WEITEK

# POWER 9100 GRAPHICS CONTROLLER

## PRELIMINARY DATA

November 1993

The WEITEK Power 9100 is a high-performance display controller for use with graphical user interfaces such as Microsoft Windows. It combines an extremely high-speed frame buffer controller with a workstation-style accelerated display controller and a local-bus host interface for maximum performance.

## Contents

Power 9100 Data Book (Preliminary Data)
November 1993



WEITEK Corporation
1060 East Arques Avenue
Sunnyvale, California 94086
Telephone (408) 738–8400



Written by Claire-Marie Costanza
Edited by Robert Plamondon
Additional writing by Robert Plamondon, D.R. Sevedge, and Allen Samuels.
Illustrated by Claire-Marie Costanza, Robert Plamondon, Allen Samuels, and D.R. Sevedge

Printed in the United States of America
93 94 6 5 4 3 2 1

4710–9302–00 Rev. A

# Contents

## Chapter 4. Native Mode Registers, 29



## Chapter 5. Commands, 59



## Chapter 6. Host Interface, 67

# Chapter 1. Technical Overview

### *Single-Chip 2-D Graphics Accelerator*

◇ Ultra-high-speed local-bus display controller uses workstation display technology to give maximum acceleration with graphical user interfaces such as Microsoft Windows

◇ Sophisticated architecture, full-custom chip design, and advanced 0.8 micron process technology deliver maximum power at an affordable cost

◇ On-chip SVGA unit for DOS compatibility

◇ 208-pin plastic QFP package

### *High Bandwidth*

◇ Interleaved VRAM controller uses a PLL timing generator to extract 100% of VRAM bandwidth

◇ Sophisticated local-bus controller extracts full bandwidth from PCI and VL host buses

◇ Supports 32- and 64-bit RAMDACs at speeds up to 200 MHz for maximum display bandwidth

◇ Pipelined internal architecture is 32 bits wide throughout to ensure maximum throughput

### *Full Software Support*

◇ Drivers for Windows 3.1, Windows NT, OS/2 Presentation Manager, and AutoCAD R10–R12

◇ VESA-compatible BIOS

◇ DOS application drivers

### *Powerful Graphics Features*

◇ Supports high- and true-color at large screen sizes:

- 24-bit true-color to 1280x1024 pixels
- 16-bit high-color to 1600x1200 pixels

◇ Fully accelerates 8-, 15-, 16-, 24- and 32-bits/pixel

◇ Draws lines and polygons (with optional pattern fill) at full VRAM bandwidth

◇ Performs bit block-transfer (BitBlt) from screen to screen at VRAM bandwidth, and from host to screen (with color expansion) at host bus bandwidth

◇ Provides automatic clipping against window and screen edges

◇ Supports patterning, plane masking, and boolean operations of pixels during drawing; supports polylines and mesh polygons; supports pick mode

◇ Directly mapped linear frame buffer allows efficient CPU access to frame-buffer memory without banking

◇ Multimedia support via frame-buffer coprocessor interface

◇ Supports VESA power management

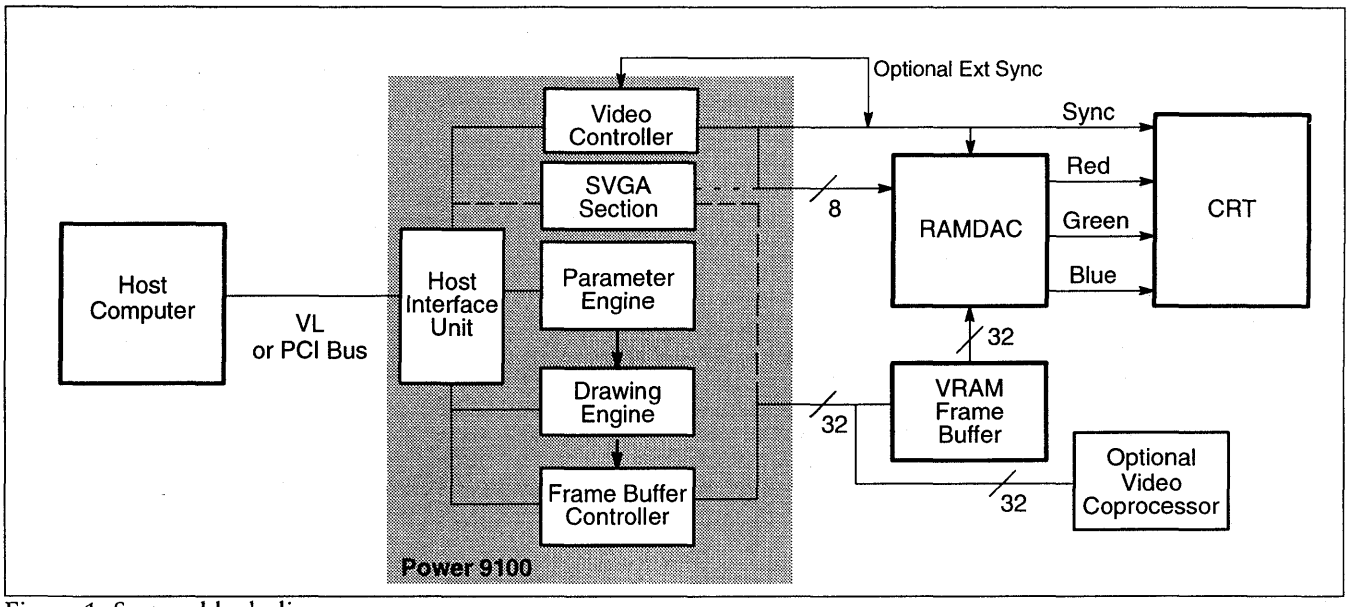



Figure 1. System block diagram

## 1.1. Architecture

The WEITEK Power 91000 is a hard-wired display processor. Designed specifically for high-performance personal computers, its architecture takes full advantage of the speed of video RAM and high-speed local-bus interfaces.

Based on WEITEK's workstation display controllers, the Power 9100 is the latest in a series of controllers designed from an architectural strategy that can be summarized in two sentences:

1. Get the highest possible bandwidth.
2. Use all of it.

We have found that these simple statements capture the essence of high-performance controller design.

The following sections describe how these principles were applied in the main sections of the Power 9100:

1. Frame buffer controller
2. Parameter engine
3. Drawing engine
4. Host bus interface
5. Video controller

The Power 9100 contains an additional unit:

6. SVGA unit

for backward compatibility.

### 1.1.1. FRAME BUFFER CONTROLLER

Capturing the highest possible bandwidth starts at the frame buffer. The Power 9100 uses three techniques to achieve (and use) the highest possible frame buffer bandwidth:

1. VRAM frame buffer
2. Interleaving
3. PLL-controlled memory timing

*VRAM FRAME BUFFER*

The Power 9100 uses a VRAM frame buffer for maximum performance. While VRAM is more expensive than DRAM, it is dual-ported; the stream of pixel data needed to refresh the display does not detract from the drawing rate. DRAM systems, in contrast, divide their bandwidth between these two functions, causing drawing rates to degrade as screen sizes increase.

The Power 9100 supports screen sizes of up to four megapixels, and full drawing bandwidth is retained even at these large screen resolutions.

*INTERLEAVING*

The Power 9100's two 32-bit banks of VRAM are interleaved for maximum performance.

Interleaving is a technique that takes advantage of the fact that RAM access times are much longer than their write-enable times. Two banks of RAM are placed in parallel, with common address and data lines, but separate write enable lines. One bank contains the even-numbered data words; the other contains the odd-numbered words. With interleaved RAM, writing two words of data only takes one cycle longer than writing one, giving the performance of 64-bit memory, while saving pins and giving a better architectural match to a 32-bit controller.

*PLL-CONTROLLED MEMORY TIMING*

To get maximum speed out of memory devices, you have to precisely match a large number of timing specifications. Typical memory controllers approximate this with timing patterns based on the system clock. Since memory speeds and system clock speeds are similar, this approach is too granular to match the memory parameters closely, and it results in significant performance loss.

The Power 9100 can match memory parameters exactly, using a PLL (phase-locked loop) timing generator and a programmable memory controller. This allows the frame buffer to be used at its full theoretical capacity.

### 1.1.2. PARAMETER ENGINE

The graphics core of the Power 9100 is crucial to using all of the frame buffer's bandwidth. In addition to providing a low-latency path between the host and the frame buffer, the core must provide large amounts of acceleration if the full frame buffer bandwidth is to be used. The importance of acceleration stems from three factors:

1. Local-bus interfaces, while fast, are not as fast as the Power 9100's frame buffer.
2. General-purpose CPUs are inefficient at simple drawing operations, and cannot perform them at full host-bus bandwidth (let alone full frame-buffer bandwidth).
3. The user is better served if the CPU is running applications, not serving as an inefficient graphics controller.

## 1.1. Architecture, continued

The Power 9100's graphics accelerator is divided into two loosely coupled graphics engines: the *parameter engine* and the *drawing engine*. The *parameter engine* prepares drawing operations for execution by the drawing engine (which is described in the next section). The parameter engine's basic function is to take input coordinates from the host and convert them to a form usable by the drawing engine. The input parameters include the *x,y* vertices of polygons and the corners of bit block-transfer *(BitBlt)* regions. The parameter engine tests the vertices against window and screen boundaries, tests for exceptions, and performs trivial rejection. Finally, it transfers commands that pass these tests to the drawing engine to be rendered into the frame buffer.

The parameter engine works independently of the drawing engine; the parameters for a new operation can be loaded while the drawing engine is busy.

The parameter engine prepares four kinds of "polygons" for drawing: quadrilaterals, triangles, lines, and points." It also handles screen-to-screen BitBlt and host-to-screen BitBlt.

The parameter engine handles all exception testing, trivial rejection, status reporting, and access to parameter engine registers. Once the parameter engine verifies that a drawing command should be performed (that is, it has valid parameters and has not been trivially rejected), it passes the operation to the drawing engine. The parameter engine's exception testing is very fast, completing in a few cycles.

### 1.1.3. DRAWING ENGINE

The drawing engine performs three basic functions:

1. It draws quadrilaterals, triangles, and lines (the *quad* operation).
2. It performs screen-to-screen BitBlt (the *blit* operation).
3. It performs host-to-screen BitBlt (the *pixel1* and *pixel8* operations).

The quad operation draws quadrilaterals, triangle, lines, and points (the last three being treated by the hardware as special cases: quadrilaterals with one or more identical vertices). Triangles, lines, and points can always be rendered correctly, but the drawing engine cannot draw horizontally convex quadrilaterals. That is, it cannot cross from the inside to the outside of the same object more than once per scan line. This means that "bow ties" cannot be drawn (such quads are decomposed into two triangles by the driver software), though "hourglasses" can. See figure 2.

The blit operation copies a rectangular area of the display from one screen location to another.

The pixel1 operation takes monochrome, one-bit-per-pixel data from the host, expands the pixels internally to the current pixel depth (typically by assigning the foreground and background colors to one and zero bits), and writes them to the frame buffer. Up to 32 pixels can be transferred to the Power 9100 in a single word.

The pixel8 operation takes color pixels of 8, 16, or 32 bits, packed as four, two, or one pixel per word, respectively, and writes them to the frame buffer.

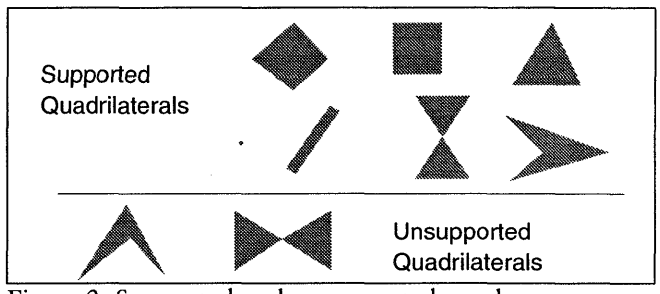



Figure 2. Supported and unsupported quads

### 1.1.4. HOST BUS INTERFACE

The Power 9100 connects directly to PCI and VESA local buses. With small amounts of glue logic it can be connected directly to processor buses.

The Power 9100 itself runs asynchronously to its bus clock. It supports both big-endian and little-endian address formats.

The Power 9100 is memory-mapped; each command has a unique memory address. In many cases, this means that command and data are given in a single write operation: the address specifies what operation is to be performed on the data being transferred.

## 1.1. Architecture, continued

### 1.1.5. VIDEO CONTROLLER

A typical board design feeds the VRAM shift registers into a RAMDAC, which converts digital pixel data into an analog RGB video signal. The host initializes the control registers and look-up tables in the RAMDAC through the Power 9100's RAMDAC access instructions.

The Power 9100 also generates horizontal and vertical synchronization and blanking signals, and controls the clocking of the video data. The timing of these signals is programmed through the video controller registers. The Power 9100's divided dot clock is completely asynchronous to the Power 9100's main system clock.

The Power 9100 supports 32- and 64-bit RAMDACs, such as the IBM RGB525, the Brooktree Bt 484 and Bt 485, and the Texas Instruments TVP3010 and TVP3020. By supporting RAMDACs with wide input buses at speeds of up to 200 MHz, the Power 9100 has no difficulty supporting large screens at high color depths and ergonomic refresh rates.

### 1.1.6. SVGA UNIT

For compatibility with older applications, the Power 9100 contains an on-chip SVGA unit which supports screen sizes to 1280x1024, and color depths of up to 24 bits. This SVGA unit is independent of the main graphics engine of the Power 9100, although the two share bus, frame buffer, and video interfaces. Control can be switched between the two graphics units under software control.

## 1.2. Major Differences Between the Power 9100 and the Power 9000

The Power 9100 builds upon the strengths of its predecessor, the Power 9000. This section summarizes the most important differences between the two.

### 1.2.1. ADDITIONAL FEATURES

1. On-chip VESA Local (VL) Bus interface
2. On-chip PCI Bus interface
3. On-chip SVGA unit
4. Native and SVGA modes
5. Full acceleration of 16- through 32-bit graphics
6. Higher clock speeds
7. Four-color pattern RAM (but reduced from 16x16 to 8x8) can be used for four-color dither at full speed
8. Aperture memory mapping (frame buffer can be mapped into a 64 KB aperture in addition to its usual 4 MB linear mapping)
9. Video coprocessor support
10. ROM BIOS control logic
11. Configuration EEPROM control logic
12. Clock generator control logic
13. Serial clock and serial enable generation for VRAMs
14. Text transparency
15. 256 raster-ops

### 1.2.2. ADDRESSING

Power 9100 addressing and general address formats are different from those on the Power 9000.

## 1.3. Details of Graphics Operation

### 1.3.1. THE GRAPHICS PIPELINE

Graphics operations flow through the *graphics pipeline,* first through the parameter engine, and then through the stages of the drawing engine.

#### *THE PARAMETER ENGINE*

*Functions.* The parameter engine determines what will happen as the result of each drawing operation. It calculates status information, including whether the operation is clipped by the viewing window or the screen edge, and whether the operation can be drawn at all before passing the operation to the drawing engine. If the drawing engine is busy, or if the request contains illegal parameters, the parameter engine does not pass on the request. (The parameter engine performs these tests only on coordinate register loads and drawing commands. Other operations, such as setting color registers or video timing registers, bypass the parameter engine.)

The parameter engine performs clipping calculations on each *x,y* vertex. It compares these points against all four edges of the screen and viewing window, and against each other. It also tests for trivial rejection and trivial acceptance.

The parameter engine detects, but cannot correct an illegal request, such as a request for a horizontally convex quad. Such quadrilaterals must be rendered in software. The status register flags such problems. In addition, the interrupt signal can be used to interrupt the host when such exceptions occur.

*Access.* All accesses to parameter engine functions and registers complete in a few cycles; the parameter engine is always accessible.

#### *THE DRAWING ENGINE*

*Functions.* The drawing engine accepts one drawing operation at a time from the parameter engine. When processing a drawing operation, it first determines which pixels are "touched" by the drawing process *(scan conversion)*, then determines the color value of each touched pixel *(raster ops)*.

*Scan conversion.* Two line-drawing engines follow the left and right edges of the quadrilateral on each scan line, and a pixel-processing engine fills the region between these edges. (Pixel1, pixel8, and blit operations are limited to rectangular areas, while the quad operation can have edges at any angle).

*Raster ops.* The color of each touched pixel is determined by the raster-op function, which is further conditioned by the contents of other registers (see section 1.3.5).

*Access.* The drawing engine can remain busy for long periods when drawing large quads or performing a blit operation on a large portion of the screen. The quad and blit operations are started by a read operation. The read requests that the operation take place, and returns the contents of the status register, which indicates whether or not the request has been granted. The Power 9100 does not accept a request if the drawing engine is busy, or if an exception has occurred, nor does it queue requests. Therefore, the software must check the status register, which contains both exception and drawing engine status bits, and resubmit any quad or blit request that is not accepted. While no new drawing operations can be started until the drawing engine is idle, the host can load the parameter engine's coordinate registers with the vertices of a new operation while the drawing engine is busy, thereby overlapping the processing of two objects.

### 1.3.2. DRAWING QUADRILATERALS

The drawing engine assigns the two edges at either the topmost or bottom-most vertex to its two Bresenham *line-drawing engines.* These engines do not actually draw anything in the frame buffer, but they traverse the boundaries of the quadrilateral on each scan line. The *pixel-processing engine* then fills in the pixels between the boundaries found by the line-drawing engines. (This filling operation is also clipped against the clipping window and screen boundaries.) See figure 3. When a line-drawing engine reaches another vertex, it starts down the new edge.

In *oversized* mode, the Power 9100 draws perimeter pixels according to the Bresenham algorithm. Oversized mode must be selected to draw points and lines, as X11's drawing rules do not "touch" pixels in zero-width objects (meaning nothing is drawn).

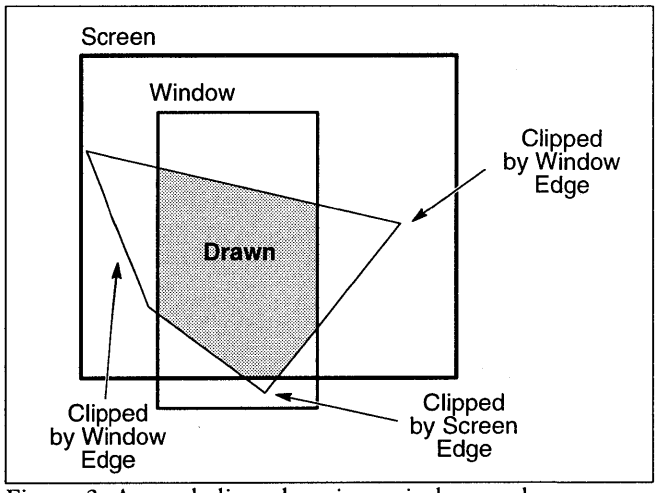



Figure 3. A quad clipped against window and screen

## 1.3. Details of Graphics Operation, continued

In *X11 mode,* the Power 9100 draws only those pixels whose centers lie within the perimeter of the quad. Pixels whose centers lie precisely on the perimeter are drawn in accordance with X11's tie-breaker rules. (See figure 4.) Coordinates can either be specified relative to the clipping window or relative to the previous vertex.

Polylines, meshed triangles, and meshed quadrilaterals are all supported; they are drawn by transferring the new vertices and issuing another draw command.

### 1.3.3. DRAWING BIT MAPS

To transfer a bit map to the screen, the host first sets up the *x,y* coordinates of the destination, and then transfers data to the Power 9100 with the pixel1 and pixel8 operations. The Power 9100 draws the pixels on the screen, auto-incrementing the current *x,y* location after each pixel. Bit-map drawing proceeds from left to right; when it reaches the right-hand edge of the target bit map, the Power 9100 automatically wraps to the next line.

The pixel8 operation draws colored pixels; up to four eight-bit pixels (or two 16-bit pixels, or one 32-bit pixel) are transferred and drawn through a single bus transfer from the host. This mode is used to transfer color images.

The pixel1 operation, which draws monochrome bit maps, is ideal for text transfer. Its basic operation is similar to pixel8, but instead of transferring, say, four eight-bit pixels per operation, it transfers up to 32 one-bit pixels. The Power 9100 expands the one and zero bits of the data word into pixels at the current color depth. (The Power 9100 expands these values according to the raster-op function described in section 1.3.5.)

Pixel data must be padded out to multiples of 32 bits per scan line for pixel8 transfers; leftover pixels in the current word do not wrap to the next line. Pixel1 allows the left-over pixels to wrap, however, making it especially suitable for sending narrow blocks of monochrome data, such as character bit maps.

### 1.3.4. BIT BLOCK TRANSFER

The blit operation moves a rectangular block of pixels from one part of the screen to another. It handles overlapping source and destination blocks properly; the source block arrives unchanged at the destination, as if it were moved off-screen, then copied back at its new destination.

Like quad, blit is a "fire-and-forget" operation; once initiated, it runs to completion without additional attention from the host. As a large blit can involve over a million pixels, the host driver code should test the busy bit in the Power 9100's status register before attempting another drawing operation when a blit could be in progress.

### 1.3.5. COLOR SELECTION

Once a pixel has been touched, there still remains the question of what color it will be. Screen color is selected at five levels:

1. the initial (source) color
2. the pattern color
3. the raster-op color
4. the plane mask
5. the RAMDAC look-up table color (8-bit modes only; higher pixel depths use direct color)

The Power 9100 applies each of these in turn. The *pattern RAM* allows imposition of an 8x8-bit repeating pattern on the data. See figure 5. The *raster-op* function is a three-input boolean function controlled by an eight-bit minterm array. The three inputs are:

1. The source color
2. The destination color (the color value currently at the *x,y* location being written)
3. The color registers color[3..0].

The Power 9100 applies the same function to each of the bits in the pixel.

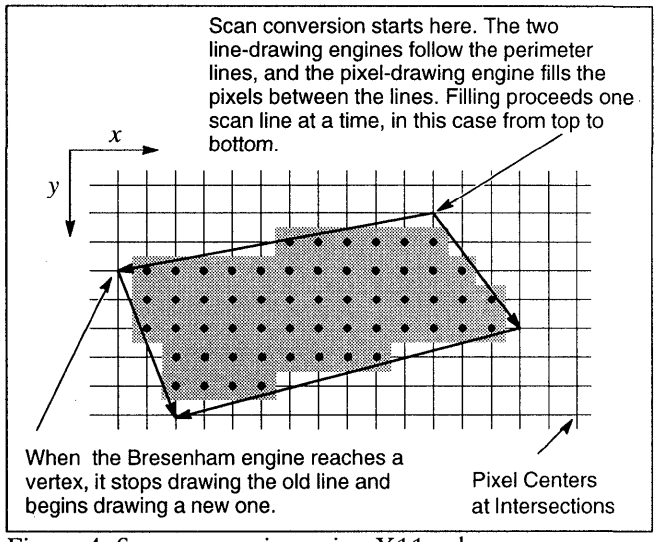



Figure 4. Scan conversion using X11 rules

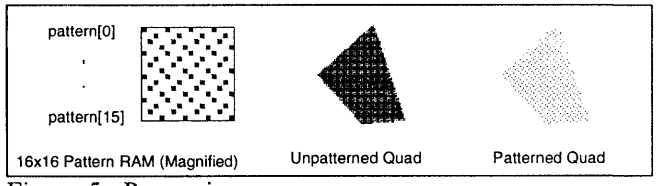



Figure 5. Patterning

## 1.4. Host Bus Interface

The Power 9100 supports the VL and PCI buses directly, with no glue logic. See figures 6 through 9. Other buses can be accommodated with a small amount of external glue logic.

Some pins on the Power 9100 change function depending on which bus is selected. Rather than introduce a confusing third signal nomenclature to that of PCI and VL, we have provided two pin configuration diagrams: one for each bus, each using that bus' signal naming conventions. See section 13.4 for the pin configuration.

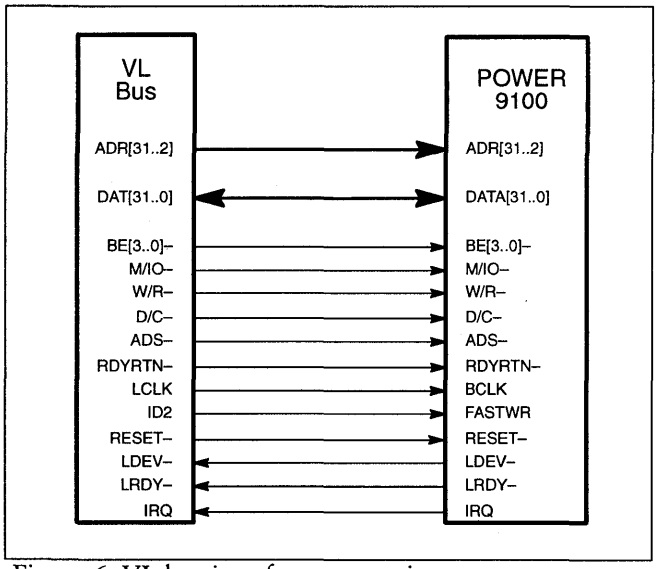



Figure 6. VL bus interface connection

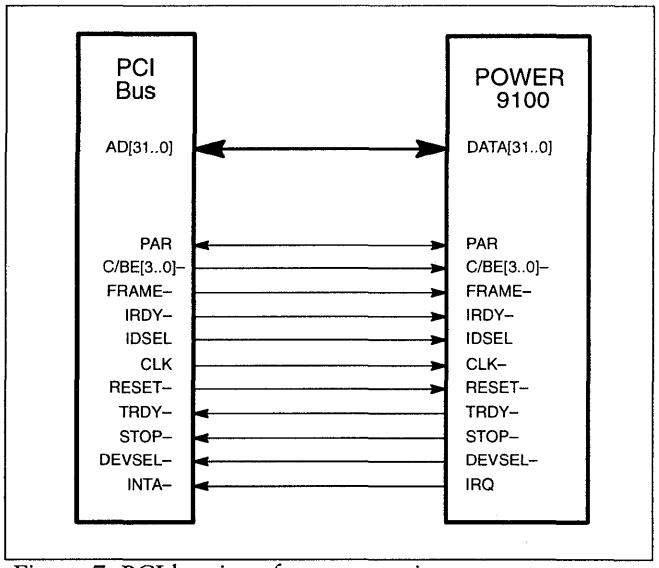



Figure 7. PCI bus interface connection

| VL Bus | | |
|---|---|---|
| **Signal** | **I/O** | **Description** |
| DATA[31..0] | I/O | Data bus |
| ADR[31..2] | Input | Address bus |
| BE[3..0]– | Input | Byte enable |
| W/R– | Input | Write or read status |
| M/IO– | Input | Memory or I/O status |
| FASTWR | Input | ID2 identifier |
| ADS– | Input | Address data strobe |
| LDEV– | Output | Local device |
| LRDY– | Output | Local ready |
| RDYRTN– | Input | Ready return |
| IRQ | Output | Interrupt request |
| BCLK | Input | VL clock |
| RESET– | Input | Reset |
| D/C– | Input | Data or code status |

Figure 8. VL bus interface signals

| PCI Bus | | |
|---|---|---|
| **Signal** | **I/O** | **Description** |
| DATA[31..0] | I/O | Address and data bus |
| C/BE[3..0]– | Input | Bus command/byte enable |
| PAR | I/O | Parity |
| IDSEL | Input | Initialization device select |
| STOP– | Output | Stop |
| FRAME– | Input | Cycle frame |
| DEVSEL– | Output | Device select |
| TRDY– | Output | Target ready |
| IRDY– | Input | Initiator ready |
| IRQ | Output | Interrupt request |
| CLK– | Input | PCI clock |
| RESET– | Input | Reset |

Figure 9. PCI bus interface signals

## 1.5. Frame Buffer and Video Interfaces

Figure 10 illustrates Power 9100 RAMDAC, frame buffer, and video pin connections.

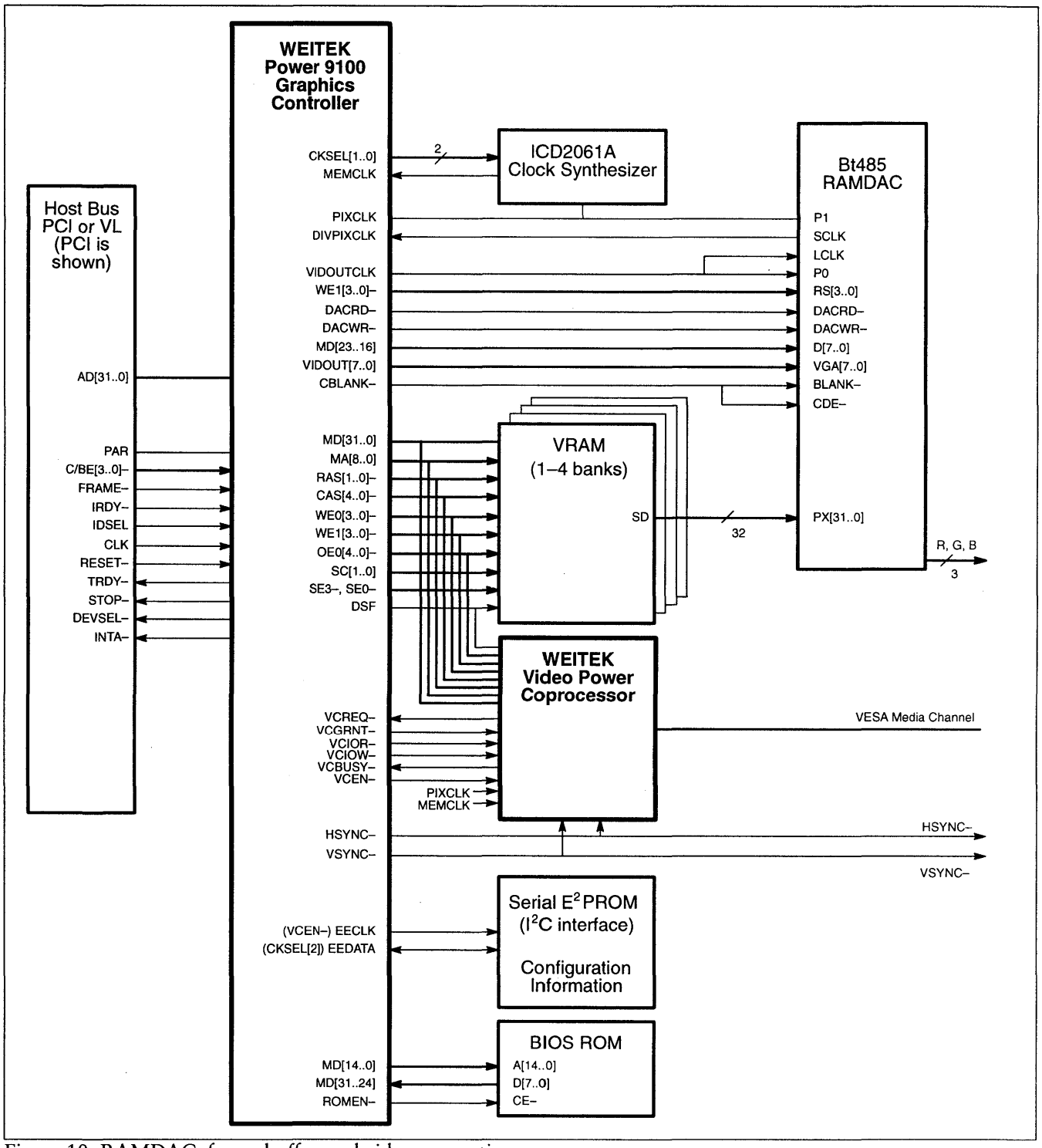



Figure 10. RAMDAC, frame buffer, and video connections

## 1.5. Frame Buffer and Video Interfaces, continued

### 1.5.1. SIGNAL DESCRIPTIONS

| Signal | Type | Description |
|---|---|---|
| MD[31..0] | I/O | Memory data bus |
| MA[8..0] | Output | Memory address bus |
| RAS[1..0]– | Output | Row address strobes |
| CAS[4..0]– | Output | Column address strobes. CAS[0]– controls the least-significant 16 bits of bank 0; CAS[4]– controls the most-significant bits of bank 0 (necessary for VGA transfers). Note that in two-bank systems, bank 1 is controlled by CAS[3]–, not CAS[1]– |
| WE0[3..0]– | Output | Write-enable for the individual bytes in bank 0 (2-bank systems), or banks 0–1 (4-bank systems) |
| WE1[3..0]– | Output | Write-enable for the individual bytes in bank 1 (2-bank systems), or banks 2–3 (4-bank systems) |
| OE[4..0]– | Output | Output Enables. OE[0]– controls the least-significant 16 bits of bank 0; OE[4]– controls the most-significant bits of bank 0 (necessary for VGA transfers) |
| DSF | Output | VRAM special function pin |
| DACRD– | Output | RAMDAC control line |
| DACWR– | Output | RAMDAC write control signal |
| MEMCLK | Input | Main chip clock |
| ROMEN– | Output | BIOS ROM enable |
| EECK | Output | EEPROM clock control signal |
| EEDA | I/O | EEPROM data control signal |
| VDDPLL | Input | Supply voltage for on-chip clock generator |
| VSSPLL | Output | Ground for on-chip clock generator |

Figure 11. Memory control signals

| Signal | Type | Description |
|---|---|---|
| SE[3]–, SE[0]– | Output | VRAM serial shift enable |
| SC[1..0] | Output | VRAM serial shift clock |
| VIDOUT[7..0] | Output | Video data out (SVGA modes) |
| VIDOUTCLK | Output | Video data clock out |
| HSYNC– | I/O | Horizontal synchronization. Normally an output; can also be used as an input for external sync |
| VSYNC– | I/O | Vertical synchronization. Normally an output; can also be used as an input for external sync |
| BLANK– | Output | Blanking interval |
| PIXCLK | Input | Pixel clock |
| DIVPIXCLK | Input | Divided pixel clock |
| CKSEL[2..0] | Output | Frequency synthesizer control |

Figure 12. Video control signals

## 1.5. Frame Buffer and Video Interfaces, continued

| Signal | Type | Description |
|---|---|---|
| VCBUSY– | Input | Video coprocessor busy. Video coprocessor is busy; I/O reads and writes will not succeed. This signal should be pulled up to VCC with a 10 KΩ resistor |
| VCEN– | Output | Video coprocessor enable. When asserted, video coprocessor is active. Resets video coprocessor when de-asserted |
| VCGRNT– | Output | Video coprocessor bus grant. Signals that the Power 9100 has released the frame buffer |
| VCIOR– | Output | Video coprocessor I/O read. Requests a read from a video coprocessor register |
| VCIOW– | Output | Video coprocessor I/O write. Requests a write to a video coprocessor register |
| VCREQ– | Input | Video coprocessor bus request. Requests that the video coprocessor be given control of the bus. This signal should be pulled up to VCC with a 10 KΩ resistor |

Figure 13. Video coprocessor signal description

## 1.6. Video Coprocessor Interface

The video coprocessor interface allows a separate coprocessor to share the Power 9100's frame buffer and host interface. Such a coprocessor could provide features to accelerate still pictures, video animation, or 3-D rendering.

The video coprocessor interface allows the host to read and write data from the video coprocessor, while the video coprocessor grant and release functions allow the video coprocessor to take and relinquish control of the frame buffer. The video coprocessor pre-empt function lets the Power 9100 regain control to perform high-priority tasks such as memory refresh.

## 1.7. Related Documents

For availability of Power 9100 documentation, see your WEITEK sales representative. Upcoming titles include:

*Power 9100 Programmer's Reference Manual.* Full information on registers and commands for both P9100 and SVGA modes.

*Power 9100 Application Notes.* Practical techniques for Power 9100 design, including examples of complete board designs for both VL and PCI.

*Power 9100 Manufacturing Kits.* Artwork, programmable logic equations, and manufacturing drawings for tested, cost-effective Power 9100 designs.

# Chapter 2. Quick Reference

*[Editor's Note: Will necessarily be completed last.]*

## 1.2. Major Differences between the Power 9100 and the Power 9000, continued

| Registers (# of regs) | Power 9000 or Power 9100 | New/Changed/ Replaced | Notes |
|---|---|---|---|
| Configuration (8) | Power 9100 | New | These registers specify device and vendor identifiers; enable the RAMDAC pallet, memory space access, and I/O space access; specify base addresses; enable VGA; and select P9100 or SVGA mode. |
| Color (color[0] through color[3]) | Power 9100 | New | color[0] replaces Power 9000 foreground register; color[1] replaces Power 9000 background register. Both are used in two-color patterns. Color[2] and color[3] are used in new four-color patterns. Each register is 32 bits wide |
| Pattern (pattern[0] through pattern[3] | Power 9100 | New | Replace Power 9000 pattern registers. (Each register is 32 bits wide.) |
| alt_read_bank alt_write_bank | Power 9100 | New | These registers specify the high-order address bits (in register bits 22–14) when alternate aperture frame buffer banking logic is used to read/write directly from the frame buffer |
| Software (4) | Power 9100 | New | These registers are reserved for the system software to use. |
| SVGA | Power 9100 | New | The Power 9100 has all of the registers found on the W5286 on its SVGA unit; they are identical to the W5286 registers. |
| System configuration (sysconfig) | Both | Changed | New field: bits 28–26 specify 8, 16, or 32 bits per pixel for the drawing engine<br>Deleted field: version (bits 2–0), which is now contained in a configuration register; bits 2–0 are always zero |
| Memory configuration (mem_config) | Both | Changed | New fields:<br>bits 31–29 are reserved<br>bits 28–27 control the delay for blank generation<br>bit 26 inverts the MUXSEL pin<br>bits 25–24 set the timing patterns for shift clocks<br>bits 23–22 set timing patterns for shift clocks as well<br>bits 21–19 set the frequency of the DAC load clock<br>bit 18 controls the skew between the DAC load clock and the VRAM shift clocks<br>bits 17–14 set the frequency of the internal crtc divided dot clock (DDOTCLK)<br>bits 13–10 set the frequency of the VRAM shift clock state machine<br>bit 9 forces memory and video into reset state<br>bit 8 is reserved<br>bit 7 enables RAMDAC back-to-back transfer checking<br>bit 6 specifies that VCP has higher priority than the drawing engine<br>bit 5 specifies an adjustment for VRAM write timing<br>bit 4 specifies an adjustment for VRAM read timing<br>bit 3 specifies an adjustment for VRAM row miss timing<br>bit 2 specifies the number of bits for the CAS section of the VRAMs<br>bits 1–0 specify the number of VRAM banks connected |

Figure 14. Register differences between the Power 9100 and the Power 9000

## 1.2. Major Differences between the Power 9100 and the Power 9000, continued

| Register(s) | Power 9000 or Power 9100 | New/Changed/ Replaced | Notes |
|---|---|---|---|
| Raster | Both | Changed | New fields:<br>bit 15 enables pixel1 transparent mode<br>bit 14 specifies pattern depth (2-color or 4-color pattern)<br><br>Changed fields:<br>bit 17 enables transparent pattern<br>bits 7–0 minterms (bits 15–0 on Power 9000)<br>bits 12–8 now always zero |
| Foreground (fground)<br>Background (bground) | Power 9000 | Replaced | Replaced by new Power 9100 color registers. |
| Pattern | Power 9000 | Replaced | Replaced by new Power 9100 pattern registers. |

Figure 14, continued. Register differences between the Power 9100 and the Power 9000

# Chapter 3. Memory Map

## 3.1. Overview

The host controls the Power 9100 by setting registers, loading pseudo-registers, and issuing commands. Power 9100 *registers*, all of which are 32 bits wide, contain the parameters for Power 9100 operations and maintain status information. *Pseudo-registers* minimize the number of parameters that must be specified per drawing operation. *Commands* initiate the drawing operations whose parameters are defined by register and pseudo-register settings.

The Power 9100 is memory-mapped: The address is divided into control fields that determine the actions to be performed. The sections of memory reserved for Power 9100 addresses must be marked as non-cacheable; because the Power 9100 alters its own register contents, memory caching would give invalid data.

The Power 9100 performs all tasks sequentially. The host must ensure that requests for drawing operations are accepted, either by checking Power 9100 status to verify that the previous drawing operation has completed before issuing a new request, or, for operations that return status, by checking the status of a requested operation before issuing a new request. The status register contents reflect status conditions, as defined in section 4.4.2. The host processor is also responsible for proper byte alignment of data it sends across the bus to the Power 9100.

This chapter describes all Power 9100 address formats and registers. Subsequent chapters describe the Power 9100 registers, pseudo-registers, and commands, and define the specific address format for accessing each. For illustrations and examples of Power 9100 operation, see Chapter 5.

### 3.1.1. BIG-ENDIAN AND LITTLE-ENDIAN MODES

The Power 9100 data bus connects to the host bus so that transfer of a 32-bit word preserves the significance of the data bits (bit 0 is the least significant bit and bit 31 is the most significant bit). Because the Power 9100 stores all data in big-endian mode, it is necessary to convert the mode when transferring data to and from a host system that uses little-endian mode. The H, B, and b bits enable swapping of half-words, bytes, and bits respectively. This is used most often when accessing the frame buffer. Figure 15 illustrates the swapping effect of setting the H, B, and b bits. The H, B and b bits are derived from different registers depending on the address format used.

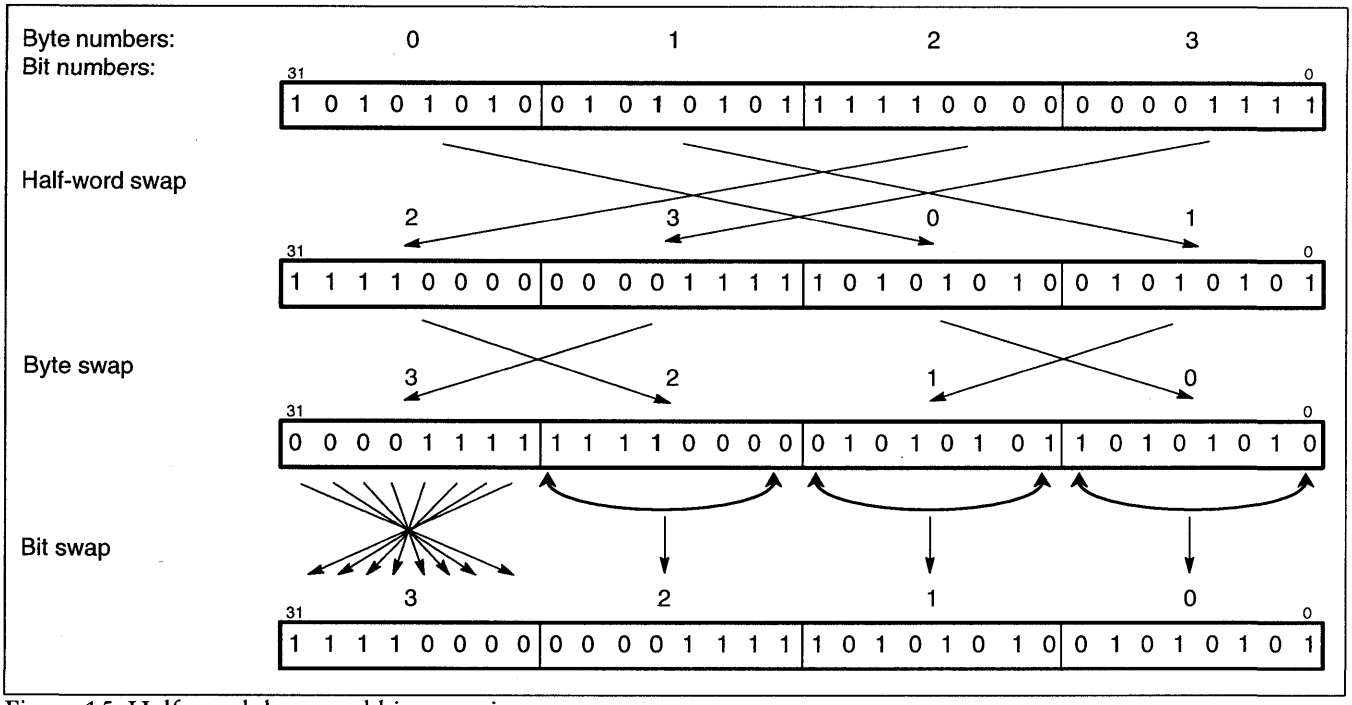



Figure 15. Half-word, byte, and bit swapping

## 3.2. Conventions and Notation

The conventions and notation defined in this section are used throughout this chapter.

All discussions of registers and address formats define all 32 bits. Illustrations appear as shown in the example in figures 16 and 17, with bit numbers identified across the top of the illustration and field sizes called out below. The field contents are identified in the illustration and defined in a table that accompanies the illustration.

The prefix "c." on a field name indicates that the most-significant bit in the field is a field write control bit. When writing to a register, it is not always desirable to replace the entire register contents. For example, clearing a bit in the interrupt register is a function that needs to be done without altering the information contained in other register fields. When writing to a register, setting a field write control bit to 1 replaces that field with the new data; setting a field write control bit to 0 leaves the contents of the field unaltered. On a register read, field write control bits are always 1, which makes it easy to save and restore all of the fields of a register.

