ITU-R-656 Master Mode timing parameters

Symbol	Parameter	Min.	Max.	Unit	Notes
t_3	**VID[7:0]** hold from **PIXCLK** high	0		ns	
t_4	**VID[7:0]** setup to **PIXCLK** high	5		ns	
t_5	**PIXCLK** cycle time	35		ns	

NOTE

1 **VACTIVE** indicates that valid pixel data is being transmitted across the video port.

Table 17. YUV (YCbCr) byte ordering

1st byte	2nd byte	3rd byte	4th byte	5th (next dword)	6th byte	7th byte
U[7:0]	Y0[7:0]	V[7:0]	Y1[7:0]	U[7:0]	Y0[7:0]	V[7:0]
Cb[7:0]	Y0[7:0]	Cr[7:0]	Y1[7:0]	Cb[7:0]	Y0[7:0]	Cr[7:0]

8.5 VBI HANDLING IN THE VIDEO PORT

RIVA128ZX supports two basic modes for VBI data capture. VBI mode 1 is for use with the Philips SAA7111A digitizer, VBI mode 2 is for use with the Samsung KS0127 digitizer.

In VBI mode 1, the region to be captured as VBI data is set up in the SAA7111A via the serial interface, and in the RIVA128ZX under software control. The SAA7111A responds by suppressing generation of SAV and EAV codes for the lines selected, and sending raw sample data to the port. The RIVA128ZX Video Port capture engine starts capturing VBI data at an EAV code in the line last active and continues to capture data without a break until it detects the next SAV code. VBI capture is then complete for that field.

In VBI mode 2, the region to be captured as VBI data is set up in a similar manner. The KS0127 responds by enabling VBI data collection only during the lines specified and framed by normal ITU-R-656 SAV/EAV codes. The RIVA128ZX Video Port capture engine starts capturing data at an SAV code controlled by the device driver, and continues capturing data under control of SAV/EAV codes until a specific EAV code identified by the device driver is sampled. VBI capture is then complete for that field. The number of bytes collected will vary depending on the setup of the KS0127.

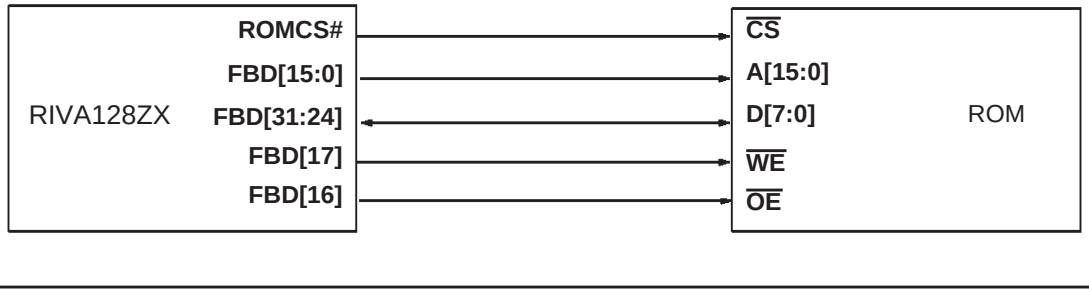
8.6 SCALING IN THE VIDEO PORT

The RIVA128ZX Video Port allows any arbitrary scale factor between 1 and 31. For best results the scale factors of 1, 2, 3, 4, 6, 8, 12, 16, and 24 are selected to avoid filtering losses. The Video Port decimates in the y-direction, dropping lines every few lines depending on the vertical scaling factor. The intention is to support filtered downscaling in the attached video decoder.

9 BOOT ROM INTERFACE

BIOS and initialization code for the RIVA128ZX is accessed from a 32KByte ROM. The RIVA128ZX memory bus interface signals **FBD[15:0]** and **FBD[31:24]** are used to address and access one of 64KBytes of data respectively. The unique decode to the ROM device is provided by the **ROMCS#** chip select signal.

Figure 63. ROM interface



ROM interface timing specification

Figure 64. ROM interface timing diagram

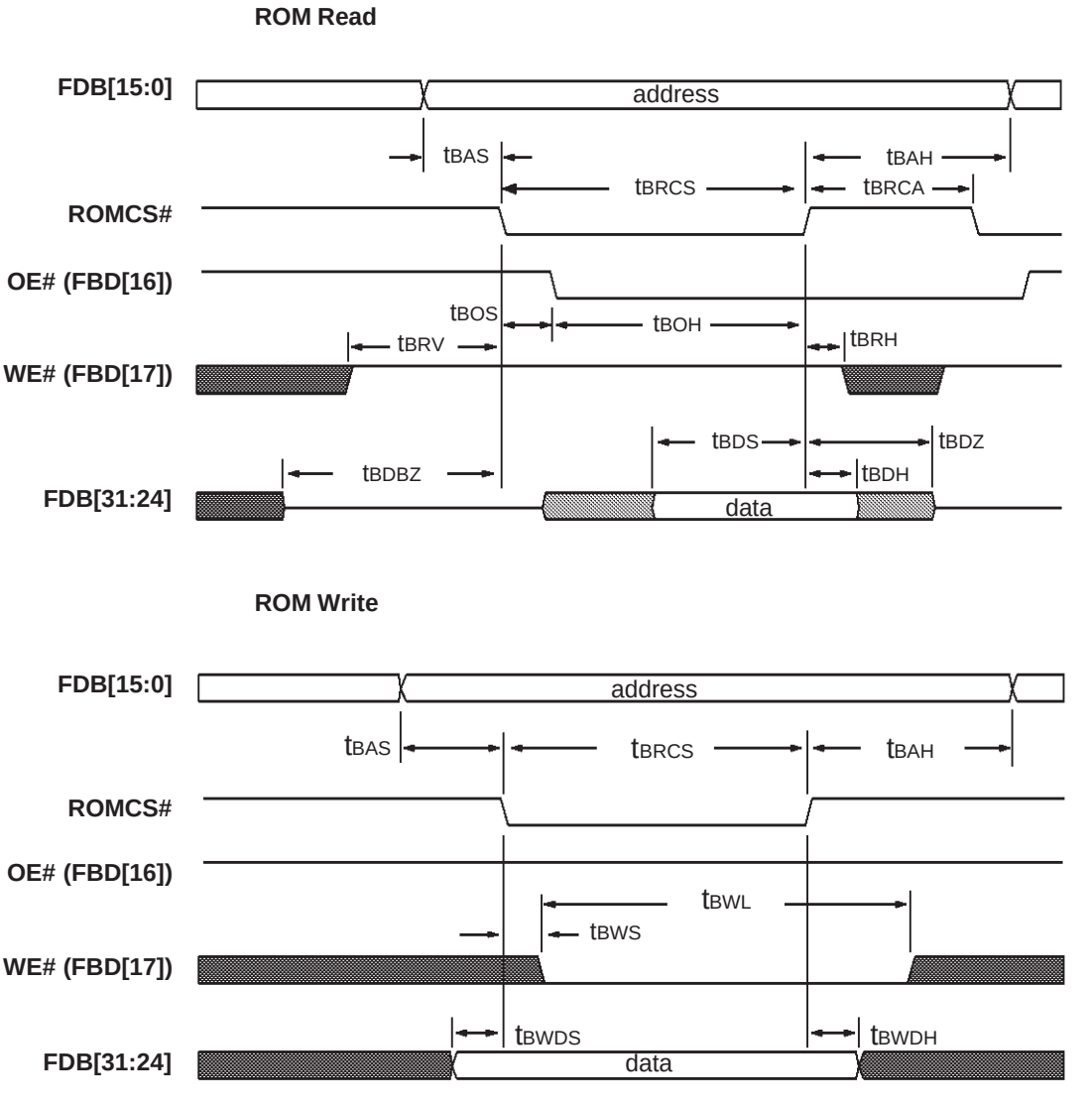


Table 18. ROM interface timing parameters

Symbol	Parameter	Min.	Max.	Unit	Notes
t_{BRCS}	**ROMCS#** active pulse width	20T_{MCLK}-5		ns	
t_{BRCA}	**ROMCS#** precharge time	T_{MCLK}-5		ns	
t_{BRV}	Read valid to **ROMCS#** active	T_{MCLK}-5		ns	
t_{BRH}	Read hold from **ROMCS#** inactive	T_{MCLK}-5		ns	
t_{BAS}	Address setup to **ROMCS#** active	T_{MCLK}-5		ns	
t_{BAH}	Address hold from **ROMCS#** inactive	T_{MCLK}-5		ns	
t_{BOS}	OE# low from **ROMCS#** active			ns	2
t_{BOH}	OE# low to **ROMCS#** inactive			ns	3
t_{BWS}	WE# low from **ROMCS#** active			ns	2
t_{BWL}	WE# low time			ns	3
t_{BDBZ}	Data bus high-z to **ROMCS#** active	T_{MCLK}-5		ns	
t_{BDS}	Data setup to **ROMCS#** inactive	10		ns	
t_{BDH}	Data hold from **ROMCS#** inactive	0		ns	
t_{BDZ}	Data high-z from **ROMCS#** inactive		T_{MCLK}-5	ns	
t_{BWDH}	Write data hold from **ROMCS#** inactive	0.5T_{MCLK}-5		ns	
t_{BWDS}	ROM write data setup to **ROMCS#** active	T_{MCLK}-5		ns	

NOTE

1 T_{MCLK} is the period of the internal memory clock.
2 This parameter is programmable in the range 0 - 3 MCLK cycles
3 This parameter is programmable in the range 0 - 15 MCLK cycles

10 POWER-ON RESET CONFIGURATION

The RIVA128ZX latches its configuration on the trailing edge of **RST#** and holds its system bus interface in a high impedance state until this time. To accomplish this, pull-up or pull-down resistors are connected to the **FBA[9:0]** pins as appropriate. Since there are no internal pull-up or pull-down resistors and the address bus should be floating during reset, a resistor value of up to 47KΩ, but no less than 10KΩ, should be sufficient.

The power-on reset configuration seen by the chip may be overwritten. The external **FBA[9:0]** configuration is stored in latches. An additional writable register, containing all of the configuration bits plus an additional STRAP_OVERWRITE bit (register bit [11]), exists in parallel with the latches to allow the host to overwrite the external value. Writing to address BOOT_0 (0x00101000) writes into this register.