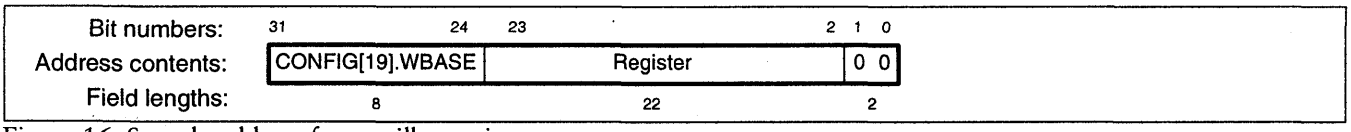


Figure 16. Sample address format illustration

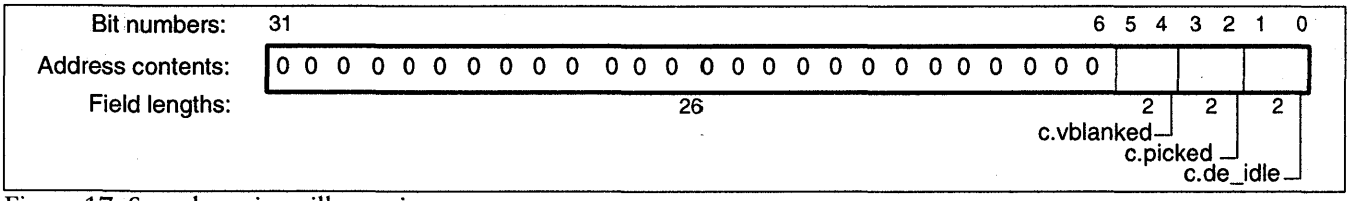


Figure 17. Sample register illustration

## 3.3. Configuration Registers

The PCI bus specification requires a group of configuration registers be present in the configuration address space. These registers are used to control memory and I/O mapping as well as various other control functions. The Power 9100 fully supports the requirements of the PCI specification.

When operating in the VL bus mode the same functionality is supported by mapping the configuration registers into the I/O address space. Two double words of I/O address space are used to access the configuration registers. The first is used as an index into the configuration address space, the second is used as the data transfer register. These configuration registers are present independent of the mode (VGA or native) that the device is currently operating in. The index registers can be placed at several locations in the I/O address space. The actual location of these registers is controlled by the setting of the PU_CONFIG.CFGBA field. The value of this field is determined during the reset sequence (see section 3.3.1).

Configuration registers that are not implemented must follow the absent PCI configuration register convention – ignore all writes, read as all zeroes. When accessed through I/O space all operations must be a single byte, no assembly or disassembly of data is supported.

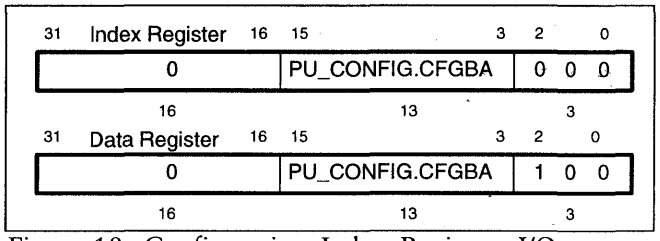
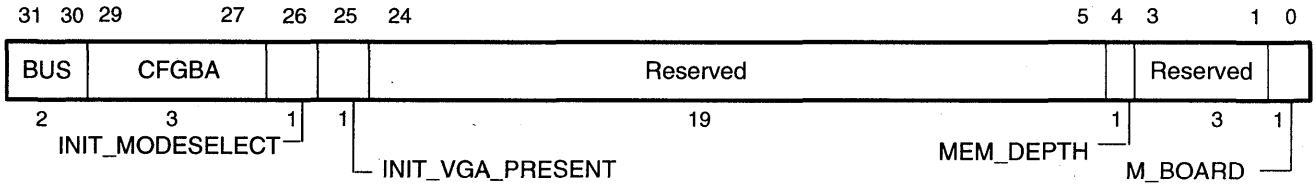



Figure 18. Configuration Index Registers I/O space mapping (VL bus mode)

### 3.3.1. POWER-UP CONFIGURATION

The deassertion of reset– forces the system to sample the frame buffer data bus to determine its power-up configuration. Data bus drivers have built-in pull-down resistors causing the bus to eventually float to a low state. By placing large (~10KΩ) pull-up resistors on the data bus selected bits can be forced into the high state. The initial settings of the data bus is preserved in the read–only PU_CONFIG register.

**Power-up Configuration Bits (PU_CONFIG register)**




| Field | Bits | Contents |
|---|---|---|
| BUS | 31–30 | The bus signalling protocol.<br>11 = Reserved<br>10 = VESA Local Bus<br>01 = PCI Bus<br>00 = Reserved (internal Testing mode) |
| CFGBA | 29–27 | I/O address for configuration register index and data registers.<br>000 = $9100, $9104 001 =$9108, $910C<br>010 = $9110, $9114 011 = $9118, $911C<br>100 = $9120, $9124 101 =$9128, $912C<br>110 = $9130, $9134 111 = $9138, $913C |
| INIT_MODESELECT | 26 | The initial value for CONFIG[65].MODESELECT |
| INIT_VGA_PRESENT | 25 | The initial value for CONFIG[10].VGA_PRESENT |
| MEM_DEPTH | 4 | Depth of memory Chips.<br>0 = 256K 1 = 128K |
| MBOARD | 0 | Motherboard VGA address decode control.<br>0 = On motherboard 1 = On add–in board |

Figure 19. Power-up Configuration bits. (PU_CONFIG register)

## 3.3. Configuration Registers, continued

### 3.3.2. CONFIG[0] REGISTER

**CONGIG[0] Register (Read Only)**

| 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 |

| Field | Bits | Definition |
|---|---|---|
| | 7–0 | Low order byte of Vendor ID (0E, $0E, or 0x0E) |

Figure 20. CONFIG[0] register (Read Only)

### 3.3.3. CONFIG[1] REGISTER

**CONFIG[1] Register (Read Only)**

| 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |

| Field | Bits | Definition |
|---|---|---|
| | 7–0 | High order bye of Vendor ID (10, $10, or 0x10) |

Figure 21. CONFIG[1] register (Read Only)

### 3.3.4. CONFIG[2] REGISTER

**CONFIG[2] Register (Read Only)**

| 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Field | Bits | Definition |
|---|---|---|
| | 7–0 | Low order bye of Vendor ID ($00) |

Figure 22. CONFIG[2] register (Read Only)

Device?

### 3.3.5. CONFIG[3] REGISTER

**CONFIG[3] Register (Read Only)**

| 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |

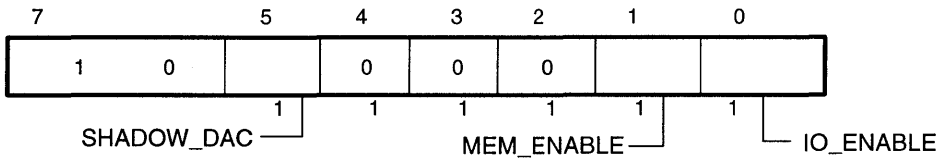
| Field | Bits | Definition |
|---|---|---|
| | 7–0 | High order byte of Device ID (91) |

Figure 23. CONFIG[3] register (Read Only)

## 3.3. Configuration Registers, continued

### 3.3.6. CONFIG[4] REGISTER

**CONFIG[4] Configuration Register**

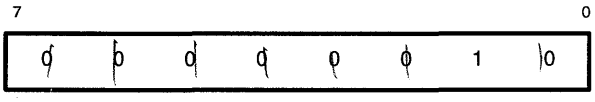



| Field | Bits | Contents |
|---|---|---|
| SHADOW_DAC | 5 | Shadow access to the RAMDAC palette.<br>0 = shadow disabled, Power 9100 responds to palette accesses.<br>1 = shadow enabled, Power 9100 does not respond to palette write accesses (default). |
| MEM_ENABLE | 1 | Memory space enable.<br>0 = disabled, Power 9100 doesn't respond to any memory accesses. (default)<br>1 = enabled, Power 9100 responds to memory space accesses. |
| IO_ENABLE | 0 | I/O space enable.<br>0 = disabled, Power 9100 does not respond to any I/O space accesses. (PCI default)<br>1 = enabled, Power 9100 responds to I/O space accesses. (VL default) |

Figure 24. CONFIG[4] configuration register

### 3.3.7. CONFIG[7] REGISTER

**CONFIG[7] Register (Read Only)**

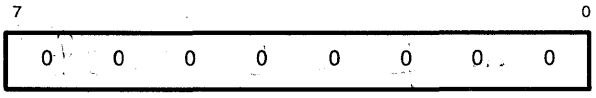



| Field | Bits | Definition |
|---|---|---|
| | 7–0 | Status Register bits |

Figure 25. CONFIG[7] register (Read Only)

### 3.3.8. CONFIG[8] REGISTER

**CONFIG[8] Register (Read Only)**




| Field | Bits | Definition |
|---|---|---|
| | 7–0 | Revision ID. Current revision ID. |

Figure 26. CONFIG[8] register (Read Only)

## 3.3. Configuration Registers, continued

### 3.3.9. CONFIG[10] REGISTER

**CONFIG[10] Register (Read Only)**

| 7 | 6 | | | | | | 0 |
|---|---|---|---|---|---|---|---|
| vga_present | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Field | Bits | Definition |
|---|---|---|
| VGA_PRESENT | 6 | 0 = VGA present in emulation mode<br>1 = VGA absent in emulation mode<br>This bit is set by PU_CONFIG[25] on reset (see figure 19). |

Figure 27. CONFIG[10] register (read only)

### 3.3.10. CONFIG[11] REGISTER

**CONFIG[11] Register (Read Only)**

| 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |

| Field | Bits | Definition |
|---|---|---|
| | 1–0 | Display controller.<br>These bits set to 11 in accordance with PCI 2.0. |

Figure 28. CONFIG[11] register (read only

### 3.3.11. CONFIG[19] REGISTER

**CONFIG[19] Register**

| 7 | 0 |
|---|---|
| WBASE | |

| Field | Bits | Definition |
|---|---|---|
| WBASE | 7–0 | Base for Native mode. Memory space address is relocatable every 16 MB.<br>VL bus – set through I/O read/write<br>PCI bus – set through configuration space read/write<br>See figures 37 and 42. |

Figure 29. CONFIG[19] register

## 3.3. Configuration Registers, continued

### 3.3.12. CONFIG[48] REGISTER

**CONFIG[48] Register**

| 7 | | | | | | 1 | 0 |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | rom_enable |

| Field | Bits | Definition |
|---|---|---|
| ROM_ENABLE | 0 | 0 = BIOS ROM decoding disabled (default for PCI)<br>1 = BIOS ROM decoding enabled (default for VL) |

Figure 30. CONFIG[48] register

### 3.3.13. CONFIG[49] REGISTER

**CONFIG[49] Register**

| 7 | 6 | | | | | | 0 |
|---|---|---|---|---|---|---|---|
| rom_base[0] | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Field | Bits | Definition |
|---|---|---|
| ROM_BASE[0] | 7 | Bit 0 of the ROM base address (default is 0 after reset) |

Figure 31. CONFIG[49] register

## 3.3. Configuration Registers, continued

### 3.3.14. CONFIG[50] REGISTER

**CONFIG[50] Register**

| 7 | 0 |
|---|---|
| rom_base[8..1] | |

| Field | Bits | Definition |
|---|---|---|
| ROM_BASE[8..1] | 7–0 | Bits 8..1 of the ROM base address (default is 0x0C after reset) |

Figure 32. CONFIG[50] register

### 3.3.15. CONFIG[51] REGISTER

**CONFIG[51] Register**

| 7 | 0 |
|---|---|
| rom_base[16..9] | |

| Field | Bits | Definition |
|---|---|---|
| ROM_BASE[16..9] | 7–0 | Bits 16..9 of the ROM base address (default is 0x00 after reset) |

Figure 33. CONFIG[51] register

### 3.3.16. CONFIG[64] REGISTER

**CONFIG[64] Register (Read Only)**

| 7 6 | 5 3 | 2 1 | 0 |
|---|---|---|---|
| BUS | CFGBA | 0 0 | EEDAIN |
| 2 | 3 | 2 | 1 |

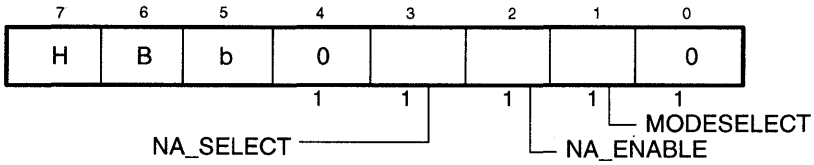
| Field | Bits | Contents |
|---|---|---|
| BUS | 7–6 | Read-only copy of PU_CONFIG.BUS<br>11 = Reserved<br>10 = VESA Local Bus<br>01 = PCI Bus<br>00 = Reserved (internal Testing mode) |
| CFGBA | 5–3 | Read–only copy of PU_CONFIG.CFGBA<br>000 =$9100, $9104 001 = $9108, $910C<br>010 = $9110, $9114 011 = $9118, $911C<br>100 = $9120, $9124 101 = $9128, $912C<br>110 = $9130, $9134 111 = $9138, $913C |
| EEDAIN | 0 | Current state of CKSEL[2] pin. |

Figure 34. CONFIG[64] register (Read Only)

## 3.3. Configuration Registers, continued

### 3.3.17. CONFIG[65] REGISTER

**CONFIG[65] Register**

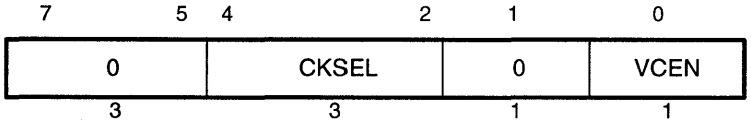



| Field | Bits | Contents |
|---|---|---|
| MODESELECT | 1 | Native/emulation mode.<br>This bit is set by PU_CONFIG[26] on reset (see figure 19).<br>0 = Power 9100 is in native mode.<br>1 = Power 9100 is in emulation (VGA) mode. |
| NA_ENABLE | 2 | Native mode. Respond to address range 000A0000–000AFFFF.<br>This bit is set to 0 (default) on reset.<br>0 = disabled.<br>1 = enabled. |
| NA_SELECT | 3 | Native mode. Select Coprocessor or frame buffer to respond to address range 000A0000–000AFFFF<br>This bit is set to 0 (default) on reset.<br>0 = Frame Buffer<br>1 = Coprocessor |
| b | 5 | Bit swap for commands through 000A0000–000AFFFF |
| B | 6 | Byte swap for commands through 000A0000–000AFFFF |
| H | 7 | Halfword swap for commands through 000A0000–000AFFFF |

Figure 35. CONFIG[65] register

### 3.3.18. CONFIG[66] REGISTER

**CONFIG[66] Register**




| Field | Bits | Contents |
|---|---|---|
| CKSEL | 4–2 | Sets the current state of pin CKSEL[2..0] in native mode. (default 0)<br>(CKSEL[4] also used as EEDATA; for writing, must be read through CONFIG[64]) |
| VCEN | 0 | The inverse of the level of the VCEN– pin.<br>This bit is set to 0 (default) on reset.<br>(Also used as EECLK) |

Figure 36. CONFIG[66] register

## 3.4. General Address Formats

The following sections define each of the general address formats. Formats for accessing specific registers, pseudo-registers, and commands are all variations on these general formats, and are defined in chapters 4 and 5.

### 3.4.1. LINEAR FRAME BUFFER ADDRESSING

The pixel address format, illustrated in figure 37, allows direct access to the pixel map. Half-word, byte, and bit swapping for endian control is specified in the system configuration (sysconfig) register (see section 4.3). Data in the frame buffer is stored in big-endian format.

When the host issues a pixel address, the Power 9100 accesses the display memory as if it were normal memory. Read accesses do not modify the pixel data; the current settings of the pixel processing registers do not affect the pixel data. Accessing non-existent pixels yields undefined results. Pixel accesses complete within a few clock cycles; the worst case includes the time required for a VRAM refresh, a shift register load, a row miss, and the actual read or write.

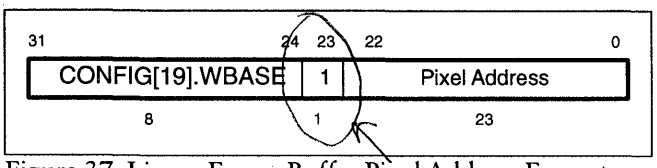



Figure 37. Linear Frame Buffer Pixel Address Format (CONFIG[19].WBASE, see figure 29)

### 3.4.2. ALTERNATE FRAME BUFFER APERTURE MAPPING

The aperture address format, illustrated in figure 38, allows indirect access to the pixel map. Half-word, byte, and bit swapping for endian control is specified in the system configuration (sysconfig) register (see section 4.3). Data in the frame buffer is stored in big-endian format. Access to the frame buffer through this mechanism is controlled by CONFIG[65].NA_ENABLE and CONFIG[65].NA_SELECT. By setting the alt_read_bank and alt_write_bank registers, this address format can be used to access a contiguous 64KB block of frame buffer memory on any 64KB boundary. The contents of the alt_read_bank and alt_write_bank registers are merged with the offset from the address field to compute the addressed byte of the frame buffer.

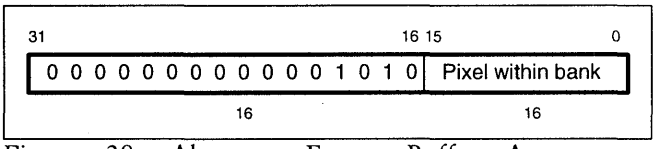



Figure 38. Alternate Frame Buffer Aperture (CONFIG[65].NA_ENABLE = 1, CONFIG[65].NA_SELECT = 0)

### 3.4.3. ROM BIOS ACCESS

The ROM BIOS access format, illustrated in figure 39, allows direct access to the ROM BIOS. Data in the ROM BIOS is stored in little-endian format and is not subject to H, B and b swapping. Access to the ROM BIOS through this mechanism is controlled by the rom_base field of configuration registers 49, 50 and 51. Both the CONFIG[4].mem_enable field and the CONFIG[48].rom_enable fields must be a 1 in order to enable BIOS ROM access.

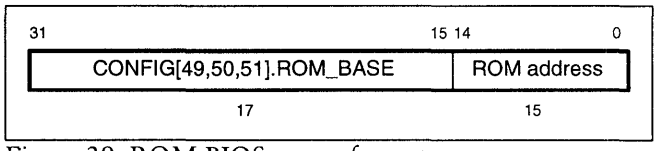



Figure 39. ROM BIOS access format

### 3.4.4. COPROCESSOR ACCESS

Expect for configuration registers, frame buffer memory and ROM BIOS access to all remaining native mode items are controlled through the coprocessor access protocol. See figure 40 for a list of the items that are accessible.

There are two access methods: direct and indirect. In both access modes data swapping is available. Figure 42 shows the direct access protocol. With this protocol the H, B and b swapping is directly specified within the address itself.

Figure 41 shows the indirect access protocols. With these protocols the H, B and b swapping are specified indirectly, they are taken from CONFIG[65].H, CONFIG[65].B and CONFIG[65].b. The indirect access format is only available if CONFIG[65].NA_ENABLE is a 1.

## 3.4. General Address Formats, continued

| Address Format | Described in Section | Page |
|---|---|---|
| RAMDAC access | 3.4.5 | 26 |
| Video Coprocessor Access | 3.4.6 | 26 |
| System control registers | 4.3 | 33 |
| Video control registers | 4.6 | 51 |
| VRAM control registers | 4.7 | 55 |
| Status register | 4.4.2 | 39 |
| Parameter engine registers | 4.4 | 38 |

| Address Format | Described in Section | Page |
|---|---|---|
| Drawing engine registers | 4.5 | 46 |
| Load coordinate pseudo-registers | 5.1 | 60 |
| Quad command | 5.2 | 60 |
| Blit command | 5.3 | 61 |
| Pixel8 command | 5.4 | 62 |
| Pixel1 command | 5.5 | 63 |
| Next_pixels command | 5.6 | 64 |

Figure 40. Specific Coprocessor address formats

**Coprocessor Access Protocol (Indirect)**

HBb are from config[65]

| 31 | | | | | | | | | | | | | | | | 15 | 14 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | – | Coprocessor Address | |
| 17 | | | | | | | | | | | | | | | | | 15 | |
| Prefix | | | | | | | | | | | | | | | | | | |

| Field | Bits | Definition |
|---|---|---|
| Prefix | 31–15 | Indirect access prefix used in address formats.<br>See figures 43, 44, 45, 46, 49, 54, 56, 58, 66, 72, 77, 80, 81, 82, 83, 85, and 87.<br>(CONFIG[65].NA_ENABLE = 1, CONFIG[65].NA_SELECT = 1; see figure 35.) |

Figure 41. Coprocessor access protocol (indirect)

**Coprocessor Access Protocol (Direct)**

| 31 | 24 | 23 | 19 | 18 | 17 | 16 | 15 | 14 | 0 |
|---|---|---|---|---|---|---|---|---|---|
| CONFIG[19].WBASE | | 0 | | H | B | b | 0 | Coprocessor Address | |
| 8 | | 5 | | 1 | 1 | 1 | 1 | 15 | |
| Prefix | | | | | | | | | |

| Field | Bits | Definition |
|---|---|---|
| Prefix | 31–15 | Direct access prefix used in address formats.<br>See figures 43, 44, 45, 46, 49, 54, 56, 58, 66, 72, 77, 80, 81, 82, 83, 85, and 87.<br>(CONFIG[19].WBASE, see figure 29.) |
| H | 18 | Half-word swap |
| B | 17 | Byte within half-word swap |
| b | 16 | Bit within byte swap |

Figure 42. Coprocessor access protocol (direct)

H = B = b = 0

## 3.4. General Address Formats, continued

### 3.4.5. RAMDAC ADDRESS FORMAT

The RAMDAC address format, illustrated in figure 43, allows direct access to the RAMDAC. RAMDAC reads and writes are initiated immediately and completed within a few clock cycles; the longest possible wait time for a direct RAMDAC access is the time required to complete a VRAM refresh, plus the time required to complete a shift register reload, plus the time required for the access. The 8-bit data transfer to or from the RAMDAC occurs across the eight MD bus bits that are connected to the RAMDAC data bus (see chapter 5).

### 3.4.6. VIDEO COPROCESSOR INTERFACE

The video coprocessor address format, illustrated in figure 44, allows direct access to the video coprocessor. Video coprocessor reads and writes are initiated immediately and completed within a few clock cycles. The video coprocessor interface is only enabled when operating in native mode. Many of the emulation mode pins are shared.

**RAMDAC Address Format**

| 31–15 | 14–6 | 5–2 | 1–0 |
|---|---|---|---|
| Prefix | 0 0 0 0 0 1 0 0 0 | CTL | 0 0 |
| 17 | 9 | 4 | 2 |

| Field | Bits | Definition |
|---|---|---|
| | 31–15 | Prefix. See figures 41 and 42. |
| | 5–2 | Controls transfer as defined by RAMDAC specifications |

Figure 43. RAMDAC address format

**Video Coprocessor Interface Format**

| 31–15 | 14–10 | 9–2 | 1–0 |
|---|---|---|---|
| Prefix | 0 0 0 0 1 | REG | 0 0 |
| 17 | 5 | 8 | 2 |

| Field | Bits | Definition |
|---|---|---|
| | 31–15 | Prefix. See figures 41 and 42. |
| | 9–2 | The register of the video coprocessor |

Figure 44. Video coprocessor interface format

## 3.4. General Address Formats, continued

| Prefix* | Address Bits | | | | | | | | | | | | | Specific Address | Section |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 31 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | | |
| | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | – | – | – | – | – | System control registers | 4.3 |
| | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | – | – | – | – | – | Video control registers | 4.6 |
| | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | – | – | – | – | – | VRAM control registers | 4.7 |
| | 0 | 0 | 0 | 0 | 0 | 1 | – | – | – | – | – | – | – | RAMDAC control | 3.4.5 |
| | 0 | 0 | 0 | 0 | 1 | – | – | – | – | – | – | – | – | Video Coprocessor Interface | 3.4.6 |
| | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | Pixel8 command | 5.4 |
| | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | Next_pixels command | 5.6 |
| | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | – | – | – | – | – | Pixel1 command | 5.5 |
| | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | – | – | – | – | – | Parameter engine control registers | 4.4.3 |
| | 0 | 1 | 0 | 0 | 0 | 1 | – | – | – | – | – | – | – | Drawing engine pixel processing registers | 4.5 |
| | 0 | 1 | 1 | 0 | 0 | 0 | 0 | – | – | – | – | – | – | Parameter engine coordinate registers | 4.4.3 |
| | 0 | 1 | 1 | 0 | 0 | 1 | – | – | – | – | – | – | – | Load coordinates pseudo-registers | 5.1 |
| | 1 | – | – | – | – | – | – | – | – | – | – | – | – | Pixel8 command | 5.4 |

* See figures 41 and 42.

Figure 45. Decoding of addresses for write operations

| Prefix* | Address Bits | | | | | | | | | | | | | Specific Address | Section |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 31 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | | |
| | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | – | – | – | – | – | System control registers | 4.3 |
| | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | – | – | – | – | – | Video control registers | 4.6 |
| | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | – | – | – | – | – | VRAM control registers | 4.7 |
| | 0 | 0 | 0 | 0 | 0 | 1 | – | – | – | – | – | – | – | RAMDAC control | 3.4.5 |
| | 0 | 0 | 0 | 0 | 1 | – | – | – | – | – | – | – | – | Video Coprocessor Interface | 3.4.6 |
| | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | Status register | 4.4.2 |
| | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | Blit command | 5.3 |
| | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | Quad command | 5.2 |
| | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | – | – | – | – | – | Parameter engine control registers | 4.4.3 |
| | 0 | 1 | 0 | 0 | 0 | 1 | – | – | – | – | – | – | – | Drawing engine pixel processing registers | 4.5 |
| | 0 | 1 | 1 | 0 | 0 | 0 | 0 | – | – | – | – | – | – | Parameter engine coordinate registers | 4.4.3 |
| | 0 | 1 | 1 | 0 | 0 | 1 | – | – | – | – | – | – | – | Load coordinates pseudo-registers | 5.1 |

* See figures 41 and 42.

Figure 46. Decoding of addresses for read operations

# Chapter 4. Native Mode Registers

## 4.1. Overview

The 32-bit Power 9100 registers specify parameters for an operation, control the operation, and maintain status information. Figure 47 summarizes the Power 9100 registers. All registers except those designated "read only" are read/write registers.

The register descriptions in this section define the register functions, present the address format for accessing each register, and provide a complete definition of each register field.

## 4.2. Register Summary

| Subgroup | Register | Name/Function | Accessed via | Effect of reset | See section |
|---|---|---|---|---|---|
| *System Control Registers* | | | | | |
| | sysconfig | System configuration. Specifies system configuration information. | video and system control address format | set to zero (Host Reset) | 4.3 |
| | interrupt | Interrupt. Records interrupt conditions. | | set to zero | |
| | interrupt_en | Interrupt enable. Enables interrupts upon occurrence of interrupt conditions. | | set to zero | |
| *Parameter Engine Registers* | | | | | |
| Device coordinate | X[0],Y[0]<br>X[1],Y[1]<br>X[2],Y[2]]<br>X[3],Y[3] | Device coordinate. Supplies screen coordinates for drawing operation. | user register address format | Not changed | 4.4 |
| Status | status | Status. Read only. Records status of drawing engine and coordinate register clip checks. | user register address format | Not changed | |
| Control and condition | oor | Out of range. Read only. Records out of range x,y values. | user register address format | Not changed | |
| | cindex | Index. Supplies current index into x and y coordinates | | Not changed | |
| | w_off.x/y | Window offset. Supplies offset of current window on the display. | | Not changed | |
| | pe_w_min<br>pe_w_max | Parameter engine window minimum, parameter engine window maximum. Read only. Record the contents of the drawing engine window minimum (w_min) and window maximum (w_max) registers. | | Not changed | |
| | xclip<br>yclip | Xclip, yclip. Read only. Record results of clip checks on x and y coordinates. | | Not changed | |
| | xedge_lt<br>xedge_gt<br>yedge_lt<br>yedge_gt | Vertex relationship. Read only. Record results of vertex relationship checking. | | Not changed | |

Figure 47. Register summary

## 4.2. Register Summary, continued

| Subgroup | Register | Name/Function | Accessed via | Effect of reset | See section |
|---|---|---|---|---|---|
| *Drawing Engine Registers* | | | | | |
| Pixel processing | color[0..3] | Color registers | user register address format | Not changed | 4.5 |
| | pmask | Plane mask. Specifies plane mask. | | Not changed | |
| | draw_mode | Draw mode. Specifies control for writing within a picked window and selects the destination buffer for drawing operations. | | Not changed | |
| | pat_originx pat_originy | X pattern origin, y pattern origin. Specify x and y screen coordinates for pattern origin. | | Not changed | |
| | raster | Raster. Specifies parameters for a raster operation. | | Not changed | |
| | pixel8_reg | Pixel8. Stores excess pixel8 operation data bits. | | Not changed | |
| | p_w_min b_w_min | Window minimum. Specifies minimum x,y values for window. | | Not changed | |
| | p_w_max b_w_min | Window maximum. Specifies maximum x,y values for window. | | Not changed | |
| | pattern [0..1][0..3] [0..7] | Pattern. Specify pattern. | | Not changed | |
| *Video Control Registers* | | | | | |
| Horizontal | hrzc | Horizontal counter. Read only. | video and direct control address format | Set to zero | 4.6 |
| | hrzt | Horizontal length. | | Set to 0xFFF | |
| | hrzsr | Horizontal sync rising edge. | | Set to 0xFFF | |
| | hrzbr | Horizontal blank rising edge. | | Set to 0xFFF | |
| | hrzbf | Horizontal blank falling edge. | | Set to 0xFFF | |
| | prehrzc | Horizontal counter preload. | | Set to zero | |
| Vertical | vrtc | Vertical counter. Read only. | | Set to zero | |
| | vrtt | Vertical length. | | Set to 0xFFF | |
| | vrtsr | Vertical sync rising edge. | | Set to 0xFFF | |
| | vrtbr | Vertical blank rising edge. | | Set to 0xFFF | |
| | vrtbf | Vertical blank falling edge. | | Set to 0xFFF | |
| | prevrtc | Vertical counter preload. | | Set to zero | |
| Repaint | srtctl | Screen repaint timing control. | | Set to zero | |
| | srtctl2 | Screen repaint timing control. | | Set to zero | |
| | qsfcounter | QSF counter. | | Set to zero | |

Figure 47, continued. Register summary

## 4.2. Register Summary, continued

| Subgroup | Register | Name/Function | Accessed via | Effect of reset | See section |
|---|---|---|---|---|---|
| *VRAM Control Registers* | | | | | |
| | mem_config | Memory configuration. | video and direct control address format | Set to zero | 4.7 |
| | rfperiod | Refresh period. | | Set to 0x3FF | |
| | rfcount | Refresh counter. | | Set to zero | |
| | rlmax | RAS low maximum. | | Set to 0x3FF | |
| | rlcur | RAS low current. | | Set to zero | |

Figure 47, continued. Register summary

## 4.3. System Control Registers

The system control registers, summarized in figure 48, control all but the video portions of the Power 9100. They can be read or written at any time. Figure 49 defines the specific format for accessing these registers.

| Register | Function |
|---|---|
| sysconfig | System configuration. Specifies shift control, swapping for endian control on pixel accesses, buffer selection for pixel accesses, and the current version of the Power 9100. This register is only reset by a host reset. |
| interrupt | Interrupt. Indicates the occurrence of various Power 9100 interrupt conditions. This register is reset by a mode change (native mode/VGA mode). |
| interrupt_en | Interrupt enable. Enables/disables the assertion of the interrupt signal upon the occurrence of specific interrupt conditions. This register is reset by a mode change (native mode/VGA mode). |
| alt_read_bank | Alternate bus read bank select. Read/write.<br>Controls banking of the frame buffer. This register is reset by a mode change (native mode/VGA mode). |
| alt_write_bank | Alternate bus write bank select. Read/write.<br>Controls banking of the frame buffer. This register is reset by a mode change (native mode/VGA mode). |

Figure 48. System control registers

**Address Format for System Control Register Access**

| 31 … 15 | 14 … 7 | 6 … 2 | 1 | 0 |
|---|---|---|---|---|
| Prefix | 0 0 0 0 0 0 0 0 | Register | 0 | 0 |
| 17 | 8 | 5 | 2 | |

| Field | Bits | Entry | Register Selected |
|---|---|---|---|
| Prefix | 31–15 | | See figures 41 and 42. |
| Register | 6–2 | 00001 | sysconfig HWBase Addr + 4 |
| | | 00010 | interrupt |
| | | 00011 | interrupt_en |
| | | 00100 | alt_write_bank |
| | | 00101 | alt_read_bank |
| | | 00110–11111 | not used |

Figure 49. Address format for system control register access

## 4.3. System Control Registers, continued

### 4.3.1. ALTERNATE READ BANK REGISTER

The alternate read bank register (alt_read_bank) specifies the high order address bits when the alternate aperture frame buffer banking logic is used to read directly from the frame buffer. See figure 50 for details.

### 4.3.2. ALTERNATE WRITE BANK REGISTER

The alternate write bank register (alt_write_bank) specifies the high order address bits when the alternate aperture frame buffer banking logic is used to write directly from the frame buffer. See figure 50 for details.

### 4.3.3. SYSTEM CONFIGURATION REGISTER

The system configuration (sysconfig) register provides various system configuration controls, as defined in figure 51. Bits 13 through 11 determine half-word, byte, and bit swapping control.

The shift control fields of the sysconfig register are set to indicate the size of a drawing engine scanline in bytes. This is true independent of the pixel_size field.

**Format of alt_write_bank and alt_read_bank**

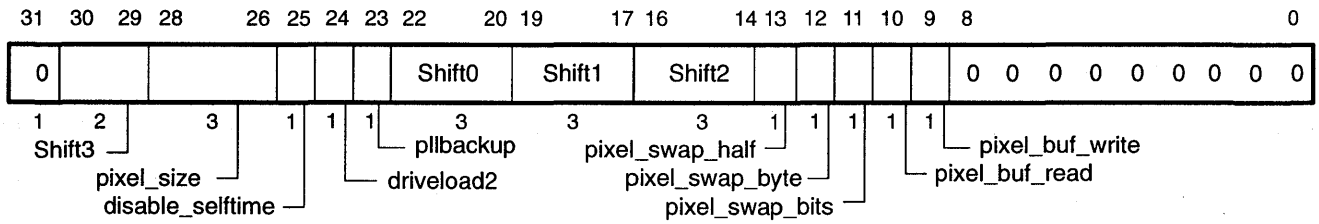
| 31–23 | 22–16 | 15–0 |
|---|---|---|
| 0 | BANK | 0 |
| 10 | 6 | 16 |

| Field | Bits | Contents |
|---|---|---|
| BANK | 22–16 | Bank for alternate frame buffer aperture |

Figure 50. Format of alt_write_bank and alt_read_bank

## 4.3. System Control Registers, continued

**System Configuration Register (sysconfig)**




| Field | Bits | Contents |
|---|---|---|
| | 31 | Reserved |
| Shift3 | 30–29 | Shift Control 3.<br>00 = SHIFT_0 (no add) 01 = SHIFT_1024 (multiply by 1024)<br>10 = SHIFT_2048 (multiply by 2048) 11 = SHIFT_4096 (multiply by 4096) |
| pixel_size | 28–26 | Pixel size for drawing engine<br>010 = 8 bits per pixel (8bpp) 011 = 16 bits per pixel (16bpp)<br>111 = 24 bits per pixel (24bpp) 101 = 32 bits per pixel (32bpp) |
| disable_selftime | 25 | Disables internal selftiming on FBC address |
| driveload2 | 24 | Frame buffer controller drive load selection:<br>0 = Normal drive load 1 = Double drive load |
| pllbackup | 23 | Overrides internal PLL to use external clocking circuit.<br>0 = Normal operation 1 = External PLL clocking. |
| Shift0 | 22–20 | Shift Control 0.<br>000 = SHIFT_0 (no add) 011 = SHIFT_128 (multiply by 128)<br>100 = SHIFT_256 (multiply by 256) 101 = SHIFT_512 (multiply by 512)<br>110 = SHIFT_1024 (multiply by 1024) 111 = SHIFT_2048 (multiply by 2048) |
| Shift1 | 19–17 | Shift Control 1.<br>000 = SHIFT_0 (no add) 010 = SHIFT_64 (multiply by 64)<br>011 = SHIFT_128 (multiply by 128) 100 = SHIFT_256 (multiply by 256)<br>101 = SHIFT_512 (multiply by 512) 110 = SHIFT_1024 (multiply by 1024) |
| Shift2 | 16–14 | Shift Control 2.<br>000 = SHIFT_0 (no add) 001 = SHIFT_32 (multiply by 32)<br>010 = SHIFT_64 (multiply by 64) 011 = SHIFT_128 (multiply by 128)<br>100 = SHIFT_256 (multiply by 256) 101 = SHIFT_512 (multiply by 512) |
| pixel_swap_half | 13 | Half-word swap selection for endian control for frame buffer accesses:<br>0 = do not swap half words on frame buffer accesses.<br>1 = swap half words on frame buffer accesses |
| pixel_swap_byte | 12 | Byte swap selection for endian control for frame buffer accesses:<br>0 = do not swap bytes on frame buffer accesses<br>1 = swap bytes on frame buffer accesses |
| pixel_swap_bits | 11 | Bit swap selection for endian control for frame buffer accesses:<br>0 = do not swap bits on frame buffer accesses<br>1 = swap bits on frame buffer accesses |
| pixel_buf_read | 10 | Buffer selection for read pixel accesses |
| pixel_buf_write | 9 | Buffer selection for write pixel accesses |
| | 8–0 | Reserved |

Figure 51. System configuration (sysconfig) register

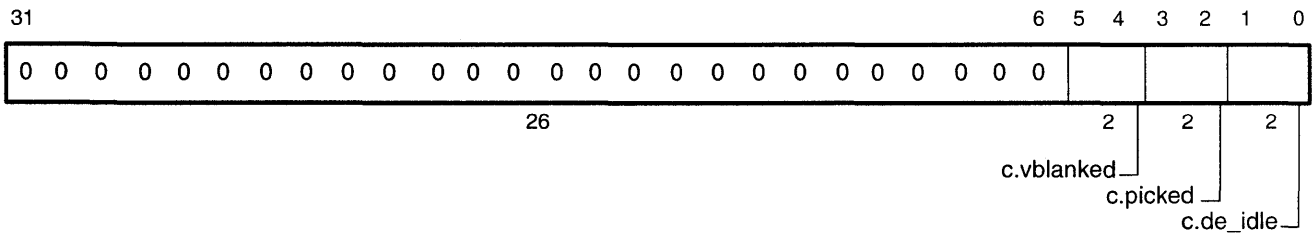
## 4.3. System Control Registers, continued

### 4.3.4. INTERRUPT REGISTER

The interrupt register is a read/write register that accumulates status information. The Power 9100 sets bits in the interrupt register to indicate the occurrence of specific interrupt conditions, as shown in figure 52. The Power 9100 generates a hardware interrupt (that is, asserts the INTR– signal) only when the corresponding bit is set in the interrupt enable (interrupt_en) register.

The interrupt register bits are "sticky;" they remain set after the conditions they reflect are gone. They must be cleared by a host register transfer. Each condition bit is preceded by a field write control bit which must be set to 1 to write into the field or 0 to leave the field as it stands when writing to the register to clear interrupt bits. Also, if the host sets an interrupt bit in the interrupt register and enables that interrupt via the interrupt enable register, the Power 9100 generates the interrupt.

**Interrupt Register**




| Field | Bits | Contents |
|---|---|---|
| vblanked | 5–4 | Vblank:<br>Bit 5: field write control bit<br>Bit 4: 0 = no vblank done<br>1 = vblank done |
| picked | 3–2 | Pick:<br>Bit 3: field write control bit<br>Bit 2: 0 = no pick done<br>1 = pick done |
| de_idle | 1–0 | Drawing engine idle:<br>Bit 1: field write control bit<br>Bit 0: 0 = drawing engine busy<br>1 = drawing engine idle |

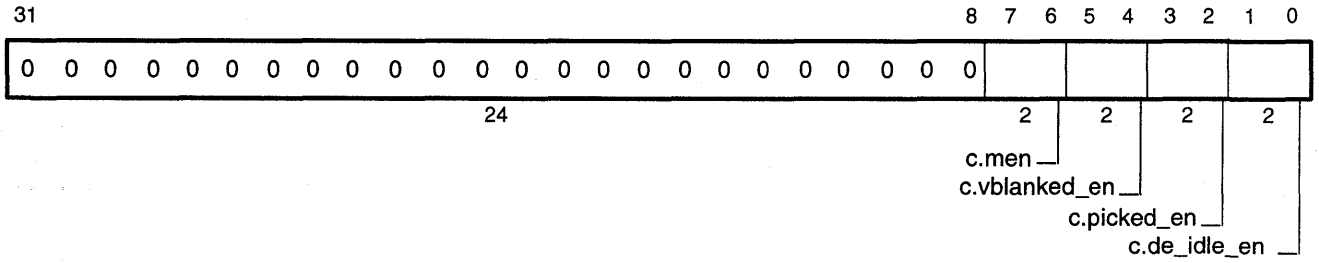
Figure 52. Interrupt register

## 4.3. System Control Registers, continued

### 4.3.5. INTERRUPT ENABLE REGISTER

The interrupt enable (interrupt_en) register specifies the conditions under which the Power 9100 asserts the INTR– interrupt signal. Figure 53 defines this register. Note that each interrupt enable bit is preceded by a field write control bit which must be set to 1 to write into the field (or 0 to leave the field as it stands) when writing to the register. Also, if the host sets an interrupt bit in the interrupt register and enables that interrupt by setting the corresponding bit in the interrupt enable register, the Power 9100 generates the interrupt.

**Interrupt Enable Register (interrupt_en)**




| Field | Bits | Contents |
|---|---|---|
| men | 7–6 | Master enable:<br>Bit 7: field write control bit<br>Bit 6: 0 = disable all interrupts<br>1 = enable interrupts as specified by individual interrupt enables (bits 5–0) |
| vblanked_en | 5–4 | Vblank:<br>Bit 5: field write control bit<br>Bit 4: 0 = do not generate interrupt when a vblank occurs<br>1 = generate interrupt when a vblank occurs |
| picked_en | 3–2 | Pick:<br>Bit 3: field write control bit<br>Bit 2: 0 = do not generate interrupt when a pick occurs<br>1 = generate interrupt when a pick occurs |
| de_idle_en | 1–0 | Drawing engine status:<br>Bit 1: field write control bit<br>Bit 0: 0 = do not generate interrupt when drawing engine goes idle<br>1 = generate interrupt when drawing engine goes idle |

Figure 53. Interrupt enable (interrupt_en) register

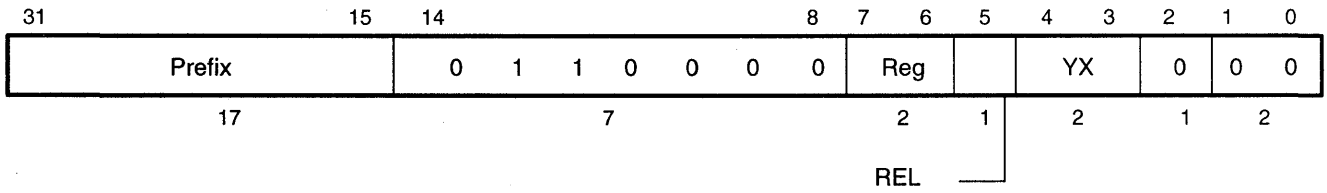
## 4.4. Parameter Engine Registers

The parameter engine registers, summarized in figure 55, supply the coordinates for a drawing operation, provide control for the drawing operation, and maintain status information. The host can read or write these registers at any time. A system reset leaves them in an undefined state.

### 4.4.1. DEVICE COORDINATE REGISTERS

The device coordinate registers supply the screen coordinates for a drawing operation. The device coordinate registers supply the four screen coordinates required for each drawing operation. There are four X registers and four Y registers, one for each *x* and *y* value required. *X* and *y* values can be supplied as 32-bit or 16-bit values, as indicated by the YX field in the address format. When supplied as 16-bit values, the *x* and *y* values are packed and transferred in a single 32-bit read or write. Figure 54 defines the specific user register address format for accessing these registers.