The STRAP_OVERWRITE bit controls whether the latches or the parallel writable register are selected. When STRAP_OVERWRITE is set to 0, the latched **FBA[9:0]** configuration is selected. When set to 1, the chip uses the register value. Reading from address BOOT_0 reads either the configuration latches or the parallel register depending on the STRAP_OVERWRITE setting.

Power-on reset FBA[9:0] bit assignments

9	8	7	6	5	4	3	2	1	0
AGP Mode	TV Mode		Crystal	Host Interface	RAM Width	ACPI Supported	RAM Type	Sub-Vendor	Bus Speed

[9] AGP Mode. This bit selects whether the AGP bus 1X or 2X mode is enabled.
0 = AGP 2X mode enabled
1 = AGP 1X mode enabled

[8:7] TV Mode. These bits select the timing format when TV mode is enabled.
00 = Reserved
01 = NTSC
10 = PAL
11 = TV mode disabled

[6] Crystal Frequency. This bit should match the frequency of the crystal or reference clock connected to **XTALOUT** and **XTALIN**.
0 = 13.500MHz (used where TV output may be enabled)
1 = 14.31818MHz

[5] Host Interface
0 = PCI
1 = AGP

[4] RAM Width
0 = 64-bit framebuffer data bus width (the upper 64-bit data bus and byte selects are tri-state)
1 = 128-bit framebuffer data bus width
Although a 10KΩ resistor is used to strap the default state of this bit in RIVA 128 and RIVA128ZX reference designs, it is ignored by the video BIOS which auto-detects the memory type and configures the RIVA128ZX appropriately.

[3] APCI Supported
0 = ACPI registers not supported. This provides compatibility with the PCI register configuration as implemented by RIVA 128. The RIVA128ZX will be identified by DEVICE_ID_CHIP = 0x18 at PCI Configuration offset 0x0.
1 = ACPI registers supported. Power management registers supported in PCI configuration space. The RIVA128ZX will be identified by DEVICE_ID_CHIP = 0x19 at PCI Configuration offset 0x0.

[2] RAM Type

0 = 16Mbit, 2 or 4 internal bank SGRAM or 16Mbit SDRAM. The BIOS should test the memory to determine whether it supports 2 or 4 internal banks.

1 = 8Mbit, 2 internal bank SGRAM.

Although a 10KΩ resistor is used to strap the default state of this bit in RIVA 128 and RIVA128ZX reference designs, it is ignored by the video BIOS which auto-detects the memory type and configures the RIVA128ZX appropriately.

[1] Sub-Vendor. This bit indicates whether the PCI Subsystem Vendor field is located in the system motherboard BIOS or adapter card VGA BIOS. If the Subsystem Vendor field is located in the system BIOS it must be written by the system BIOS to the PCI configuration space prior to running any PnP code.

0 = System BIOS (Subsystem Vendor ID and Subsystem ID set to 0x0000)

1 = Adapter card VGA BIOS (Subsystem Vendor ID and Subsystem ID read from ROM BIOS at location 0x54 - 0x57)

[0] Bus Speed. This bit indicates the value returned in the 66MHz bit in the PCI Configuration registers.

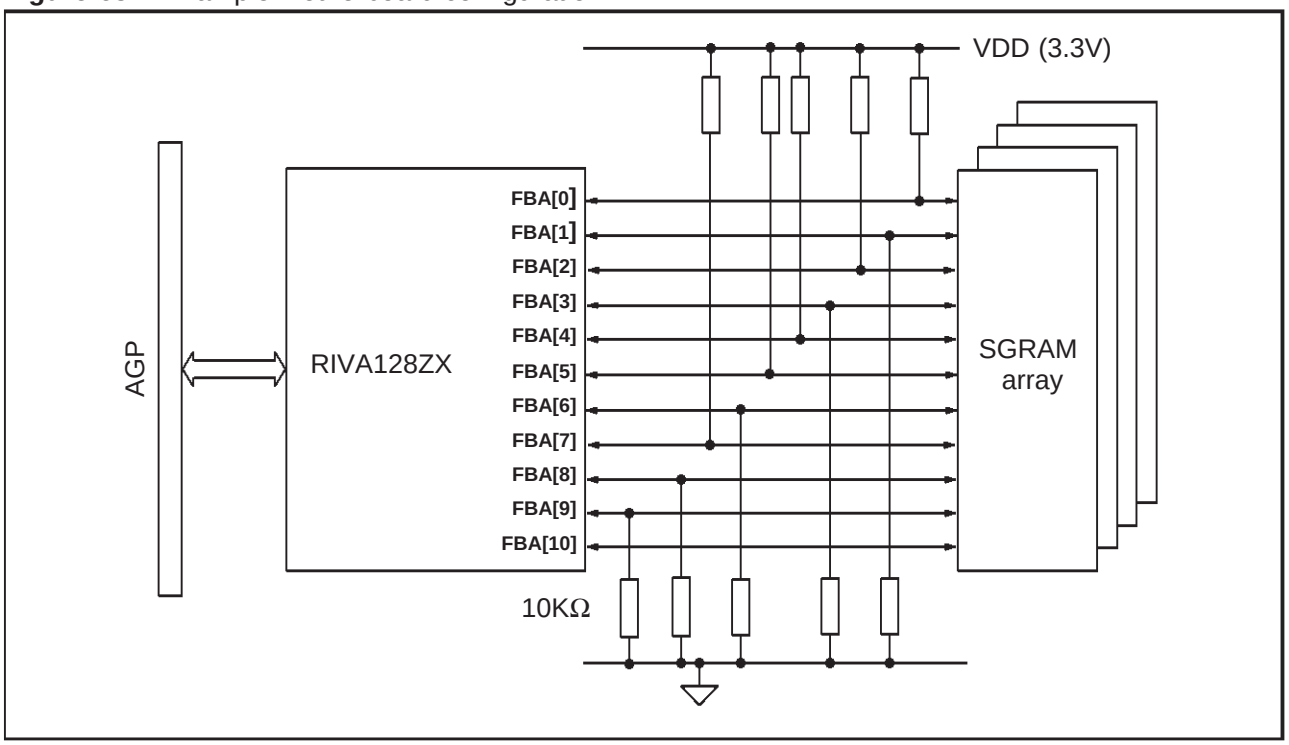
0 = RIVA128ZX PCI interface is 33MHz

1 = RIVA128ZX is 66MHz capable

The following example configuration is shown in Figure 65:

- Subsystem Vendor ID initialized to 0 and writeable by system BIOS (see Appendix A, page 76)
- 8Mbit, 2 internal bank, SGRAM
- 128-bit framebuffer interface
- AGP 2X mode enabled including 66MHz PCI 2.1 compliant subset
- Using 13.5000MHz crystal
- TV output mode is NTSC

Figure 65. Example motherboard configuration



11 DISPLAY INTERFACE

11.1 PALETTE-DAC

The Palette-DAC integrated into the RIVA128ZX supports a traditional pixel pipeline with the following enhancements:

- Support for 10:10:10, 8:8:8, 5:6:5 and 5:5:5 direct color pixel modes
- Support for dynamic gamma correction on a pixel by pixel basis
- Support for mixed indexed color and direct color pixels
- 256 x 24 LUT for 8-bit indexed modes
- High quality video overlay
  - Accepts interlaced video fields allowing a reduction in memory buffering requirements while incorporating temporal averaging
  - Line buffer for horizontal and vertical interpolation of video streams up to square pixel PAL resolution
  - 3x2 multitap interpolating filter with image sharpening
  - Color key in all color pixel modes
  - High quality YUV to RGB conversion with chrominance control.

11.2 PIXEL MODES SUPPORTED

8-bit indexed color

In the 8-bit indexed color each 32-bit word contains four 8-bit indexed color pixels, each comprising bits b[7:0] as shown below.

Pixel formats (FBD[31:0])																															
31	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Pixel 3								Pixel 2								Pixel 1								Pixel 0							
b7	b6	b5	b4	b3	b2	b1	b0	b7	b6	b5	b4	b3	b2	b1	b0	b7	b6	b5	b4	b3	b2	b1	b0	b7	b6	b5	b4	b3	b2	b1	b0

NOTE

1 This 32-bit representation can be extended to 64-bit and 128-bit widths by duplicating the 32-bit word in little-endian format.

16-bit direct color modes (5:6:5 direct and 5:5:5 with and without gamma correction)

In 5:5:5 color modes bit 15 of each pixel can be enabled to select whether pixel data bypasses the LUT to feed the DACs directly, or indirectly, through the LUT, to allow gamma correction to be applied. If not enabled then the bypass mode will always be selected, and the LUT powered down. The 16-bit modes include a 5:6:5 format which always bypasses the LUT.

5:6:5 mode	**Pixel formats (FBD[31:0])**																														
	Pixel 1																Pixel 0														
	31	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
0	0	Red gamma					Green gamma					Blue gamma					0	Red gamma					Green gamma					Blue gamma				
0	1	Red bypass					Green bypass					Blue bypass					1	Red bypass					Green bypass					Blue bypass				
1	Red bypass						Green bypass					Blue bypass					Red bypass						Green bypass						Blue bypass			

NOTE

1 This 32-bit representation can be extended to 64-bit and 128-bit widths by duplicating the 32-bit word in little-endian format.

32-bit direct color (8:8:8 with gamma correction or 10:10:10 direct)

In 32-bit color mode bit 31 of each pixel selects whether pixel data bypasses the LUT, to feed the DACs directly or indirectly, through the LUT, to allow gamma correction to be applied. In the table below the Red, Green and Blue bypass bits are shown individually as R[9:0], G[9:0], and B[9:0] because, in the bypass

mode pixel format, the least significant bits of each color are located separately in the top byte of the pixel. This also permits an 8:8:8 mode without gamma with <1% error if desired.

Use of pixel input pins (FBD[31:0])																													
Pixel 0																													
31	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
0	X	X	X	X	X	X	X	Red gamma								Green gamma								Blue gamma							
1	X	R1	R0	G1	G0	B1	B0	R9	R8	R7	R6	R5	R4	R3	R2	G9	G8	G7	G6	G5	G4	G3	G2	B9	B8	B7	B6	B5	B4	B3	B2

NOTE

This 32-bit representation can be extended to 64-bit and 128-bit widths by duplicating the 32-bit word in little-endian format.

Limitations on line lengths

Table 19. Permitted line length multiples

bpp	8	16	32
Number of pixels that the line length must be a multiple of	4	2	1

11.3 HARDWARE CURSOR

The RIVA128ZX supports a 32x32 15bpp full color hardware cursor as defined by Microsoft Windows.

- Full color 5:5:5 format
- Pixel complement
- Transparency
- Pixel inversion

The cursor pattern is stored in a 2KByte bitmap located in off-screen framestore. Details of programming the hardware cursor are given in the RIVA128ZX *Programming Reference Manual* [2]. Registers control cursor enabling/disabling, location of cursor bitmap and cursor display coordinates. The cursor data and it's position should only be changed during frame flyback. The cursor should be disabled when not being used.

Cursor format

15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
A	Red					Green					Blue				

The 5-bit RGB color components are expanded to 10 bits per color before combining with the graphics display data. The expanded 10-bit color is composed of the 5-bit cursor color replicated in the upper and lower 5-bit fields. The cursor pixels are combined such that the cursor will overlay a video window if present.

Cursor pixel bit 15 (A) is the replace mode bit. When A=1, the cursor pixel replaces the normal display pixel. If A=0, the expanded 30 bits of the cursor color are XORed with the display pixel to provide the complement of the background color.

Cursor pixels can be made transparent (normal display pixel is unmodified) by setting to a value of 0x0000. To invert the bits of the normal display pixel, the cursor pixel should be set to 0x7FFF.

11.4 SERIAL INTERFACE

The RIVA128ZX serial interface supports connection to DDC1/2B, DDC2AB and DDC2B+ compliant monitors and to serial interface controlled video decoders and tuners. Supported video decoder chips include Philips SAA7110, SAA7111A, ITT 3225 and Samsung KS0127. For details of address locations and protocols applying to specific parts refer to the appropriate manufacturer's datasheet.

The serial interface in RIVA128ZX requires operation under software control to provide emulation of the interface standard. RIVA128ZX can act as a master for communication with slave devices like those mentioned above. It also acts as a master when interfacing to a DDC1/2 compatible monitor. Although it is not Access.bus compatible, it can communicate with a DDC2AB compatible monitor via the DDC2B+ protocol. (No other Access.bus peripherals can be attached although other serial devices may co-reside on the DDC bus). The RIVA128ZX can clock stretch incoming messages in the event that the software handler is interrupted by another task.

11.5 ANALOG INTERFACE

Figure 66. Recommended circuit (crystal circuit is for designs not supporting TV out)

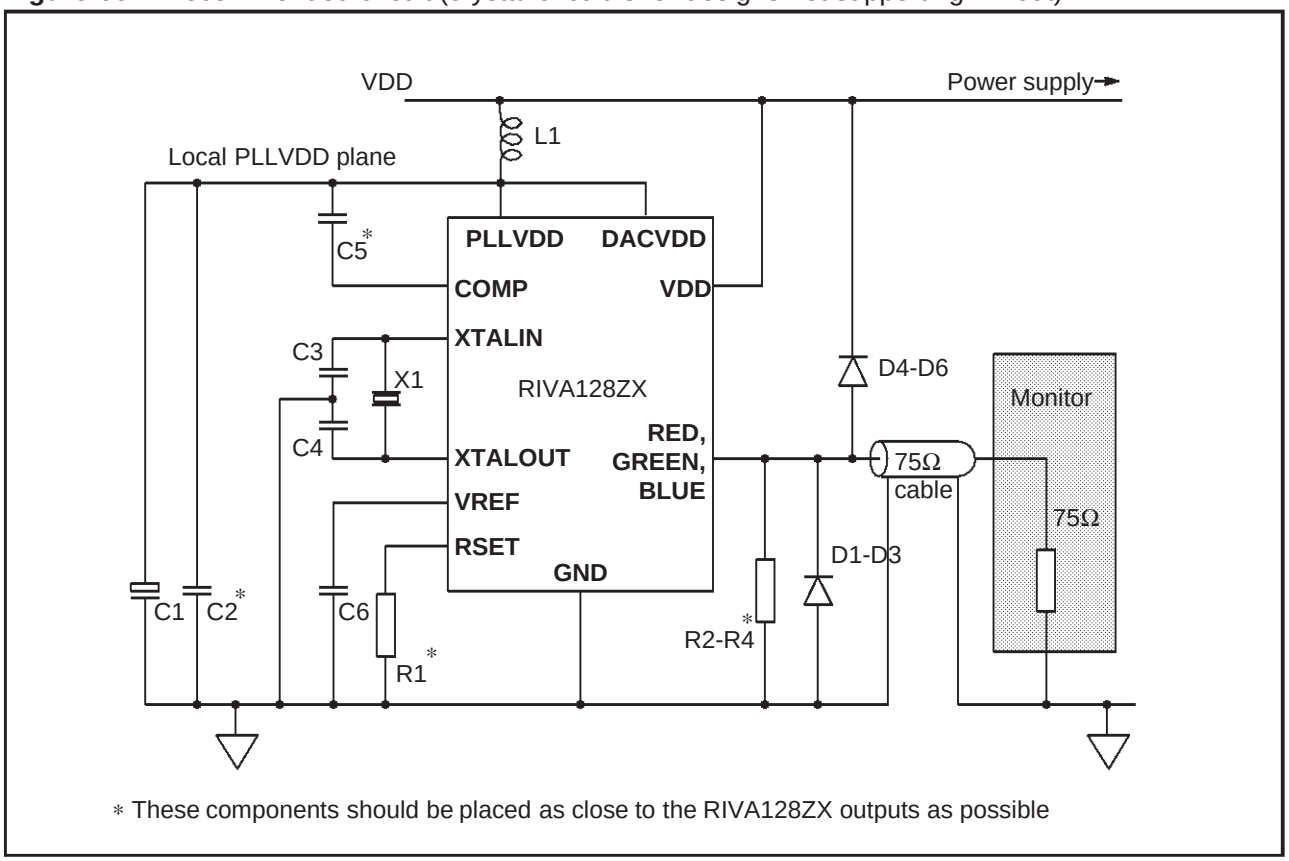


Table 20. Table of parts for recommended circuit (Figure 66)

Part number	Value	Description
C1	22µF	tantalum capacitor
C2	100nF	surface mount capacitor
C3, C4	10pF	surface mount capacitor
C5, C6	10nF	surface mount capacitor
R1	147Ω	1% resistor
R2-R4	75Ω	1% resistor
D1-D6	1N4148	protection diodes
L1	1µH	inductor
X1	13.50000MHz	series resonant crystal (used where TV output may be required)
	14.31818MHz	series resonant crystal

11.6 TV OUTPUT SUPPORT

Reference clock options

The RIVA128ZX supports two synthesizer reference clock frequencies; 13.5MHz and 14.31818MHz. The reference clock frequency is determined by a crystal or reference clock connected to the **XTALIN** and **XTALOUT** pins. Where TV-out is supported, **XTALOUT** should be driven by a 13.5MHz reference clock derived from an external NTSC/PAL clock source as illustrated in Figure 67. The clock frequency should match the power-on configuration setting described in Section 10, page 55.

PAL/NTSC TV interface

The RIVA128ZX supports TV output through an external Analog Devices AD722 PAL/NTSC RGB encoder chip as shown in Figure 67. A MicroClock MK2715 NTSC/PAL clock chip provides a common source for synchronization of the pixel and subcarrier clocks. In TV output modes the RIVA128ZX **XTALOUT** pin must be externally driven from the MK2715 reference clock output, with **XTALIN** tied to GND.

The MK2715 requires a number of external components for proper operation. For crystal input a parallel resonant 13.5000MHz crystal is recommended, with a frequency tolerance of 50ppm or better. Capacitors should be connected from X1 and X2 to GND as shown in Figure 67. Alternatively a clock input (e.g. ClockCan) can be connected to X1, leaving X2 disconnected. Further details are given in the MK2715 datasheet [8].