**Address Format for Device Coordinate Register Access**




| Field | Bits | Entry | Register Selected |
|---|---|---|---|
| Prefix | 31–15 | | See figures 41 and 42. |
| Reg | 7–6 | 00 | X[0]/Y[0] |
| | | 01 | X[1]/Y[1] |
| | | 10 | X[2]/Y[2] |
| | | 11 | X[3]/Y[3] |
| REL | 5 | 0 | Perform absolute screen addressing |
| | | 1 | Perform window-relative screen addressing (write only) |
| YX | 4–3 | 00 | Not used |
| | | 01 | Read or write 32-bit X value |
| | | 10 | Read or write 32-bit Y value |
| | | 11 | Read or write 16-bit X value (high 16 bits) and 16-bit Y value (low 16 bits) |

Figure 54. Address format for device coordinate register access

| Register Category | Register | Function |
|---|---|---|
| Device coordinate | X[0]/Y[0]<br>X[1]/Y[1]<br>X[2]/Y[2]<br>X[3]/Y[3] | Supply the screen coordinates for the drawing operation. |
| Status | status | Records the current status of the drawing engine and results of coordinate register clip checks. |
| Control and condition | clip | Clip check. Records the results of clip checks. |
| | oor | Out of range. Records the x,y values that are out of range for the current drawing operation. |
| | w_off.X/Y | Window offset. Supplies the offset of the current window on the display for coordinate computation of window-relative addressing. (This provides a translation only, no clip or scale; it is an offset only.) |
| | cindex | Index. Supplies the current index into the x,y coordinates for load coordinate computation (see section 5.1). |

Figure 55. Parameter engine registers

## 4.4. Parameter Engine Registers, continued

### 4.4.2. THE STATUS REGISTER

The status register is a read-only register whose contents reflect the current status of the drawing engine and coordinate register clip checks. Access to the status register can be obtained by issuing a quad, blit or pixel8 command. Figure 56 defines the specific address format for accessing the status register. Figure 57 defines the status register. The Power 9100 updates this register every time the *x* and *y* coordinates are changed.

Bit 31 must be zero to successfully initiate another a quad draw operation or a blit operation; bit 30 must be zero when issuing the first in a series of pixel1 or pixel8 operations or writing to any drawing engine register (see section 5.7.5).

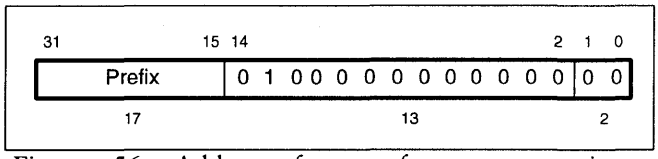
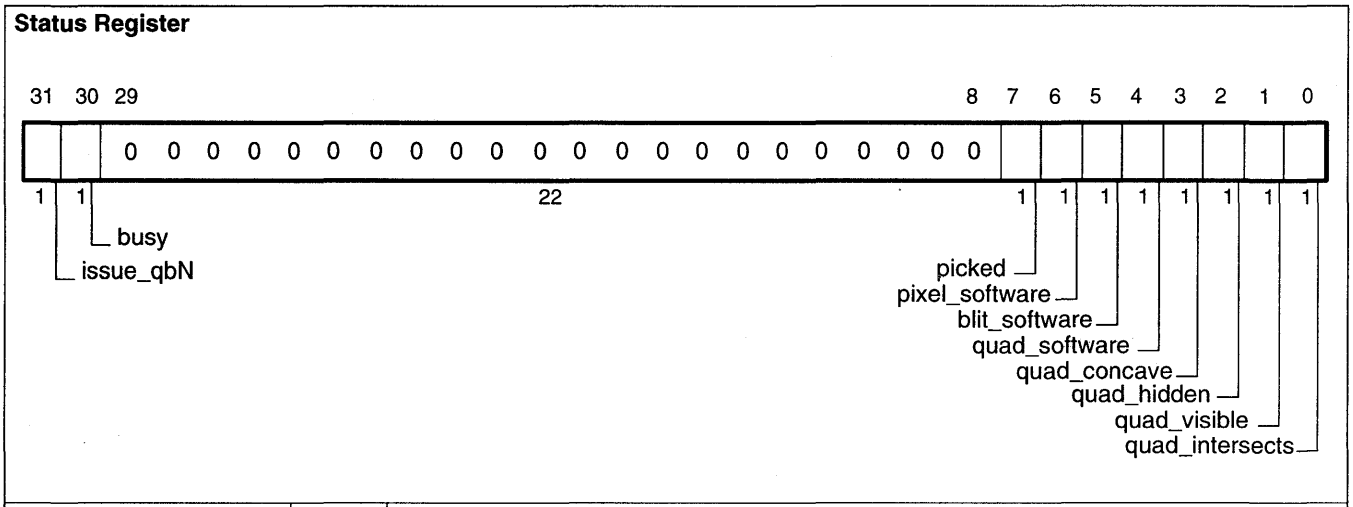



Figure 56. Address format for status register access (prefix: see figures 41 and 42)




| Field | Bits | Contents |
|---|---|---|
| issue_qbN | 31 | Quad/blit initiate another:<br>0 = may start another quad operation if previous was also a quad operation or may start another blit operation if previous was also a blit operation<br>1 = may not start quad or blit operation |
| busy | 30 | The drawing engine is busy. |
| picked | 7 | Pick detected. (This field is a copy of the picked field in the interrupt register.) |
| pixel_software | 6 | Exception encountered for pixel command; the operation cannot be performed and must be done in software. |
| blit_software | 5 | Exception encountered for blit command; the operation cannot be performed and must be done in software. |
| quad_software | 4 | Exception encountered for quad command; the operation cannot be performed and must be done in software. |
| quad_concave | 3 | The requested quad is concave. |
| quad_hidden | 2 | The source coordinates for a quad draw are entirely outside the clipping window. |
| quad_visible | 1 | The source coordinates for a quad draw are entirely inside the clipping window. |
| quad_intersects | 0 | The source coordinates for a quad draw straddle the clipping window. |

Figure 57. Status register

### 4.4.3. PARAMETER ENGINE CONTROL AND CONDITION REGISTERS

The parameter engine control and condition registers control and monitor Power 9100 operations, as summarized in figure 55. Figure 58 defines the specific user register address format for accessing these registers.

*Out of Range Register.* The read-only out of range register (oor), illustrated in figure 59, identifies $x$ and $y$ values that are out of range of the drawing engine for the current drawing operation. The Power 9100 updates this register every time the $x$ and $y$ coordinate registers are modified.

*Index Register.* The index register (cindex) supplies the current index into the $x$ and $y$ coordinates as a 2-bit binary integer. The value is used for load coordinate computation (see section 5.1). This is a read/write register.

*Window Offset Register.* The window offset register (w_off.xy) supplies the offset of the current window on the display as two packed 16-bit two's complement binary integers, representing the $x$ and $y$ coordinates. This information is required by coordinate computation operations to calculate addressing relative to the window. It has no effect on clipping. This is a read/write register.

*Pixel Window Minimum and Maximum Registers.* The pixel valued window minimum (p_w_min) and window maximum (p_w_max) registers are read through this address format. See section 4.5.7 for a complete description of the window maximum and minimum registers.

*Clip Registers.* The read-only clip registers (xclip and yclip) define the results of clip checks, as summarized and illustrated in figures 60 and 61. The Power 9100 updates these registers for every new set of $x$ and $y$ values loaded into the coordinate registers.

*Vertex Relationship Checking Registers.* The read-only vertex relationship checking registers (xedge_lt, xedge_gt, yedge_lt, and yedge_gt) define the results of checks on vertices to verify that the requested drawing operation is acceptable, as summarized and illustrated in figures 62 through 65. The Power 9100 updates these registers for every new set of $x$ and $y$ values loaded into the coordinate registers.

## 4.4. Parameter Engine Registers, continued

**Address Format for Parameter Engine Control and Condition Register Access**

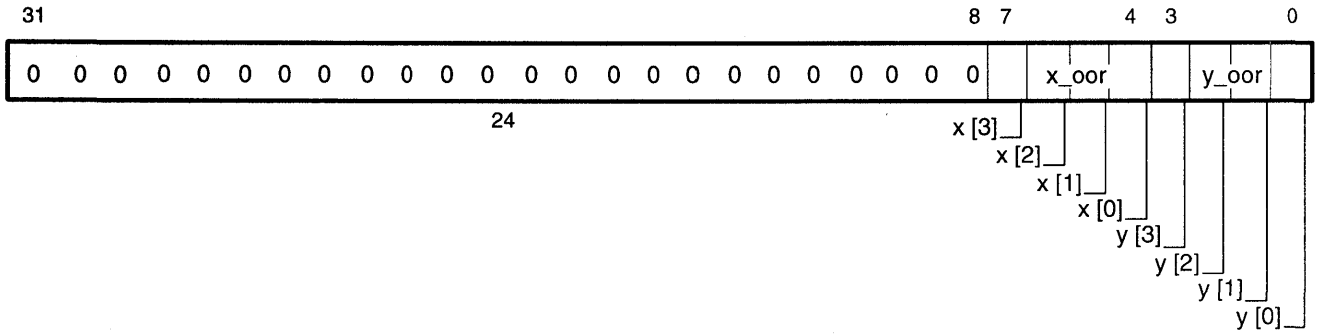
| 31 – 15 | 14 – 7 | 6 – 2 | 1 – 0 |
|---|---|---|---|
| Prefix | 0 1 0 0 0 0 1 1 | Register | 0 0 |
| 17 | 8 | 5 | 2 |

| Field | Bits | Entry | Register Selected |
|---|---|---|---|
| Prefix | 31–15 | | See figures 41 and 42. |
| Register entry | 6–2 | 00000 | not used |
| | | 00001 | oor |
| | | 00010 | not used |
| | | 00011 | cindex |
| | | 00100 | w_off.xy |
| | | 00101 | p_w_min (Read only) |
| | | 00110 | p_w_max (Read only) |
| | | 00111 | not used |
| | | 01000 | yclip |
| | | 01001 | xclip |
| | | 01010 | xedge_lt |
| | | 01011 | xedge_gt |
| | | 01100 | yedge_lt |
| | | 01101 | yedge_gt |
| | | 01110–11111 | not used |

Figure 58. Address format for parameter engine control and condition register access

## 4.4. Parameter Engine Registers, continued

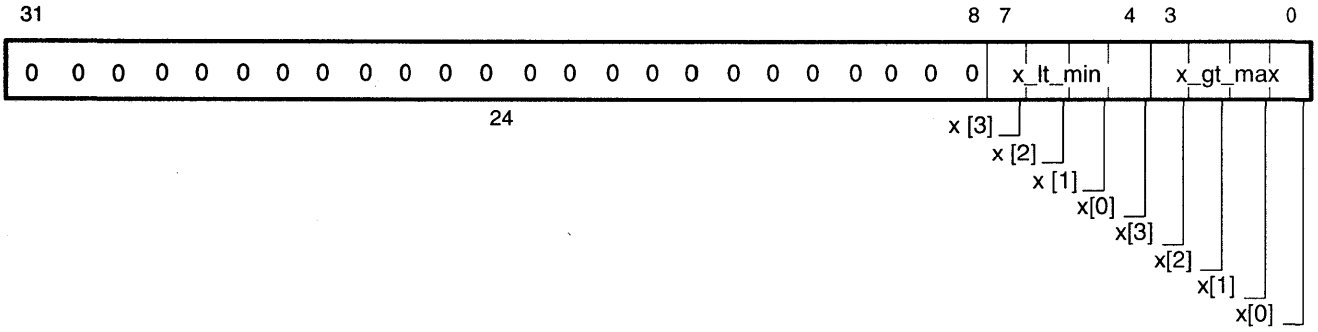
**Out of Range (oor) Register**




| Field | Bits | Contents |
|---|---|---|
| x_oor | 7–4 | Each bit indicates whether or not one x value is out of range of the drawing engine (msb = x[3]...lsb = x[0]):<br>0 = The x value is not out of range<br>1 = The x value is out of range |
| y_oor | 3–0 | Each bit indicates whether or not one y value is out of range of the drawing engine (msb = y[3]...lsb = y[0]):<br>0 = The y value is not out of range<br>1 = The y value is out of range |

Figure 59. Out of range (oor) register

**Xclip Register**

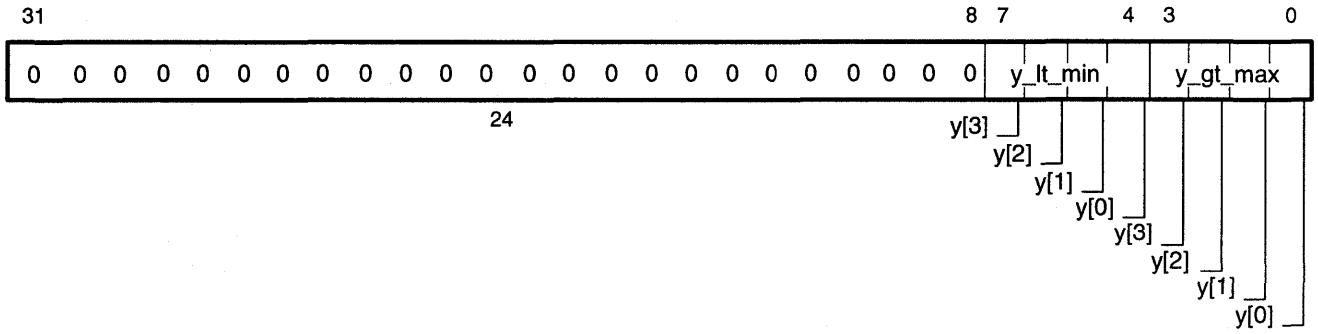



| Field | Bits | Contents |
|---|---|---|
| x_lt_min | 7–4 | Each bit indicates whether or not one x value is less than the window minimum (w_min.x) (msb = x[3]...lsb = x[0]):<br>0 = The x value is not less than w_min.x<br>1 = The x value is less than w_min.x |
| x_gt_max | 3–0 | Each bit indicates whether or not one x value is greater than the window maximum (w_max.x) (msb = y[3]...lsb = y[0]):<br>0 = The x value is not greater than w_max.x<br>1 = The x value is greater than w_max.x |

Figure 60. Xclip register

## 4.4. Parameter Engine Registers, continued

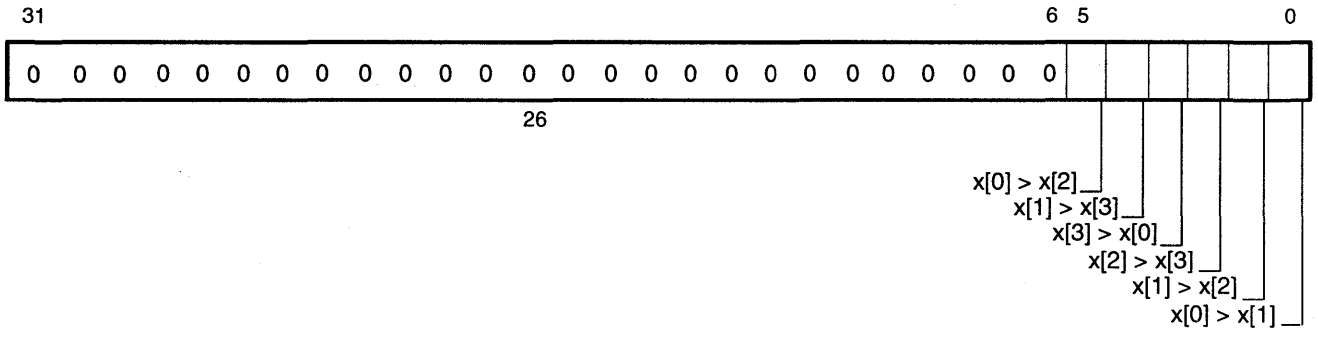
**Yclip Register**




| Field | Bits | Contents |
|---|---|---|
| y_lt_min | 7–4 | Each bit indicates whether or not one y value is less than the window minimum (w_min.y) (msb = y[3]...lsb = y[0]):<br>0 = The y value is not less than w_min.y<br>1 = The y value is less than w_min.y |
| y_gt_max | 3–0 | Each bit indicates whether or not one y value is greater than the window maximum (w_max.y) (msb = y[3]...lsb = y[0]):<br>0 = The y value is not greater than w_max.y<br>1 = The y value is greater than w_max.y |

Figure 61. Yclip register

**Xedge_lt Register**

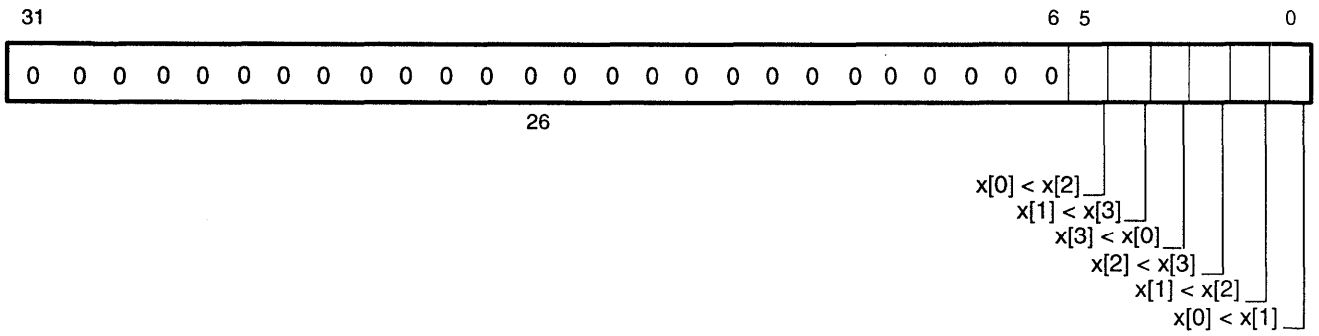



| Field | Bits | Contents |
|---|---|---|
| | 5–0 | When the condition specified in the bit definition is met, the bit is set to one; when the condition is not met, the bit is set to zero. |

Figure 62. Xedge_lt register

## 4.4. Parameter Engine Registers, continued

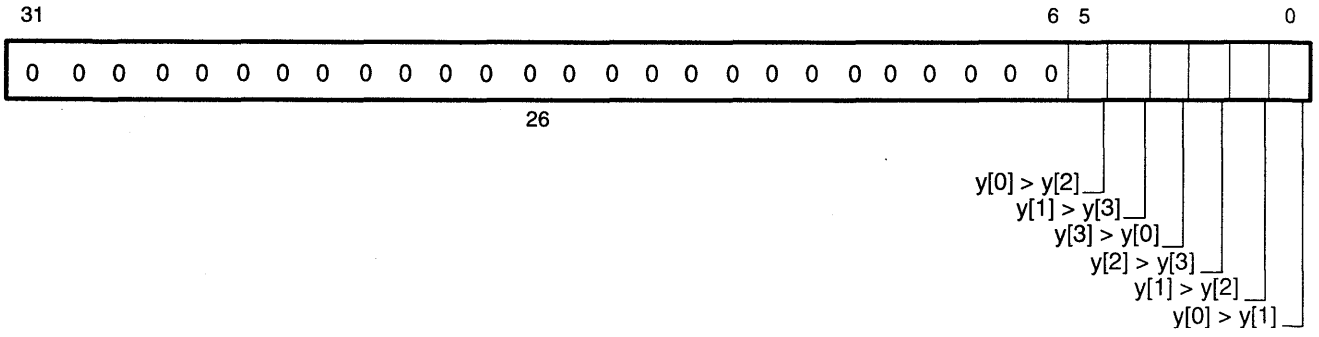
**Xedge_gt Register**




| Field | Bits | Contents |
|---|---|---|
| | 5–0 | When the condition specified in the bit definition is met, the bit is set to one; when the condition is not met, the bit is set to zero. |

Figure 63. Xedge_gt register

**Yedge_lt Register**

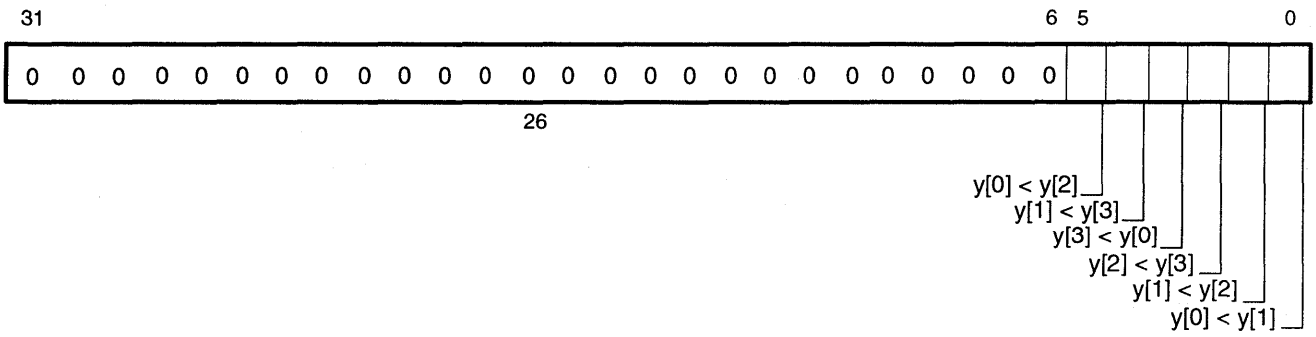



| Field | Bits | Contents |
|---|---|---|
| | 5–0 | When the condition specified in the bit definition is met, the bit is set to one; when the condition is not met, the bit is set to zero. |

Figure 64. Yedge_lt register

## 4.4. Parameter Engine Registers, continued

**Yedge_gt Register**




| Field | Bits | Contents |
|---|---|---|
| | 5–0 | When the condition specified in the bit definition is met, the bit is set to one; when the condition is not met, the bit is set to zero. |

Figure 65. Yedge_gt register

## 4.5. Drawing Engine Registers

The drawing engine registers are pixel processing registers that provide storage and define the foreground and background colors, plane mask, pattern and pattern origins, pixel drawing window limits, and raster operation parameters. These registers are accessed as illustrated in figure 66. The host can read or write these registers only when the drawing engine is idle (when the busy bit in the status register is set to zero); attempting to do so when the drawing engine is busy produces undefined results.

### 4.5.1. COLOR REGISTERS

There are four 32-bit color registers: color[0], color[1], color[2] and color[3]. These registers contain the colors used by the pattern and pixel1 logic. When operating in 8-bpp and 16-bpp the requested color must be replicated four and two times respectively to fill the entire register.

### 4.5.2. PLANE MASK REGISTER

The plane mask register (pmask) supplies the plane mask. When operating in 8-bpp and 16-bpp the requested color must be replicated four and two times respectively to fill the entire register.

**Address Format for Drawing Engine Pixel Processing Register Access**

| 31–15 | 14–9 | 8–2 | 1–0 |
|---|---|---|---|
| Prefix (17) | 0 1 0 0 0 1 (6) | Register (7) | 0 0 (2) |

| Field | Bits | Entry | Register Selected |
|---|---|---|---|
| Prefix | 31–15 | | See figures 41 and 42. |
| Register | 8–2 | 0000000 | color[0] |
| | | 0000001 | color[1] |
| | | 0000010 | Plane mask (pmask) |
| | | 0000011 | Draw mode (draw_mode) |
| | | 0000100 | Pattern origin x (pat_originx) |
| | | 0000101 | Pattern origin y (pat_originy) |
| | | 0000110 | Raster (raster) |
| | | 0000111 | Pixel8 (pixel8_reg) |
| | | 0001000 | Pixel Window min (p_w_min) (Write only) |
| | | 0001001 | Pixel Window max (p_w_max) (Write only) |
| | | 0001010–0001011 | Reserved |
| | | 0001110 | color[2] |
| | | 0001111 | color[3] |
| | | 0001010–0011111 | Reserved |
| | | 0100000 | Pattern0 |
| | | 0100001 | Pattern1 |
| | | 0100010 | Pattern2 |
| | | 0100011 | Pattern3 |
| | | 0100100–0100111 | Reserved for Software |
| | | 0101000 | Byte Window min (b_w_min) |
| | | 0101001 | Byte Window max (b_w_max) |

Figure 66. Address format for drawing engine pixel processing register access

## 4.5. Drawing Engine Registers, continued

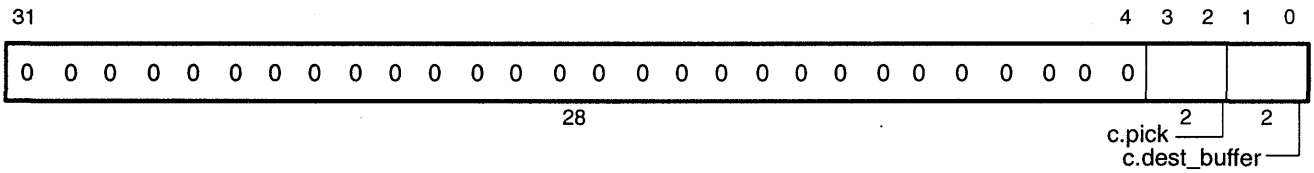
### 4.5.3. DRAW_MODE REGISTER

The draw_mode register controls writing within a picked window and selects the destination buffer for drawing operations, as defined in figure 67.

### 4.5.4. PATTERN ORIGIN REGISTERS

The two pattern origin registers (pat_originx and pat_originy) supply the 3-bit origin of the pattern to be used by the quad command to fill polygons on the display screen. The *x* and *y* origins are supplied in the lower three bits of each register.

**Draw_mode Register**




| Field | Bits | Contents |
|---|---|---|
| pick | 3–2 | Pick mode enable.<br>Bit 3: field write control bit<br>Bit 2: 0 = Suppress any write operation outside the window; allow a write inside the window<br>1 = Suppress all write operations; set the picked bit in the interrupt register (see figure 52) if the suppressed write is within the window |
| dest_buffer | 1–0 | Destination buffer.<br>Bit 1: field write control bit<br>Bit 0: 0 = Select buffer 0 as the destination buffer for all drawing engine operations<br>1 = Select buffer 1 as the destination buffer for all drawing engine operations |

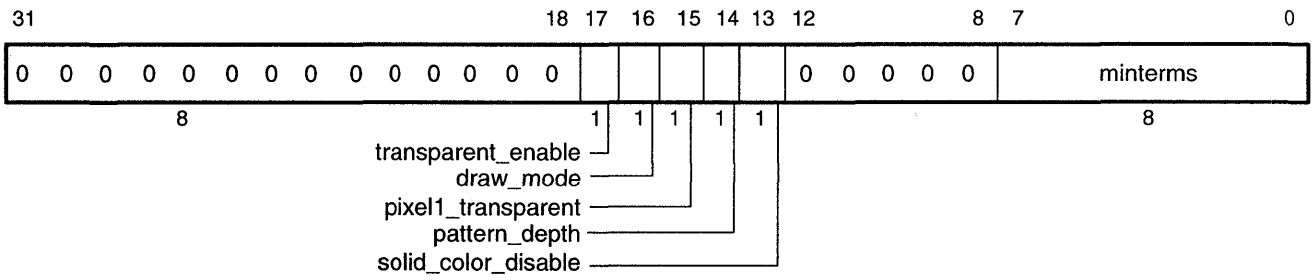
Figure 67. Draw_mode register

## 4.5. Drawing Engine Registers, continued

### 4.5.5. RASTER REGISTER

The raster register specifies the parameters for a raster operation, as defined in figure 68.

**Raster Register**




| Field | Bits | Contents |
|---|---|---|
| transparent_enable | 17 | Transparent Pattern Enable<br>0 = Disable transparent pattern<br>1 = Enable transparent pattern |
| draw_mode | 16 | Quad draw mode (see section 5.7.2):<br>0 = X11 mode<br>1 = Oversized mode |
| pixel1_transparent | 15 | Transparent Pixel1 Enable<br>0 = Disable pixel1 transparent mode<br>1 = Enable pixel1 transparent mode |
| pattern_depth | 14 | Pattern Depth<br>0 = 2 color pattern<br>1 = 4 color pattern (8 bpp only) |
| solid_color_disable | 13 | Solid color disable<br>0 = Solid color enabled, Pattern disabled<br>1 = Solid color disabled, Pattern enabled |
| minterms[0..7] | 7–0 | Minterms for boolean raster operation<br>Bit 7 = Pattern • Source • Destination<br>Bit 6 = Pattern • Source • $\overline{\text{Destination}}$<br>Bit 5 = Pattern • $\overline{\text{Source}}$ • Destination<br>Bit 4 = Pattern • $\overline{\text{Source}}$ • $\overline{\text{Destination}}$<br>Bit 3 = $\overline{\text{Pattern}}$ • Source • Destination<br>Bit 2 = $\overline{\text{Pattern}}$ • Source • $\overline{\text{Destination}}$<br>Bit 1 = $\overline{\text{Pattern}}$ • $\overline{\text{Source}}$ • Destination<br>Bit 0 = $\overline{\text{Pattern}}$ • $\overline{\text{Source}}$ • $\overline{\text{Destination}}$ |

Figure 68. Raster register

## 4.5. Drawing Engine Registers, continued

### 4.5.6. PIXEL8 REGISTER

The pixel8 register (pixel8_reg) provides temporary storage for excess bit data from a pixel8 operation (see section 5.4). The data is stored in the lower end of the register. This register is generally of interest to the programmer only when a context switch is being performed in the middle of a pixel8 operation.

### 4.5.7. WINDOW MINIMUM/WINDOW MAXIMUM REGISTERS

The window clipping registers in the drawing engine are used to control the clipping of drawing operations. A single rectangle is supported by specifying the minimum and maximum X and Y values. Clipping is always enabled and imposes no performance penalty. There are two sets of clipping registers: pixel valued (p_w_min and p_w_max) and byte valued (b_w_min and b_w_max). The pixel valued registers contain the X and Y values encoded in pixels and scanlines. The byte valued registers contain the X and Y values in bytes and scanlines. The pixel valued registers are loaded through the drawing engine address format but are read back through the parameter engine address format (see section 4.5). Figure 69 illustrates the contents of these registers.

### 4.5.8. PATTERN REGISTERS

The pattern registers are read/write registers by which the host specifies the pattern for quad fill. The index 1 registers: pattern[1][0..7][0..7] are only used with the special four color pattern in 8-bpp mode. Figure 70 shows the format of the pattern registers.

### 4.5.9. RESERVED FOR SOFTWARE

These 4 registers are reserved for use by the system software to store anything that it wants to.

**Window Minimum (p_w_min and b_w_min) and Window Maximum (p_w_max and b_w_max) Registers**

| 31 29 | 28 16 | 15 13 | 12 0 |
|---|---|---|---|
| 0 0 0 | x value | 0 0 0 | y value |

| Field | Bits | Contents |
|---|---|---|
| x value | 28–16 | Minimum x value in w_min register; maximum x value in w_max register. |
| y value | 12–0 | Minimum y value in w_min register; maximum y value in w_max register. |

Figure 69. Window minimum (p_w_min and b_w_min) and window maximum (p_w_max and b_w_max) registers

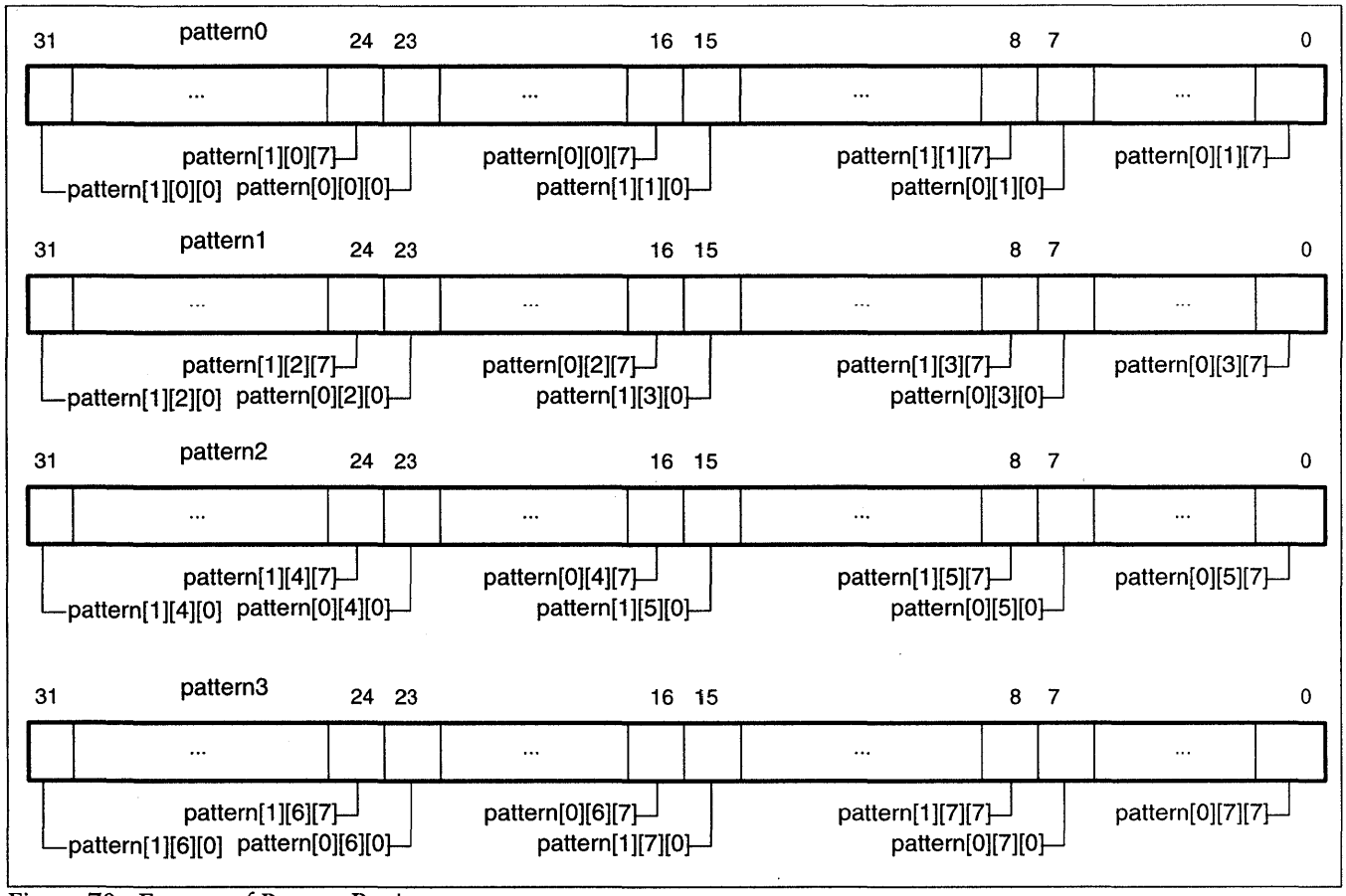



Figure 70. Format of Pattern Registers

## 4.6. Video Control Registers

Figures 73 and 74 summarize the video control registers. They can be accessed at any time. Figure 72 defines the specific address format for accessing these registers.

Most of the registers in the video section are operate in terms of CRTC clocks (CRTC_CLCKs). The CRTC clock is generated by the programmable dividing down of the video clock input to the Power 9100. Usually, the RAMDAC divides the clock. (The divide ratio of the dot clock can also be controlled by mem_config.crtc_freq and mem_config.video_clk_sel.)

QSF counter position. This value determines which bit in the qsfcounter register is used to generate internal shift register reload requests. This field should be set to four less than the log (base 2) of the length of one-half of the shift register, as counted by CRTC_CLCK. A value of zero in this field indicates a shift register of half-length $2^{0+4}$ or 16 CRTC_CLCKs, with a total shift register length of 32 CRTC_CLCKs.

| Memory Configuration | qsfselect value CRTC_CLCK= 1/4 dot clock | qsfselect value CRTC_CLCK= 1/8 dot clock in 8bpp mode |
|---|---|---|
| 1 | 4 | 3 |
| 2 | 4 | 3 |
| 3 | 5 | 4 |
| 4 | 5 | 4 |
| 5 | 6 | 5 |

Figure 71. QSF counter position

**Address Format for Video Control Register Access**

| 31 … 15 | 14 … 7 | 6 … 2 | 1 | 0 |
|---|---|---|---|---|
| Prefix | 0 0 0 0 0 0 1 0 | Register | 0 | 0 |
| 17 | 8 | 5 | 2 | |

| Field | Bits | Entry | Register Selected |
|---|---|---|---|
| Prefix | 31–15 | | See figures 41 and 42. |
| Register | 6–2 | 00001 | hrzc |
| | | 00010 | hrzt |
| | | 00011 | hrzsr |
| | | 00100 | hrzbr |
| | | 00101 | hrzbf |
| | | 00110 | prehrzc |
| | | 00111 | vrtc |
| | | 01000 | vrtt |
| | | 01001 | vrtsr |
| | | 01010 | vrtbr |
| | | 01011 | vrtbf |
| | | 01100 | prevrtc |
| | | 01101 | sraddr |
| | | 01110 | srtctl |
| | | 01111 | sraddr_inc |
| | | 10000 | srtctl2 |
| | | 10001–11111 | Not used |

Figure 72. Address format for video control register access

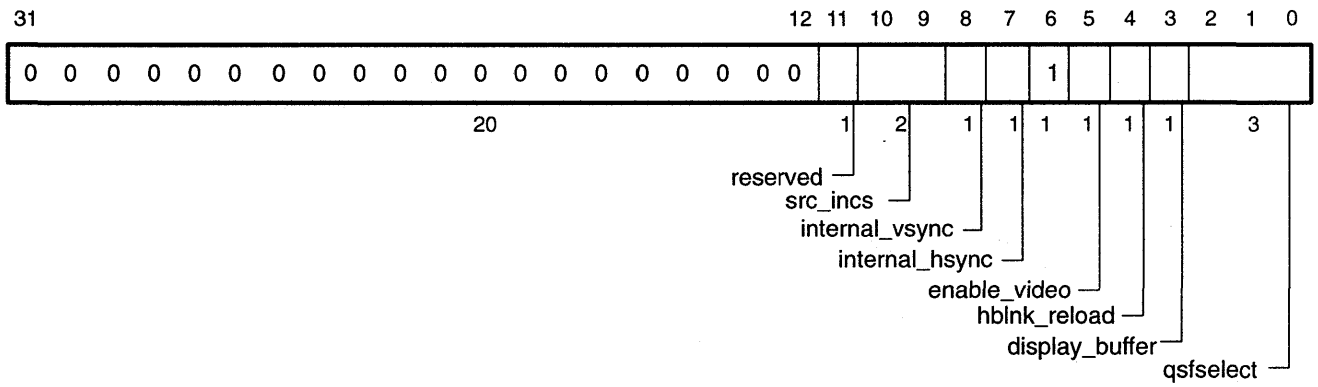
## 4.6. Video Control Registers, continued

| Register | Description/Function |
|---|---|
| *HORIZONTAL TIMING* | |
| hrzc | Horizontal counter. Read only. Specifies the current pixel position along a horizontal sweep; the Power 9100 increments this counter, which is originally set by the host, upon each occurrence of CRTC_CLCK. The value occupies the lower 12 bits of the register, which is set by the Power 9100. |
| hrzt | Horizontal length. Read/write. Specifies the length of a horizontal scan line. The value occupies the lower 12 bits of the register, which is set by the host. The Power 9100 compares the current hrzc value (the current pixel position) to this value to determine when to wrap around. |
| hrzsr | Horizontal sync rising edge. Read/write. Specifies the location along a horizontal sweep which defines the horizontal sync rising edge. The value occupies the lower 12 bits of the register, which is set by the host. |
| hrzbr | Horizontal blank rising edge. Read/write. Specifies the location along a horizontal sweep which defines the horizontal blank rising edge. The value occupies the lower 12 bits of the register, which is set by the host. |
| hrzbf | Horizontal blank falling edge. Read/write. Specifies the location along a horizontal sweep which defines the horizontal blank falling edge. The value occupies the lower 12 bits of the register, which is set by the host. |
| prehrzc | Horizontal counter preload value. Read/write. Specifies the value with which to preload hrzc upon receipt of an internal or external HSYNC– or an external VSYNC–; allows synchronization of the Power 9100 with external video sources, whatever the combination of internal or external delays. The value occupies the lower 12 bits of the register, which is set by the host. Set this register to zero when using only internal syncs. |
| *VERTICAL TIMING* | |
| vrtc | Vertical counter. Read only. Specifies the current line position along a vertical sweep; the Power 9100 increments this counter upon each occurrence of HSYNC–. The value occupies the lower 12 bits of the register, which is set by the Power 9100. |
| vrtt | Vertical length. Read/write. Specifies the number of lines in a vertical sweep. The value occupies the lower 12 bits of the register, which is set by the host. The Power 9100 compares the current vrtc value (the current line position) to this value to determine when to wrap around. |
| vrtsr | Vertical sync rising edge. Read/write. Specifies the location along a vertical sweep which defines the vertical sync rising edge. The value occupies the lower 12 bits of the register, which is set by the host. |
| vrtbr | Vertical blank rising edge. Read/write. Specifies the location along a vertical sweep which defines the vertical blank rising edge. The value occupies the lower 12 bits of the register, which is set by the host. |
| vrtbf | Vertical blank falling edge. Read/write. Specifies the location along a vertical sweep which defines the vertical blank falling edge. The value occupies the lower 12 bits of the register, which is set by the host. |
| prevrtc | Vertical counter preload value. Read/write. Specifies the value with which to preload vrtc upon receipt of an internal or external VSYNC–; allows synchronization of the Power 9100 with external video sources, whatever the combination of internal or external delays. The value occupies the lower 12 bits of the register, which is set by the host. Set this register to zero when using only internal syncs. |
| *SCREEN REPAINT* | |
| srtctl2 | Screen refresh timing control. Read/write. Specifies controls for screen refresh, as set by the host. Figure 75 defines this register. |
| srtctl | Screen refresh timing control. Read/write. Specifies controls for screen refresh, as set by the host. Figure 74 defines this register. |
| qsfcounter | QSF counter. Read only. Used to determine when to generate a shift register load operation. It is a duplicate of the QSF signal from the VRAMs. It keeps track of which part of the SAM is being shifted out. It is loaded with a zero after every read transfer and incremented by CRTC_CLCK. |

Figure 73. Video control registers

## 4.6. Video Control Registers, continued

**Screen Repaint Timing control (srtctl) Register**

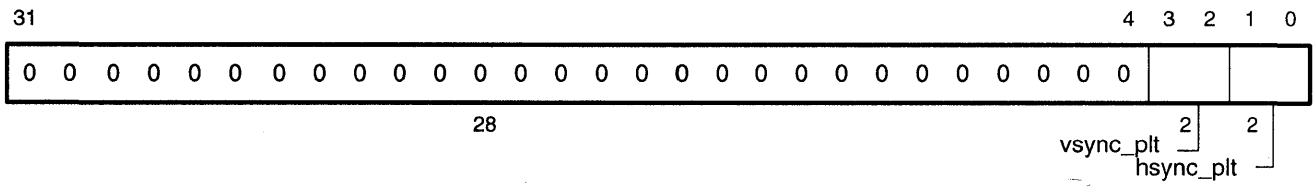



| Field | Bits | Contents |
|---|---|---|
| reserved | 11 | Reserved, set to 0 |
| src_incs | 10–9 | Amount to increment DRADDR register (see section 8.1.4)<br>00 = 256<br>01 = 512<br>10 = 1024 |
| internal_vsync | 8 | Internal vertical synchronization:<br>0 = Select external VSYNC–<br>1 = Select internal VSYNC– |
| internal_hsync | 7 | Internal horizontal synchronization:<br>0 = Select external HSYNC–<br>1 = Select internal HSYNC– |
| enable_video | 5 | Enable video:<br>0 = Blanks asserted<br>1 = Normal operation |
| hblnk_reload | 4 | Screen repaint mode (see section 8.1.9):<br>0 = Normal (split shift mode)<br>1 = Restricted (HBLNK– reload mode) |
| display_buffer | 3 | Buffer for display,:Any change to this field takes effect only after the next VBLNK–.<br>0 = Buffer 0<br>1 = Buffer 1 |
| qsfselect | 2–0 | QSF counter position. See section 4.6 |

Figure 74. Screen repaint timing control (srtctl) register

## 4.6. Video Control Registers, continued

**Screen Repaint Timing Control (srtctl2) Register**




| Field | Bits | Contents |
|---|---|---|
| hsync_plt | ~~3–2~~ 1-0 | External HSYNC polarity control.<br>00 = Low true polarity<br>01 = High true polarity<br>10 = Forced low<br>11 = Forced high |
| vsync_plt | ~~1–0~~ 3-2 | External HVYNC polarity control.<br>00 = Low true polarity<br>01 = High true polarity<br>10 = Forced low<br>11 = Forced high |

Bit fields conflict with picture

*Corrected - now accurate based on later Writer's Doc.

Figure 75. Screen repaint timing control (srtctl2) register

## 4.7. VRAM Control Registers

The VRAM control registers, which specify the timings and configurations associated with the VRAMs, are summarized in figure 76. Figure 77 defines the specific address format for accessing these registers; aside from the standard general format, bits 8 through 7 always contain the value 3, and bits 6 through 2 select the VRAM control register.

### 4.7.1. MEMORY CONFIGURATION REGISTER

The memory configuration register (mem_config) defines how the frame buffer is configured, as defined in figure 78. See chapter 7 for definitions of the memory configurations.

Setting mem_config.vram_read_sample = 1 and mem_config.slow_host_hifc = 0 is not supported.

### 4.7.2. REFRESH PERIOD REGISTER

The refresh period (rfperiod) register defines the length of the refresh period, which is the number of cycles between memory refreshes. The register field containing the refresh period is the least significant 10 bits of the register; the rest of the register must be zeroed. A system reset initializes the contents of this register to 0x3FF.