Figure 67. TV output implementation

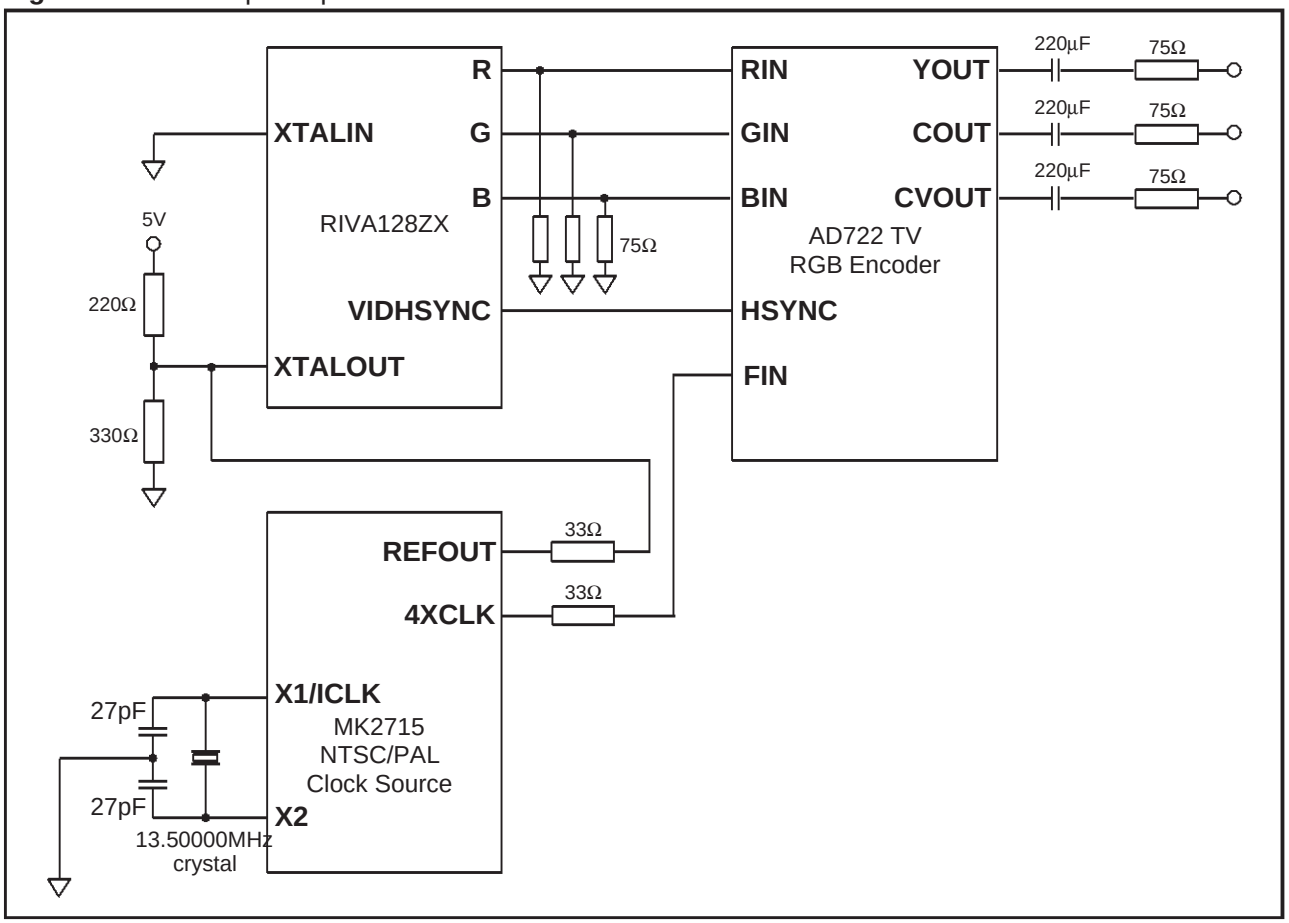
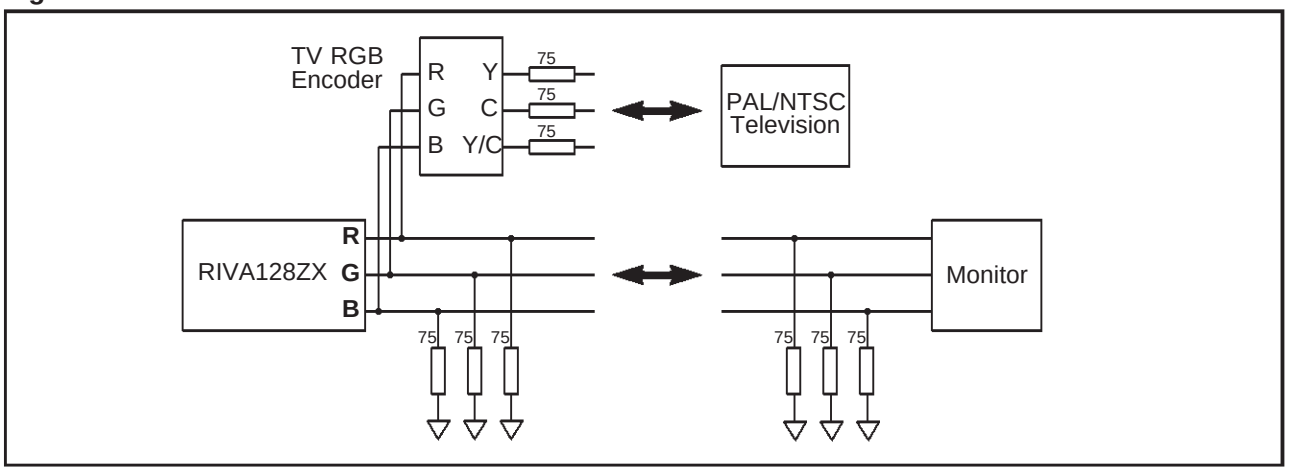


Figure 68. Interface to monitor or television



Monitor detection

Figure 68 shows the typical connection of a television or computer monitor to the RIVA128ZXs' DAC outputs. The RIVA128ZX expects only one output display device to be connected at a time and does not support simultaneous output to both the monitor and television.

During system initialization, the BIOS detects if a monitor is connected by sensing the doubly-terminated 75Ω load (net 37.5Ω). When no monitor is connected, only the local 75Ω load is detected and the RIVA128ZX switches to television output mode. The BIOS sets the CRTC registers to generate the appropriate timing for the local television standard and the DACs are adjusted to compensate for the single 75Ω load.

Monitor mode is always selected if a monitor is detected since it is assumed to be the output device of choice, having a higher display fidelity than television.

Timing generation

Televisions contain two Phase-Locked Loops (PLLs). One PLL locks the horizontal frequency and is used to synchronize horizontal and vertical flyback, and to keep the active video region stable and centered. The second PLL locks the color subcarrier frequency (NTSC 3.5794545MHz or PAL 4.43361875MHz). The color subcarrier is used as a phase reference to extract the color information from the television signal.

The RIVA128ZX encodes horizontal and vertical timing on a composite sync signal. Using a 13.5000MHz reference clock, the RIVA128ZX timing generator creates ITU-R-601 NTSC and PAL compliant horizontal timing with only ppm (parts per million) error. The RIVA128ZX does not use the color subcarrier clock internally. The reference clock source can be located on the television upgrade module with the video encoder and TV output connectors, thus lowering the base system cost.

Flicker filter

RIVA128ZX provides an optional flicker filtering feature for TV and interlaced displays.

Without flicker filtering, elements of an image present on either the odd or the even field, but not both, are seen to flicker or shimmer obtrusively. This is a problem especially with 1-pixel-wide horizontal lines often originating from computer generated GUI displays.

Flicker filtering causes a slight smearing of pixels in the vertical direction. This trades off image quality versus flicker. The displayed pixel contains a proportion of the data for the pixel on that line, plus a smaller proportion of the data of the equivalent pixel on the line above and on the line below. Overall, the proportions add up to 1 so that the brightness of the screen does not alter and the pixel data does not get clipped.

Flicker filtering only takes place on pixel data from the framestore - the pattern written into the cursor already has flicker removed. No flicker removal is performed on video images.

Overscan and underscan

The RIVA128ZX supports overscan and underscan in the horizontal and vertical directions using hardware scaling. Underscan allows 640x480 resolution to fit onto NTSC displays and 800x600 resolution to fit onto PAL displays. Scaling can be adjusted and controlled by software to suit specific TV requirements. The TV output image position is also software controllable.

12 IN-CIRCUIT BOARD TESTING

The RIVA128ZX has a number of features designed to support in-circuit board testing. These include:

- Dedicated test mode input and dual-function test mode select pins selecting the following modes:
  - Pin float
  - Parametric NAND tree
  - All outputs driven high
  - All outputs driven low
- Checksum test
- Test registers

12.1 TEST MODES

Primary test control is provided by the dedicated **TESTMODE** input pin. The RIVA128ZX is in normal operating mode when this pin is deasserted. When **TESTMODE** is asserted, **MP_AD[3:0]** are reassigned as **TESTCTL[3:0]** respectively. Test modes are selected asynchronously through a combination of the pin states shown in Table 21.

Table 21. Test mode selection and descriptions

Test mode	TESTCTL[3:0]				Description
	3	2	1	0	
Parametric NAND tree	1	0	1	0	A single parametric NAND tree is provided to give a quiescent environment in which to test VIL and VIH without requiring core activity. This capability is provided in the pads by chaining all I and I/O paths together via two input NAND gates. The chain begins with one input of the first NAND gate tied to VDD while the other input is connected to the first device pin on the NAND tree. The output of this gate then becomes the input of the next NAND gate in the tree and so on until all pad input paths have been connected. The final NAND gate output is connected to an output-only pin whose normal functionality is disabled in NAND tree mode. The NAND tree length is therefore equal to the number of I and I/O pins in the RIVA128ZX. Output -only pins are not connected into the NAND tree.
Pin float	1	1	0	0	All pin output drivers are tristated in this test mode so that pin leakage current (IIL,IIH,IOZL,IOZH) can be measured.
Outputs high	1	1	1	0	All pin output drivers drive a high output state in this test mode so that output high voltage (VOH at IOH) can be measured.
Outputs low	1	1	1	1	All pin output drivers drive a low output state in this test mode so that output low voltage (VOL at IOL) can be measured.

12.2 CHECKSUM TEST

The RIVA128ZX hardware checksum feature supports testing of the entire pixel datapath at full video rates from the framebuffer through to the DAC inputs. Each of the three RGB colors can be tested to provide a correlation between the intended and actual display. Checksums are accumulated during active (unblanked) display. Note that the checksum mechanism does not check the DAC outputs (i.e. what is physically being displayed on the monitor).

For a given image (which can be a real application's image or a specially prepared test card), theoretically derived checksum values can be calculated for a selected RGB color, which are then compared with the RIVA128ZX hardware checksum value. Alternatively the checksum value from a known good chip can be used as the reference.

Hardware checksum accumulation is not affected by the horizontal and vertical synchronization waveforms or timings. Any discrepancy between the calculated and RIVA128ZX hardware accumulated checksum values therefore indicates a problem in the device or system being tested. Details of programming the RIVA128ZX checksum are given in the RIVA128ZX *Programming Reference Manual* [2].

13 ELECTRICAL SPECIFICATIONS

13.1 ABSOLUTE MAXIMUM RATINGS[1]

Symbol	Parameter	Min.	Max.	Units	Notes
VDD/AVDD	DC supply voltage		3.6	V	
	Voltage on input and output pins	GND-1.0	VDD+0.5	V	2
TS	Storage temperature (ambient)	-55	125	°C	
TA	Temperature under bias	0	85	°C	
	Analog output current (per output)		45	mA	
	DC digital output current (per output)		25	mA	

NOTES

1 Stresses greater than those listed under 'Absolute maximum ratings' may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect reliability.

2 For 3V tolerant pins VDD = 3.3V ± 0.3V, for 5V tolerant pins (PCI, Video Port and Serial interfaces) VDD = 5V ± 0.5V

13.2 OPERATING CONDITIONS

Symbol	Parameter	Min.	Typ.	Max.	Units	Notes
TC	Case temperature			120	°C	

13.3 DC SPECIFICATIONS

Table 22. DC characteristics

Symbol	Parameter	Min.	Typ.	Max.	Units	Notes
VDD	Positive supply voltage	3.135	3.3	3.465	V	
IIN	Input current (signal pins)			±10	µA	1, 2
	Power dissipation			3.7	W	

NOTES

1 Includes high impedance output leakage for all bi-directional buffers with tri-state outputs

2 VDD = max, GND ≤ VIN ≤ VDD

Table 23. Parameters applying to PCI and AGP interface pins

Symbol	Parameter	Min.	Typ.	Max.	Units	Notes
CIN	Input capacitance	5		10	pF	1
COUT	Output load capacitance	5		50	pF	1
Parameters for 5V signaling environment only:						
VIH	Input logic 1 voltage	2.0		5.75	V	
VIL	Input logic 0 voltage	-0.5		0.8	V	
VOH	Output logic 1 level	2.4			V	
VOL	Output logic 0 level			0.55	V	
IOH	Output load current, logic 1 level			-2	mA	
IOL	Output load current, logic 0 level			3 or 6	mA	2
Parameters for 3.3V and AGP signaling environments only:						
VIH	Input logic 1 voltage	0.475VDD		VDD+0.5	V	

Symbol	Parameter	Min.	Typ.	Max.	Units	Notes
VIL	Input logic 0 voltage	-0.5		0.325VDD	V	
VOH	Output logic 1 level	0.9VDD			V	
VOL	Output logic 0 level			0.1VDD	V	
IOH	Output load current, logic 1 level			-0.5	mA	
IOL	Output load current, logic 0 level			1.5	mA	

NOTE

1 Tested but not guaranteed.

2 3mA for all signals except **PCIFRAME#**, **PCITRDY#**, **PCIIRDY#**, **PCIDEVSEL#** and **PCISTOP#** which have IOL of 6mA.