### 4.7.3. REFRESH COUNT REGISTER

The refresh count (rfcount) register keeps track of the number of cycles between refreshes. The Power 9100 loads this register with the value contained in the rfperiod register, and then decrements the rfcount register contents on each SYSCLK. When the rfcount register contents reach zero, the Power 9100 reloads the register with the current value in the rfperiod register and makes a refresh request. The register field containing the refresh count is the least significant 10 bits of the register; the rest of the register contains zeroes.

### 4.7.4. RAS LOW REGISTER

The RAS low maximum (rlmax) register specifies the maximum amount of time that the RAS– signal can be asserted. The register field containing the time is the least significant 10 bits of the register; the rest of the register contains zeroes. A system reset initializes the contents of this register to 0x3FF.

### 4.7.5. LOW CURRENT REGISTER

The RAS low current (rlcur) register controls the amount of time that the RAS– signal can be held low when it is asserted. When the Power 9100 asserts the RAS– signal, it also loads the rlcur register with the value in the rlmax register. The Power 9100 then decrements the contents of the rlcur register on each SYSCLK until either RAS– is deasserted or the rlcur value reaches zero. If the register contents reach zero while RAS– remains asserted, the Power 9100 performs a refresh sequence. The register field containing the time count is the least significant 10 bits of the register; the rest of the register contains zeroes.

### 4.7.6. POWER-UP CONFIGURATION REGISTER

The PU_CONFIG register is a read-only copy of the value that is on the 32-bits of the frame buffer data bus when the system is reset. As well as setting certain default modes within the Power 9100 it can also be used to control certain BIOS settings.

| Register | Function |
|---|---|
| mem_config | Memory configuration. Read/write. Specifies how the frame buffer is configured. |
| rfperiod | Refresh period. Read/write. Controls the frequency of frame buffer refresh. |
| rfcount | Refresh counter. Read only. Controls the number of frame buffer refreshes. |
| rlmax | RAS low maximum. Read/write. Specifies the maximum amount of time the Power 9100 can assert the RAS– signal. |
| rlcur | RAS low current. Read only. Controls the maximum amount of time the RAS– signal is asserted. |
| alt_read_bank | Alternate bus read bank select. Read/write, Controls banking of the frame buffer. |
| alt_write_bank | Alternate bus read bank select. Read/write Controls banking of the frame buffer. |

Figure 76. VRAM control registers

## 4.7. VRAM Control Registers, continued

**Address Format for VRAM Control Register Access**

| 31–15 | 14–7 | 6–2 | 1–0 |
|---|---|---|---|
| Prefix | 0 0 0 0 0 0 1 1 | Register | 0 0 |
| 17 | 8 | 5 | 2 |

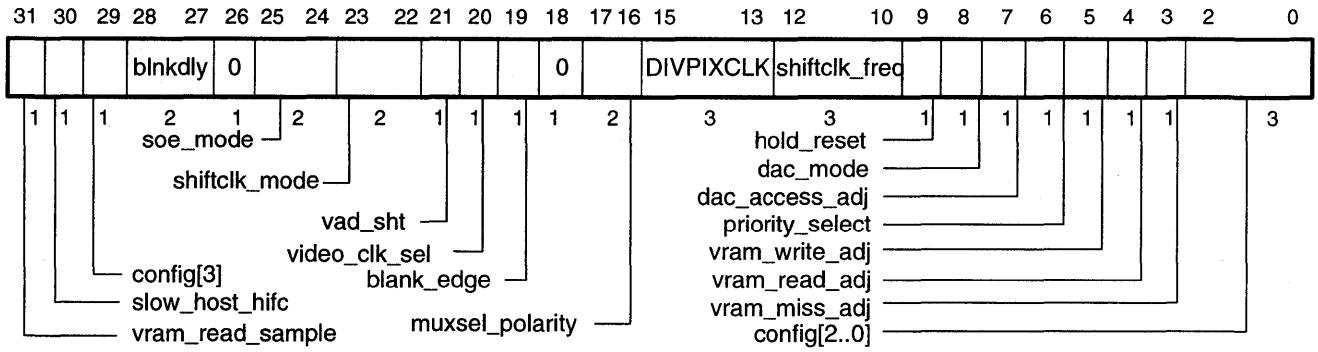
| Field | Bits | Entry | Register Selected |
|---|---|---|---|
| Prefix | 32–15 | | See figures 41 and 42. |
| Register | 6–2 | 00001 | mem_config |
| | | 00010 | rfperiod |
| | | 00011 | rfcount |
| | | 00100 | rfmax |
| | | 00101 | rlcur |
| | | 00110 | pu_config |
| | | 00111–11111 | Not used |

Figure 77. Address format for VRAM control register access

mem_config = Prefix + 0000001, 0000100
Prefix + 0 1 8 4
Prefix = baseAddr from Node + ~~Endian~~ 0

## 4.7. VRAM Control Registers, continued

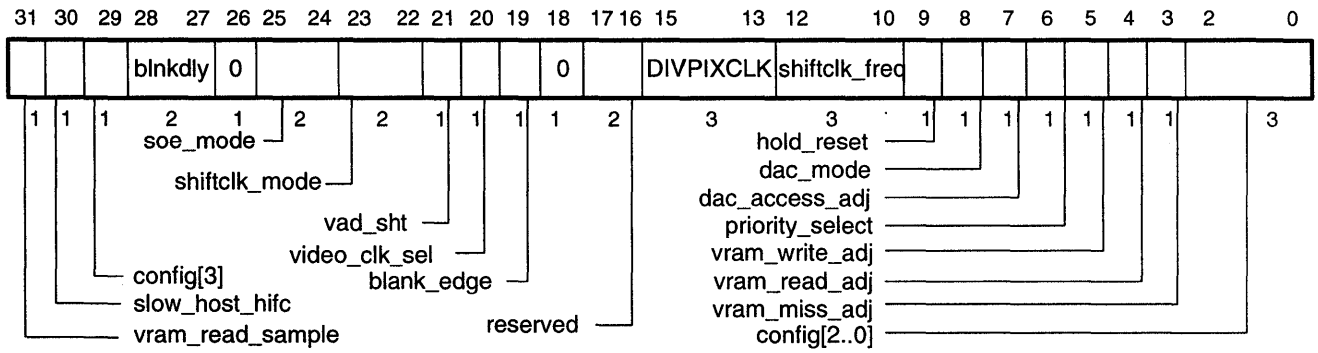
**Memory Configuration (mem_config) Register**




| Field | Bits | Contents |
|---|---|---|
| config[2..0] | 2–0 | Low order 3 bits of memory configuration. See Figures 105 through 112 |
| vram_miss_adj | 3 | Adjustment for row miss timing template, see Figures 113 and 114 |
| vram_read_adj | 4 | Adjustment for read timing template, see Figures 115 and 116 |
| vram_write_adj | 5 | Adjustment for VRAM write timing template, see Figures 119 and 120 |
| priority_select | 6 | 0 = Coprocessor has lower priority than drawing engine, 1 = higher |
| dac_access_adj | 7 | Enable RAMDAC back to back transfer checking, see Figures 139 and 140 |
| dac_mode | 8 | Controls dac read/write signalling mode, see FIgures 139 and 140 |
| hold_reset | 9 | 1 Forces memory and video into reset state, 0 for normal operation |
| shiftclk_freq | 12–10 | Sets the frequency of the RAMDAC latch clock state machine.<br>Normally set to 000 because the RAMDAC generates the divided clock.<br>000 = pixel_clk / 1 001 = pixel_clk / 2 010 = pixel_clk / 4<br>011 = pixel_clk / 8 100 = pixel_clk / 16 |
| crtc_freq | 15–13 | Sets the frequency of the internal CRTC divided dot clk.<br>Normally set to 000 because the RAMDAC generates the divided clock.<br>000 = pixel_clk / 1 001 = pixel_clk / 2 010 = pixel_clk / 4<br>011 = pixel_clk / 8 100 = pixel_clk / 16 |

Figure 78. Memory configuration (mem_config) register (1 of 2)

**Memory Configuration (mem_config) Register (continued)**




| Field | Bits | Contents |
|---|---|---|
| reserved | 17–16 | Reserved, set to 0. |
| | 18 | Not used; set to 0. |
| blank_edge | 19 | Clock edge that HSYNC and VSYNC are synchronized to<br>0 = Falling edge of video_clk 1 = Rising edge of video_clk |
| video_clk_sel | 20 | Selects the video_clk source for the video section.<br>0 = PIXCLK 1 = DIVPIXCLK |
| vad_sht | 21 | Additional divide for Video Transfer address (see section 8.1.4)<br>0 = Don't divide by two 1 = divide by two |
| sjftckl_mode | 23–22 | Sets timing patterns for shift clocks (see figure 126):<br>00 = 1 bank style 01 = 2 bank style 10 = 4 bank style |
| soe_mode | 25–24 | Sets timing patterns for serial output enables:<br>00 = 1 bank style 01 = 2 bank style 10 = 4 bank style |
| blnkdly | 28–27 | Controls delay for blank generation in units of shiftclk_freq.<br>00 = 0 shiftclk_freq delay 01 = 1 shiftclk_freq delay<br>10 = 2 shiftclk_freq delay 11 = 3 shiftclk_freq delay |
| config[3] | 29 | High order bit of memory configuration. See Figures 105 through 112 |
| slow_host_hifc | 30 | Slow host interface adjust. |
| vram_read_sample | 31 | Adjustment for read timing template, see Figures 115 through 118. |

Figure 78. Memory configuration (mem_config) register (2 of 2)

# Chapter 5. Commands

Figure 79 summarizes the commands that initiate functions on the Power 9100; it also summarizes the command functions and indicates the section of this data book that describes each command. All commands are accessed via the general user register address format, with the exception of the load coordinates command.

Note that Pixel8 and Blit cannot perform 24-bit processing with patterns. For Pixel8 and Blit, the X address must be programmed in BYTE.

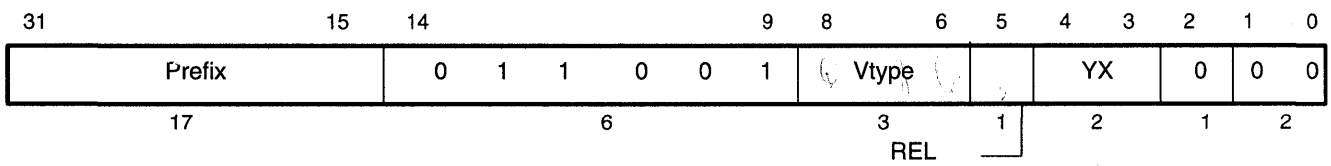
| Command | Function | Accessed via | Described in section |
|---|---|---|---|
| Load Coordinates | Load quad coordinates via shorthand method | user register load coordinates format | 5.1 |
| Quad | Draw a quadrilateral. | user register address format | 5.2 |
| Blit | Copy a rectangular area of the display form one screen location to another. | user register address format | 5.3 |
| Pixel8 | Transfer one word of pixel data from a linear host memory array to a rectangular display memory array in the frame buffer. | user register address format | 5.4 |
| Pixel1 | Transfer up to 32 pixels from a a linear host memory array to a rectangular display memory array in the frame buffer. | user register address format | 5.5 |
| Next_pixels | Advance the drawing area for a pixel8 or pixel1 operation. | user register address format | 5.6 |

Figure 79. Command summary

## 5.1. Load Coordinates Command

Using the load coordinates command allows you to provide the minimal number of coordinates required for a drawing operation. You specify the type of quadrilateral to draw and specify only the required coordinates. The load coordinates command enables loading of coordinates for quadrilaterals and degenerate quadrilaterals in a shorthand notation that performs the drawing operation with the fewest data transfers. It also facilitates drawing polylines and meshed quads. The load coordinate command is meaningful for write operations only. Figure 80 defines the specific user register address format for accessing the load coordinates pseudo-registers.

**Address Format for Load Coordinates Pseudo-register Access**




| Field | Bits | Entry | Contents |
|---|---|---|---|
| Prefix | 31–15 | | See figures 41 and 42. |
| Vtype | 8–6 | 000 | Point (specify one coordinate) |
| | | 001 | Line (specify two coordinates) |
| | | 010 | Triangle (specify three coordinates) |
| | | 011 | Quadrilateral (specify four coordinates) |
| | | 100 | Rectangle (specify four coordinates) |
| | | 101 | not used |
| | | 110 | not used |
| | | 111 | not used |
| REL | 5 | 0 | Perform screen addressing relative to the window offset |
| | | 1 | Perform screen addressing relative to the previous vertex |
| YX | 4–3 | 00 | Not used |
| | | 01 | Load or read 32-bit X value |
| | | 10 | Load or read 32-bit Y value |
| | | 11 | Load or read 16-bit X value (high 16 bits) and 16-bit Y value (low 16 bits) |

Figure 80. Address format for load coordinates pseudo-register access

## 5.2. Quad Command

The quad command draws and fills the quadrilateral defined by the vertices specified in the coordinate registers (see section 4.4.1). This command sets the busy bit in the status register and performs the drawing operation, scan line by scan line, applying the current pattern, with the left and right Bresenham engines establishing the outlines of the left and right boundaries as the fill engine fills the quadrilateral.

The quad command requires the four vertices of the quadrilateral (x[0],y[0]; x[1],y[1]; x[2],y[2]; and x[3],y[3]). Four vertices are also required for a triangle, line and point. For a triangle, two vertices specify the same coordinate; for a point, all four vertices specify the same coordinate. (You can also use the load coordinates pseudo-register access, see figure 80).

This command can fail for either of two reasons: the quad is concave or at least one coordinate is out of range of the drawing engine. If the quad cannot be drawn, quad sets the quad_sw bit in the status register (see section 4.4.2) and does not perform the operation.

Figure 81 defines the address format for initiating the quad command.

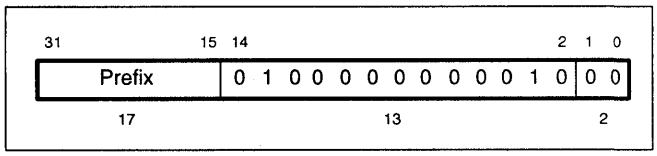



Figure 81. Quad command address format (prefix: see figures 41 and 42)

## 5.3. Blit Command

The blit command copies a rectangular area of the display from one screen location to another. When the original copied area (the source) and the new area (the destination) overlap, the Power 9100 ensures accuracy by performing the copy as though the entire source were moved offscreen and then moved onscreen to the destination.

The blit command requires the vertices of the upper left (x[0],y[0]) and lower right (x[1],y[1]) corners of the source, plus the vertices of the upper left [x[2],y[2]) and lower right (x[3],y[3]) corners of the destination. Both areas must be the same size; specifying different sized areas produces undefined results. If the destination is out of range, blit sets the blit_software bit in the status register (see section 4.4.2) and does not perform the copy. Otherwise, blit determines the direction of the blit operation and initiates the copy.

Figure 82 defines the address format for initiating the blit command.

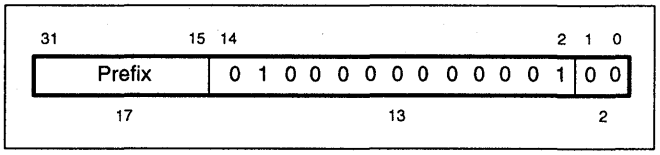



Figure 82. Blit command address format (prefix: see figures 41 and 42)

## 5.4. Pixel8 Command

The pixel8 command transfers one word of pixel data from a linear host memory array to a rectangular display memory array in the frame buffer. This operation is useful for writing pixels of various colors to the screen. The array in host memory must be padded out to a 32-bit boundary.

The pixel8 command requires the left edge of the block to be transferred (x[0]), the point at which to begin the transfer (x[1],y[1]), the right edge of the block to be transferred (x[2]), and the increment by which to increase the *y* coordinate at the end of the scan line (y[3]). The pixel8 operation is initiated with a write operation.

This command handles a busy status by holding the processor. Figure 83 defines the address format for initiating the pixel8 command. The pixel8 command can also be accessed by the alternate I/O register encoding also shown in figure 83.

Each pixel8 command writes one word into the frame buffer unless the word being written overlaps the end of the current scan line. Excess data is saved in the pixel8_reg.

Figure 84 illustrates the pixel8 operation with an example.

X coordinates for the pixel8 command must be scaled by the software so that is correctly represents the appropriate number of bytes.

There are two address formats for the pixel8 command. The first format is provided for backwards compatibility with the W9000 product. The second form allows the use of memory to memory string move instructions since it ignores the destination offset. Enough bits are provided to allow up to 8KBs to be moved in a single string move instruction, this is enough for any single scan line.

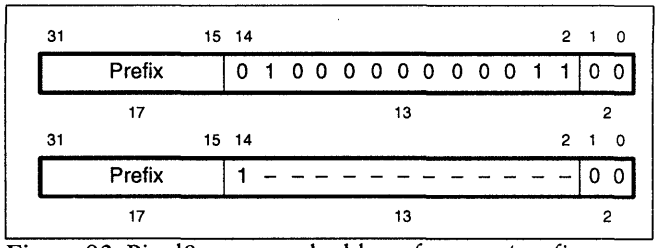



Figure 83. Pixel8 command address formats (prefix: see figures 41 and 42)

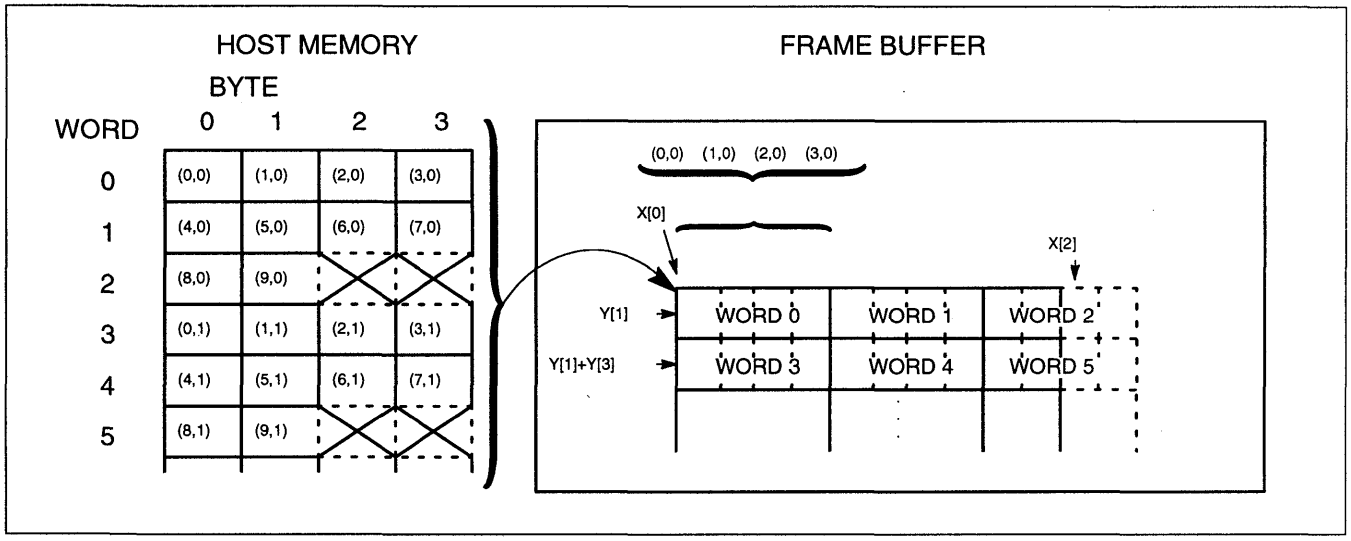



Figure 84. Sample pixel8 host memory to frame buffer transfer

## 5.5. Pixel1 Command

The pixel1 command expedites writing characters to the screen. The command transfers data for up to 32 pixels from a linear host memory array to a rectangular display memory array in the frame buffer.

The pixel1 command requires the left edge of the block to be transferred (x[0]), the point at which to begin the transfer (x[1],y[1]), the right edge of the block to be transferred (x[2]), and the increment by which to increase the y coordinate at the end of the scan line (y[3]). It also requires the total number of pixels to write (specified in the address as the number of pixels), and the pixel data.

### 5.5.1. PIXEL1 COLOR SELECTION

Each bit of the data presented to the pixel1 command is expanded into a full pixel of color information. If transparent pixel1 mode is disabled (raster.pixel1_transparent = 0) then a 0 bit is expanded into the color contained in the color[0] register and a 1 bit is expanded into the color contained in the color[1] register.

If transparent pixel1 mode is enabled (raster.pixel1_transparent = 1) a zero bit leaves the destination unmodified and a one bit is expanded into the color contained in the color[1] register.

Figure 85 defines the address format for initiating the pixel1 command.

This command handles a busy status by holding the processor.

Unlike the pixel8 command, the pixel1 command does not discard unused bits at the end of a scan line. If more bits were specified then fit on the current scan line, the system will automatically continue the pixel1 operation on the next scan line with the remaining bits.

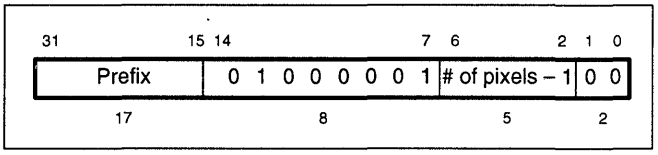



Figure 85. Pixel1 command address format (prefix: see figures 41 and 42)

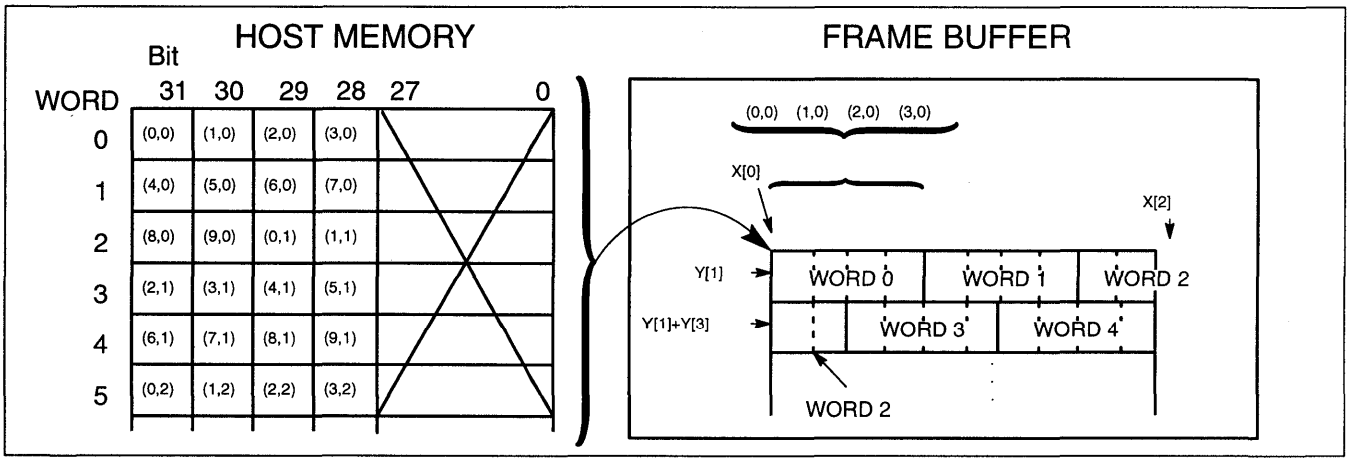



Figure 86. Sample pixel1 host memory to frame buffer data expansion and storage

## 5.6. Next_pixels Command

The next_pixels command specifies the next area of the frame buffer to load for consecutive pixel8 or pixel1 drawing operations. It advances these drawing operations to the next step in a sequence of operations.

The next_pixels command requires the right edge of the previous block transferred (x[2]) and the top edge of the previous block transferred (y[2]), plus the width of the new transfer (from the data bus).

Figure 87 defines the address format for initiating the next_pixels command.

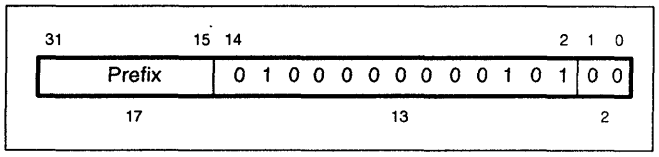



Figure 87. Next_pixels command address format (prefix: see figures 41 and 42)

## 5.7. Drawing With the Power 9100

This section describes Power 9100 operation; it presents sample code fragments that perform basic Power 9100 functions. All code samples are written in C and Q-code. Q-code macros allow direct access to Power 9100 hardware functions. Sample drawing operations are illustrated with displays from output from the functional simulator (predictor); the programs were prepared with the predictor and executed on the hardware.

For definition of the programming tools for the Power 9100, refer to the *Power 9100 Programmer's Reference Manual*, which includes descriptions of the predictor and the Q-code used in this chapter.

The Power 9100 draws quadrilaterals according to the basic rules and in the modes summarized in the following sections.

### 5.7.1. LEGAL QUADRILATERALS

The Power 9100 draws quadrilaterals and degenerate quadrilaterals (triangles, lines, and points). It handles all quadrilaterals except concave quadrilaterals. A quadrilateral is concave if any scan line crosses more than two boundaries. A request for a concave quadrilateral generates an error. Figure 88 presents examples of non-concave quadrilaterals, which the Power 9100 accepts. Figure 89 presents examples of concave quadrilaterals, which the Power 9100 rejects.

### 5.7.2. DRAWING MODES

The Power 9100 supports two drawing modes: X11 and oversized. (The drawing mode is selected via the raster register defined in section 4.5.5.)

The X11 drawing mode strictly conforms to the drawing rules employed by X-windows:

1. The boundaries of the quadrilateral are considered to be infinitely thin.
2. Only those pixels whose centers are completely within the boundaries of the quadrilateral are touched.
3. If the center of a pixel is exactly on the boundary, it is touched if and only if the interior is immediately to its right (in the increasing x direction).
4. A pixel with its center on a horizontal boundary is handled as a special case; it is touched if and only if the interior is immediately below (in the increasing y direction) and the boundary is immediately to its right (in the increasing x direction).

In addition to touching the pixels touched according to the X11 rules, the oversized drawing mode also touches the pixels along the boundary of the quadrilateral as drawn by the classic Bresenham algorithm. In this mode, the Power 9100 draws a line or a point as an infinitely thin quadrilateral. In the X11 drawing mode, no pixels would be touched because the quadrilateral has no interior. In oversized mode, however, using the classic algorithm that assumes that a line from [x0,y0] to [x1,y1] touches the same pixels as a line from [x1,y1] to [x0,y0], the pixels are touched. This method also satisfies the X11 definition of a thin line (from Gettys and Scheifler): "If a line is drawn unclipped from [x1,y1] to [x2,y2] and if another line is drawn unclipped from [x1+dx,y1+dy] to [x2+dx,y2+dy], a point [x,y] is touched by drawing the first line only if the point [x+dx,y+dy] is touched by drawing the second line."

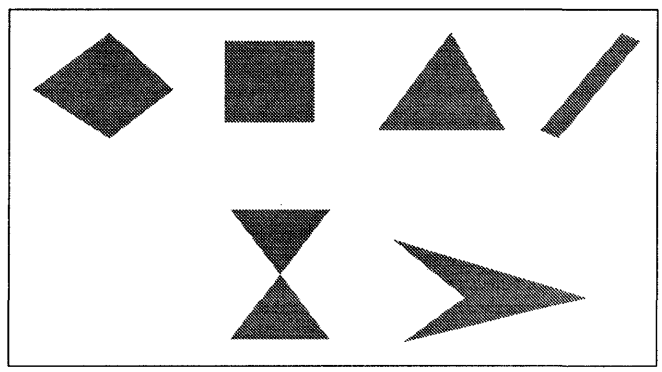


Figure 88. Samples of non-concave (legal) quadrilaterals

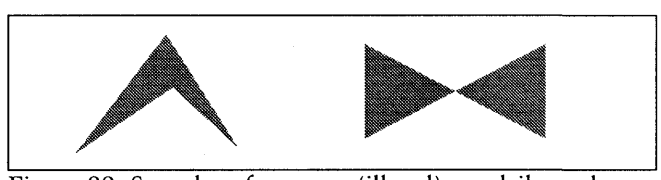


Figure 89. Samples of concave (illegal) quadrilaterals

### 5.7.3. BASIC QUAD DRAWING METHODS

To draw quadrilaterals, supply all four points of each quadrilateral in the device coordinate registers (to specify a degenerate quadrilateral, repeat one or more coordinates), or specify the quad type and its vertices via the load coordinates pseudo-registers, which enable you to specify fewer coordinates and draw a new quadrilateral sharing the last vertices of the previously drawn quadrilateral. In either case, you issue the quad command to execute the draw.

Unlike the other drawing commands, the X values for the quad command are not scaled into bytes. They are specified in pixels.

### 5.7.4. NEGATIVE COORDINATES

The Power 9100 uses data in two's complement format. Any portion of a draw operation that is in a negative quadrant, however, is not drawn. This applies to quad draw, blit destination, pixel1, and pixel8. When the source for a blit contains negative coordinates, the results are indeterminate.

### 5.7.5. FLOW CONTROL

The host processor issues read and write operations which the Power 9100 accepts. Most host buses require that any read or write operation be guaranteed to complete in a certain amount of time (for example, 10 µseconds, or about 500 cycles at 50 MHz). Because some Power 9100 operations, such as clearing the screen, can require hundreds of thousands of cycles, the host processor must be able to issue sequences of these operations in a manner that does not violate the timeout constraint. The Power 9100 handles such problems by a specific system software protocol. Protocol violations are not detected.

Accesses to non-drawing engine registers cause no problems and can be made without regard to the current state of the drawing engine.

Accesses to drawing engine registers are valid only when the drawing engine is idle (the status register busy bit is zero).

The sequence to initiate the quad and blit drawing operations comes in two flavors. The first flavor is when the software does not know what the previous drawing operation was or whether it has completed or not. In this case the software must ensure that the drawing engine is idle (with the status register busy bit set to zero) sometime before requesting the initiation of a quad or a blit operation by reading from the appropriate address.

The second flavor comes when the software knows that the previous operation is a quad or a blit and the next operation is also a quad or a blit (warning: this case only applies to quad followed by quad or blit followed by blit but not to quad followed by blit or blit followed by quad). In this case the software can request initiation of the subsequent quad or blit simply by reading the appropriate command address and checking the status.issued_qbN bit. If the drawing engine is busy, the Power 9100 ignores the request for the operation. The software is responsible for reissuing the request. It can either poll by continuously reissuing the request or it can request an interrupt when the drawing engine is ready.

The pixel1 and pixel8 operations wait for the drawing engine to become idle; sequences of these operations can be executed without checking the status of the drawing engine. It is assumed that any preceding operation is short. When this is not certain, the software should wait for the drawing engine to be idle (with the status register busy bit set to zero) before issuing the command.

Table 90 summarizes which status register bit should be checked before initiating specific operations or register writes.

### 5.7.6. EXCEPTION HANDLING

The Power 9100 determines whether a requested drawing operation is legal and within range and sets bits in the status register to indicate the result. Software can check the status register to detect an exception.

| Before performing: | Check status register: |
|---|---|
| Parameter engine register write | No status check required |
| Drawing engine register write | Busy bit (bit 30) for zero |
| Quad draw | Issue_qbN bit (bit 31) for zero |
| Blit | Issue_qbN bit (bit 31) for zero |
| First pixel1 or pixel8 in a series (when switching from reads to writes) | Busy bit (bit 30) for zero |

Figure 90. Status register checks

# Chapter 6. Host Interface

## 6.1. Signal Description Conventions

Each signal is identified with a specific type that indicates input/output status, etc.

### 6.1.1. INPUT SIGNALS

Signals of this type are never driven by the Power 9100.

### 6.1.2. OUTPUT SIGNALS

Signals of this type are always driven to a valid voltage level. The output driver is not capable of being placed into the high-impedance state. If shown as OUTPUT(HIGH) then the signal is driven HIGH whenever reset is asserted.

### 6.1.3. TRI-STATED SIGNALS

Signals of this type are can be driven high or low or placed into the high-impedance state. Generally these signals are forced into a tri-stated condition during a reset: this is indicated as TRI-STATED(TRI).

### 6.1.4. BIDIRECTIONAL SIGNALS

Signals of this type are can be either inputs or outputs.

## 6.2. Host Bus Interface

The Power 9100 directly supports two separate buses: VL and PCI. Other buses, such as 486 CPU, can easily be supported with small amounts of additional glue logic.

The bus mode is configured at reset time and cannot be changed without a complete reset. See section 3.3.1 for more information on reset configuration.

The following sections document the timing and functionality of the pins that are specific to each bus interface.

### 6.2.1. I/O ADDRESS DECODING

The PCI specification specifically states that the I/O address space is a full 32 bits. All address decodes in the Power 9100 I/O or memory are done as a full 32 bits. In VL, the host bus interface logic is required to ensure that the top 16-bits of an I/O address are ignored (i.e., treated as a zero).

The same situation is not true for memory addresses. The host interface logic assumes that all 32 bits are to be decoded.

### 6.2.2. DAC SHADOWING

DAC shadowing is an emulation mode involving DAC write operations. This mode is used only with the VGA feature connector. When this mode is enabled (CONFIG[4].SHADOW_DAC = 1) and the Power 9100 is in emulation mode (CONFIG[65].MODESELECT = 1) then write operations to the DAC addresses cause no visible response on the host bus but are actually performed internally, consequently the system is responsible for ensuring flow control for these operations. Read operations are responded to normally.

## 6.3. PCI Bus Operation

The Power 9100 directly supports the 32-bit version of the PCI bus without any external logic. The Power 9100 never becomes a bus master and therefore does not use all of the PCI signals.

### 6.3.1. PCI BUS SIGNAL LIST

The PCI bus signal list is shown in figure 91.

### 6.3.2. PCI BUS TIMINGS

Timings for the PCI bus are divided into groups as shown in figure 92.

### 6.3.3. PCI BUS CONFIGURATION REGISTERS

The PCI configuration address space is used to access the Power 9100 configuration registers. The configuration registers have been carefully arranged to conform to the PCI configuration register space standard.

| Signal | Type | Description |
|---|---|---|
| AD[31..0] | Input/Output | The address and data bus. This is tri-stated when RESET– is asserted. |
| C/BE–[3..0] | Input | Command an byte enable bus. |
| PAR | Input/Output | Even parity for the AD[31..0] and C/BE–[3..0] buses. The Power 9100 generates correct parity for read transfers. It does not check for correct parity on write transfers. |
| FRAME– | Input | Cycle frame. Indicates the beginning of a transaction. |
| TRDY– | Tri-stated (TRI) | Target ready. Indicates that the Power 9100 is ready to accept the transaction. |
| IRDY– | Input | Initiator ready. Indicates that the bus master is ready. |
| STOP– | Tri-stated (TRI) | Indicates that the Power 9100 is requesting the current transaction to stop. |
| IDSEL | Input | An alternate chip select for initialization transfers. |
| DEVSEL– | Tri-stated (TRI) | Asserted by the Power 9100 to claim a transaction. |
| IRQ | Tri-stated (TRI) | Interrupt request. |
| BCLK | Input | Bus clock. |

Figure 91. PCI bus signal list

| Group Type | Description |
|---|---|
| GROUP 1: Input Signals | C/BE–[3..0], FRAME–, IRDY–, and IDSEL. As measured relative to the 1.5V level of the rising edge of BCLK. Setup times are 7 ns and hold times are 0 ns. |
| GROUP 2: Input/Output signals | AD[31..0] and PAR. When operating as input signals they are the same as group 1: 7ns setup and 0ns hold. When operating as outputs: The turn-on time is a minimum of 2ns. The turn-off time is a maximum of 28ns (minimum is the same as the output valid time). The minimum output valid time is 2ns. |
| GROUP 3: Clock signal | A maximum frequency of 33Mhz with a minimum duty cycle of 40% for high and low portions. |

Figure 92. Timings for PCI bus

## 6.3. PCI Bus Operation, continued

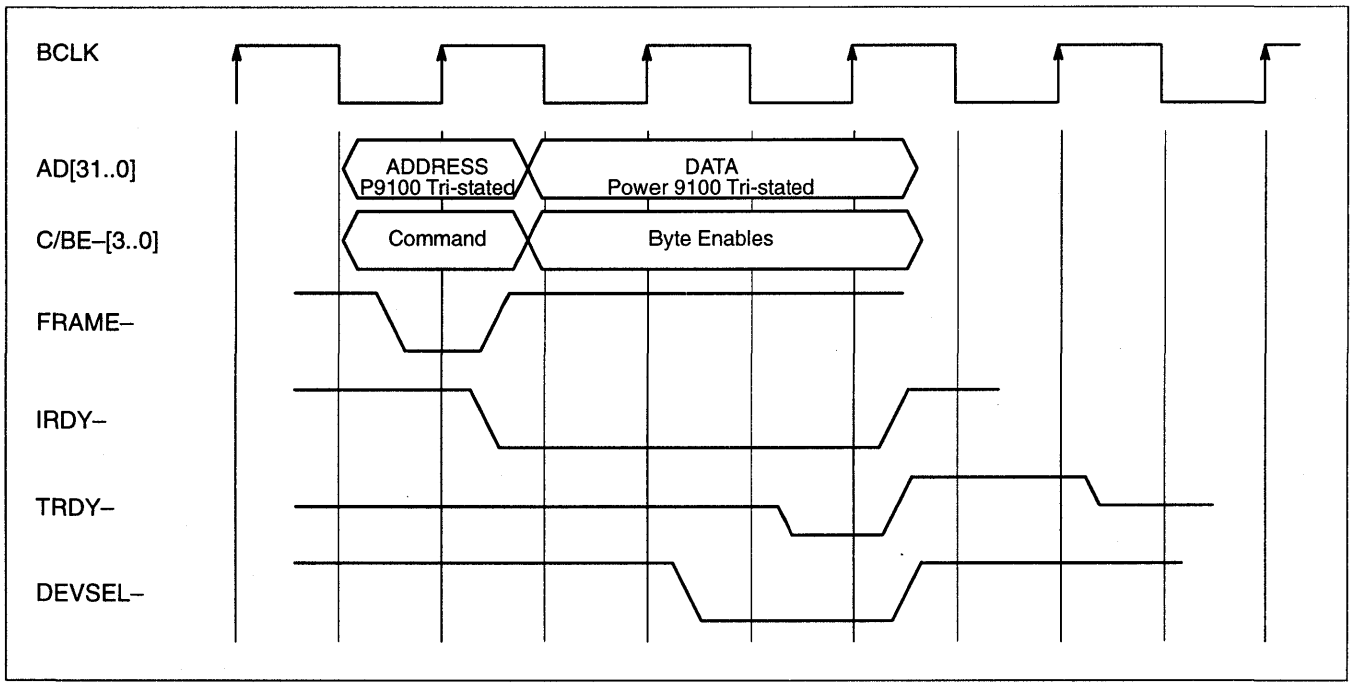



Figure 93. PCI Bus Memory Write Operation, 0 Wait States

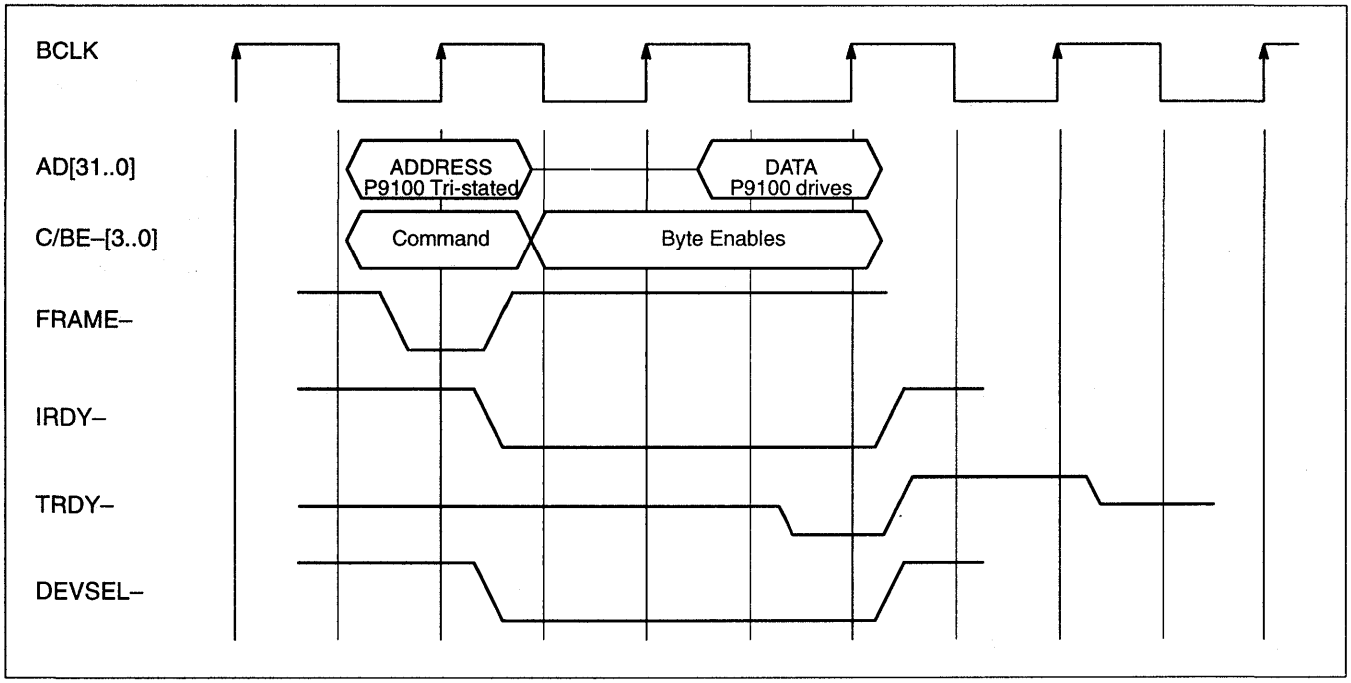



Figure 94.PCI Bus Memory Read Operation, 0 Wait States

## 6.4. VL Bus Operation

The Power 9100 supports all VL bus functionality.

The Power 9100 does not use burst mode on the VL bus. Therefore the BLAST# and BRDY# signals of the VL bus connector remain unconnected. Also, fast write mode is not supported.

### 6.4.1. VL BUS SIGNAL LIST

The VL bus signal list is shown in figure 95.

### 6.4.2. VL BUS TIMINGS

Timings for the VL bus are divide into groups as shown in figure 96.

### 6.4.3. VL BUS OPERATIONS WAVEFORMS

The LRDY– waveform shows the state of the output driver of the Power 9100. Since this signal is connected to VDD via a pull-up on the system bus, its actual state will be high when there are no buffers trying to actually drive it (show as high-impedance state on the timing diagrams).

| Signal | Type | Description |
|---|---|---|
| ADR[31..2] | Input | The address bus. |
| DATA[31..0] | Input/Output | The bidirectional data bus. |
| BE[3..0]– | Input | The byte enable lines |
| M/IO– | Input | Memory/IO address indicator |
| W/R– | Input | Read or write indicator |
| D/C– | Input | Data or code access indicator |
| ADS– | Input | Address strobe. This indicates that start of a VL bus cycle |
| LDEV– | Output (high) | Local device. This is driven by the Power 9100 to "claim" a VL bus transfer. It is driven combinatorially from ADR[31..2], M/IO– and D/C–. |
| LRDY– | Tri-stated (TRI) | Local Ready. Indicates that the Power 9100 has completed its portion of the transfer. |
| RDYRTN– | Input | Ready Return. Indicates when the system considers a read transfer to have completed. |
| IRQ | Tri-stated (TRI) | Interrupt request |
| RESET– | Input | System Reset |
| LCLK | Input | Local Bus clock. |

Figure 95. VL bus signal list

| Group Type | Description |
|---|---|
| GROUP 1: input signals to the Power 9100 | ADR[31..2], BE[3..0]–, M/IO–, W/R–, D/C–, ADS–, RESET–, RDYRTN–. These signals have a maximum setup time of 7ns. and a minimum hold time of 3ns. The timings are referenced to the 1.5V level of the rising edge of the LCLK signal. |
| GROUP 2: The LRDY signal | LRDY–. This signal has a maximum output delay of 10ns and a minimum output valid time of 3ns into a maximum load of 100pF. The timings are referenced to the 1.5V level of the rising edge of the LCLK signal. This signal also becomes tri-stated during certain operations. The turn-off time is a maximum of 7ns and is referenced to the 1.5V level of the falling edge of LCLK. |
| GROUP 3: The Input/Output data bus | DATA[31..0]. During write transfers (i.e., an input to the Power 9100) the timings for this bus are the same as the Group 1 signals: setup = 7ns, hold = 3ns. For read transfers (i.e., an output from the Power 9100), max output delay = 15ns into a 100pF load, turn off time must no more than 7ns. |
| GROUP 4: The LDEV signal | LDEV–. This signal is generated combinatorially from ADR[31..2], M/IO– and D/C–it has a maximum output delay of 15ns and a minimum output valid time of 7ns into a maximum load of 20pF. The timing is referenced relative to a change in ADR[31..2], M/IO– and D/C–. |
| GROUP 5: The clock | LCLK. The bus clock. It has a maximum frequency of 66Mhz (15ns). A minimum duty time of 40% (i.e., minimum clock low/high time is 6ns). |

Figure 96. Timings for VL bus

## 6.4. VL Bus Operation, continued

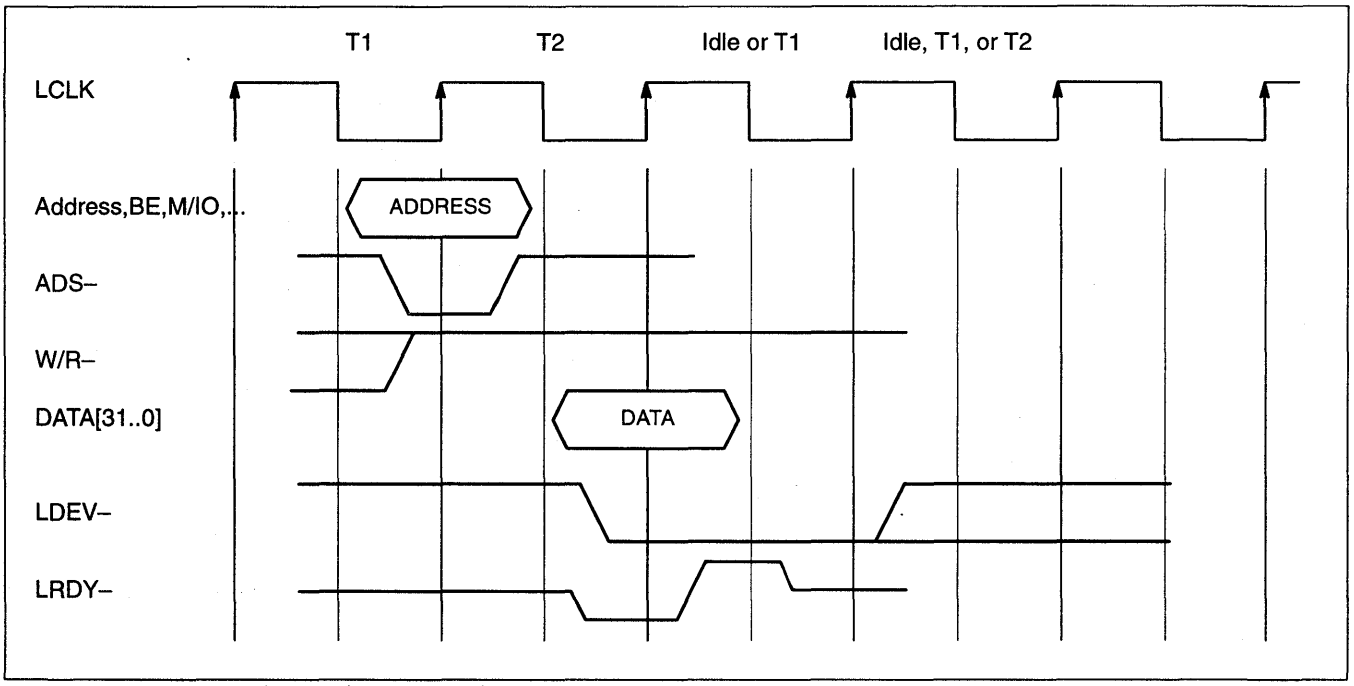



Figure 97. VL Bus High Speed Write Mode, Write, 0 Wait States

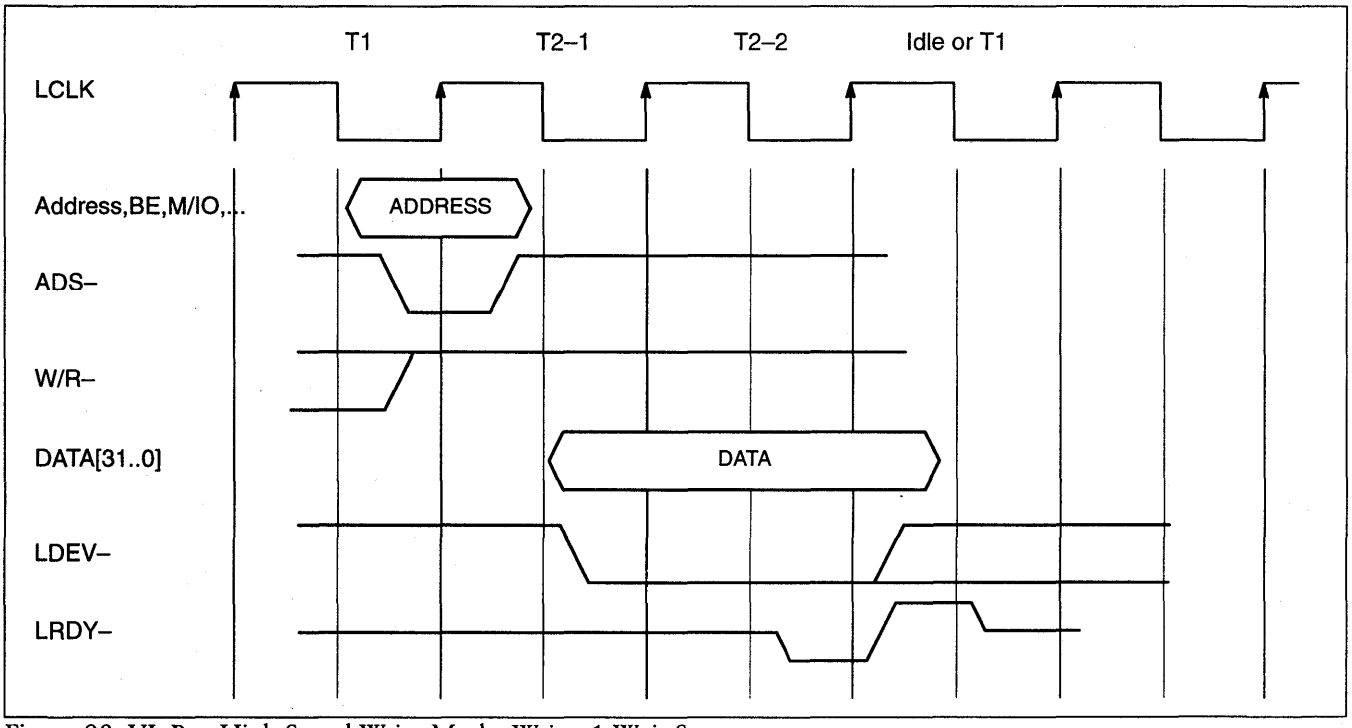



Figure 98. VL Bus High Speed Write Mode, Write, 1 Wait States

## 6.4. VL Bus Operation, continued

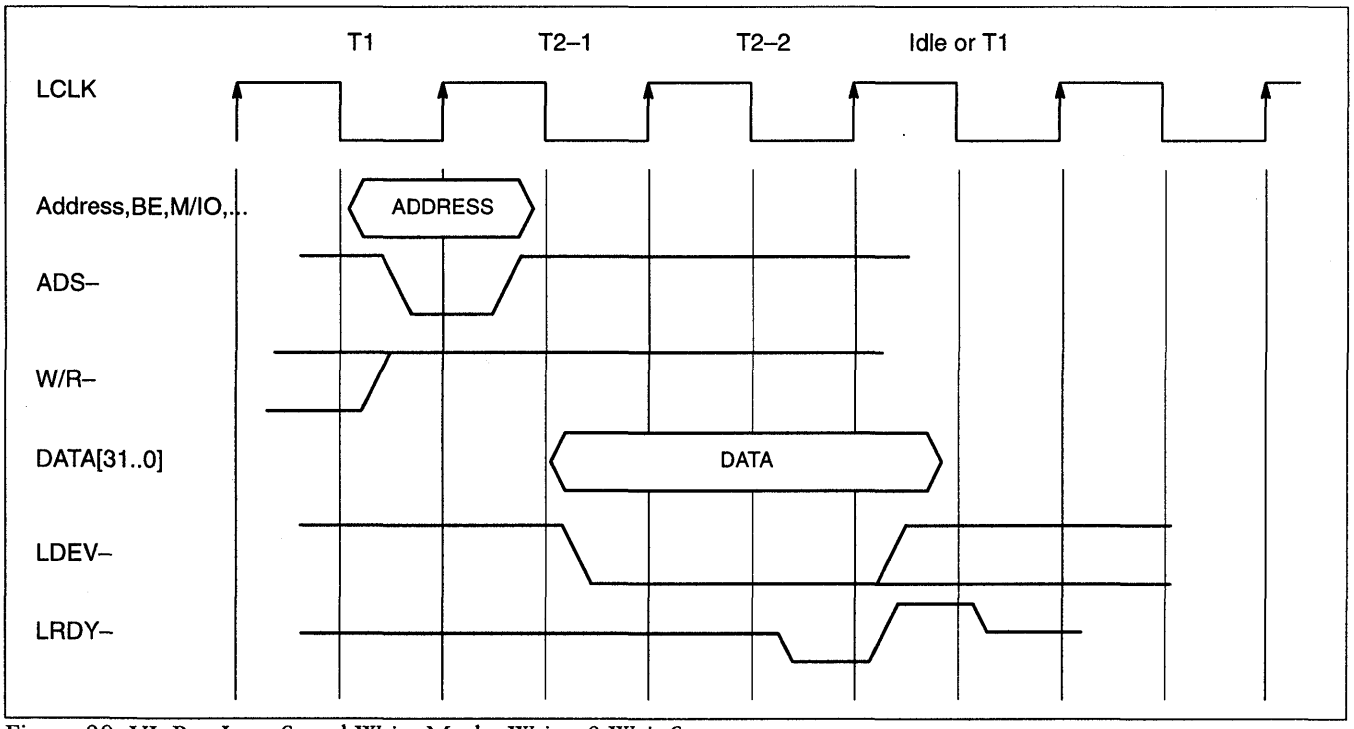



Figure 99. VL Bus Low Speed Write Mode, Write, 0 Wait States

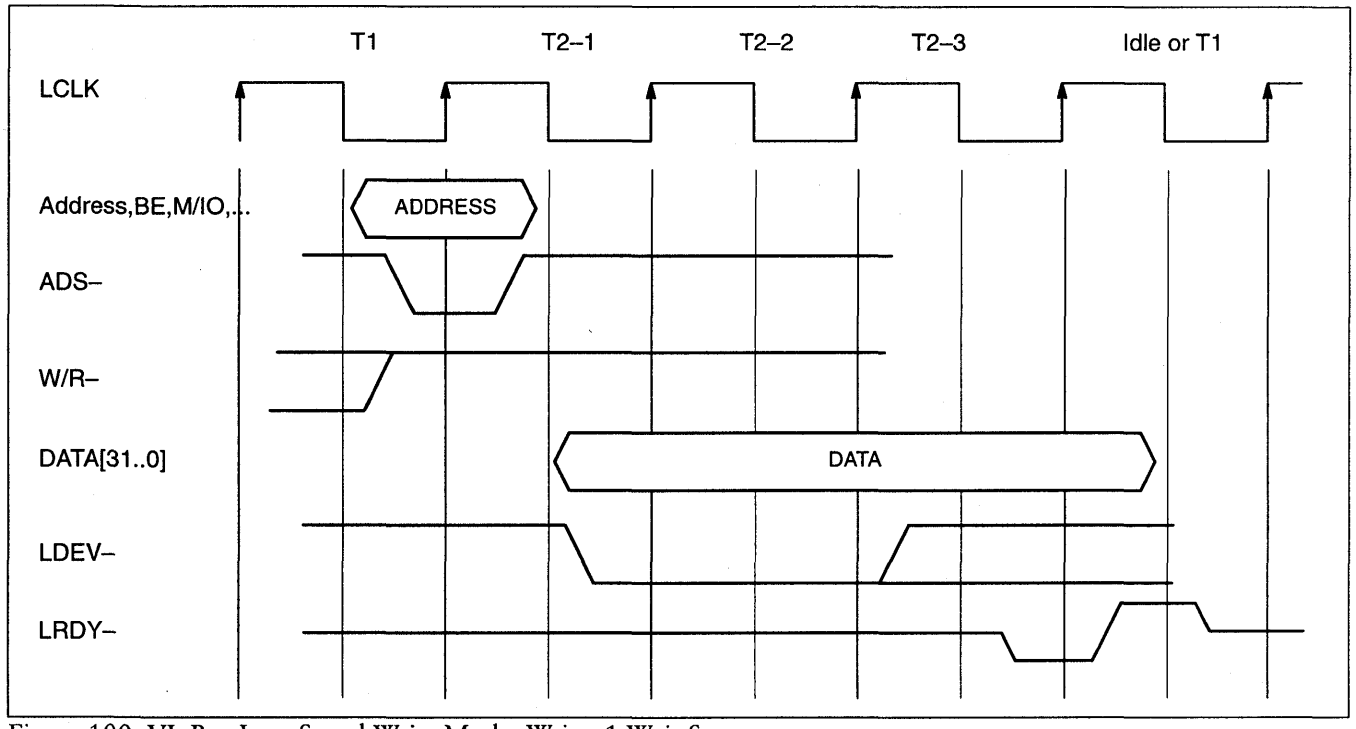



Figure 100. VL Bus Low Speed Write Mode, Write, 1 Wait States

## 6.4. VL Bus Operation, continued

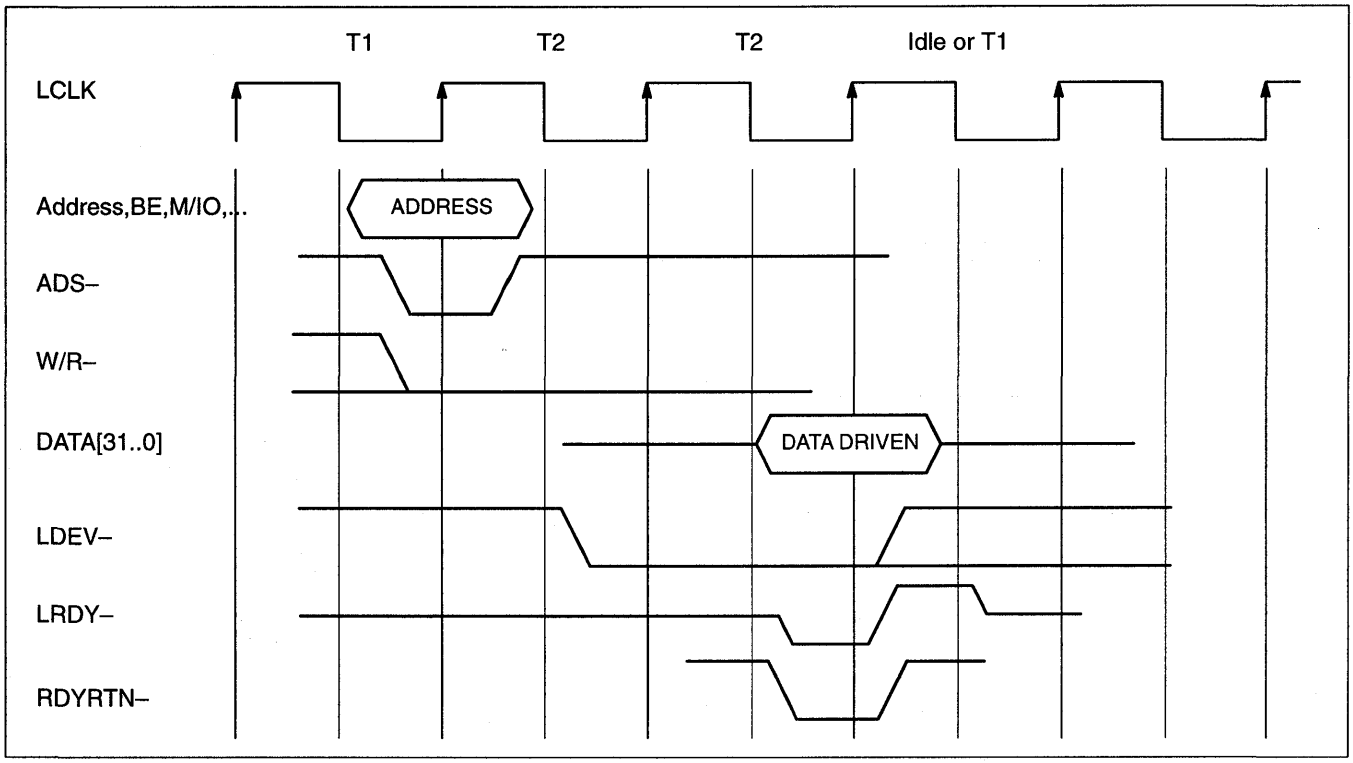



Figure 101. VL Bus Read, 0 WS, 0 Wait for RDYRTN–

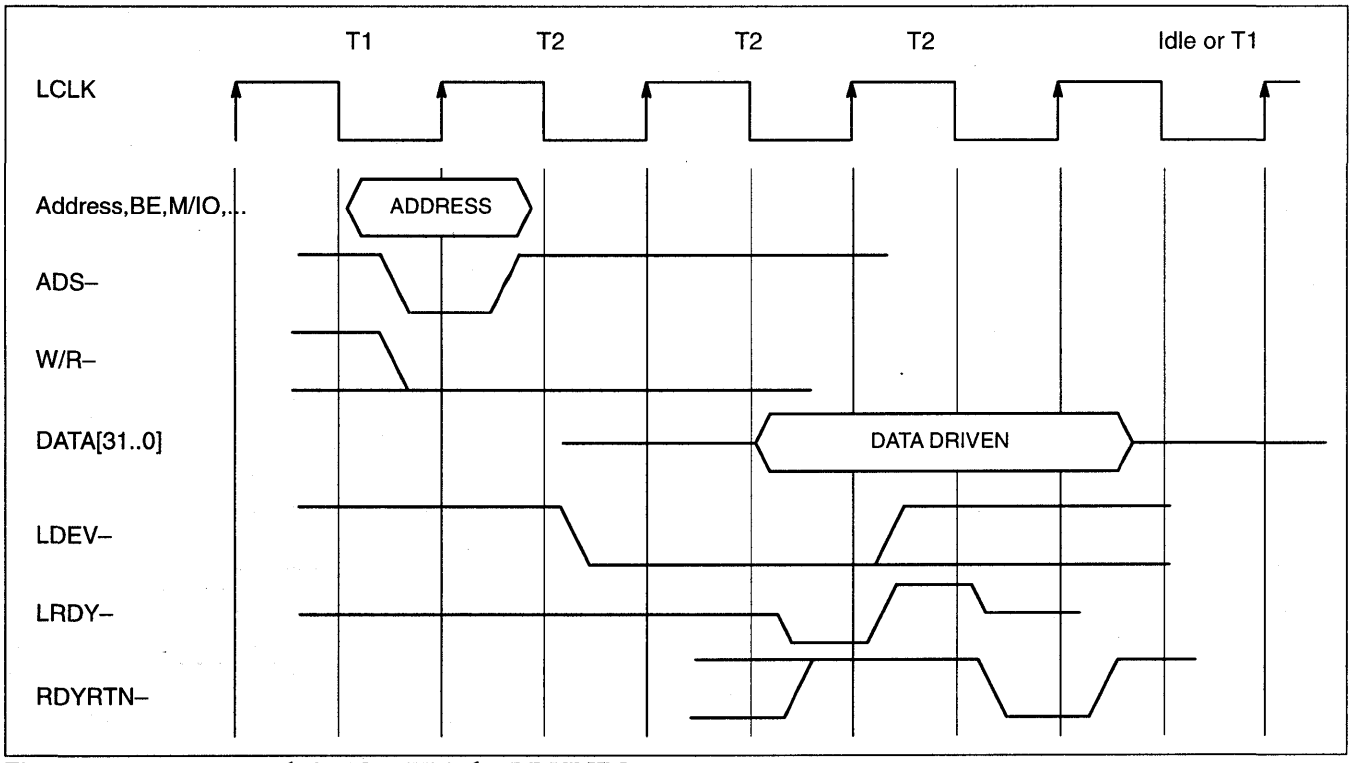



Figure 102. VL Bus Read, 0 WS, 1 Wait for RDYRTN–

## 6.4. VL Bus Operation, continued

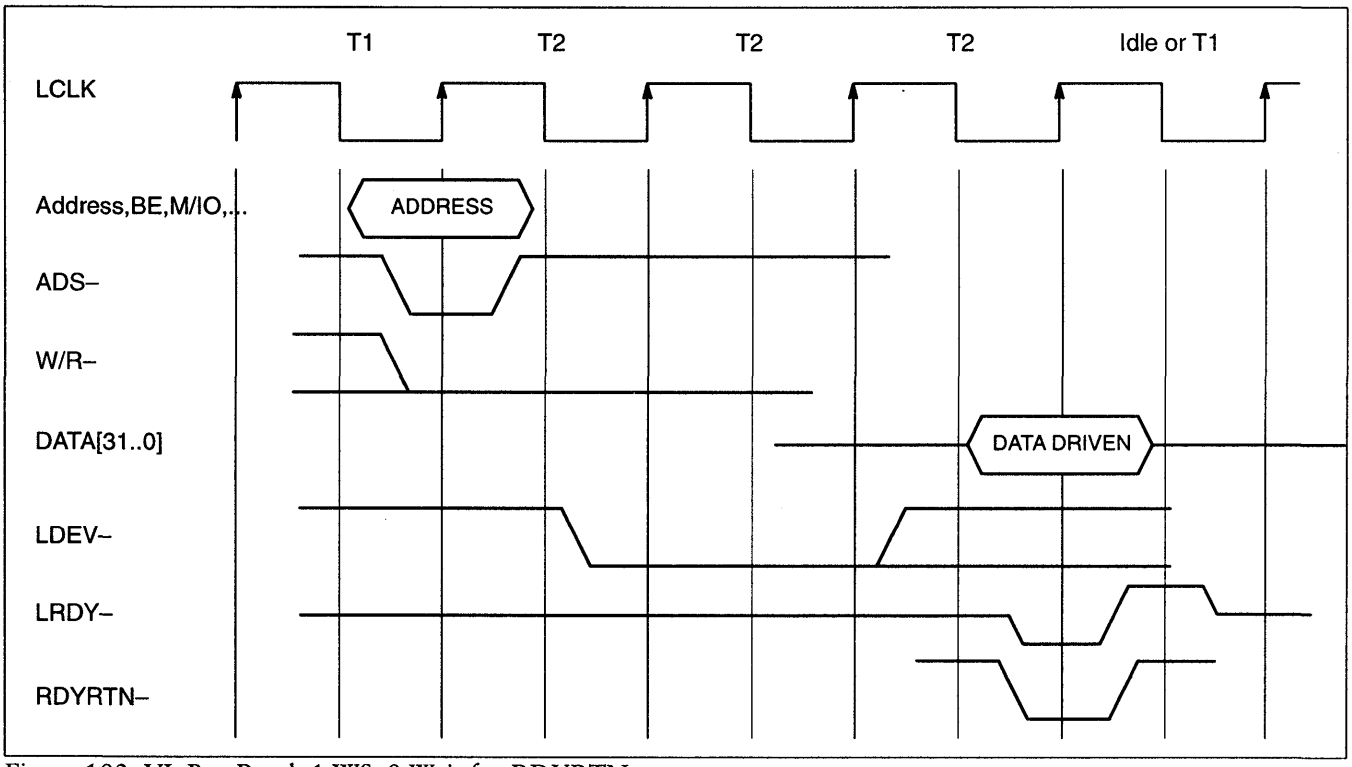



Figure 103. VL Bus Read, 1 WS, 0 Wait for RDYRTN–

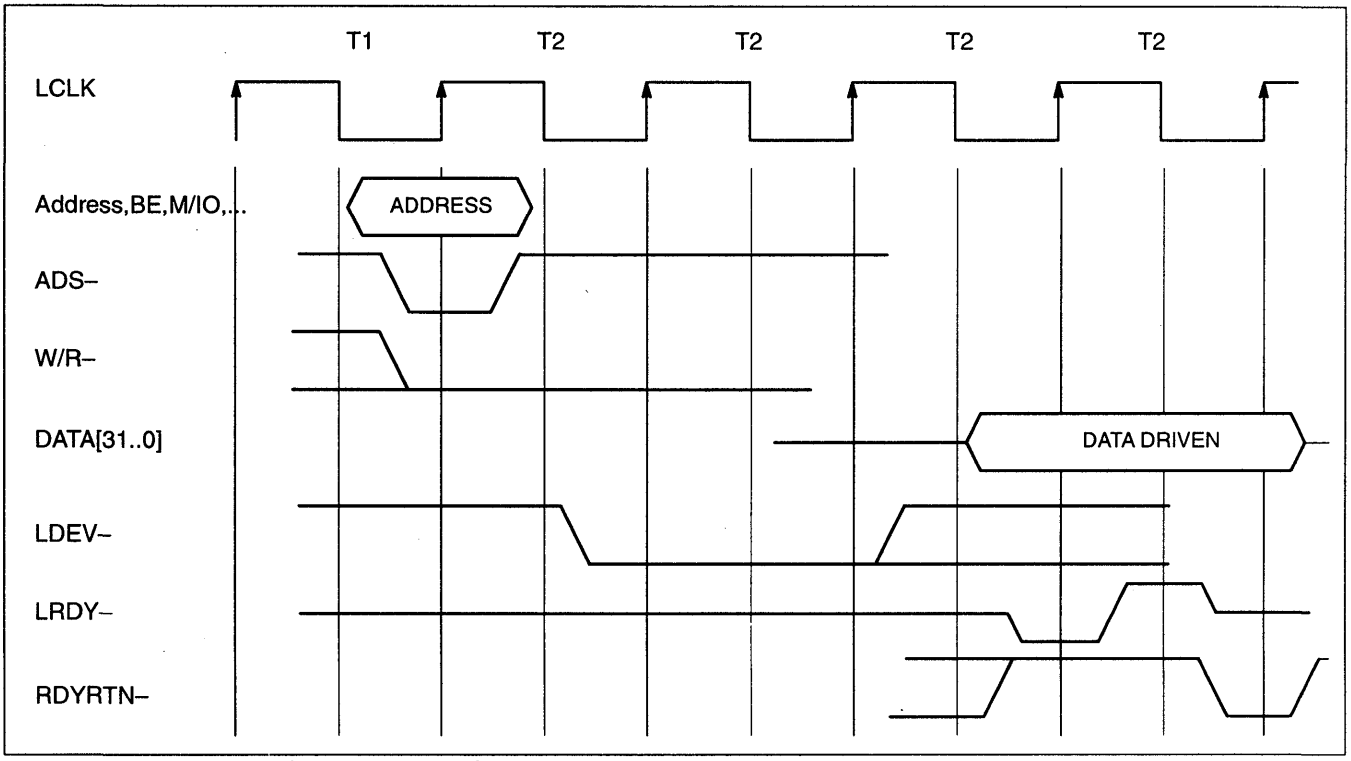



Figure 104. VL Bus Read, 1 WS, 1 Wait for RDYRTN–

# Chapter 7. Frame Buffer Interface

## 7.1. Frame Buffer Design Notes

This section presents information necessary to select, configure, and connect VRAMs. The first set of figures (105 through 112) shows how to wire together the memory chips for 1, 2 and 4 bank memory configurations.

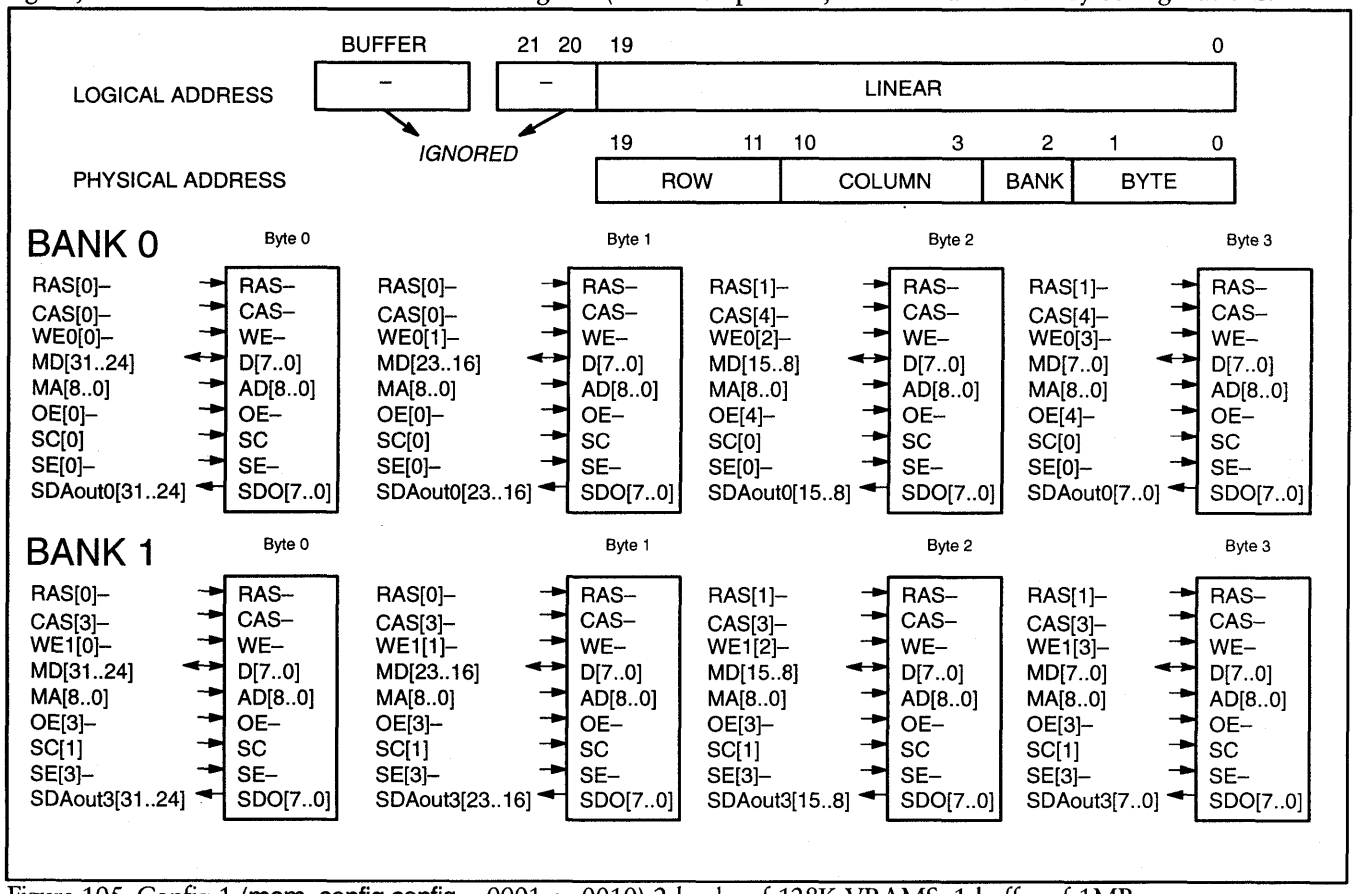



Figure 105. Config 1 (mem_config.config = 0001 or 0010) 2 banks of 128K VRAMS, 1 buffer of 1MB

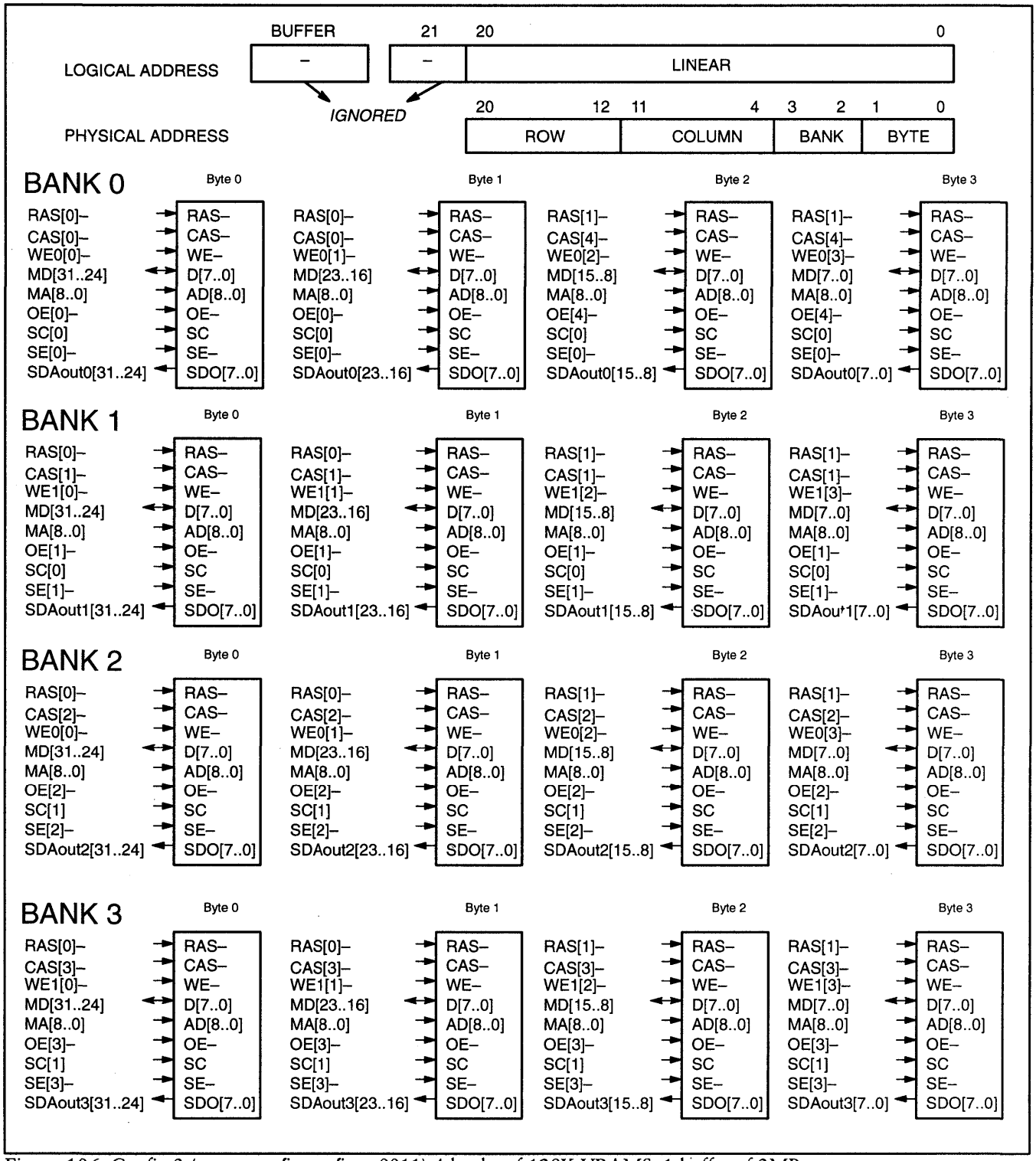



Figure 106. Config 3 (mem_config.config = 0011) 4 banks of 128K VRAMS, 1 buffer of 2MB

## 7.1. Frame Buffer Design Notes, continued

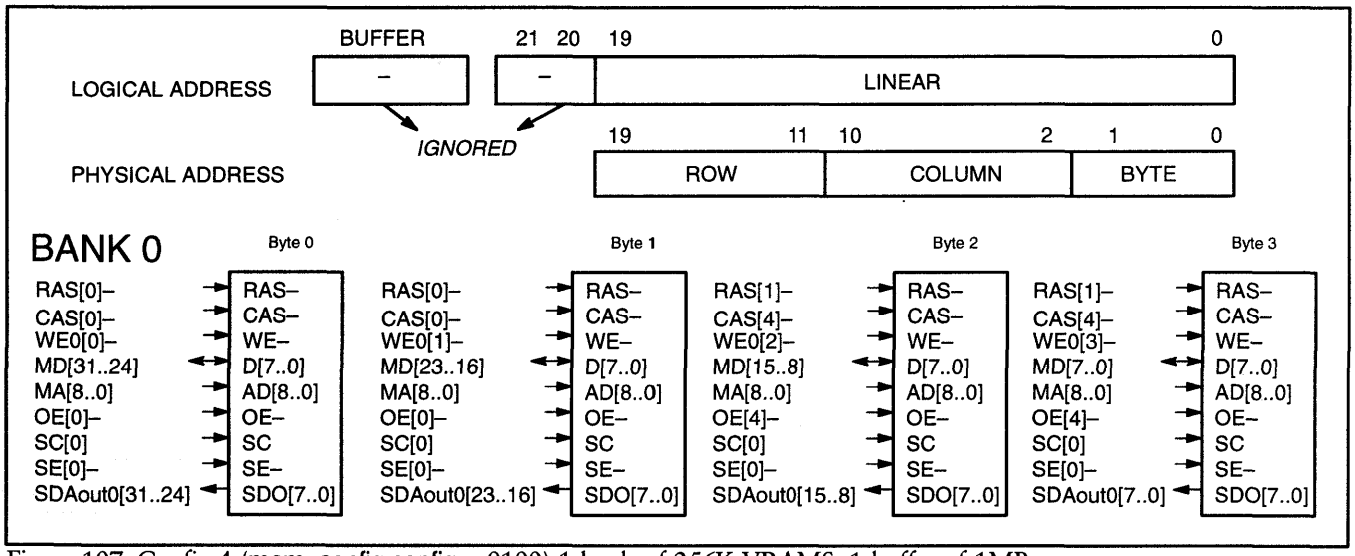



Figure 107. Config 4 (mem_config.config = 0100) 1 bank of 256K VRAMS, 1 buffer of 1MB

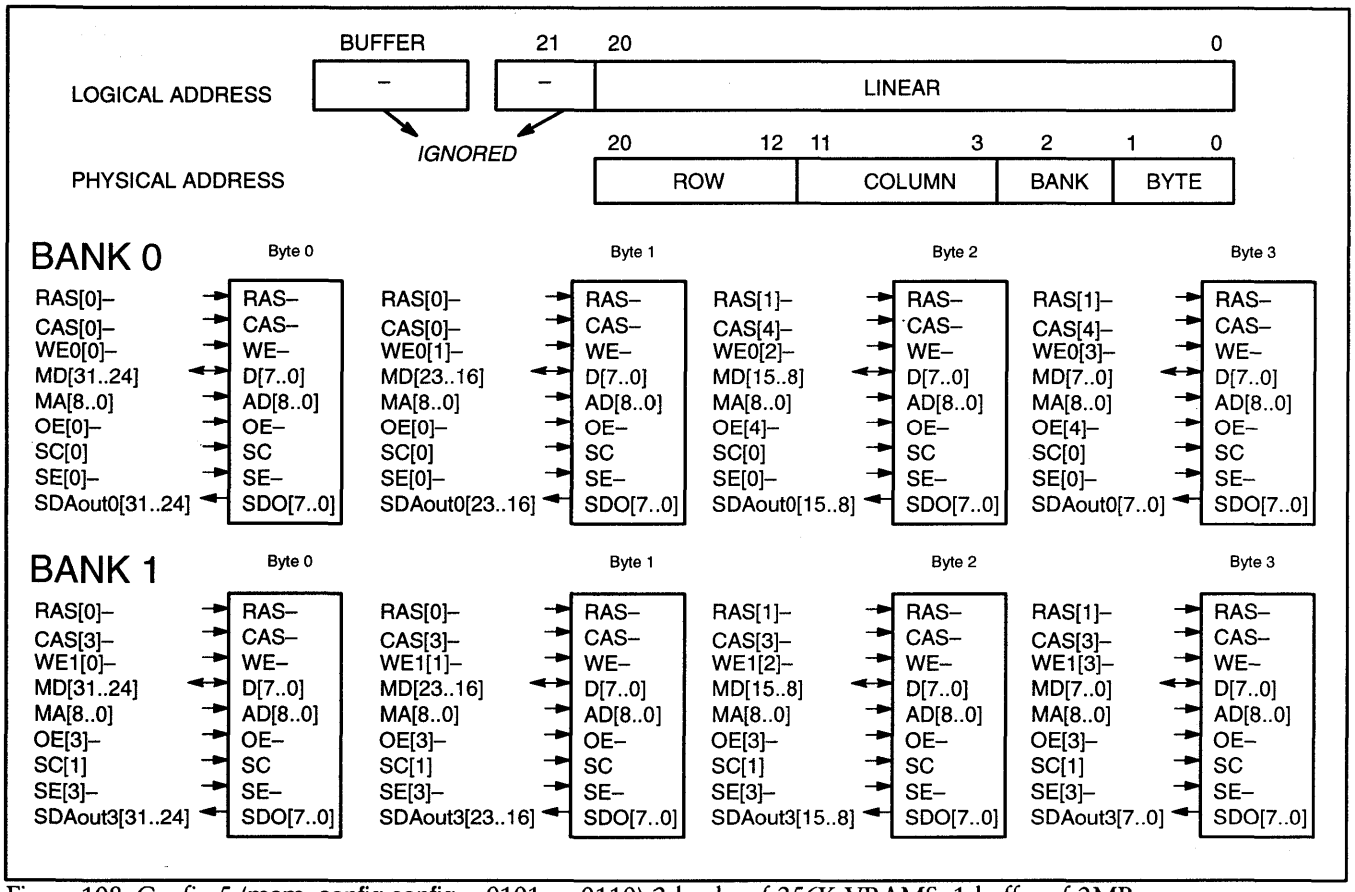



Figure 108. Config 5 (mem_config.config = 0101 or 0110) 2 banks of 256K VRAMS, 1 buffer of 2MB

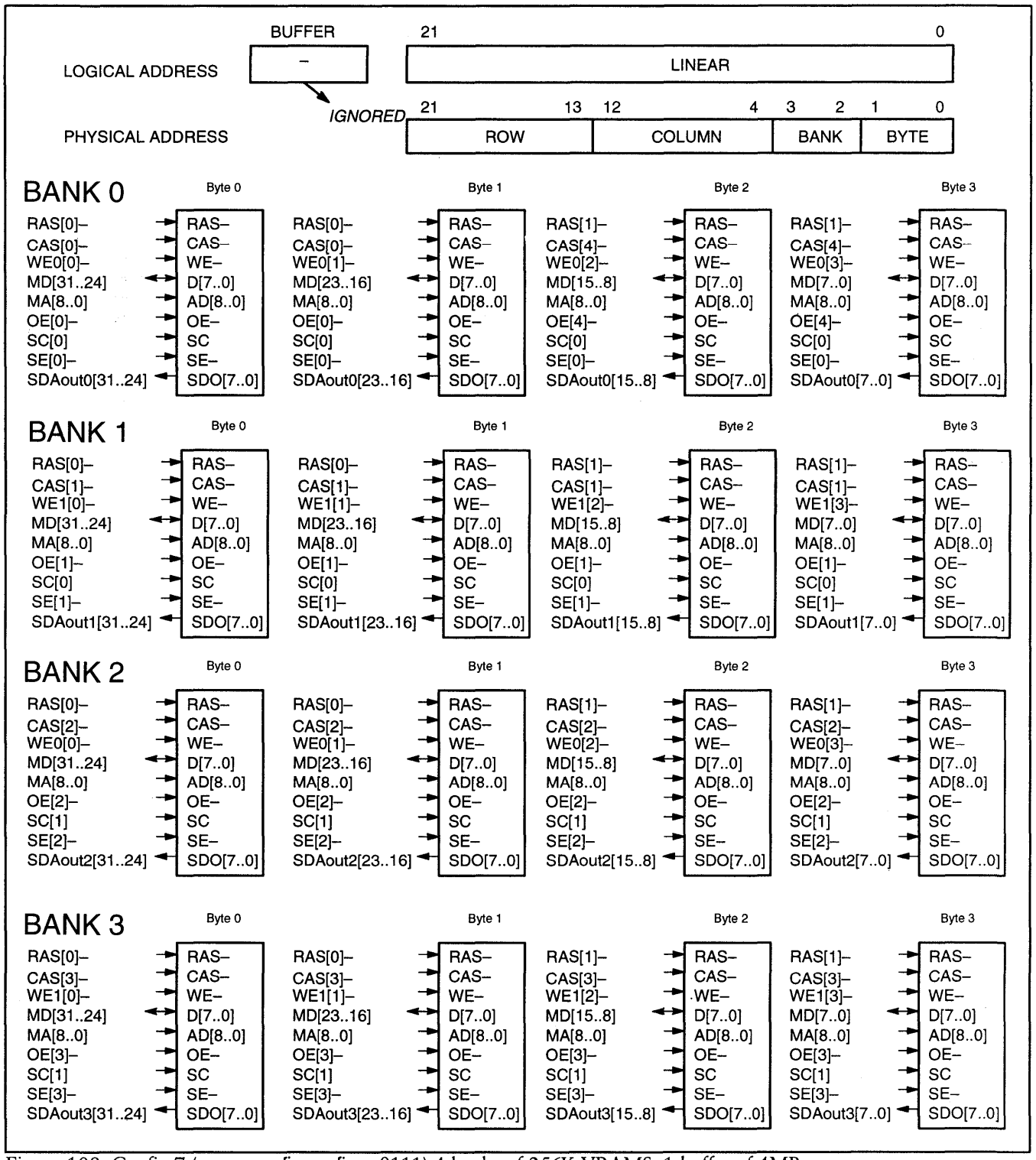



Figure 109. Config 7 (mem_config.config = 0111) 4 banks of 256K VRAMS, 1 buffer of 4MB