13.4 ELECTRICAL SPECIFICATIONS

Table 24. Parameters applying to all signal pins except PCI/AGP interfaces

Symbol	Parameter	Min.	Typ.	Max.	Units	Notes
CIN	Input capacitance		10	12	pF	1
COUT	Output load capacitance		10	50	pF	1
VIH	Input logic 1 voltage	2.0		VDD+0.5	V	2
VIL	Input logic 0 voltage	-0.5		0.8	V	
VOH	Output logic 1 level	2.4			V	
VOL	Output logic 0 level			0.4	V	
IOH	Output load current, logic 1 level			-1	mA	
IOL	Output load current, logic 0 level			1	mA	

NOTE

1 Tested but not guaranteed.

2 For 3V tolerant pins VDD = 3.3V ± 0.3V, for 5V tolerant pins (Video Port and Serial interfaces) VDD = 5V ± 0.5V

13.5 DAC CHARACTERISTICS

Parameter	Min.	Typ.	Max.	Units	Notes
Resolution		10		bits	
DAC operating frequency			250	MHz	
White relative to Black current	16.74	17.62	18.50	mA	2
DAC to DAC matching		±1	±2.5	%	2,4
Integral linearity		±0.5	±1.5	LSB_8	2,3,8
Differential linearity		±0.25	±1	LSB_8	2,3,8
DAC output voltage			1.2	V	2
DAC output impedance		20		kΩ	
Risetime (black to white level)		1	3	ns	2,5,6
Settling time (black to white)			5.9	ns	2,5,7
Glitch energy		50	100	pVs	2,5
Comparator trip voltage	280	335	420	mV	
Comparator settling time			100	µs	
Internal Vref voltage		1.235		V	
Internal Vref voltage accuracy		±3	±5	%	

NOTES

1 Blanking pedestals are not supported in TV output mode.
2 VREF = 1.235V, RSET = 147Ω
3 LSB_8 = 1 LSB of 8-bit resolution DAC
4 About the midpoint of the distribution of the three DACs
5 37.5ohm, 30pF load
6 10% to 90%
7 Settling to within 2% of full scale deflection
8 Monotonicity guaranteed

13.6 FREQUENCY SYNTHESIS CHARACTERISTICS

Parameter	Min	Typ.	Max	Units	Notes
XTALIN crystal frequency range	4		15	MHz	1
Internal VCO frequency	128		256	MHz	
Memory clock output frequency			100	MHz	
Pixel clock output frequency			250	MHz	2
Pixel clock output frequency (video displayed)			110	MHz	3
Synthesizer lock time			500	µs	

NOTE

1 A series resonant crystal should be connected to **XTALIN**
2 The pixel clock can be programmed to within 0.5% of any target frequency $10 \leq f_{pixclk} \leq 250MHz$
3 The maximum pixel clock frequency when the RIVA128ZX is displaying full motion video

14 PACKAGE DIMENSION SPECIFICATION

14.1 300 PIN BALL GRID ARRAY PACKAGE

Figure 69. RIVA128ZX 300 Plastic Ball Grid Array Package dimension reference

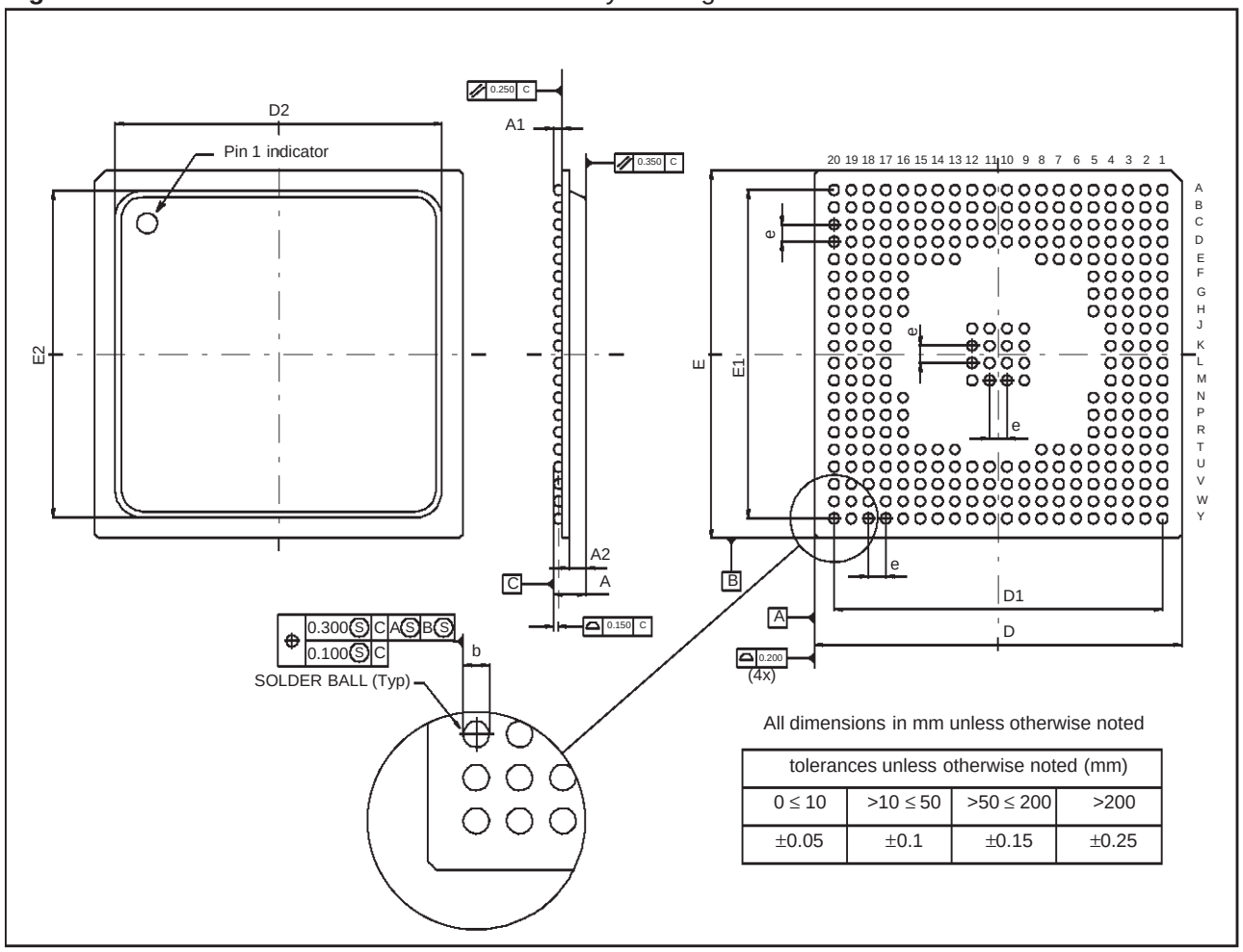


Table 25. RIVA128ZX 300 Plastic Ball Grid Array Package dimension specification

Ref.	Millimeters			Inches		
	Typ.	Min.	Max.	Typ.	Min.	Max.
A		2.125	2.595		0.083	0.102
A1		0.50	0.70		0.020	0.027
A2		1.625	1.895		0.064	0.074
b		0.60	0.90		0.024	0.035
D	27.00	26.82	27.18	1.063	1.055	1.070
D1	24.13 Basic			0.951 Basic		
D2		23.90	24.10		0.941	0.949
e	1.27 Basic			0.050 Basic		
E	27.00	26.82	27.18	1.063	1.055	1.070
E1	24.13 Basic			0.951 Basic		
E2		23.90	24.10		0.941	0.949

15 REFERENCES

1 RIVA128ZX *Turnkey Manufacturing Package TMP, Design Guide*, NVIDIA Corp./STMicroelectronics

2 RIVA128ZX *Programming Reference Manual,* NVIDIA Corp./STMicroelectronics

3 Accelerated Graphics Port Interface Specification, Revision 1.0, Intel Corporation, July 1996

4 *PCI Local Bus Specification, revision 2.1*, PCI Special Interest Group, June 1995

5 Recommendation 656 of the CCIR, Interfaces for digital component video signals in 525-line and 625-line television systems, CCIR, 1990

6 *Display Data Channel (DDC™) standard, Version 2.0, revision 0*, Video Electronics Standards Association, April 9th 1996 (Video Electronics Standards Association - http://www.vesa.org)

7 *AD722 PAL/NTSC TV Encoder Datasheet*, Analog Devices Inc., 1995

8 *MK2715 NTSC/PAL Clock Source Datasheet*, MicroClock Inc., March 1997

16 ORDERING INFORMATION

Device	Package	Supply format	Part number
RIVA128ZX	300 pin PBGA	Trays	STG3005A2S
RIVA128ZX	300 pin PBGA	Tape and reel	STG3005A2S/TR

APPENDIX

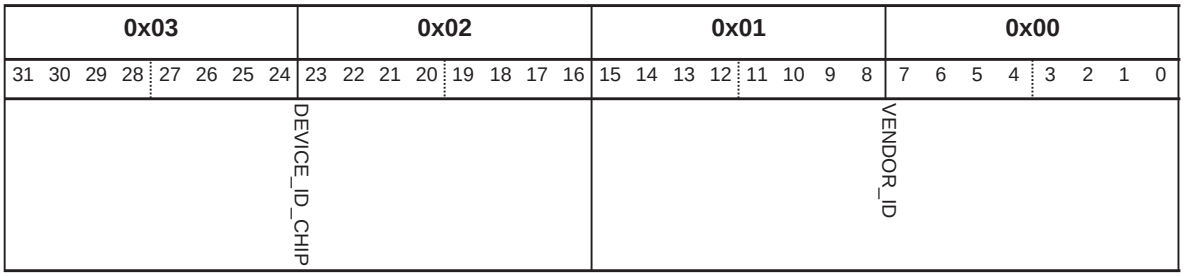
Descriptions of register contents include an indication if register fields are readable **(R)** *or writable* (**W**) *and the initial power-on or reset value of the field* (**I**). *'-' indicates not readable / writable,* X *indicates an indeterminate value, hence* I=X *indicates register or field not reset.*

A PCI CONFIGURATION REGISTERS

This section describes the 256 byte PCI configuration spaces as implemented by the RIVA128ZX. A single PCI VGA device is defined by the RIVA128ZX which decodes and acknowledges the first 256 bytes of the configuration address space. The RIVA128ZX does not respond (does not assert **DEVSEL#**) for functions 1-7.