## 7.1. Frame Buffer Design Notes, continued

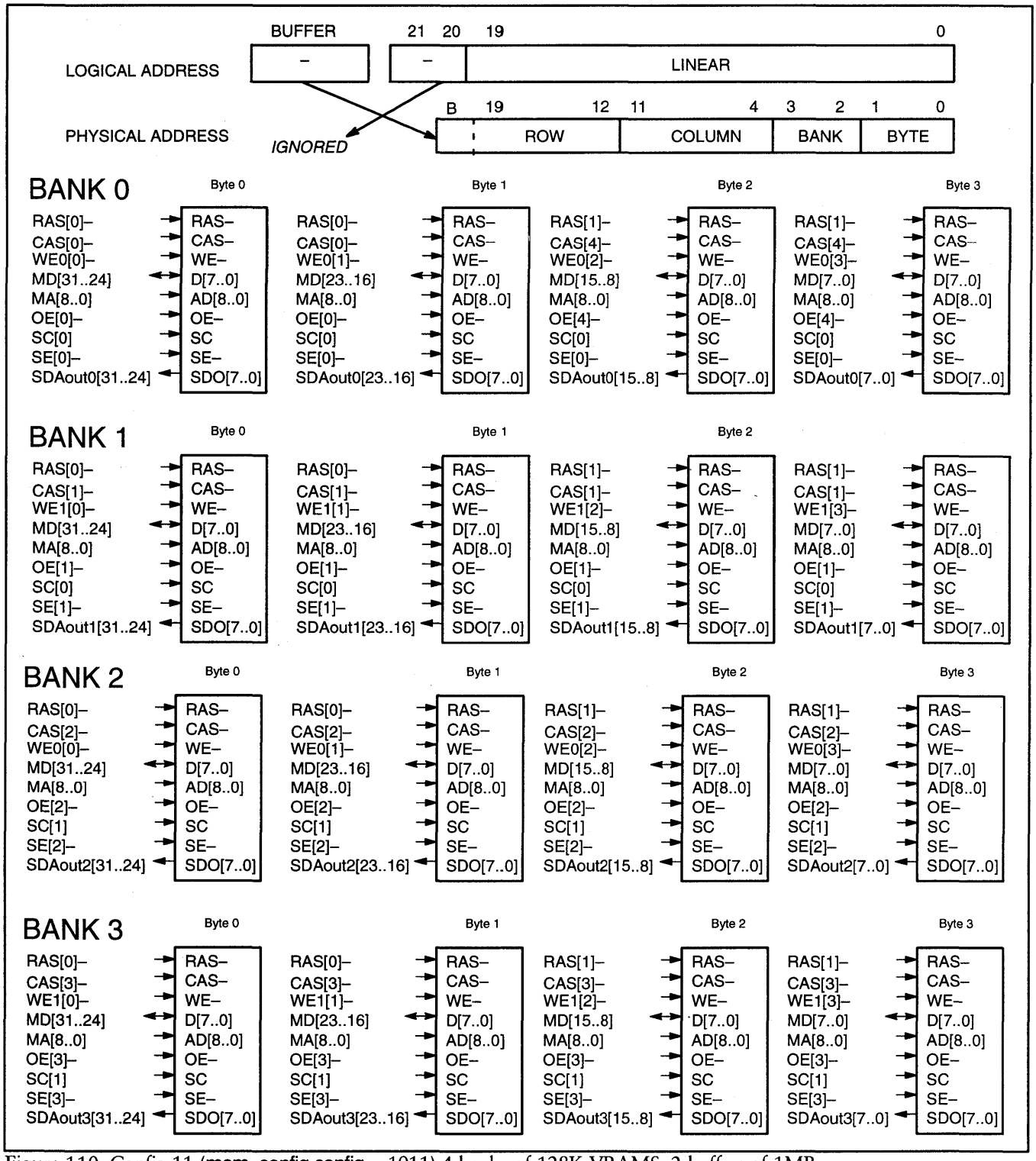



Figure 110. Config 11 (mem_config.config = 1011) 4 banks of 128K VRAMS, 2 buffers of 1MB

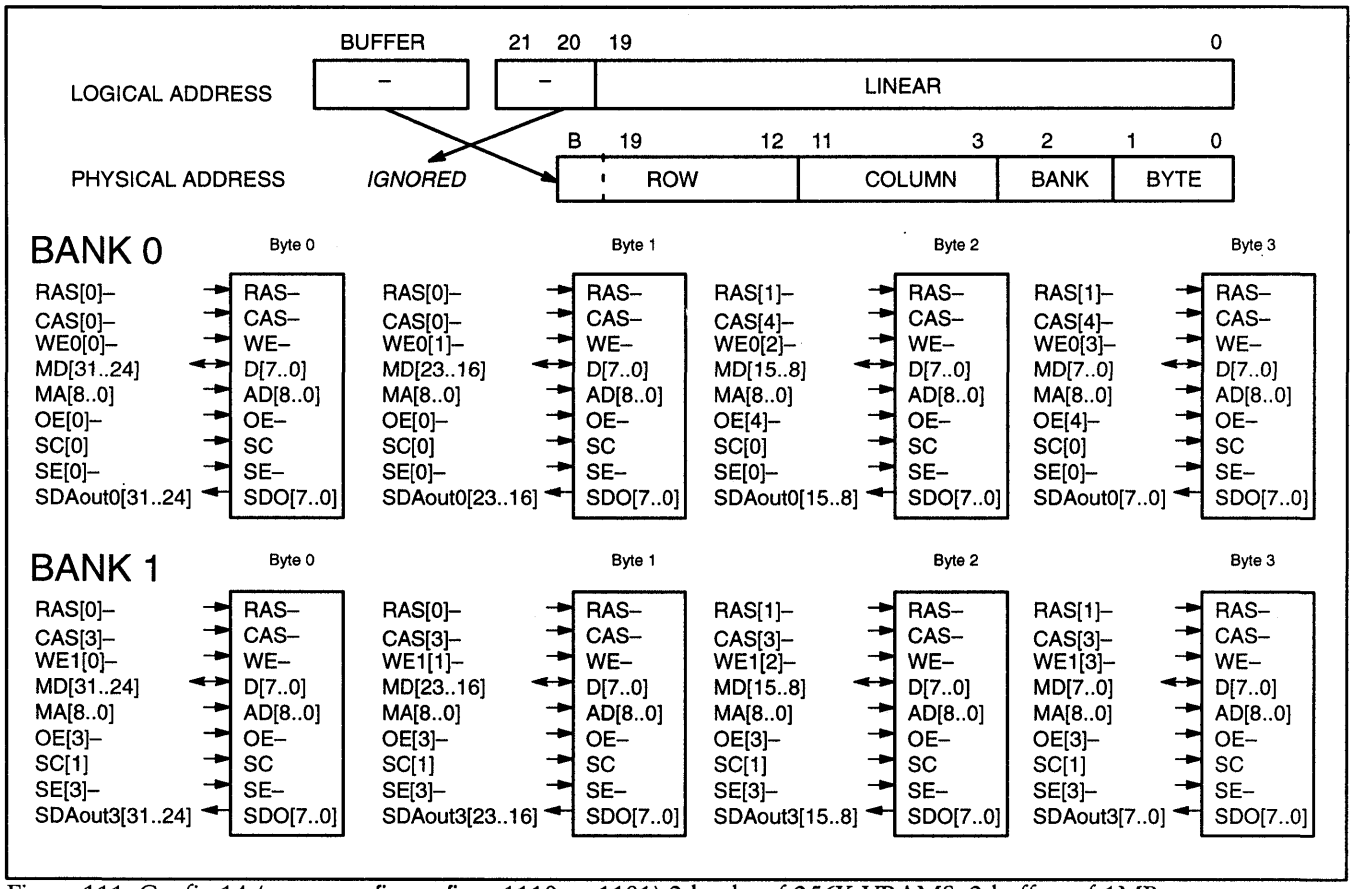



Figure 111. Config 14 (mem_config.config = 1110 or 1101) 2 banks of 256K VRAMS, 2 buffers of 1MB

## 7.1. Frame Buffer Design Notes, continued

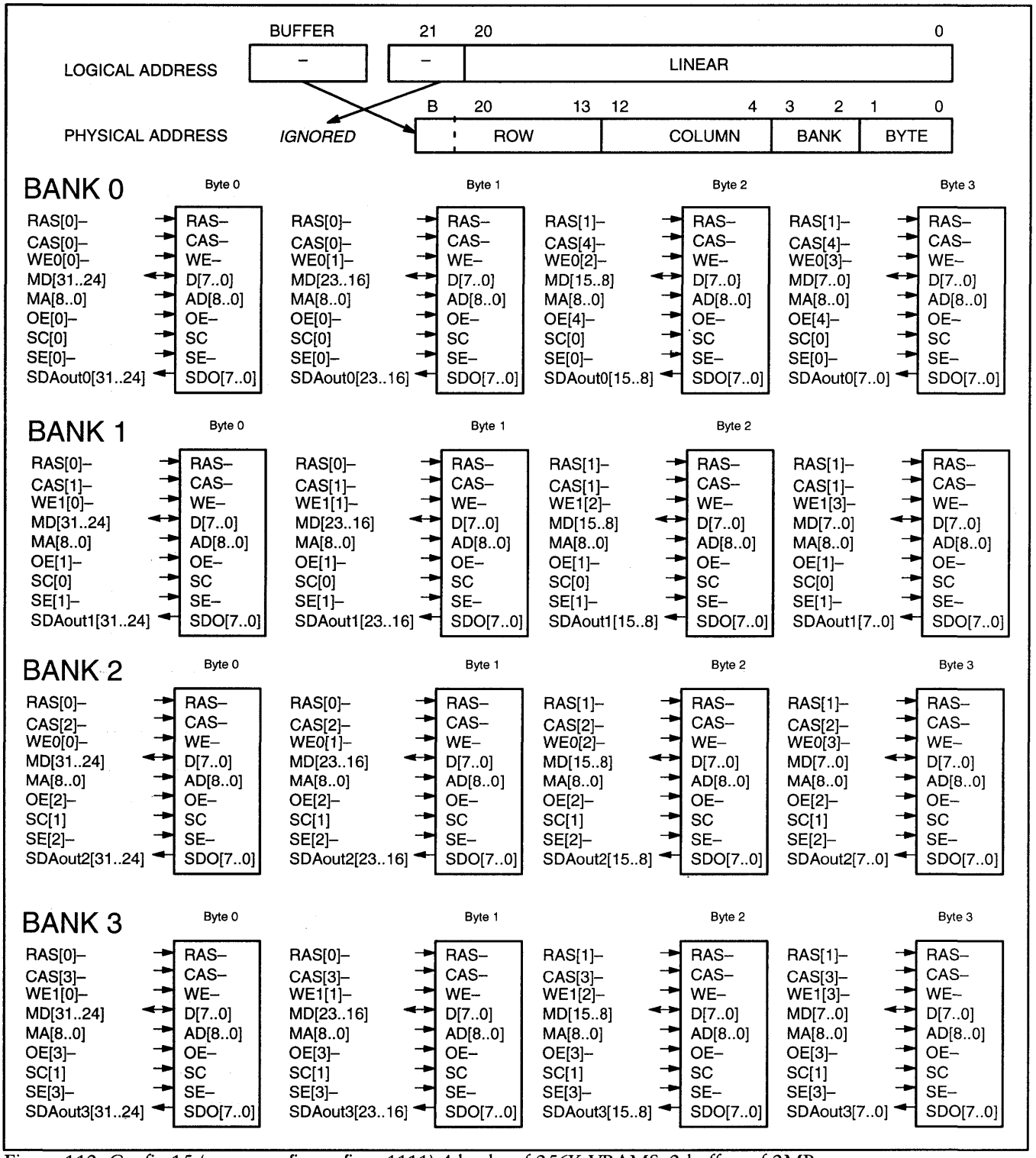



Figure 112. Config 15 (mem_config.config = 1111) 4 banks of 256K VRAMS, 2 buffers of 2MB

## 7.2. VRAM Access

All VRAM accesses can be described as sequences of the VRAM timing templates shown in figures 113 through 124.

### 7.2.1. ROW MISS

Normally the Power 9100 uses the fast page mode method to access the VRAMS. Whenever a new access is to a different page of VRAM than the previous access, the row miss timing sequence is inserted to switch the rams (both banks) to the new bank. Figures 113 and 114 shows this sequence. Note that a write mask is always loaded into the VRAMS. If the programmer has disabled the plane mask feature the Power 9100 automatically generates the all-planes-enabled mask.

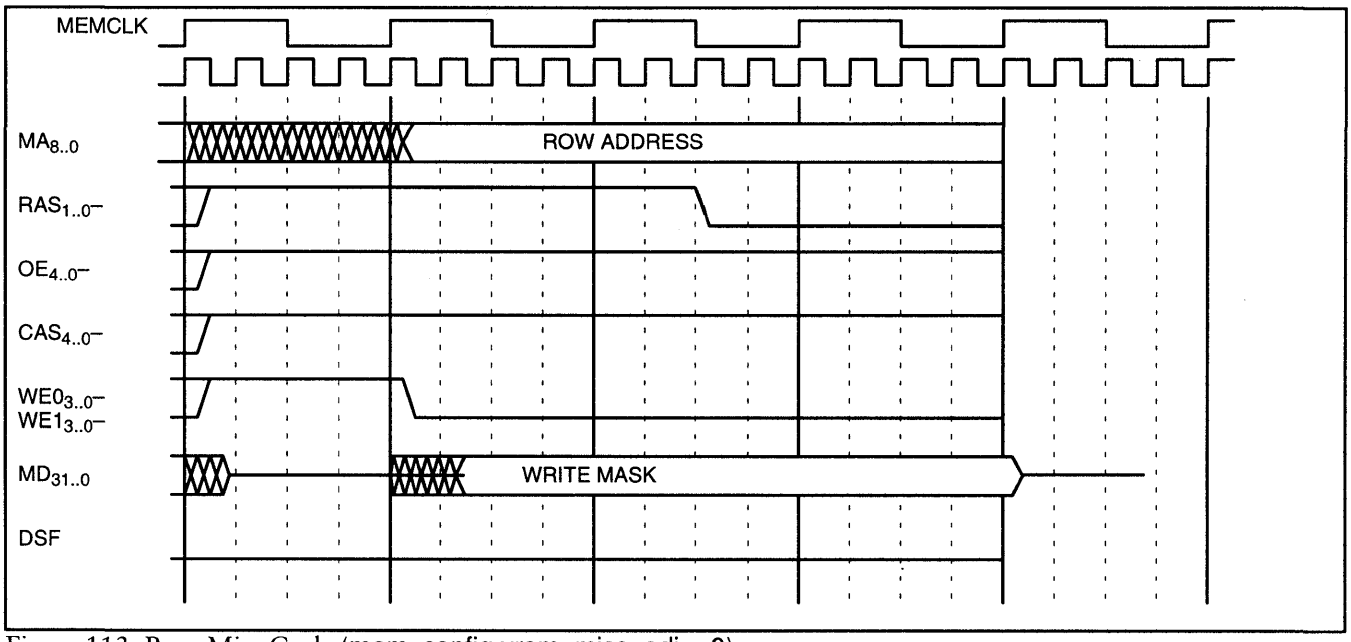



Figure 113. Row Miss Cycle (mem_config.vram_miss_adj = 0)

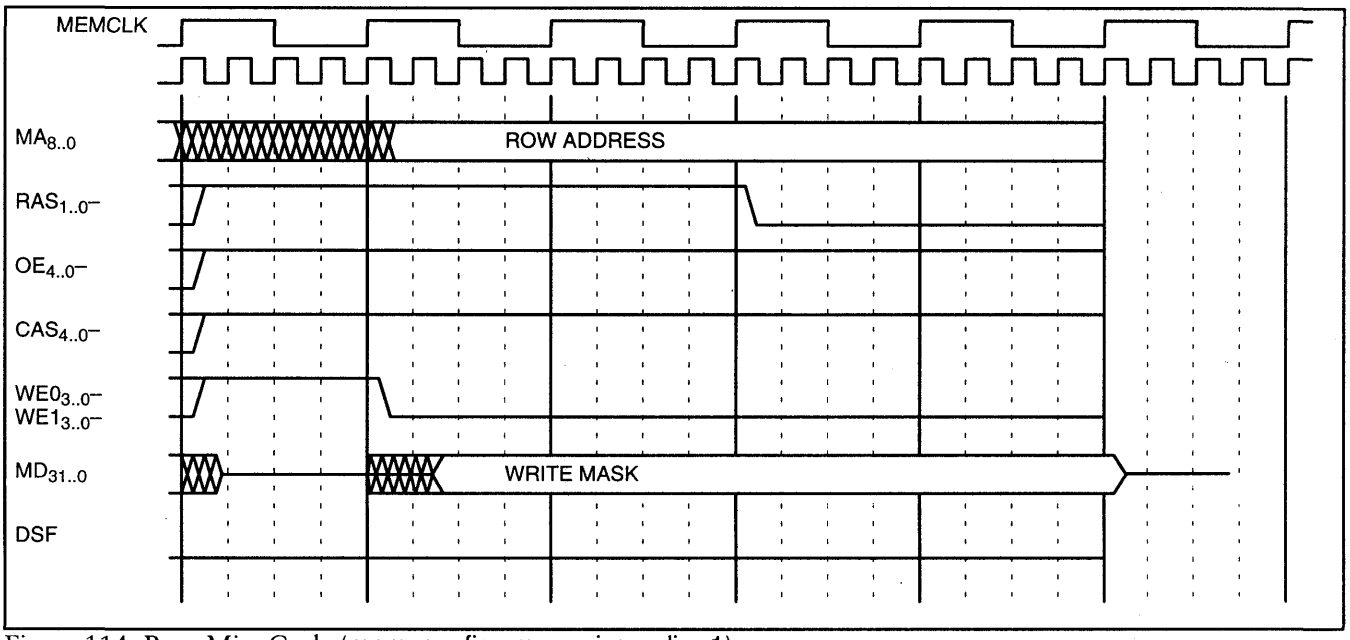



Figure 114. Row Miss Cycle (mem_config.vram_miss_adj = 1)

## 7.2. VRAM Access, continued

### 7.2.2. READ

Read operations take two cycles, this allows for the slow turn off time of VRAMS. Figures 115 and 116 shows a read operation from bank 0.

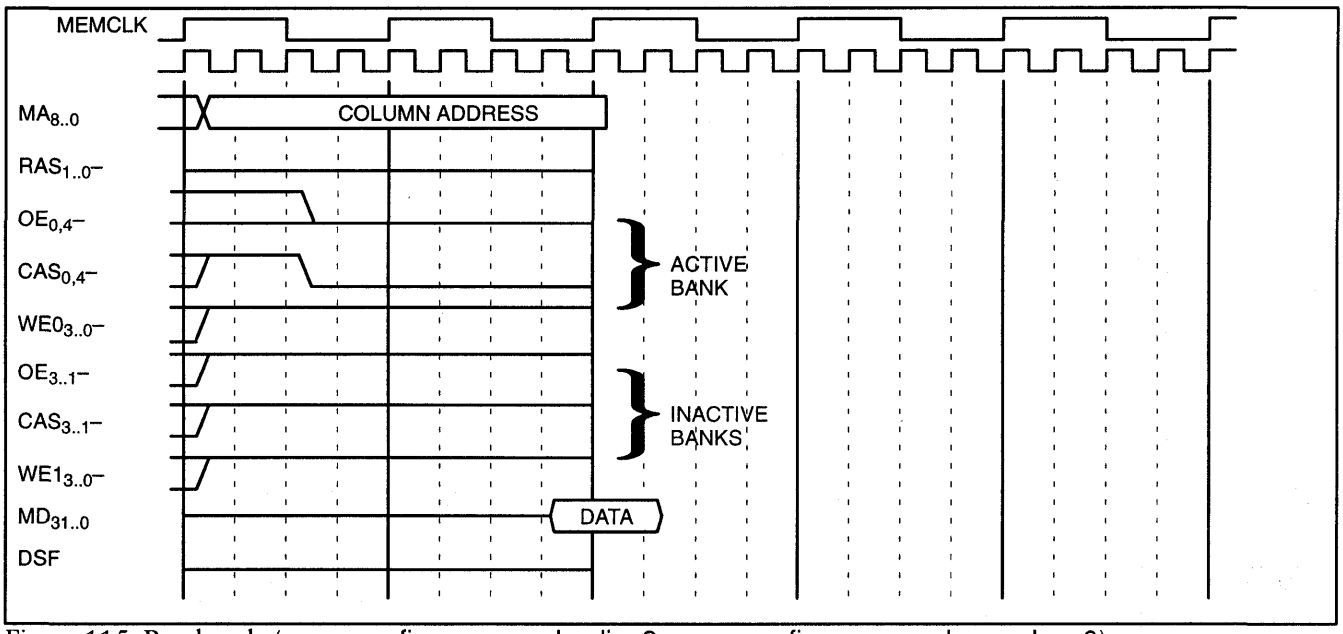



Figure 115. Read cycle (mem_config.vram_read_adj = 0, mem_config.vram_read_sample = 0)

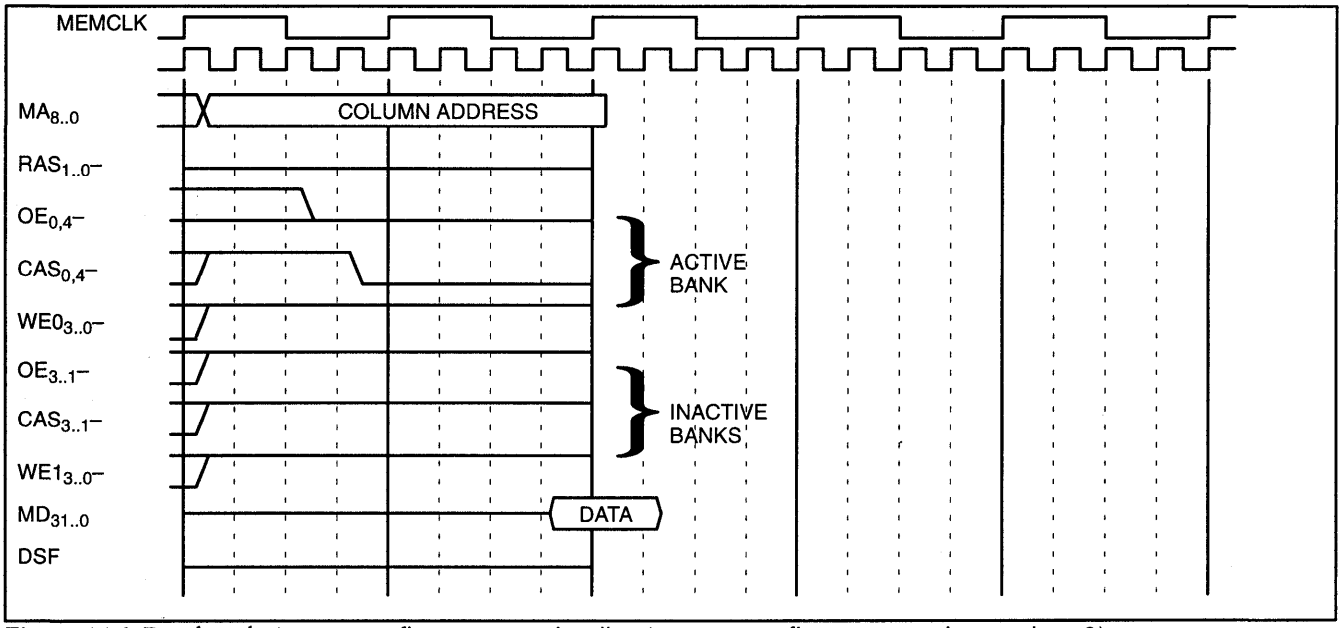



Figure 116. Read cycle (mem_config.vram_read_adj = 1, mem_config.vram_read_sample = 0)

## 7.2. VRAM Access, continued

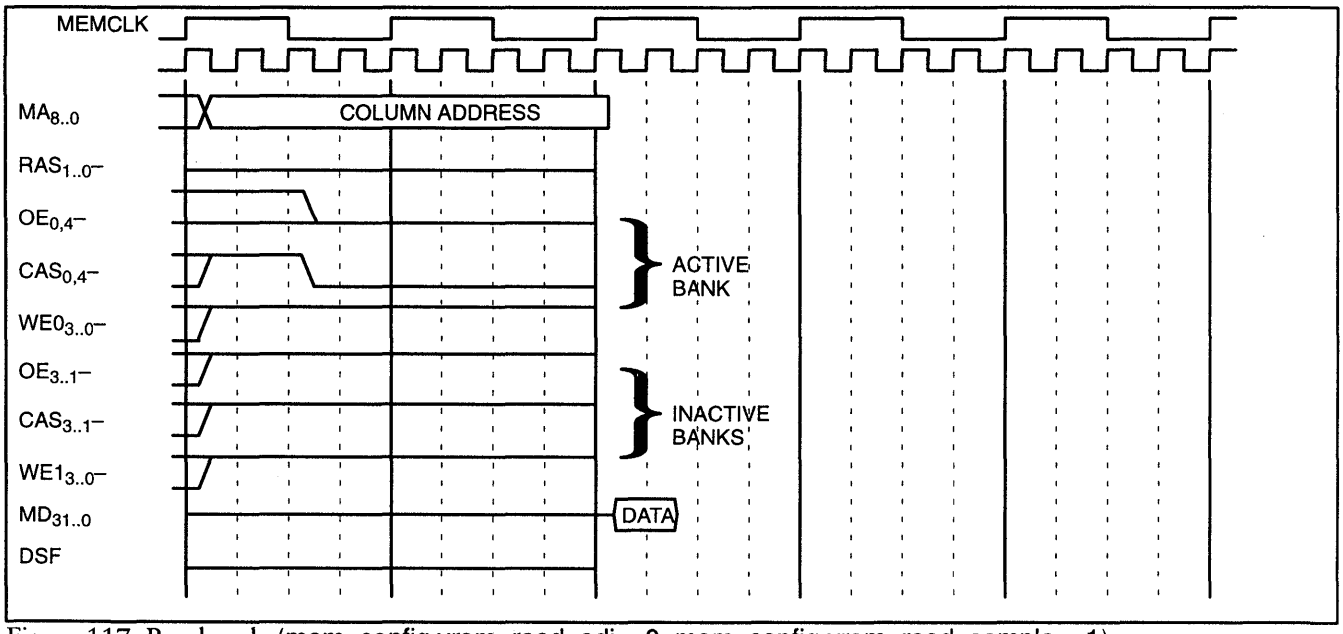



Figure 117. Read cycle (mem_config.vram_read_adj = 0, mem_config.vram_read_sample = 1)

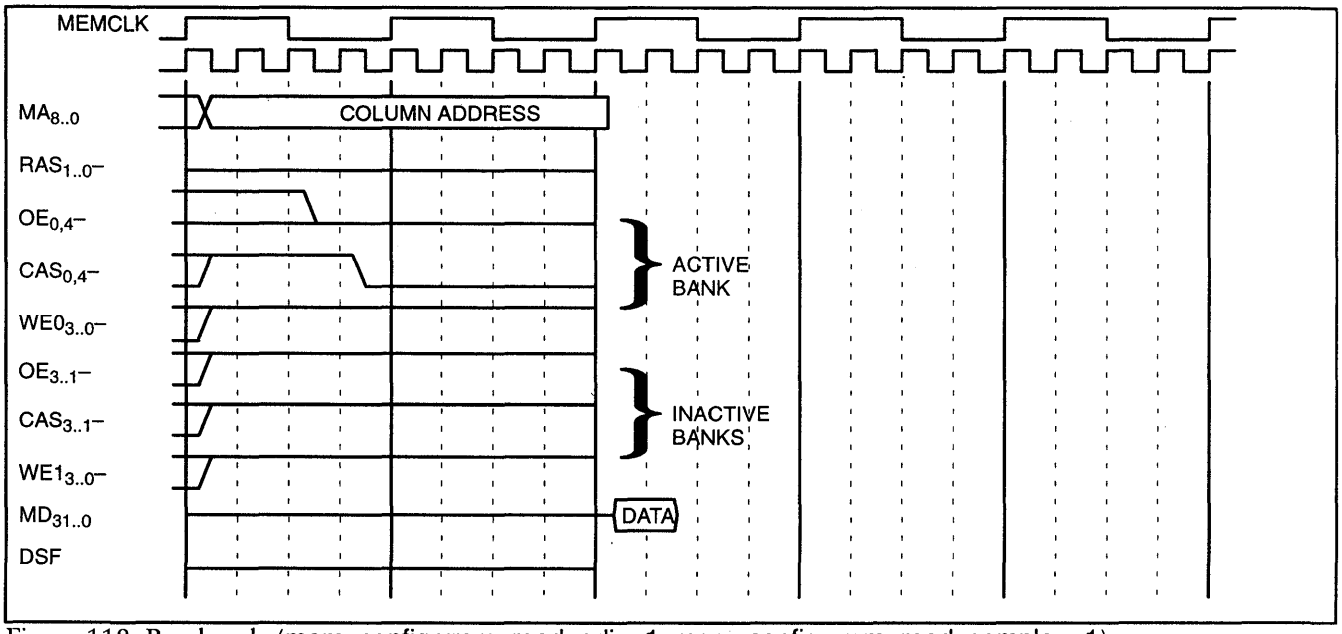



Figure 118. Read cycle (mem_config.vram_read_adj = 1, mem_config.vram_read_sample = 1)

## 7.2. VRAM Access, continued

### 7.2.3. WRITE

A write operation requires either one or two cycles. There are two basic write templates, shown in figures 119 and 120. All write operations start with the single cycle that is marked "write" in the template. If the next cycle is *not* a write to the other bank of VRAMs, then the write operation is extended to two cycles by inserting the extra cycle labelled "finish write".

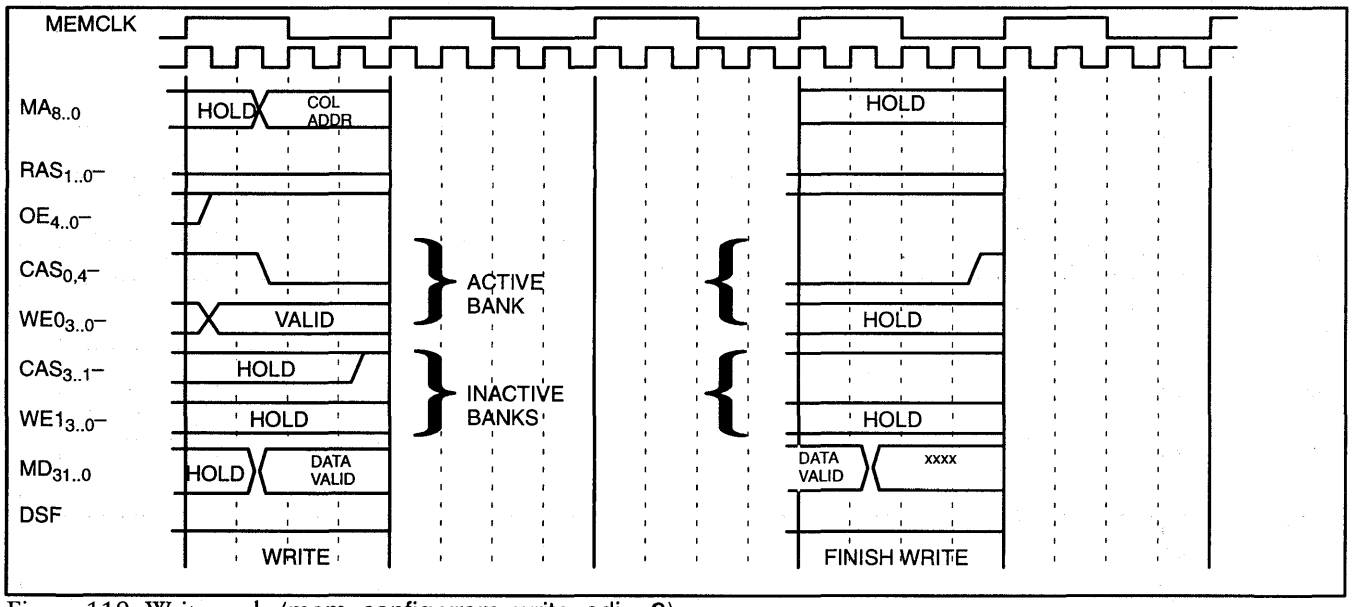



Figure 119. Write cycle (mem_config.vram_write_adj = 0)

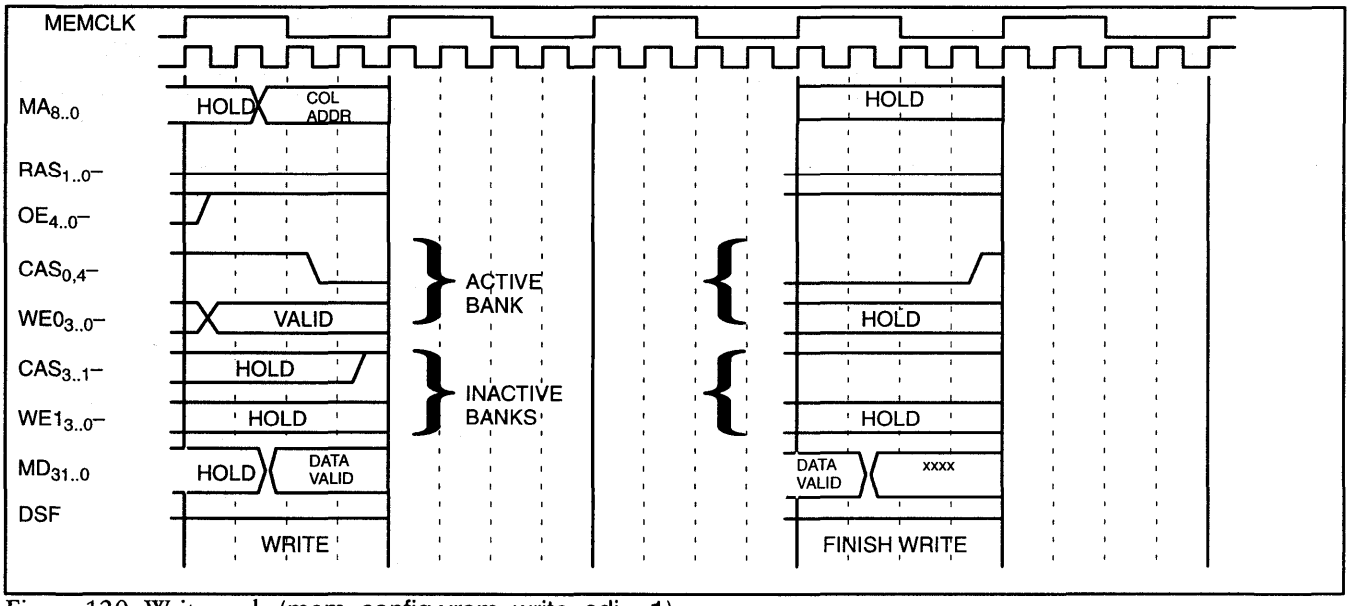



Figure 120. Write cycle (mem_config.vram_write_adj = 1)

## 7.2. VRAM Access, continued

### 7.2.4. READ TRANSFER

Figure 121 shows the template used to reload the internal VRAM video shift register. During a vertical retrace (VBLNK– asserted) the IGC will reload the entire shift register inside of the VRAMs. Thereafter the IGC will generate a split shift register reload whenever the VRAMs are more than half empty (Based upon the internally generated QSF signal). Because the IGC does not intervene between scan lines (i.e., during HBLNK–) this method requires that each scan line of the image be located in sequential linear memory and that the VRAM shift clock be inhibited during HBLNK– intervals. Therefore the physical scan line width of the monitor must match exactly the logical scan line width loaded into the drawing engine. This means that there are no extra pixels between scan lines in memory.

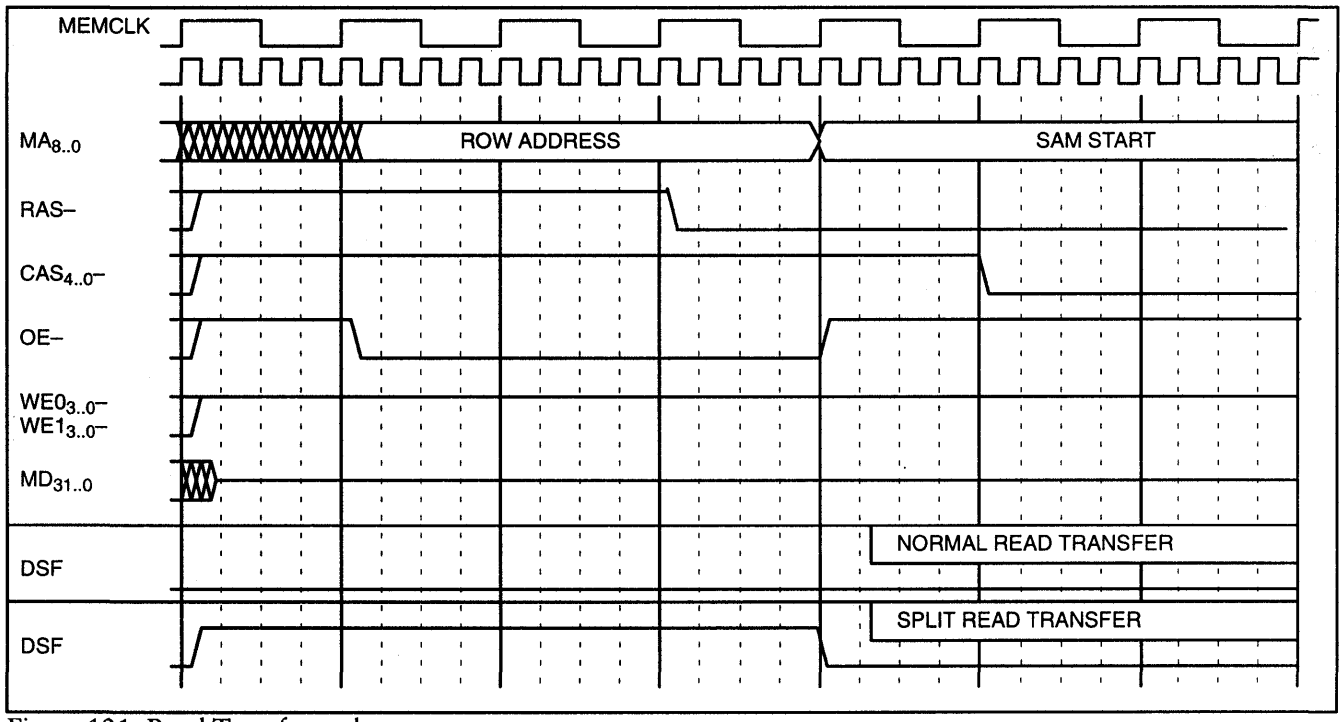



Figure 121. Read Transfer cycle

## 7.2. VRAM Access, continued

### 7.2.5. REFRESH

Figure 122 shows the template used to refresh the VRAMs. The memory controller has refresh request hold over function which the frame buffer controller may queue up to 4 memory refresh requests in order to reduce the interruption to the video processor.

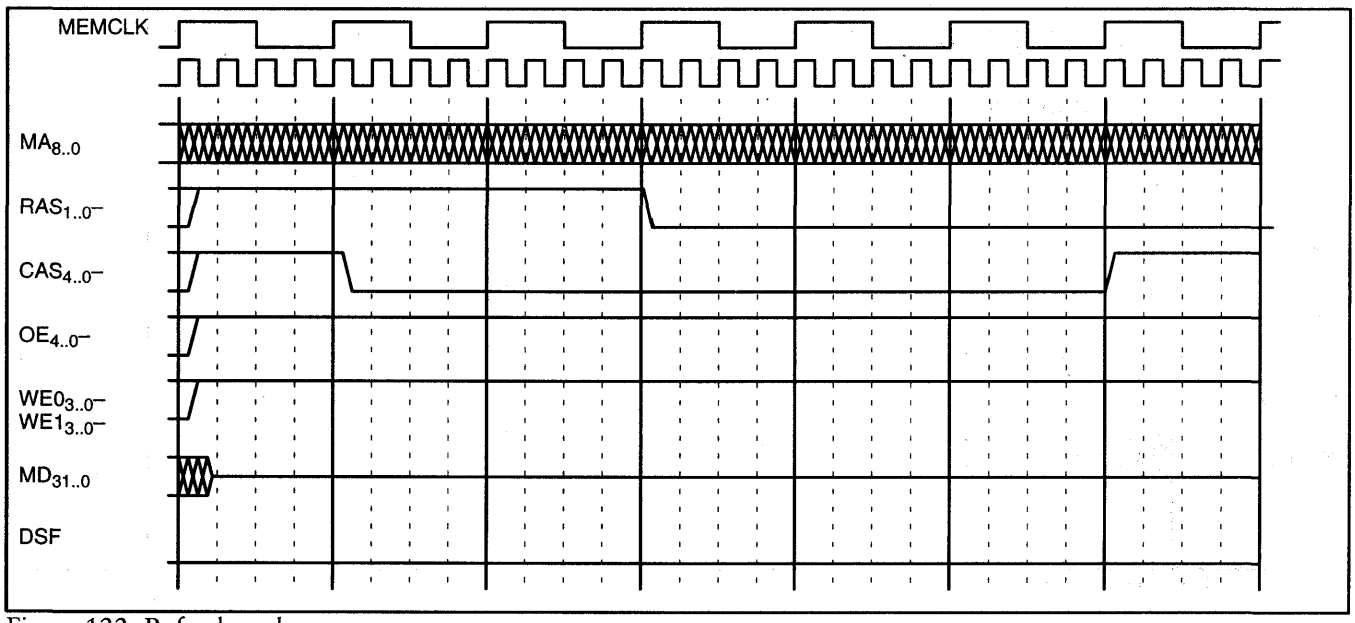



Figure 122. Refresh cycle

### 7.2.6. IDLE

Figure 123 shows the state of the VRAM controls when no ram activity is required.

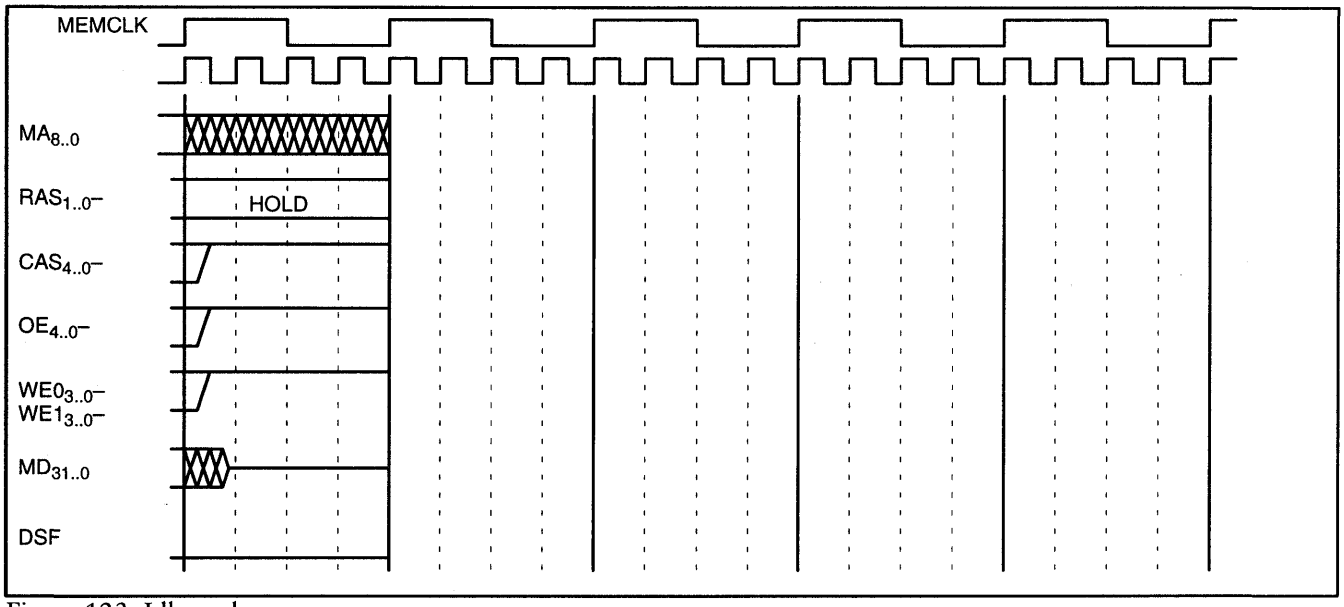



Figure 123. Idle cycle

## 7.2. VRAM Access, continued

### 7.2.7. RESET STATE

Figure 124 shows the reset state of the VRAM controls during and after power on reset, synchronous reset and asynchronous reset.

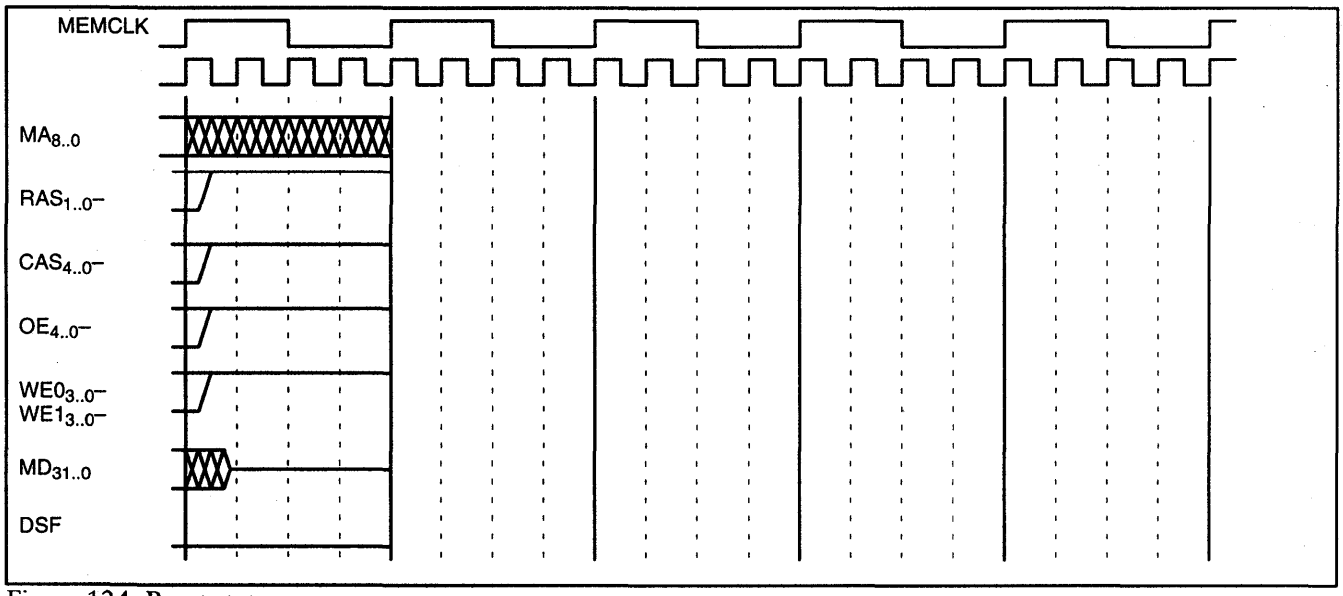



Figure 124. Reset state

# Chapter 8. Video and RAMDAC Interface

## 8.1. Video Control

This section contains information about video control issues. The topics covered are:

- Video Clock generation
- Video Shift Clock generation
- Video Serial Enable generation
- Video Address generation
- Video signals
- Video control registers
- Video timing
- External synchronization issues
- Screen repaint control issues

### 8.1.1. VIDEO CLOCK GENERATION

Figure 125 shows the different clock generation circuits for the video section. Three clocks are generated: CRTC_CLK, QSF_CLK and SHIFT_CLK. These clocks control different sections of the video display logic.

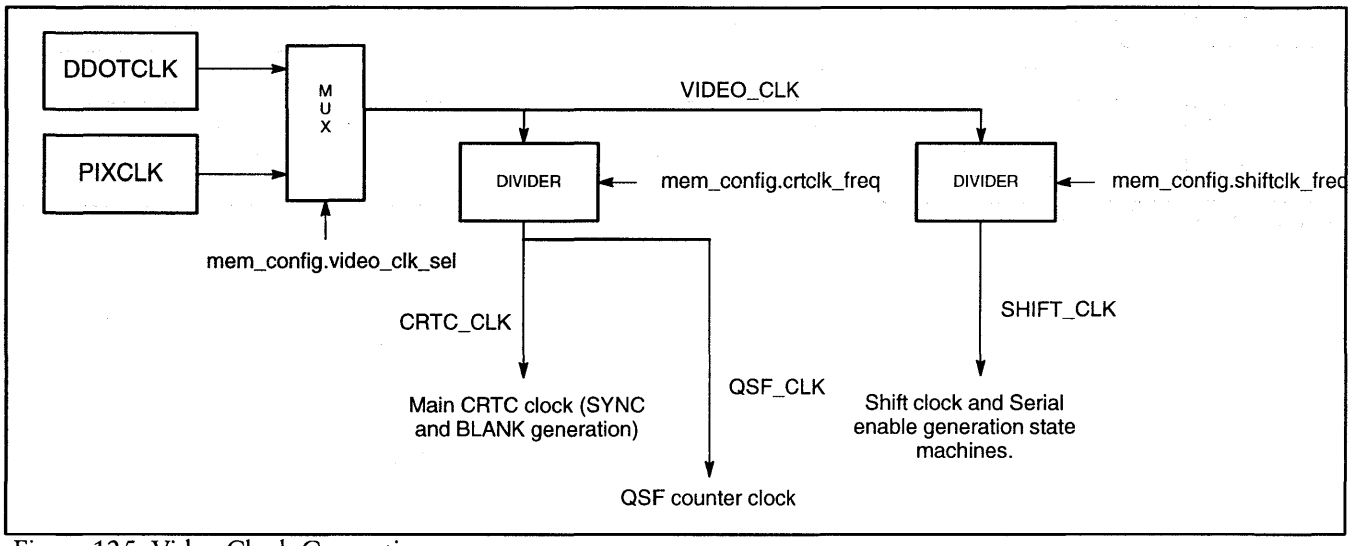



Figure 125. Video Clock Generation

## 8.1. Video Control, continued

### 8.1.2. VIDEO SHIFT CLOCK GENERATION

Figure 126 shows the different clock generation patterns for different values of mem_config.shiftclk_mode and mem_config.vad_sht

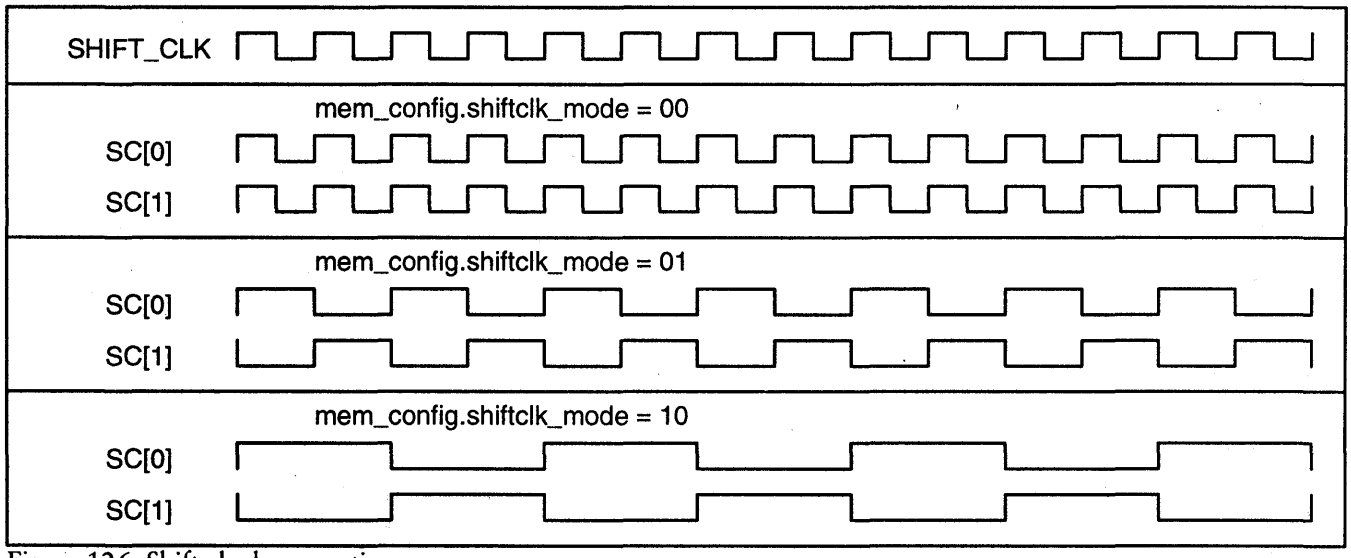



Figure 126. Shift clock generation

## 8.1. Video Control, continued

### 8.1.3. VIDEO SERIAL ENABLE GENERATION

Figure 127 shows the different serial enable generation patterns for different values of mem_config.soe_mode and mem_config.vad_sht.

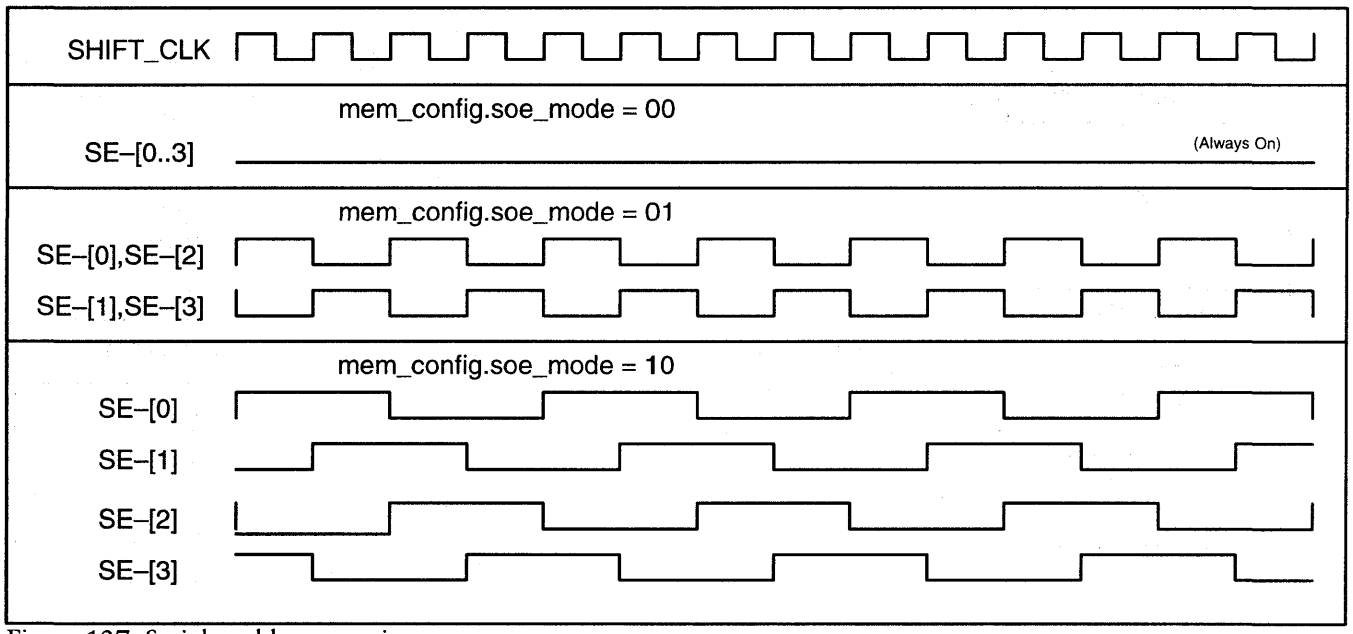
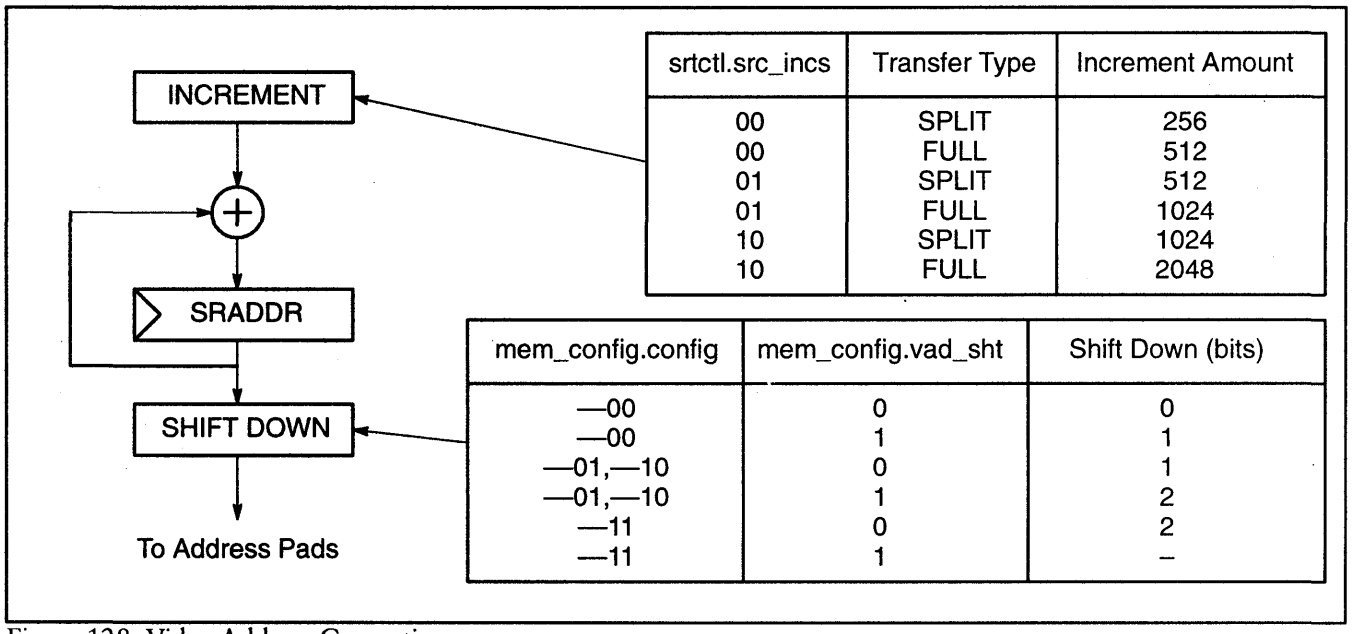



Figure 127. Serial enable generation

## 8.1. Video Control, continued

### 8.1.4. VIDEO ADDRESS GENERATION

Figure 128 shows how the VRAM shift register reload address is controlled. Since the shiftdown at the bottom of the data path is controlled somewhat by the memory configuration, the amount programmed into the srtctl.src_incs field must be computed to correct for this shiftdown.




| srtctl.src_incs | Transfer Type | Increment Amount |
|---|---|---|
| 00 | SPLIT | 256 |
| 00 | FULL | 512 |
| 01 | SPLIT | 512 |
| 01 | FULL | 1024 |
| 10 | SPLIT | 1024 |
| 10 | FULL | 2048 |

| mem_config.config | mem_config.vad_sht | Shift Down (bits) |
|---|---|---|
| —00 | 0 | 0 |
| —00 | 1 | 1 |
| —01,—10 | 0 | 1 |
| —01,—10 | 1 | 2 |
| —11 | 0 | 2 |
| —11 | 1 | – |

Figure 128. Video Address Generation

## 8.1. Video Control, continued

### 8.1.5. VIDEO SIGNALS

The Power 9100 provides two distinct video signal options: separate horizontal and vertical signals or composite (combined) horizontal and vertical signals.

To select the signal type required for a specific application, set the composite bit in the srtctl register appropriately (see figure 74). This bit setting controls the definition of the CSYNC/HBLNK– and CBLNK/VBLNK– signals. The synchronization signals selected for the separate signals option are: HSYNC–, VSYNC–, HBLNK–, and VBLNK–. The synchronization signals selected for the composite signals option are: HSYNC–, VSYNC–, CSYNC–, and CBLNK–.

The HSYNC– and VSYNC– signals operate identically in either output mode.

The bits srtctl.jam_vsync and srtctl.jam_hsync are used to implement the VESA standard monitor power management. When these bits are set the corresponding VSYNC– and HSYNC– signals are prohibited from changing. This forcing is done prior to the effect of the srtctl.vsync_polarity and srtctl.hsync_polarity. Thus when video is disabled (srtctl.enable_video = 0) the BLANK signal is asserted and the VSYNC–and HSYNC– signals are unchanging at the voltage level controlled by srtctl.vsync_polarity and srtctl.hsync_polarity respectively.

### 8.1.6. VIDEO CONTROL REGISTERS

A complete video control section resides on the Power 9100 chip. This section provides the synchronization, blanking, and timing signals for the graphics subsystem, as well as the control sections for screen refresh timing and VRAM control.

The video control section of the Power 9100 is fully programmable and is defined by the host system. For example: The video control section can be defined to generate an internal sync signal or accept an external signal to control screen refresh, thereby simplifying the task of merging Power 9100-created graphics with other video images.

This document talks consistently about VSYNC– and HSYNC–. This refers to the internal signals, not the external pins. The external pins are controlled by srtctl2.vsync_plt and srtctl2.hsync_plt. These controls allow the selection of external signal polarity and the support of VESA monitor power down modes.

For easy reference, figure 129 repeats the video control register summary presented in chapter 4.

## 8.1. Video Control, continued

| Register | Description/Function |
|---|---|
| *HORIZONTAL TIMING* | |
| hrzc | Horizontal counter. Read only. Specifies the current pixel position along a horizontal sweep; the Power 9100 increments this counter, which is originally set by the host, upon each occurrence of CRTC_CLK. The value occupies the lower 12 bits of the register, which is set by the Power 9100. |
| hrzt | Horizontal length. Read/write. Specifies the length of a horizontal scan line. The value occupies the lower 12 bits of the register, which is set by the host. The Power 9100 compares the current hrzc value (the current pixel position) to this value to determine when to wrap around. |
| hrzsr | Horizontal sync rising edge. Read/write. Specifies the location along a horizontal sweep which defines the horizontal sync rising edge. The value occupies the lower 12 bits of the register, which is set by the host. |
| hrzbr | Horizontal blank rising edge. Read/write. Specifies the location along a horizontal sweep which defines the horizontal blank rising edge. The value occupies the lower 12 bits of the register, which is set by the host. |
| hrzbf | Horizontal blank falling edge. Read/write. Specifies the location along a horizontal sweep which defines the horizontal blank falling edge. The value occupies the lower 12 bits of the register, which is set by the host. |
| prehrzc | Horizontal counter preload value. Read/write. Specifies the value with which to preload hrzc upon receipt of an internal or external HSYNC– or an external VSYNC–; allows synchronization of the Power 9100 with external video sources, whatever the combination of internal or external delays. The value occupies the lower 12 bits of the register, which is set by the host. Set this register to zero when using only internal syncs. |
| *VERTICAL TIMING* | |
| vrtc | Vertical counter. Read only. Specifies the current line position along a vertical sweep; the Power 9100 increments this counter upon each occurrence of HSYNC–. The value occupies the lower 12 bits of the register, which is set by the Power 9100. |
| vrtt | Vertical length. Read/write. Specifies the number of lines in a vertical sweep. The value occupies the lower 12 bits of the register, which is set by the host. The Power 9100 compares the current vrtc value (the current line position) to this value to determine when to wrap around. |
| vrtsr | Vertical sync rising edge. Read/write. Specifies the location along a vertical sweep which defines the vertical sync rising edge. The value occupies the lower 12 bits of the register, which is set by the host. |
| vrtbr | Vertical blank rising edge. Read/write. Specifies the location along a vertical sweep which defines the vertical blank rising edge. The value occupies the lower 12 bits of the register, which is set by the host. |
| vrtbf | Vertical blank falling edge. Read/write. Specifies the location along a vertical sweep which defines the vertical blank falling edge. The value occupies the lower 12 bits of the register, which is set by the host. |
| prevrtc | Vertical counter preload value. Read/write. Specifies the value with which to preload vrtc upon receipt of an internal or external VSYNC–; allows synchronization of the Power 9100 with external video sources, whatever the combination of internal or external delays. The value occupies the lower 12 bits of the register, which is set by the host. Set this register to zero when using only internal syncs. |
| *SCREEN REPAINT* | |
| srtctl2 | Screen refresh timing control. Read/write. Specifies controls for screen refresh, as set by the host. Figure 75 defines this register. |
| srtctl | Screen refresh timing control. Read/write. Specifies controls for screen refresh, as set by the host. Figure 74 defines this register. |
| qsfcounter | QSF counter. Read only. Used to determine when to generate a shift register load operation. It is a duplicate of the QSF signal from the VRAMs. It keeps track of which part of the SAM is being shifted out. It is loaded with a zero after every read transfer and incremented by CRTC_CLK. |

Figure 129. Video control registers

## 8.1. Video Control, continued

### 8.1.7. VIDEO TIMING

The video signals and registers work together to drive the signals that control the monitor. Remember, all counts are in units of CRTC_CLK (see figure 125).

*HORIZONTAL VIDEO TIMING*

Figure 132 presents the timing sequence for horizontal video control . All of the horizontal timing registers are loaded with counts derived from CRTC_CLK which may represent multiple pixels. Each of the values stored in the horizontal timing registers represents the total count minus one to allow for zero in the count sequence.

Figure 130 describes the functions of the horizontal timing registers.

The programmer must satisfy the following condition:

hrzsr < hrzbr < hrzbf < hrzt

| Register | Description/Function |
|---|---|
| hrzt | Defines the number of CRTC_CLKs from the falling edge of HSYNC– to the falling edge of the next HSYNC–. |
| hrzc | Counts from zero to hrzt and then repeats. |
| hrzsr | Specifies the HSYNC– active low count. |
| hrzbr | Specifies the distance from HSYNC– low to the rising edge of HBLINK–. |
| hrzbf | Specifies the distance from HSYNC– low to the falling edge of HBLINK–. |

Figure 130. Horizontal timing registers

*VERTICAL VIDEO TIMING*

Figure 133 presents the timing diagram for vertical video timing control. All of the vertical timing registers are loaded with counts that represent horizontal scan lines. Also, each number loaded into a vertical timing register represents the total count in the count sequence.

Figure 131 describes the functions of the vertical timing registers.

The programmer must satisfy the following condition:

vrtsr < vrtbr < vrtbf < vrtt

| Register | Description/Function |
|---|---|
| vrtt | In the vertical timing sequence, vrtt defines the number of horizontal lines from the falling edge of VSYNC– to the falling edge of the next VSYNC– (the number of horizontal lines in a complete vertical scan cycle). The vrtc register counts from zero to vrtt–1 and then repeats. |
| vrtsr | Specifies the VSYNC– active low count. The rising and falling transitions of VSYNC– should match those of HSYNC–. |
| vrtbr | Specifies the distance from VSYNC– low to the rising edge of VBLNK–. |
| vrtbf | Specifies the distance from VSYNC– low to the falling edge of VBLNK–. |

Figure 131. Vertical timing registers

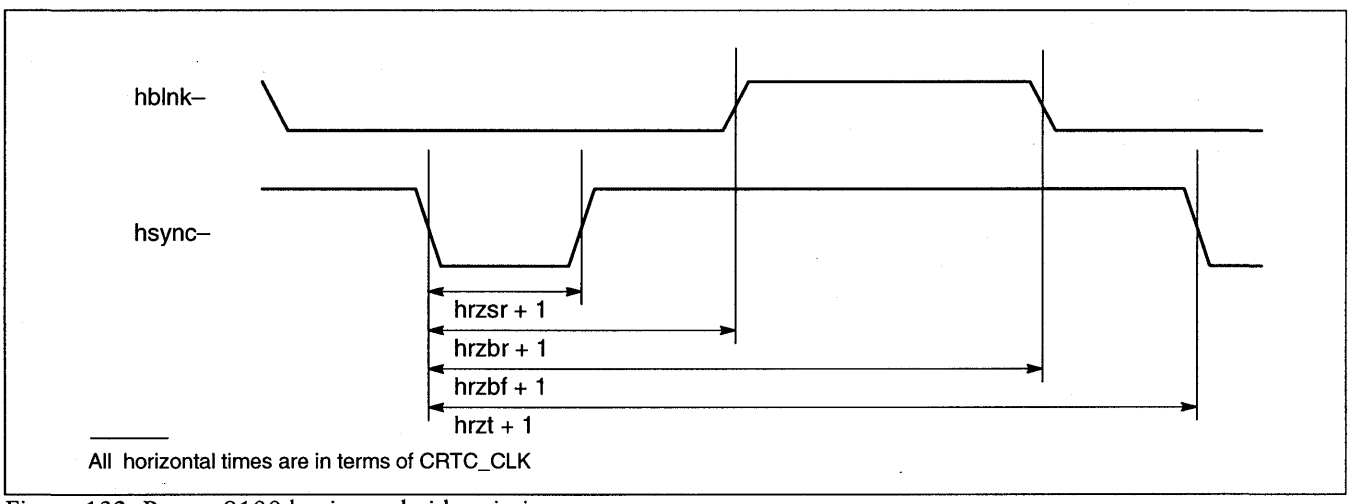



Figure 132. Power 9100 horizontal video timing parameters

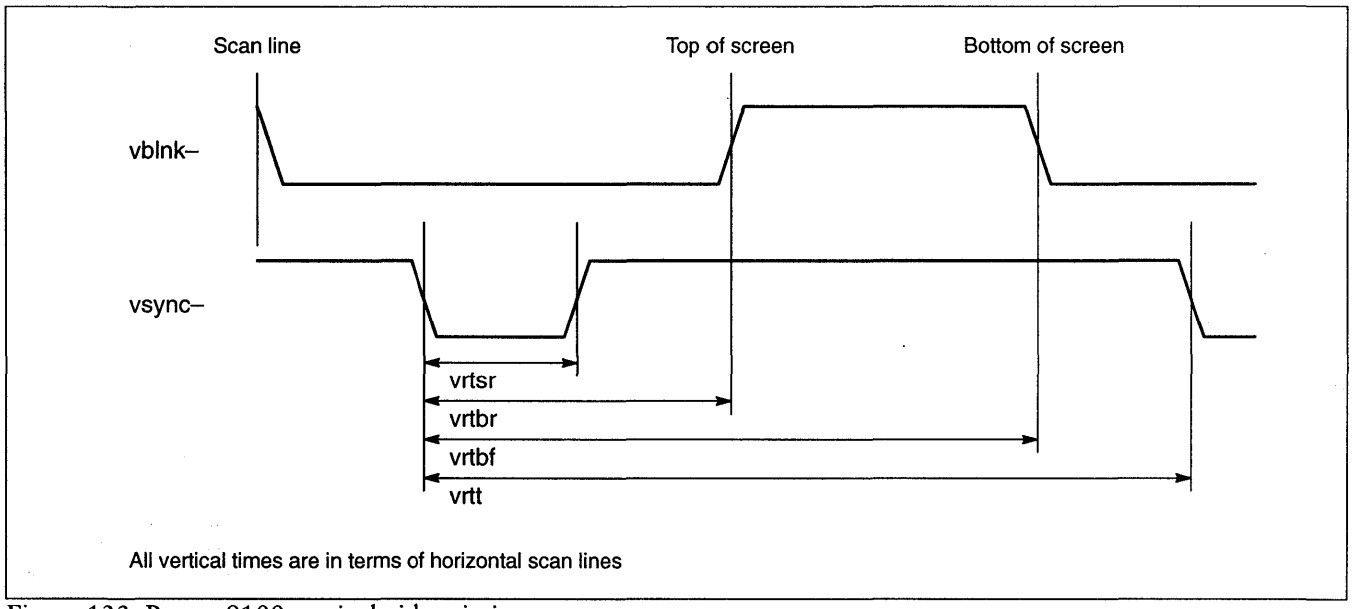



Figure 133. Power 9100 vertical video timing parameters

## 8.1.8. EXTERNAL SYNCHRONIZATION

The VSYNC– and HSYNC– pins can be defined as inputs to provide synchronization from an external source. This allows merging of Power 9100 video with video generated by an external source. With values programmed into prehrzc and prevrtc, the Power 9100 can synchronize exactly with the external source, regardless of internal and external delays in the system.

There are two standard modes for implementation of external sync timing: external VSYNC– only and external VSYNC– and HSYNC–.

With external VSYNC– only, the Power 9100 preloads both the horizontal and vertical counters (hrzc and vrtc) on the falling edge of an external VSYNC– pulse. This synchronizes both HSYNC– and VSYNC– to the external source.

With external VSYNC– and HSYNC–, the Power 9100 preloads the hrzc counter on the falling edge of an external HSYNC– and preloads the vrtc counter on the falling edge of an external VSYNC–.

HSYNC– and VSYNC– input pulses must be at least one DDOTCLK wide, but must also be shorter than the horizontal sync rising edge defined in the hrzsr register.

## 8.1.9. SCREEN REPAINT CONTROL

Screen repaint is a generic descriptor for the functions used by the frame buffer memory to provide pixel information to the "back-end" of the video subsystem to display stored information on the monitor. The Power 9100 provides two basic methods to ensure that the shift registers of the VRAMs always contain the correct data to be shifted onto the screen. These two methods are controlled by the setting of the hblnk_reload bit in the srtctl register (see figure 74).

### *NORMAL (SPLIT SHIFT MODE)*

For normal operation, the hblnk_reload bit in the srtctl register is zero. During a vertical retrace operation (VBLNK– asserted), the Power 9100 reloads the entire shift register inside the VRAMs and then performs a split shift register reload whenever the VRAM output shift registers are more than half empty, based on the internally generated QSF signal.

The Power 9100 does not intervene between scan lines (during HBLNK–), this method requires that each scan line of the image be located in sequential linear memory and that the VRAM shift clock be inhibited during HBLNK– intervals.

The physical scan line width of the monitor must exactly match the logical scan line width loaded into the drawing engine. This requires that there be no extra pixels between scan lines in memory. The initial scan-line increment after VBLNK– equals one row of VRAM addresses and subsequent increments equal one-half row.

## 8.1. Video Control, continued

*RESTRICTED (HBLNK– RELOAD MODE)*

For restricted operation, the hblnk_reload bit in the srtctl register is one. The entire VRAM shift register is reloaded for every scan line (HBLNK– asserted). In this mode, every scan line in the VRAM must be entirely contained within a single shift register. This constrains scan line length to be less than one whole, one half, or one quarter of the entire shift register length. When using this mode, the available screen resolutions are restricted, as defined in figure 134.

| VIRTUAL SCREEN SIZES 8bpp Values (Restricted Mode) (Divide by 2 for 15/16bpp, divide by 4 for 32bpp) | | | | | | |
|---|---|---|---|---|---|---|
| men_config.config | Maximum Screen Resolution | | Scan Increment | Double Buffering | Memory | Banks |
| | Horizontal | Vertical | | | | |
| 0001, 0010, 0100 | 1024 | 1024 | 1/2 row | No | 1M | 1 bank, 256K<br>2 banks, 128K |
| 0011, 0101, 0110 | 2048 | 1024 | 1/2 row | No | 2M | 4 banks, 128K<br>2 banks, 256K |
| 1011, 1110, 1101 | 1024 | 1024 | 1/4 row | Yes | 2M | 2 banks, 256K<br>4 banks, 128K |
| 1111 | 2048 | 1024 | 1/4 row | Yes | 4M | 4 banks, 256K |
| 0111 | 2048 | 2048 | 1/4 row | No | 4M | 4 banks, 256K |

Figure 134. Screen resolutions in 8 bpp values, restricted mode (divide by 2 for 15/16bpp, divide by 4 for 32bpp)

## 8.2. RAMDAC

Figure 135 shows the RAMDAC read cycle. Figure 139 shows the RAMDAC write cycle. The Power 9100 designs are expected to utilize a triple 8-bit color with VGA pseudo-color look up table and pixel unpack RAMDAC (e.g. Brooktree 485). The host has 8-bit data access to the RAMDAC through the shared frame buffer pins. Only the RAMDAC status read register is shadowed in Power 9100.

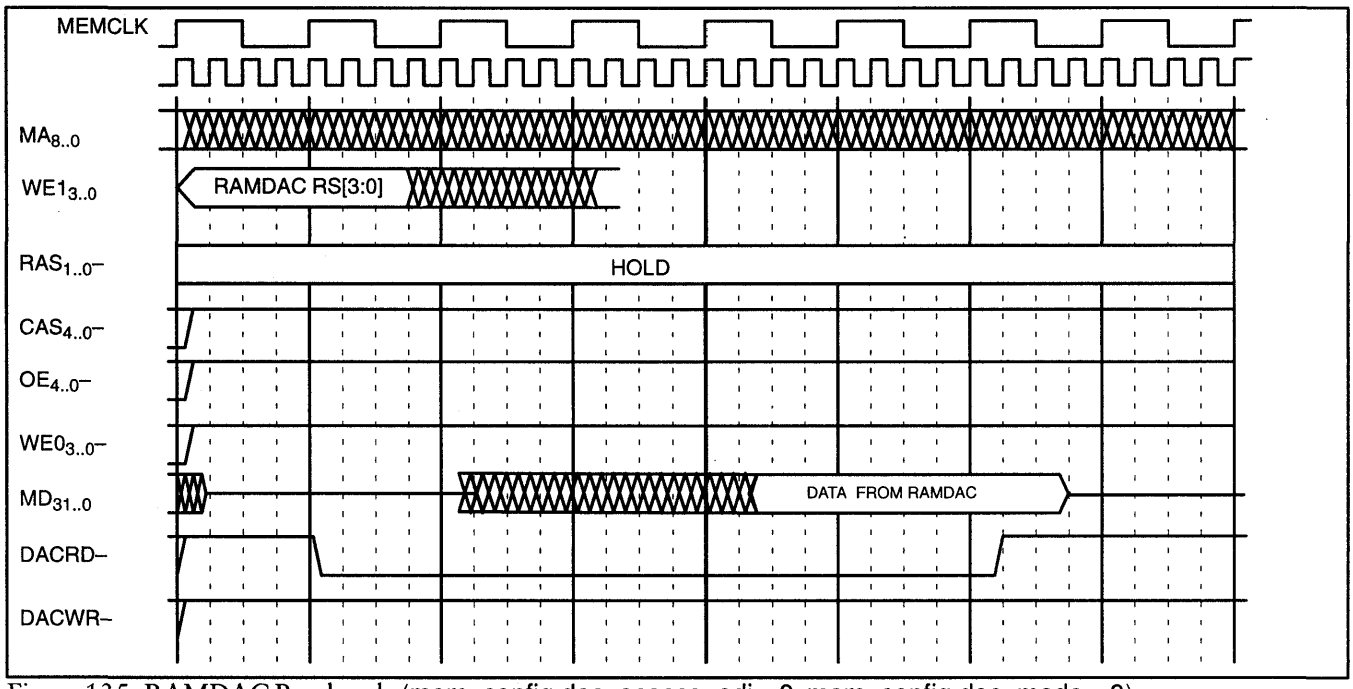



Figure 135. RAMDAC Read cycle (mem_config.dac_access_adj = 0, mem_config.dac_mode = 0)

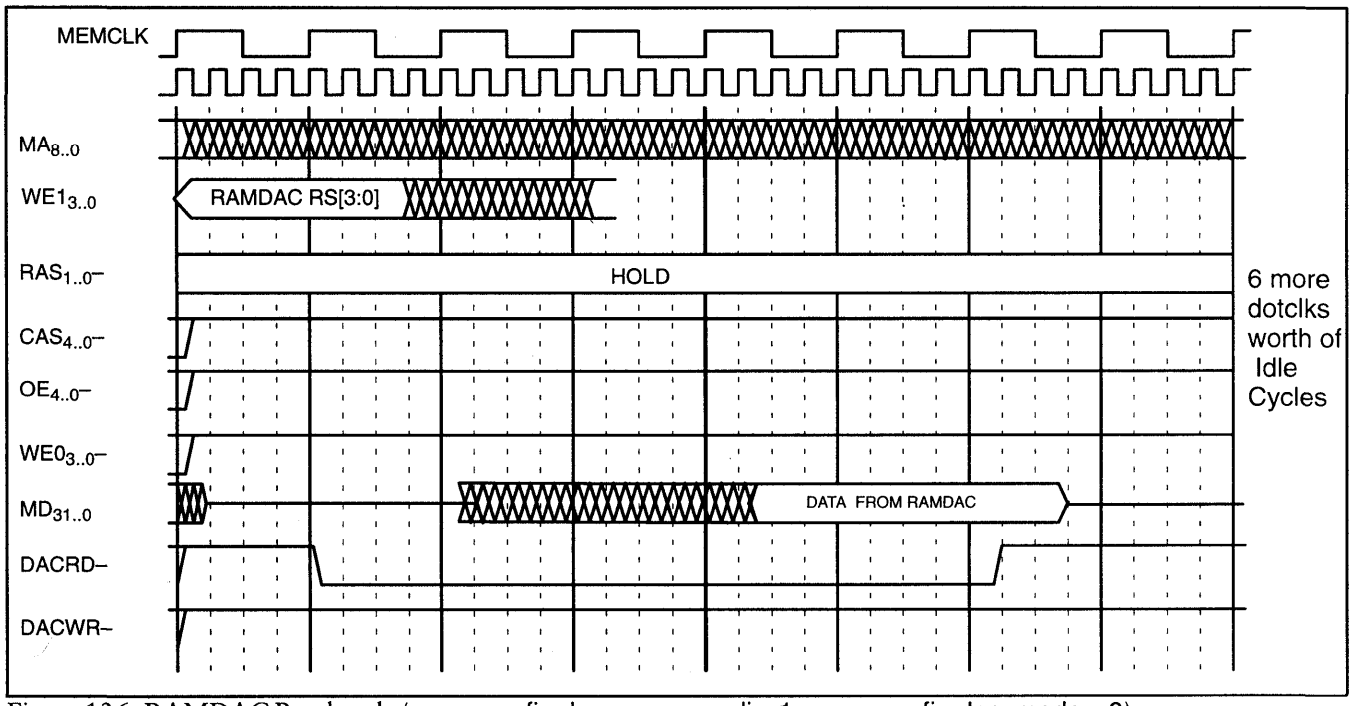



Figure 136. RAMDAC Read cycle (mem_config.dac_access_adj = 1, mem_config.dac_mode = 0)

## 8.2. RAMDAC, continued

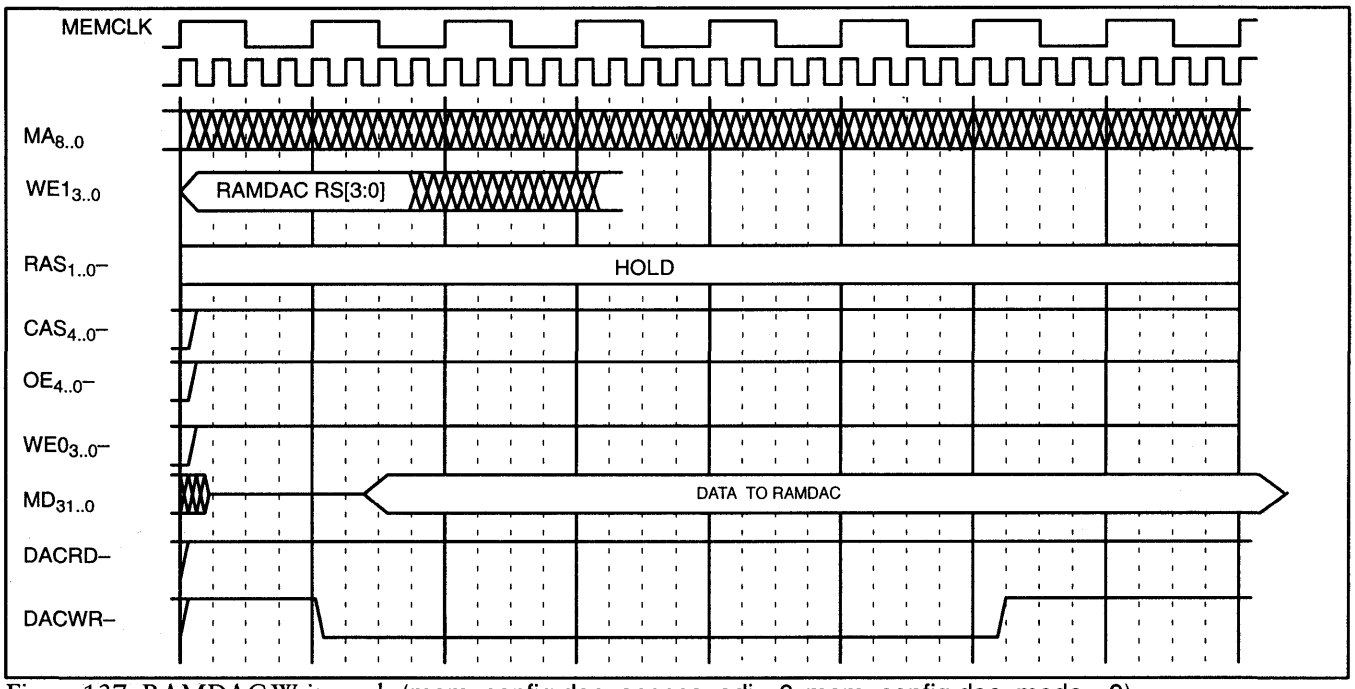



Figure 137. RAMDAC Write cycle (mem_config.dac_access_adj = 0, mem_config.dac_mode = 0)

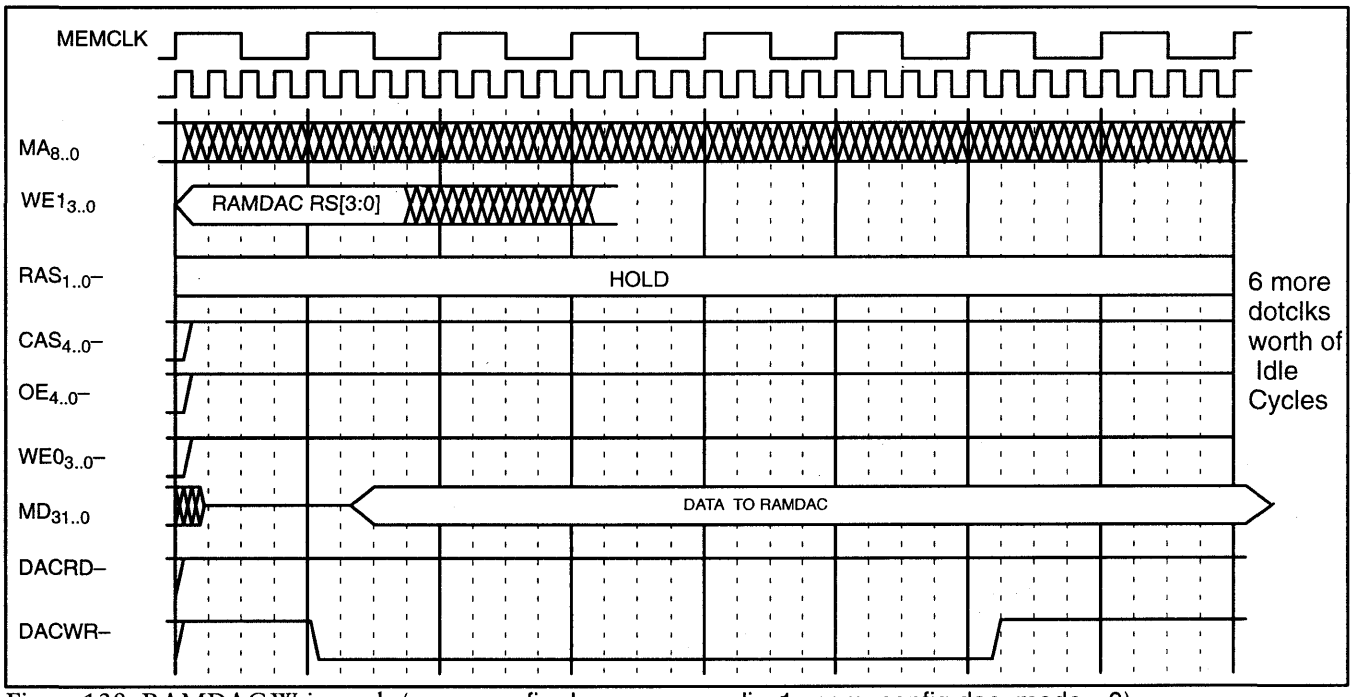



Figure 138. RAMDAC Write cycle (mem_config.dac_access_adj = 1, mem_config.dac_mode = 0)

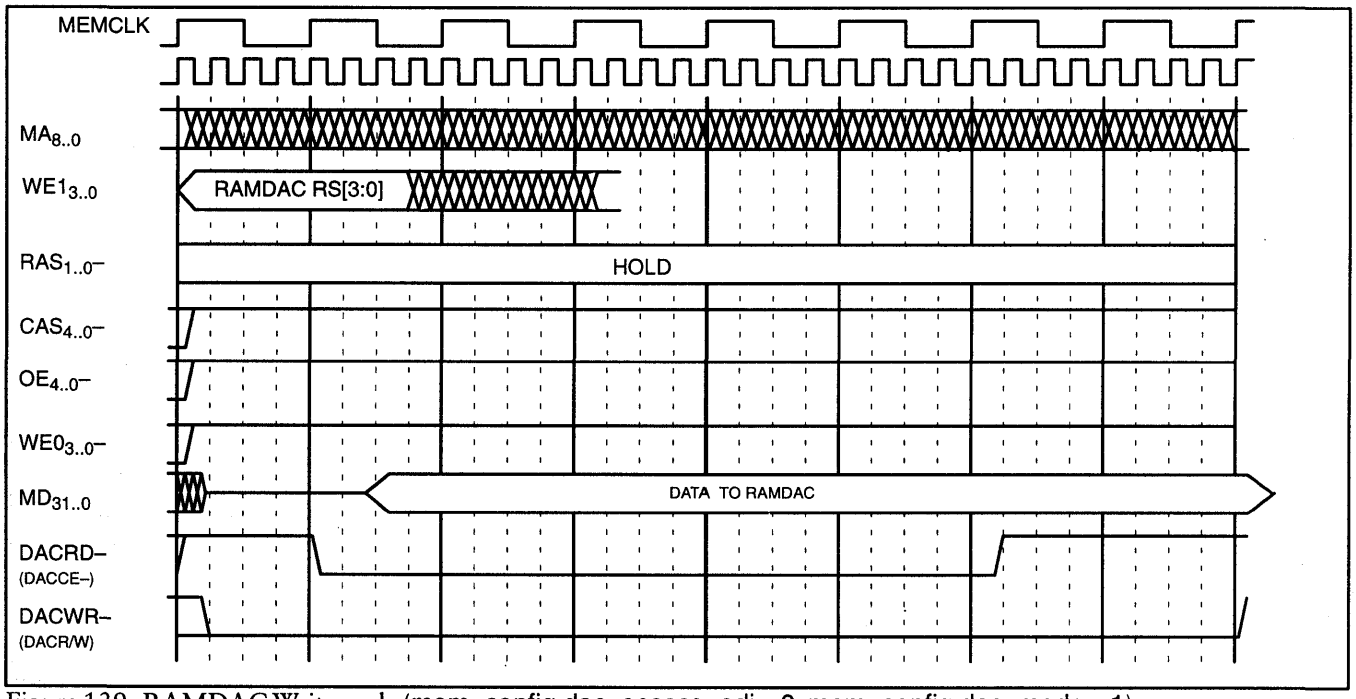



Figure 139. RAMDAC Write cycle (mem_config.dac_access_adj = 0, mem_config.dac_mode = 1)

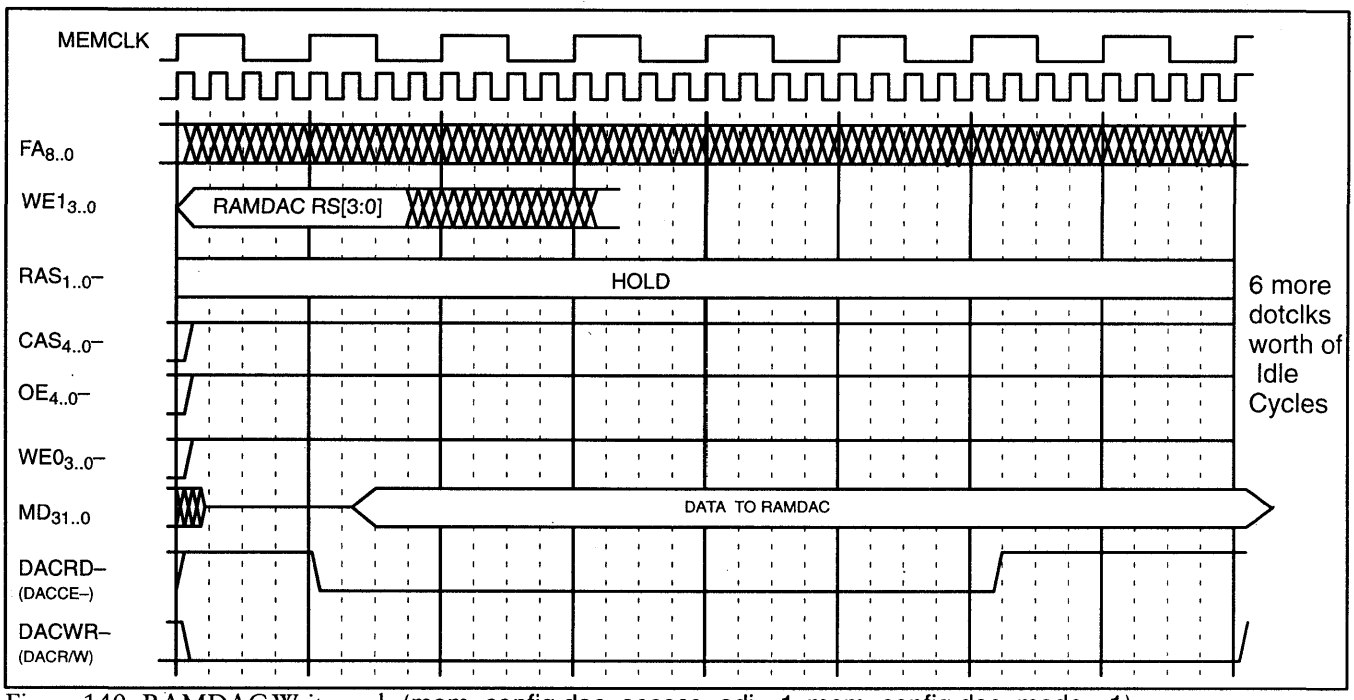



Figure 140. RAMDAC Write cycle (mem_config.dac_access_adj = 1, mem_config.dac_mode = 1)

# Chapter 9. Coprocessor Interface

The video coprocessor interface allows a separate coprocessor to share the Power 9100 host interface and frame buffer. The first set of timing diagrams allow the host to read and write control registers of the video coprocessor. The second set of diagrams specifies how the protocol for sharing the frame buffer operates. The coprocessor uses the VCEN– signal to qualify the shared VCGRNT–, VCIOR– and VCIOW– signals.