A.1 REGISTER DESCRIPTIONS FOR PCI CONFIGURATION SPACE

Byte offsets 0x03 - 0x00



Device Identification Register (0x03 - 0x02)

Bits	Function	R W I
31:16	The DEVICE_ID_CHIP bits contain the chip number allocated by the manufacturer to identify the particular device. = 0x0018 if the power-on reset configuration APCI Supported bit = 0 = 0x0019 if the power-on reset configuration APCI Supported bit = 1	R - X

Vendor Identification Register (0x01 - 0x00)

Bits	Function	R W I
15:0	VENDOR_ID bits allocated by the PCI Special Interest Group to uniquely identify the manufacturer of the device. NVIDIA/STMicroelectronics Vendor ID = 0x12D2 (4818)	R - X

Byte offsets 0x07 - 0x04

0x07								0x06								0x05								0x04							
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	SERR_SIGNALLED	RECEIVED_MASTER	RECEIVED_TARGET	Reserved	DEVSEL_TIMING			Reserved		66MHZ	CAP_LIST	Reserved				Reserved							SERR_ENABLE	Reserved		PALETTE_SNOOP	WRITE_AND_INVA:	Reserved	BUS_MASTER	MEMORY_SPACE	IO_SPACE

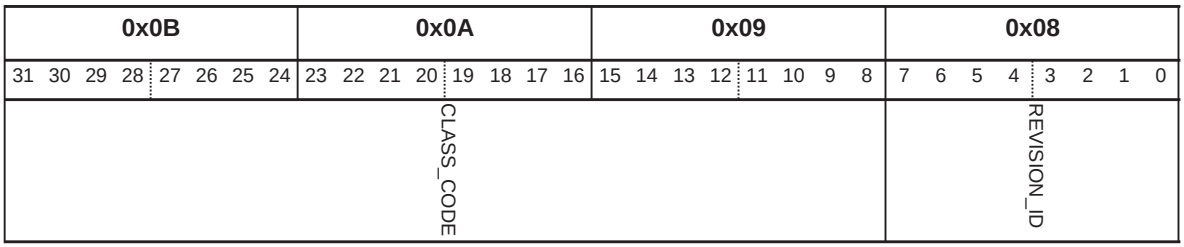
Device Status Register (0x07 - 0x06)

Bits	Function	R W I
31	Reserved	R - 0
30	SERR_SIGNALLED is set whenever the RIVA128ZX asserts SERR#.	R W 0
29	RECEIVED_MASTER indicates that a master device's transaction (except for Special Cycle) was terminated with a master-abort. This bit is clearable (=1). 0=No abort 1=Master aborted	R W 0
28	RECEIVED_TARGET indicates that a master device's transaction was terminated with a target-abort. This bit is clearable (=1). 0=No abort 1=Master received target aborted	R W 0
27	Reserved	R - 0
26:25	The DEVSEL_TIMING bits indicate the timing of **DEVSEL#**. These bits indicate the slowest time that the RIVA128ZX asserts **DEVSEL#** for any bus command except Configuration Read and Configuration Write. The RIVA128ZX responds with medium **DEVSEL#** for VGA, memory and I/O accesses. For accesses to the 16MByte memory ranges described by the BARs, the chip responds with fast decode (no wait states). 00=fast 01=medium	R - 1
24:22	Reserved	R - 0
21	66MHZ indicates that the RIVA128ZX is capable of 66MHz operation. This bit reflects the latched state of the 66MHz/33MHz strap option.	R - 1
20	CAP_LIST indicates that there is a linked list of registers containing information about new capabilities not available within the original PCI configuration structure. This bit indicates that the (byte) Capability Pointer Register located at 0x34 points to the start of this linked list. The value of CAP_LIST depends on the RIVA128ZX power-on reset configuration Host Interface and ACPI Supported settings: 0 = Host Interface is PCI and ACPI not supported 1= Host interface is AGP or ACPI supported	R - X
19:16	Reserved	R - 0

Command Register (0x05 - 0x04)

Bits	Function	R W I
15:9	Reserved	R - 0
8	SERR_ENABLE is an enable bit for the **SERR#** driver. 0=Disables the **SERR#** driver 1=Enables the **SERR#** driver	R W 0
7:6	Reserved	R - 0
5	PALETTE_SNOOP indicates that VGA compatible devices should snoop their palette registers. 0=Palette accesses treated like all other accesses 1=Enables special palette snooping behavior	R W 0
4	WRITE_AND_INVAL is an enable bit for using the Memory Write and Invalidate command. 1=The RIVA128ZX as bus master may generate the command 0=The Memory Write command must be used instead of Memory Write and Invalidate	R W 0
3	Reserved	R - 0
2	BUS_MASTER indicates that the device can act as a master on the PCI bus. 0=Disables the RIVA128ZX from generating PCI accesses 1=Allows the RIVA128ZX to behave as a bus master	R W 0
1	MEMORY_SPACE indicates that the RIVA128ZX will respond to memory space accesses. 0=Device response disabled 1=Enables response to Memory space accesses. The device will decode and respond to the 16MByte ranges as well as the default VGA memory range when it is enabled. The VGA decode range may change based upon the value in the VGA graphics Miscellaneous Register GR06, bits[3:2] and other enable bits, see RIVA128ZX *Programming Reference Manual* [2].	R W 0
0	IO_SPACE indicates that the device will respond to I/O space accesses. This bit enables I/O space accesses for the VGA function as defined in the PCI specification. These include 0x3B0 - 0x3BB, 0x3C0 - 0x3DF and their aliases.	R W 0

Byte offsets 0x0B - 0x08



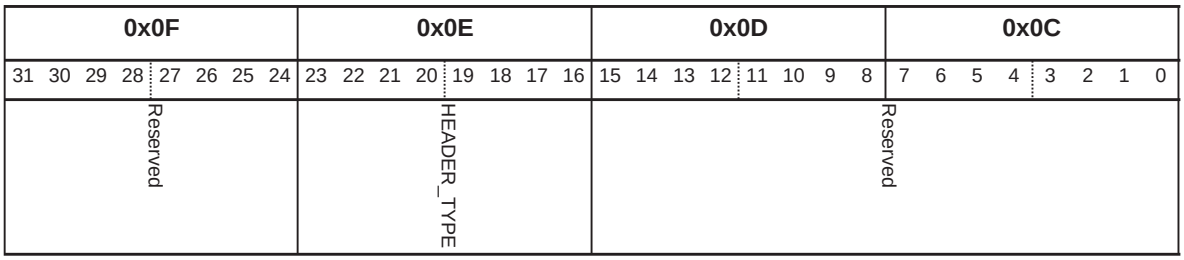
Class Code Register (0x0B - 0x09)

Bits	Function	R W I
31:8	The CLASS_CODE bits identify the generic function of the device and (in some cases) a specific register-level programming interface. The register is broken into three byte-size fields. The upper byte (at offset 0x0B) is a base class code which broadly classifies the type of function the device performs. The middle-byte (at offset 0x0A) is a sub-class code which identifies more specifically the function of the device. The lower byte (at offset 0x09) identifies a specific register-level programming interface (if any) so that device independent software can interact with the device. The VGA function responds as a VGA compatible controller. 0x030000=VGA compatible controller	R - X

Revision Identification Register (0x08)

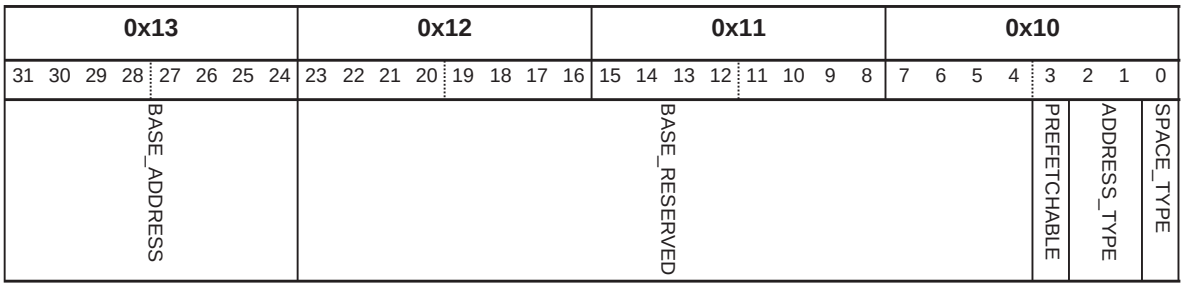
Bits	Function	R W I
7:0	The REVISION_ID bits specify a device specific revision identifier. The value is chosen by the vendor. This field should be viewed as a vendor defined extension to the DEVICE_ID. 0x01=Revision B	R - X

Byte offsets 0x0F - 0x0C



Bits	Function	R W I
31:24	Reserved	R - 0
23:16	HEADER_TYPE identifies the device as single or multi-function. RIVA128ZX responds as a single-function device. 0x00=Single function device	R - 0x00
16:00	Reserved	R - 0

Byte offsets 0x13 - 0x10



Base Memory Address Register (0x13 - 0x10)

Bits	Function	R W I
31:24	The BASE_ADDRESS bits contain the most significant bits of the base address of the device. This indicates that the RIVA128ZX requires a 16MByte block of contiguous memory beginning on a 16MByte boundary. This memory range contains memory-mapped registers and FIFOs and should not be set as part of a PentiumPro™'s write combining range.	R W 0
23:4	The BASE_RESERVED bits form the least significant bits of the base address and are hardwired to 0.	R - 0
3	The PREFETCHABLE bit indicates that there are no side effects on reads, that the device returns all bytes on reads regardless of the byte enables, and that host bridges can merge processor writes into this range without causing errors.	R - 1
2:1	The ADDRESS_TYPE bits contain the type (width) of the Base Address. 0=32-bit	R - 0
0	The SPACE_TYPE bit indicates whether the register maps into Memory or I/O space. 0=Memory space	R - 0

Byte offsets 0x17 - 0x14

0x17								0x16								0x15								0x14							
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BASE_ADDRESS								BASE_RESERVED																				PREFETCHABLE	ADDRESS_TYPE		SPACE_TYPE

Base Memory Address Register (0x17 - 0x14)

Bits	Function	R W I
31:24	The BASE_ADDRESS bits contain the most significant bits of the base address of the device. This indicates that the RIVA128ZX requires a 16MByte block of contiguous memory beginning on a 16MByte boundary. This memory range contains linear frame buffer access and may be set as part of a PentiumPro™'s write combining (wc) range.	R W 0
23:4	The BASE_RESERVED bits form the least significant bits of the base address and are hardwired to 0.	R - 0
3	The PREFETCHABLE bit indicates that there are no side effects on reads, that the device returns all bytes on reads regardless of the byte enables, and that host bridges can merge processor writes into this range without causing errors.	R - 1
2:1	The ADDRESS_TYPE bits contain the type (width) of the Base Address. 0=32-bit	R - 0
0	The SPACE_TYPE bit indicates whether the register maps into Memory or I/O space. 0=Memory space	R - 0

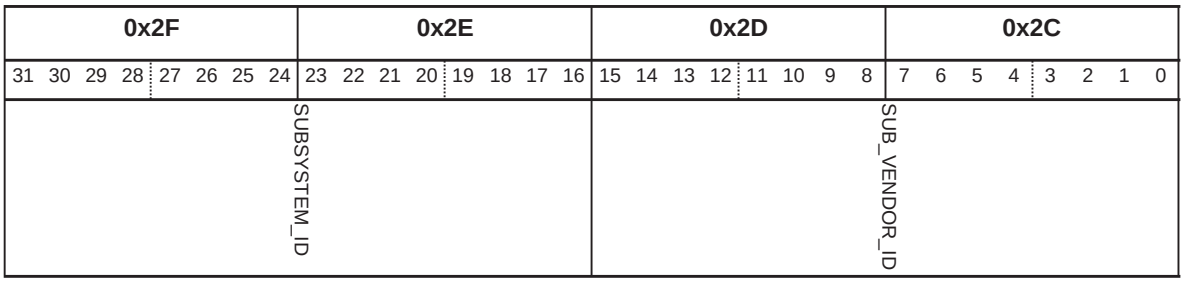
Byte offsets 0x2B - 0x18

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00000000																															

Base Address Registers (0x2B - 0x18)

Bits	Function	R W I
31:0	These bits are hardwired (read-only) to 0.	R - 0

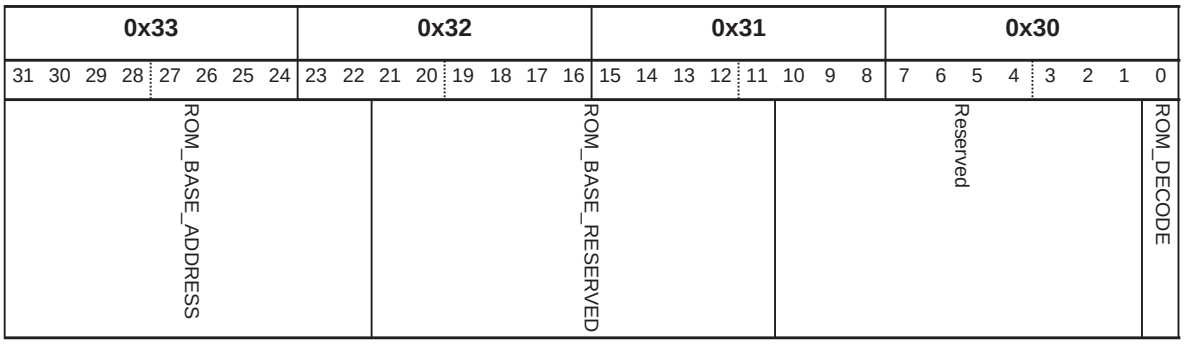
Byte offsets 0x2F - 0x2C



Subsystem Vendor ID (0x2F - 0x2C)

Bits	Function	R W I
31:16	SUBSYSTEM_ID is a unique code defined by the vendor to identify this product.	R - 0
15:0	SUB_VENDOR_ID bits allocated by the PCI Special Interest Group to uniquely identify the manufacturer of the sub-system. Based on the strapping options read from ROM during PCI reset, this field may behave in one of two ways: 1 These bytes can be read from address locations 0x54 - 0x57 of the ROM BIOS automatically during reset. This is useful for add-in card implementations. 2 These bytes may be written from PCI configuration space at locations 0x40 - 0x43.	R - 0

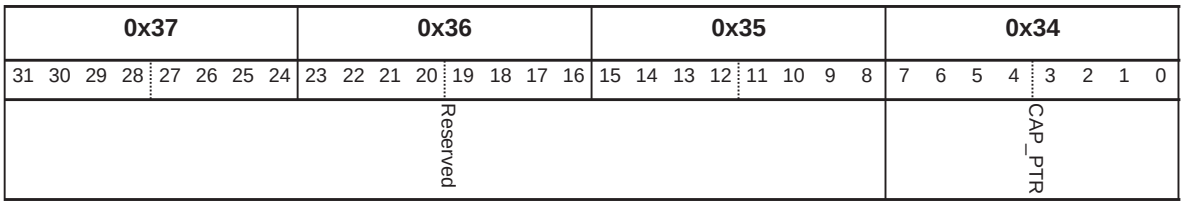
Byte offsets 0x33 - 0x30



Expansion ROM Base Address Register (0x33 - 0x30)

Bits	Function	R W I
31:22	The ROM_BASE_ADDR bits contain the base address of the Expansion ROM. The bits correspond to the upper bits of the Expansion ROM base address. This decode permits the PCI boot manager to place the expansion ROM on a 4MByte boundary. RIVA128ZX currently maps a 64KByte BIOS into the bottom of this 4MByte range. Typically the first 32K of this ROM contains the VGA BIOS code as well as the PCI BIOS Expansion ROM Header and Data Structure.	R W X
21:11	ROM_BASE_RESERVED contain the lower bits of the base address of the Expansion ROM. These bits are hardwired to 0, forcing a 4MByte boundary.	R - 0
10:1	Reserved	R - 0
0	The ROM_DECODE bit indicates whether or not the RIVA128ZX accepts accesses to its expansion ROM. When the bit is set, address decoding is enabled using the parameters in the other part of the base register. The MEMORY_SPACE bit (PCI Configuration Register 0x04, page 70) has precedence over the ROM_DECODE bit. RIVA128ZX will respond to accesses to its expansion ROM only if both the MEMORY_SPACE bit and the ROM_DECODE bit are set to 1. 0=Expansion ROM address space is disabled 1=Expansion ROM address decoding is enabled	R W 0

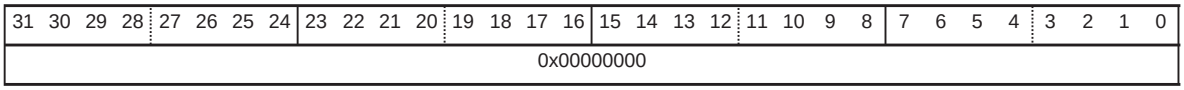
Byte offsets 0x37 - 0x34



Capabilities Pointer Register (0x37 - 0x34)

Bits	Function	R W I
31:8	Reserved	R - 0
7:0	This field contains a byte offset into this PCI configuration space containing the first item in the capabilities list. The offset returned depends on the RIVA128ZX power-on reset configuration Host Interface and ACPI Supported settings: CAP_PTR = 0x0 if Host Interface is PCI and APCI not supported CAP_PTR = 0x44 if Host Interface is AGP and APCI not supported CAP_PTR = 0x60 if APCI supported	R - X

Byte offsets 0x3B - 0x38



Reserved (0x3B - 0x38)

Bits	Function	R W I
31:0	These bits are reserved and hardwired (read-only) to 0.	R - 0

Byte offset 0x3F - 0x3C

0x3F								0x3E								0x3D								0x3C							
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MAX_LAT								MIN_GNT								INTERRUPT_PIN								INTERRUPT_LINE							

MAX_LAT Register (0x3F)

Bits	Function	R W I
31:24	The MAX_LAT bits contain the maximum time the RIVA128ZX requires to gain access to the PCI bus. This read-only register is used to specify the RIVA128ZX's desired settings for Latency Timer values. The value specifies a period of time in units of 250ns. 1=250ns	R - 1

MIN_GNT Register (0x3E)

Bits	Function	R W I
23:16	The MIN_GNT bits contain the length of the burst period the RIVA128ZX needs, assuming a clock rate of 33MHz. This read-only register is used to specify the RIVA128ZX's desired settings for Latency Timer values. The value specifies a period of time in units of 250ns. 3=750ns	R - 3

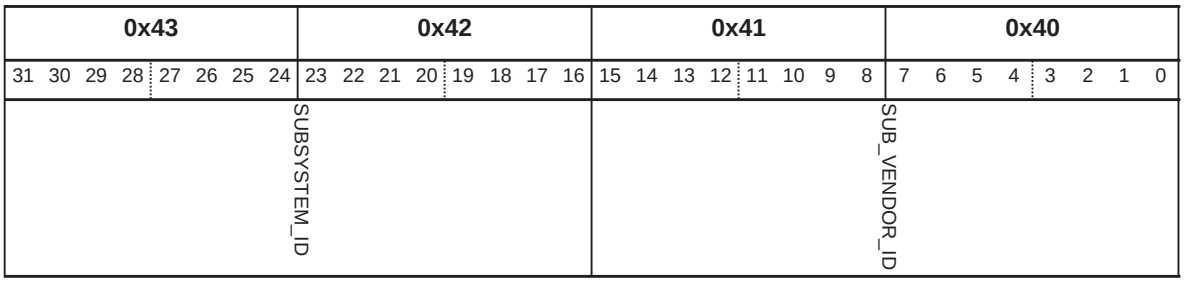
Interrupt Pin Register (0x3D)

Bits	Function	R W I
15:8	The INTERRUPT_PIN bits contain the interrupt pin the device (or device function) uses. A value of 1 corresponds to **INTA#**.	R - 1

Interrupt Line Register (0x3C)

Bits	Function	R W I
7:0	The INTERRUPT_LINE bits contain the interrupt routing information. POST software will write the routing information into this register as it initializes and configures the system. The value in this field indicates which input of the system interrupt controller(s) the RIVA128ZX's interrupt pin is connected to. Device drivers and operating systems can use this information to determine priority and vector information. INTERRUPT_LINE is initialized to 0xFF (no connection) at reset. 0=Interrupt line IRQ0 1=Interrupt line IRQ1 0xF=Interrupt line IRQ15 0xFF=No interrupt line connection (reset value)	R W 0xFF

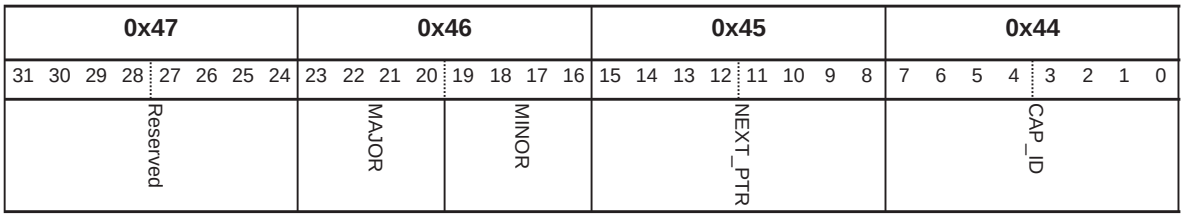
Byte offsets 0x43 - 0x40



Writeable Subsystem Vendor ID (0x43 - 0x40)

Bits	Function	R W I
31:16	This SUBSYSTEM_ID field is aliased at 0x2F - 0x2E where it is read-only. It may be modified by System BIOS for systems which do not have a ROM on the RIVA128ZX data pins. This will ensure valid data before enumeration by the operating system.	R W 0
15:0	This SUB_VENDOR_ID field is aliased at 0x2D - 0x2C where it is read-only. It may be modified by System BIOS for systems which do not have a ROM on the RIVA128ZX data pins. This will ensure valid data before enumeration by the operating system.	R W 0

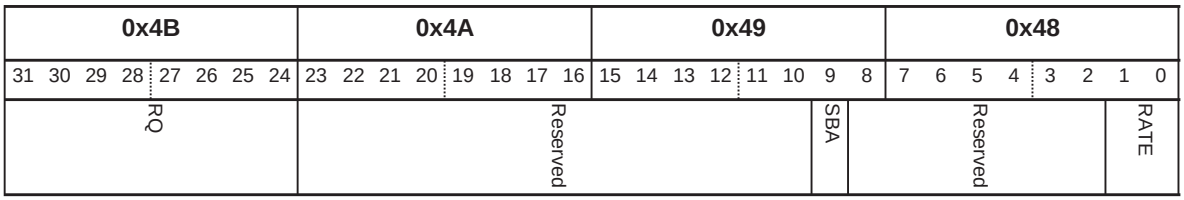
Byte offsets 0x47 - 0x44



Capabilities Identifier Register (Offset = 0x47 - 0x44 = CAP_PTR)

Bits	Function	R W I
31:24	Reserved = 0x00	R - 0
23:20	This field indicates the Major revision number of the AGP specification that the RIVA128ZX conforms to. = 0x01	R - 0x01
19:16	This field indicates the Minor revision number of the AGP specification that the RIVA128ZX conforms to. = 0x00	R - 0x00
15:8	NEXT_PTR contains the pointer to the next item in the capabilities list. This is the last entry in the capabilities list, hence it contains a null pointer = 0x00.	R - 0x00
7:0	The CAP_ID field identifies the type of capability. AGP = 0x02	R - 0x02

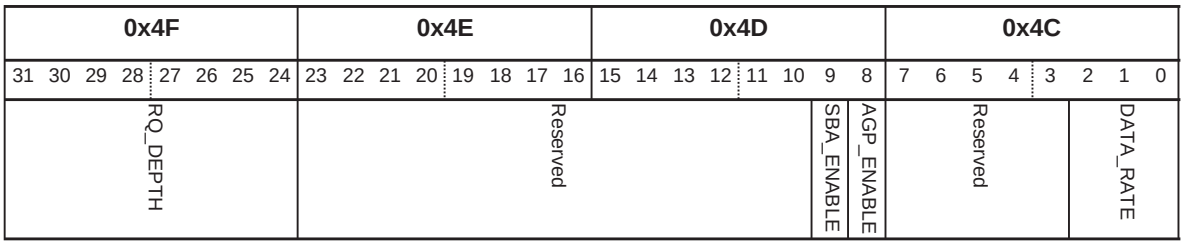
Byte offsets 0x4B - 0x48



AGP Status Register (0x4B - 0x48 = CAP_PTR+4)

Bits	Function	R W I
31:24	The RQ field contains the maximum number of AGP command requests this device can have outstanding. RQ = 0x04	R - 0x04
23:10	Reserved	R - 0
9	SBA indicates whether the RIVA128ZX supports sideband addressing. 0 = Sideband addressing not supported	R - 0
8:2	Reserved	R - 0
1:0	RATE indicates the data transfer rate(s) supported by the RIVA128ZX. 11 = 66MHz 1x supported; 133MHz 2x supported	R - 0x11

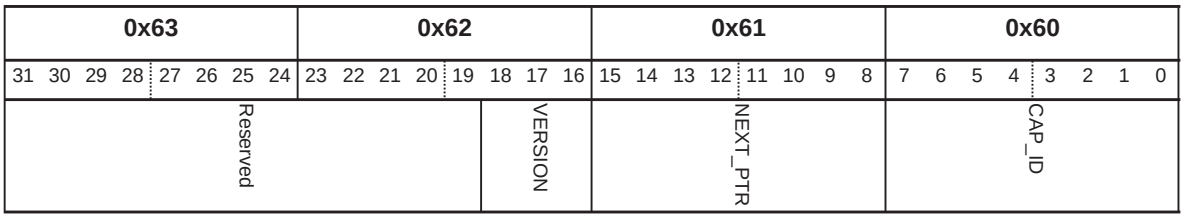
Byte offsets 0x4F - 0x4C



AGP Command Register (0x4F - 0x4C = CAP_PTR + 8)

Bits	Function	R W I
31:24	This field is set to the minimum request depth of the target as reported in its RQ field.	R W -
23:10	Reserved	R - 0
9	SBA_ENABLE enables sideband addressing when set. The RIVA128ZX does not implement sideband addressing.	R - 0
8	AGP_ENABLE allows the RIVA128ZX to act as an AGP master and initiate AGP operations. The target must be enabled before enabling the RIVA128ZX 0 = disabled 1 = AGP operations enabled	R W 0
7:3	Reserved	R - 0
2:0	The DATA_RATE field must be set to 0x01 to indicate 66MHz/1x transfer mode or 0x02 for 133MHz/2x transfer mode. This value must also be set on the target before being enabled. The initial value 0x00 indicates PCI operation.	R W 0x00

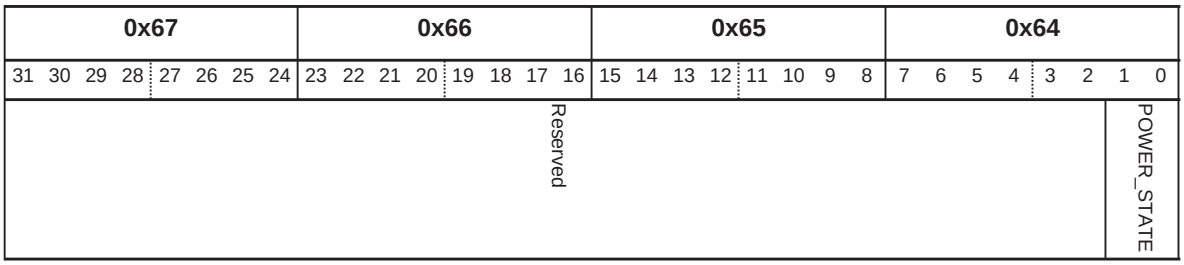
Byte offsets 0x63 - 0x60



Power Management Capabilities Register (0x63 - 0x60)

Bits	Function	R W I
31:19	Reserved	R - 0
18:16	VERSION indicates that RIVA128ZX is compliant with Revision 1.0 of the PCI Power Management Interface Specification.	R - 1
15:8	NEXT_PTR indicates the offset for the next item in the capability list. If the power-on reset configuration Host Interface bit indicates PCI then NEXT_PTR will be null (0x00), indicating there are no further items. If the power-on reset configuration indicates AGP then NEXT_PTR will indicate the next item's offset (the AGP capability, 0x44).	R - X
7:0	A value of '1' read back from CAP_ID indicates this is the Power Management Register.	R - 1

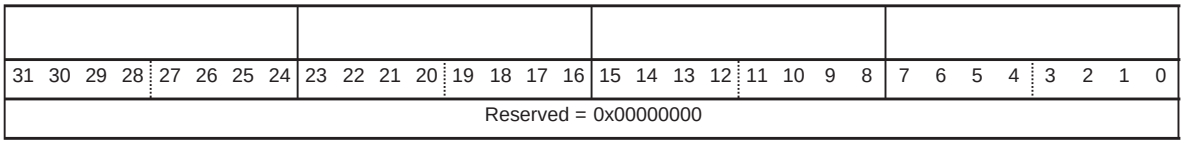
Byte offsets 0x67 - 0x64



Power Management Control/Status Register (0x67 - 0x64)

Bits	Function	R W I
31:2	Reserved	R - 0
1:0	POWER_STATE indicates and controls the current power state of RIVA128ZX. Two power states are supported D0 (full power) and D3hot (low power). 0x0 = D0 0x3 = D3hot The RIVA128ZX does not physically change its power consumption when POWER_STATE is modified. The device can be transitioned from D3 to D0 either by writing to this register or by applying a hard reset to the **PCIRST#** pin.	R W 0

Byte offset 0xFF - 0x50



STMicroelectronics GROUP OF COMPANIES

Australia - Brazil - Canada - France - Germany - Italy - Japan - Korea - Malaysia - Malta - Mexico - Morocco - The Netherlands - Singapore - Spain - Sweden - Switzerland - Taiwan - Thailand - United Kingdom - U.S.A.

Document number: 7071857 00