## 9.1. Video Coprocessor I/O Read

This template is invoked to read a 32-bit register from the video coprocessor.

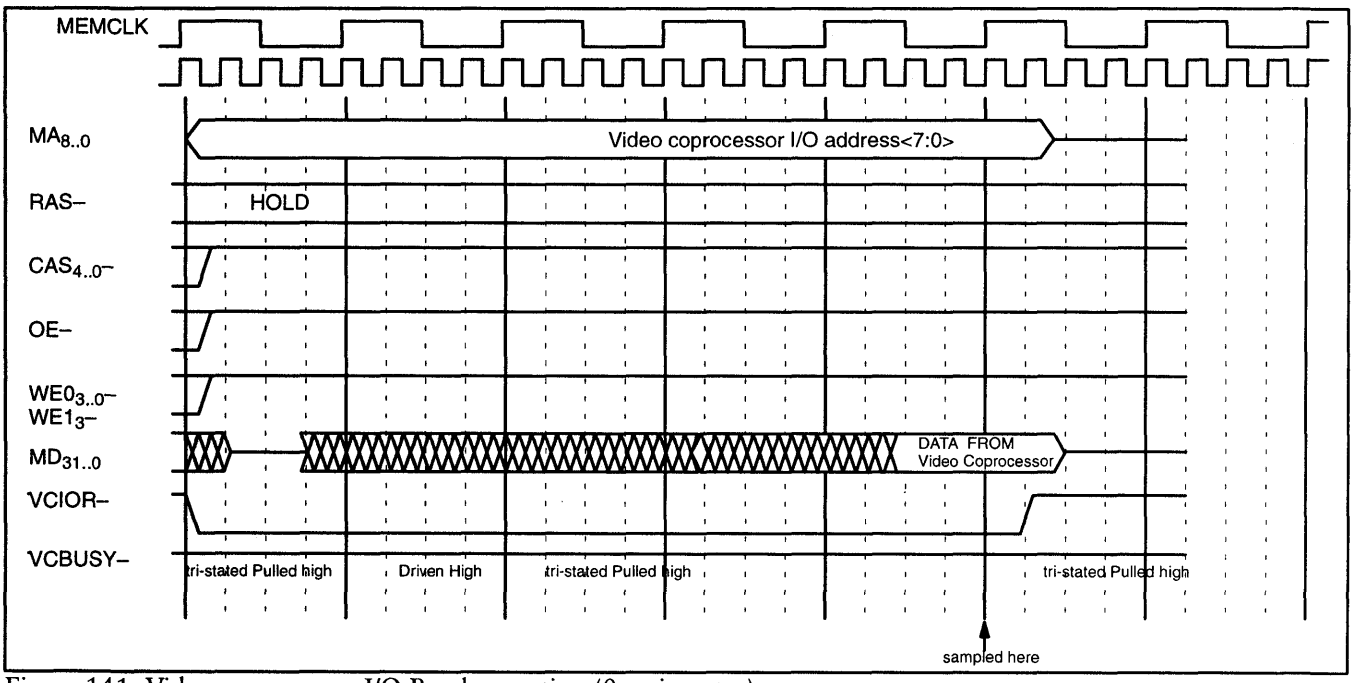



Figure 141. Video coprocessor I/O Read operation (0 wait states)

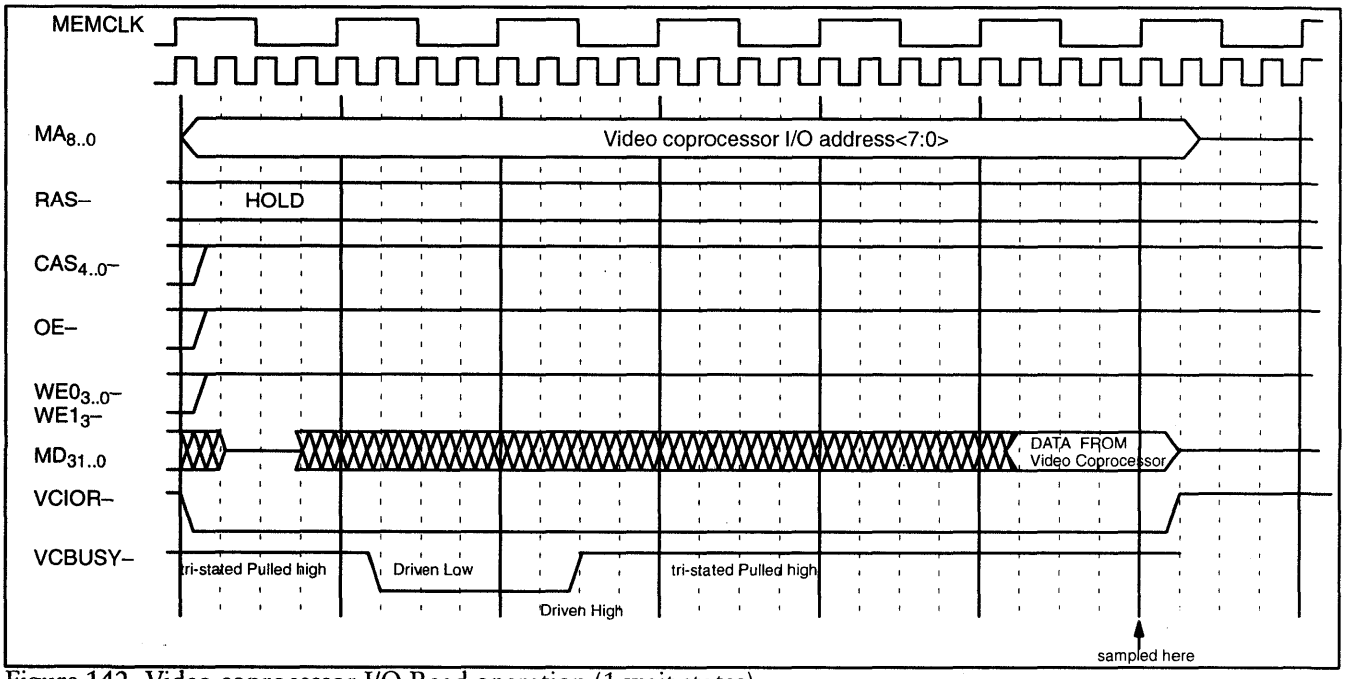



Figure 142. Video coprocessor I/O Read operation (1 wait states)

## 9.2. Video Coprocessor I/O Write

This template is invoked to write a 32-bit value to a register in the video coprocessor.

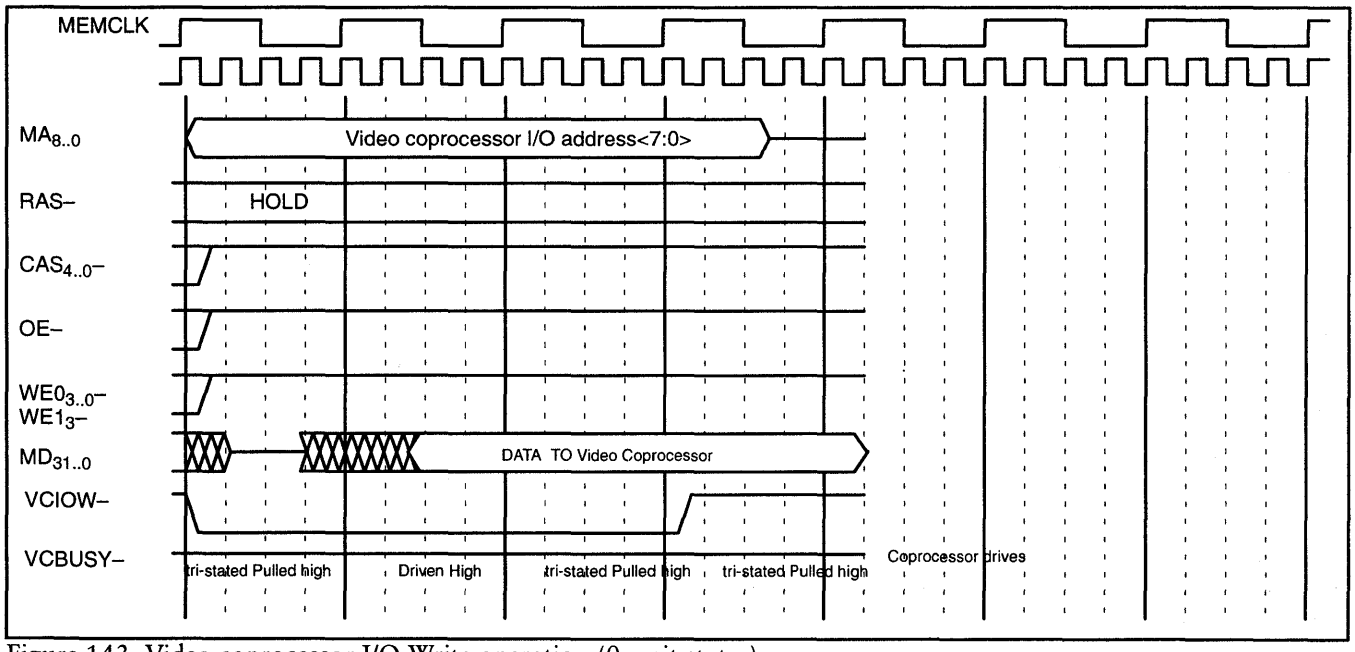



Figure 143. Video coprocessor I/O Write operation (0 wait states)

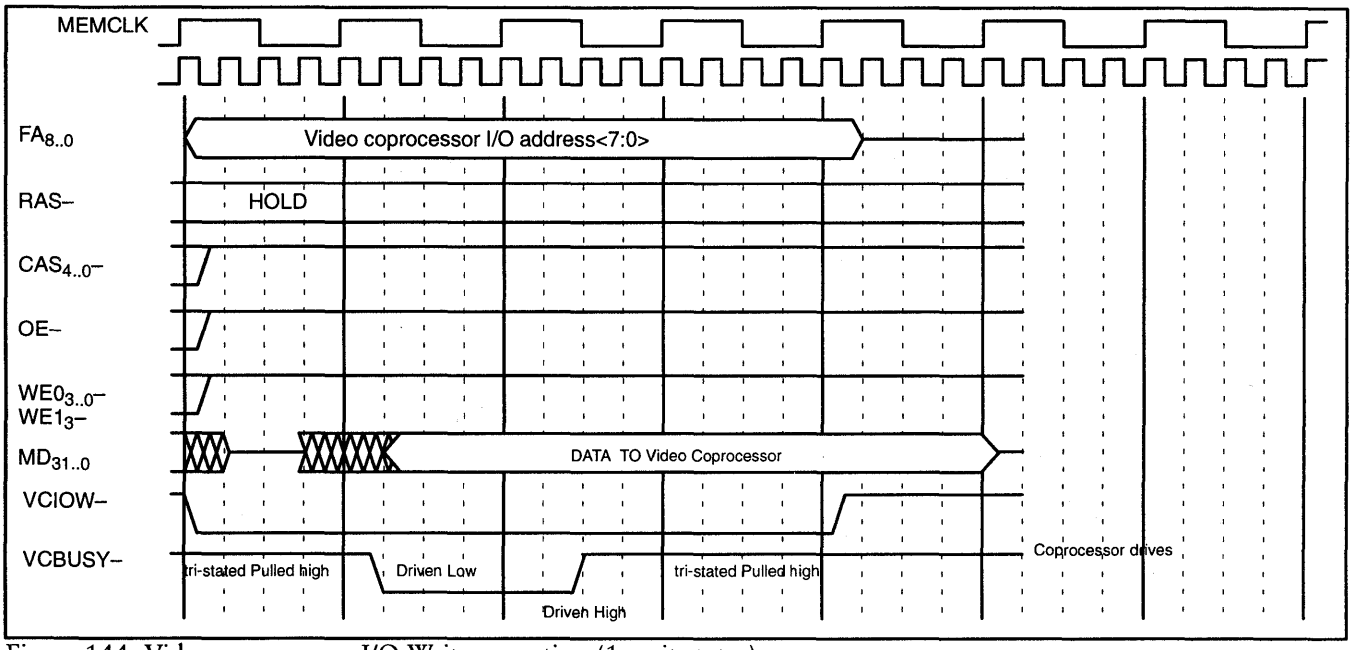



Figure 144. Video coprocessor I/O Write operation (1 wait states)

## 9.3. Video Coprocessor Grant

The video coprocessor requests direct access to the frame buffer by asserting the VCREQ– signal. After some number of cycles, the Power 9100 will grant access to the frame buffer. Figure 145 shows the Power 9100 grant sequence.

Note that once VCREQ– is asserted, it must remain asserted until a complete grant/release sequence has completed.

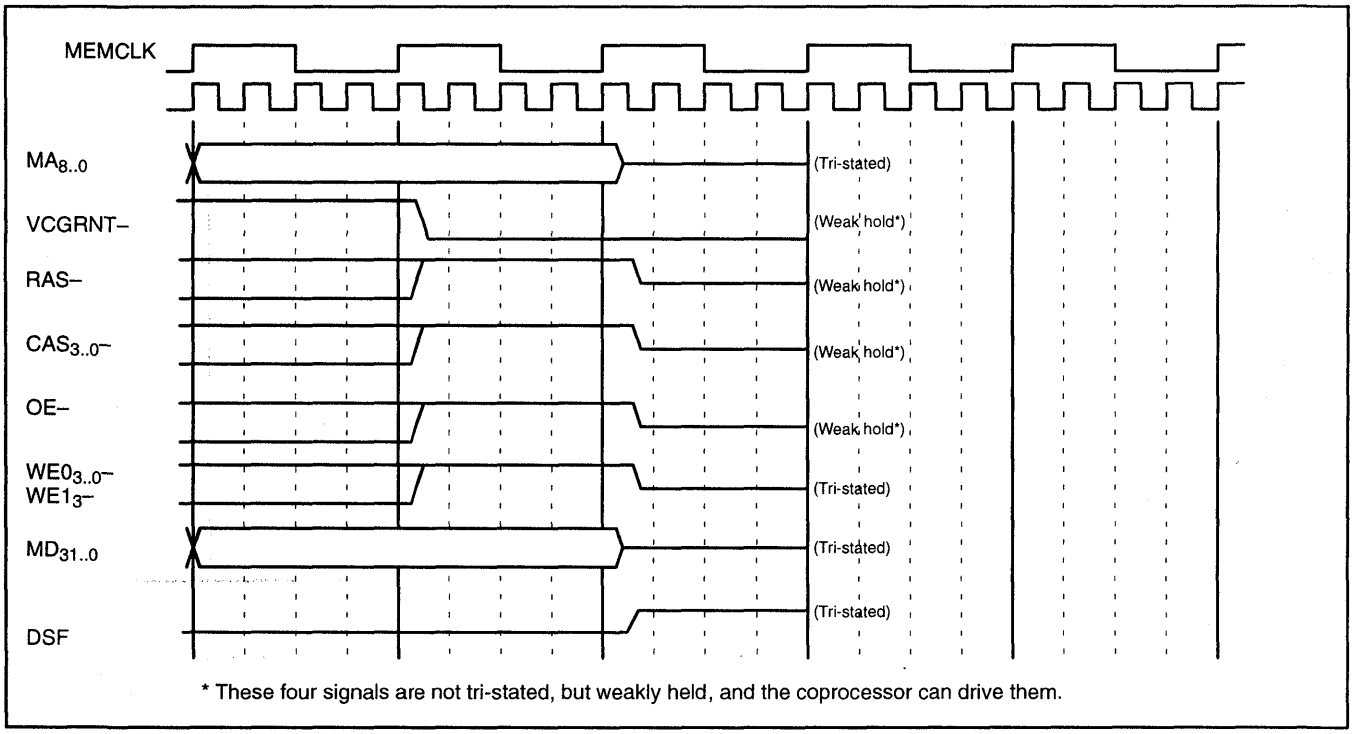
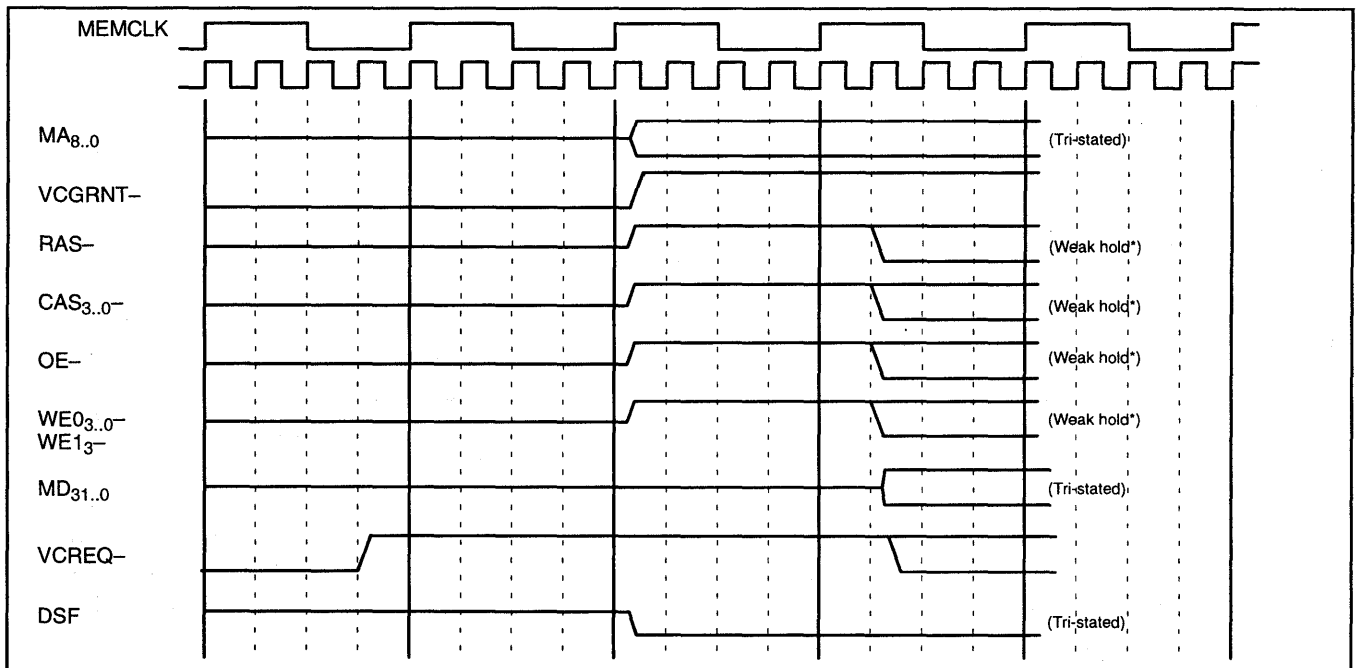



Figure 145. Video coprocessor grant operation

## 9.4. Video Coprocessor Release

When the video coprocessor wishes to release the frame buffer back to the Power 9100 is deasserts VCREQ–. The Power 9100 responds by deasserting VCGRNT– two cycles later. It also again drives all of the frame buffer control signals starting in the same cycle. The video coprocessor must retain VCREQ– deasserted for at least 2 cycles.




* These four signals are not tri-stated, but weakly held, and the coprocessor can drive them.

Figure 146. Video coprocessor release operation

## 9.5. Video Coprocessor Preempt

For higher priority operations (shift register reload and memory refresh) the Power 9100 will preempt the video coprocessor. This is done by deasserting the VCGRNT– signal. This informs the video coprocessor that a higher priority operation is pending. It must complete the current operation and release the frame buffer by deasserting the VCREQ– just like in the release operation.

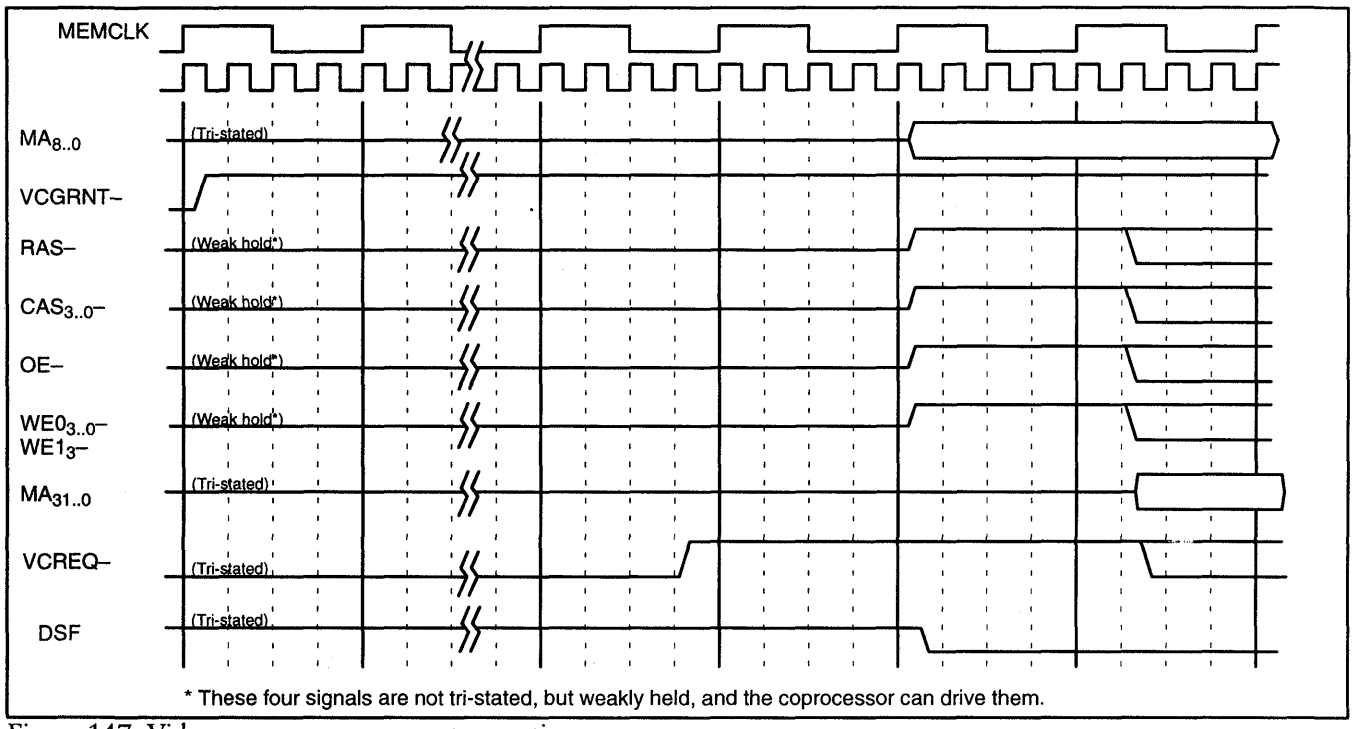



Figure 147. Video coprocessor preempt operation

# Chapter 10. Auxiliary Chip Control

## 10.1. EEPROM Control

The optional EEPROM is connected via two shared pins: CKSEL[2] and VCEN–. We use the $I^2C$ slave protocol. The software sees three bits: CONFIG[66].VCEN, CONFIG[66].CKSEL[2], and CONFIG[64].EEDAIN. The circuit schematic for connecting them is shown in figure 148

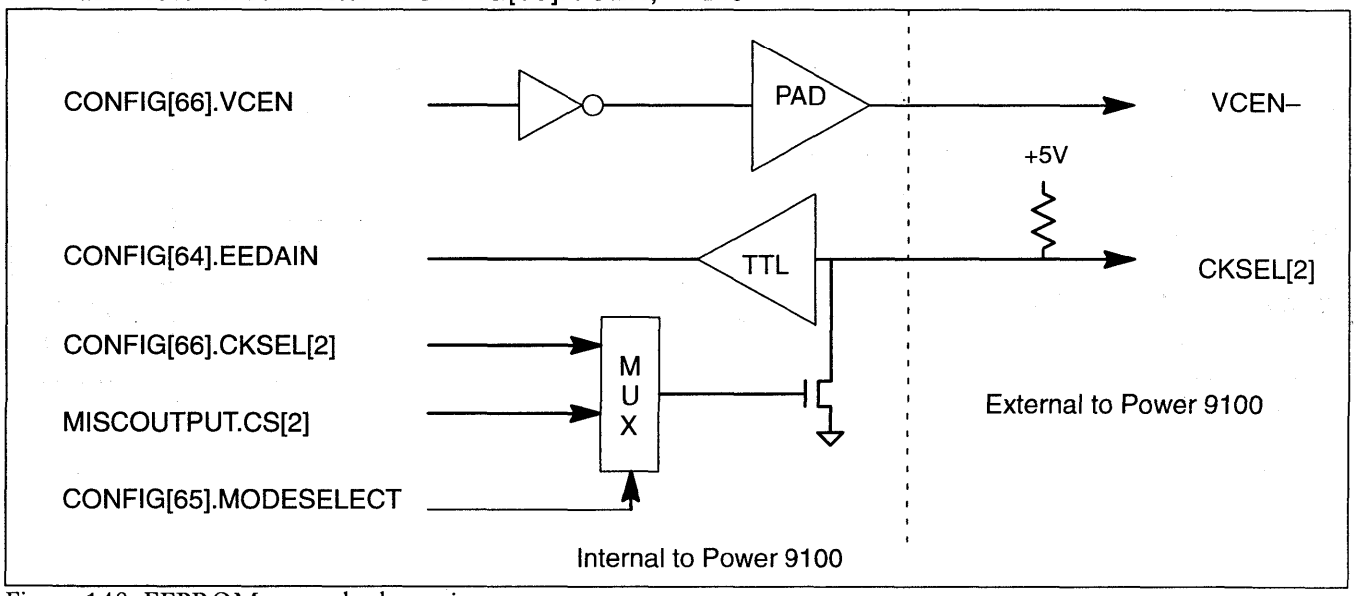



Figure 148. EEPROM control schematic.

## 10.2. Clock Synthesizer Control

The Power 9100 directly supports programmable clock synthesizers such as the ICD2016A from IC Designs. This requires 2 output pins to drive: CKSEL[1..0]. The pins are driven directly by the miscellaneous output register (I/O port 0x3CC in emulation mode) or from the CONFIG[66].CKSEL configuration bits in native mode.

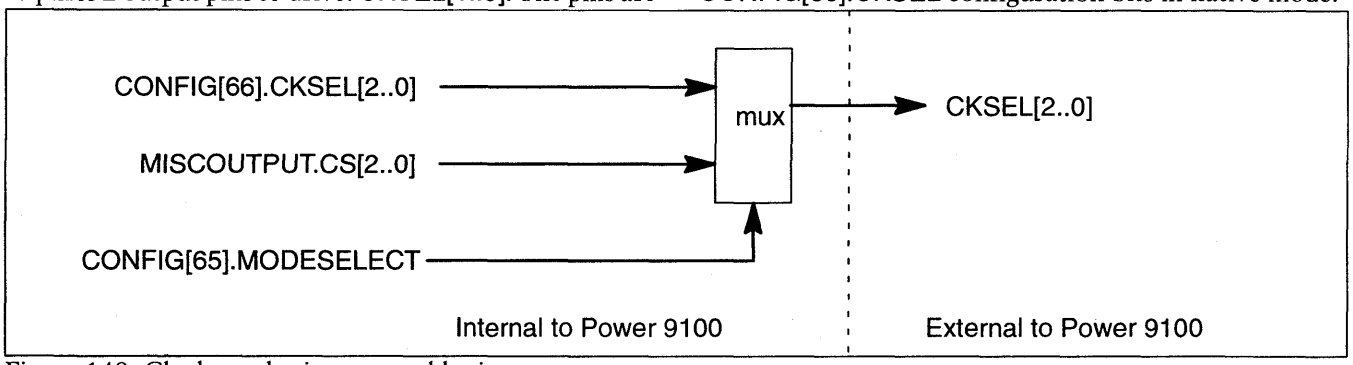



Figure 149. Clock synthesizer control logic

## 10.3. BIOS Access

Figure 150 shows the BIOS read cycle during the period when native mode is activated. It takes total of 8 memclk cycles to do one access. The last two cycles should provide enough data turn off time, which is typical for regular EPROMs. At 50 MHz, the Power 9100 will work with a 120ns EPROM.

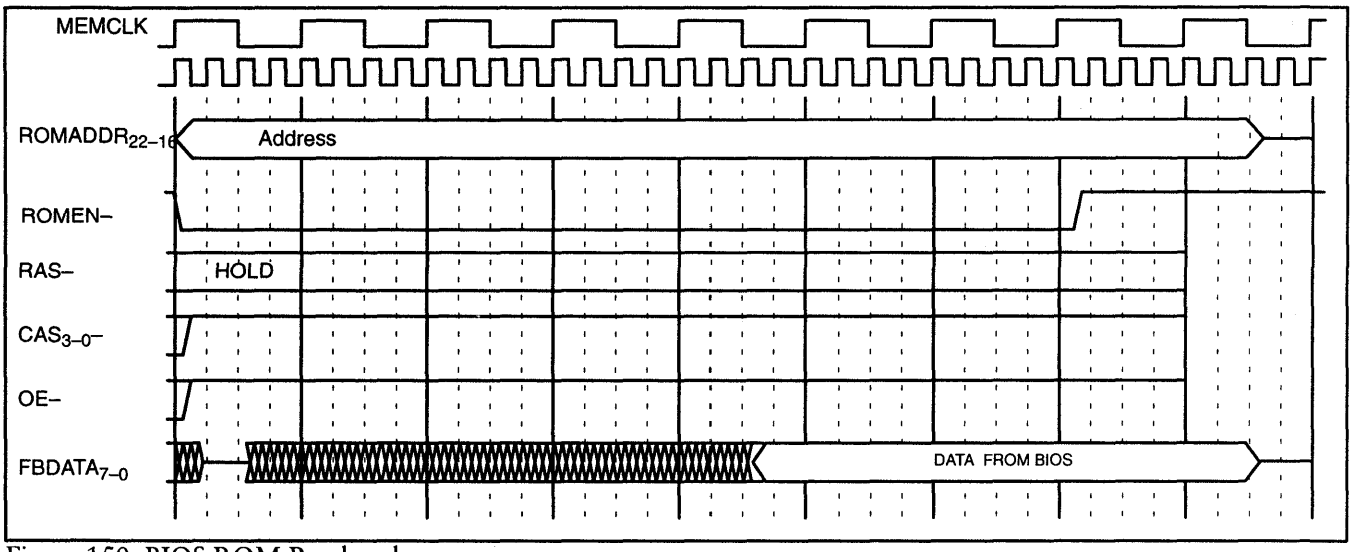



Figure 150. BIOS ROM Read cycle

# Chapter 11. SVGA Overview

## 11.1. SVGA Compatible Text/Graphics Engine

Resetting the Power 9100 sets it to VGA-emulation mode. The Power 9100 is 100% VGA-compatible in this mode.

When used as an SVGA, the Power 9100's performance is comparable to other high-performance VGA controllers on the market.

The Power 9100 supports all VGA modes, as shown in figure 151. (None of these modes are Hercules-compatible graphics modes; Hercules graphics compatibility is not part of the VGA standard.)

| VGA Name | Resolution Cols x Rows | Character Cell | Display Format | Number of Text Pages | Display Mode | Resolution (in pixels) |
|---|---|---|---|---|---|---|
| 0 | 40 x 25 | 8 x 8 | 16/256K bw | 8 | Alpha | 320 x 200 |
| 0* | 40 x 25 | 8 x 14 | 16/256K bw | 8 | Alpha | 320 x 350 |
| 0+ | 40 x 25 | 9 x 16 | 16/256K bw | 8 | Alpha | 360 x 400 |
| 1 | 40 x 25 | 8 x 8 | 16/256K | 8 | Alpha | 320 x 200 |
| 1* | 40 x 25 | 8 x 14 | 16/256K | 8 | Alpha | 320 x 350 |
| 1+ | 40 x 25 | 9 x 16 | 16/256K | 8 | Alpha | 360 x 400 |
| 2 | 80 x 25 | 8 x 8 | 16/256K bw | 8 | Alpha | 640 x 200 |
| 2* | 80 x 25 | 8 x 14 | 16/256K bw | 8 | Alpha | 720 x 400 |
| 2+ | 80 x 25 | 9 x 16 | 16/256K bw | 8 | Alpha | 720 x 200 |
| 3 | 80 x 25 | 8 x 8 | 16/256K | 8 | Alpha | 640 x 350 |
| 3* | 80 x 25 | 8 x 14 | 16/256K | 8 | Alpha | 720 x 400 |
| 3+ | 80 x 25 | 9 x 16 | 16/256K | 8 | Alpha | 320 x 200 |
| 4 | 40 x 25 | 8 x 8 | 4/256K | 1 | Graph | 320 x 200 |
| 5 | 40 x 25 | 8 x 8 | 4/256K bw | 1 | Graph | 320 x 200 |
| 6 | 80 x 25 | 8 x 8 | 4/256K bw | 1 | Graph | 640 x 200 |
| 7 | 80 x 25 | 9 x 14 | bw | 8 | Alpha | 720 x 400 |
| 7+ | 80 x 25 | 9 x 16 | bw | 8 | Alpha | 320 x 200 |
| D | 40 x 25 | 8 x 8 | 16/256K | 8 | Graph | 320 x 200 |
| E | 80 x 25 | 8 x 8 | 16/256K | 4 | Graph | 640 x 200 |
| F | 80 x 25 | 8 x 14 | bw | 2 | Graph | 640 x 350 |
| 10 | 80 x 25 | 8 x 14 | 16/256K | 2 | Graph | 640 x 350 |
| 11 | 80 x 30 | 8 x 16 | 2/256K | 1 | Graph | 640 x 480 |
| 12 | 80 x 30 | 8 x 16 | 16/256K | 1 | Graph | 640 x 480 |
| 13 | 40 x 25 | 8 x 8 | 256/256K | 1 | Graph | 320 x 200 |

Figure 151. Standard VGA display modes. All these modes are supported by the Power 9100 SVGA Mode

## 11.2. Enhanced Display Modes

| Screen Resolution | Vertical Refresh Rates Hz | Number of Display Colors * | Frame Buffer Size | DRAM32 50 MHz MEMCLK |
|---|---|---|---|---|
| | | | | |
| 640x480 | 56 to 60 | 16 | 256K | Yes |
| 640x480 | 56 to 60 | 256 | 512K | Yes |
| 640x480 | 56 to 60 | 32K | 1MB | Yes |
| 640x480 | 56 to 60 | 16M | 1MB | Yes |
| | | | | |
| 640x480 | 72 | 16 | 256K | Yes |
| 640x480 | 72 | 256 | 512K | Yes |
| 640x480 | 72 | 32K | 1MB | Yes |
| 640x480 | 72 | 16M | 1MB | Yes |
| | | | | |
| 800x600 | 56 to 60 | 16 | 256K | Yes |
| 800x600 | 56 to 60 | 256 | 512K | Yes |
| 800x600 | 56 to 60 | 32K | 1MB | Yes |
| | | | | |
| 800x600 | 72 | 16 | 256K | Yes |
| 800x600 | 72 | 256 | 512K | Yes |
| 800x600 | 60 Hz max | 32K | 1MB | No |
| | | | | |
| 1024x768 | 60 | 16 | 512K | Yes |
| 1024x768 | 60 | 256 | 1MB | Yes |
| | | | | |
| 1024x768 | 70 Hz max | 16 | 512K | Yes |
| 1024x768 | 70 Hz max | 256 | 1MB | Yes |
| | | | | |
| 1024x768 | 43.5 int | 16 | 512K | Yes |
| 1024x768 | 43.5 int | 256 | 1MB | Yes |
| | | | | |
| 1280x1024 | 43.5 int | 16 | 1MB | Yes |
| Frame buffers sizes in parenthesis indicate the smallest frame buffer configuration allowed with this type memory chips. | | | | |
| * 32K colors (15-bits per pixel) requires Hi-Color DACs and 16M colors (24-bits per pixel) requires true color DACs. | | | | |

Figure 152. Power 9100 enhanced modes and memory requirements

## 11.3. Signal Description, DRAM32 System

| Signal | Type | Description |
|---|---|---|
| BLANK– | Output | Blanking period in progress |
| CAS01– | Output | Column-address strobe for planes 0 and 1 |
| CAS23– | Output | Column-address strobe for planes 2 and 3 |
| DACRD– | Output | DAC read |
| DACWR– | Output | DAC write |
| HSYNC | Output | Horizontal retrace in progress |
| MA[8..0] | Output | Multiplexed frame buffer address bus |
| MD[31..0] | I/O | Frame buffer data bus |
| MEMCLK | Input | Memory clock |
| MWR[3..0]– | Output | Memory strobes for planes 0 through 3 |
| PIXCLK | Input | Video pixel clock input |
| RAS01– | Output | Row-address strobe for lower bank planes 0 and 1 |
| RAS23– | Output | Row-address strobe for lower bank planes 2 and 3 |
| ROMEN– | Output | ROM enable |
| SENSE | Input | SENSE signal from the RAMDAC |
| VIDOUT[7..0] | Output | Video data output |
| VIDOUTCLK | Output | Video pixel clock output |
| VSYNC | Output | Vertical retrace in progress |

Figure 153. Signal description, DRAM32 configuration

# Chapter 12. SVGA Registers

All documentation in the chapter applies only when the Power 9100 is functioning in SVGA emulation mode.

## 12.1. WEITEK-Specific Registers

The extra functionality of the Power 9100 is controlled by the Power 9100 extended registers, the Power 9100 extended bits, and a Power 9100 additional I/O port. The extended bit definitions are unused or reserved bits in existing VGA registers. Also, one EGA bit is redefined for Power 9100 use.

### 12.1.1. POWER 9100 EXTENDED REGISTERS

Figure 154 lists the extended Power 9100 registers.

| Name | Port | Index | Lock Protection* | Section |
|---|---|---|---|---|
| *Power 9100 Additional Sequencer Registers* | | | | |
| Power 9100 Control Register 0 | 03C5 | 05 | Write | 12.5.7 |
| Power 9100 Control Register 1 | 03C5 | 06 | Write | 12.5.8 |
| Power 9100 Revision | 03C5 | 07 | — | 12.5.9 |
| Power 9100 ID | 03C5 (read) | 10 | Read | 12.5.10 |
| Power 9100 Miscellaneous | 03C5 | 11 | Read/Write | 12.5.11 |
| Power 9100 Output Control | 03C5 | 12 | Read/Write | 12.5.12 |
| *Power 9100 Additional Controller Registers* | | | | |
| Power 9100 Interlace | 03x5 | 19 | — | 12.6.27 |
| Power 9100 Serial Start Address High | 03x5 | 1A | — | 12.6.28 |
| Power 9100 Serial Start Address Low | 03x5 | 1B | — | 12.6.29 |
| Power 9100 Serial Offset | 03x5 | 1C | — | 12.6.30 |
| Power 9100 Total Characters per Line | 03x5 | 1D | — | 12.6.31 |
| *Power 9100 Additional Attribute Registers* | | | | |
| Power 9100 Overscan Color High | 03C0/03C1 | 15 | — | 12.8.8 |
| *Power 9100 Additional CRT Registers* | | | | |
| Power 9100 Attributes states | 03D5 | 24 | — | 12.6.32 |

* = Lock protection provided by control register lock bit (Power 9100 miscellaneous register, bit 5)
*x* = B hex in monochrome mode and D hex in color mode.

Figure 154. WEITEK-specific extended registers

## 12.1. WEITEK-Specific Registers, continued

*LOCKING AND UNLOCKING REGISTERS*

*Locking and Unlocking.* Except for the Power 9100 revision register (which is always available), the Power 9100 extended registers must be unlocked before they can be accessed. Bit 5 of the Power 9100 miscellaneous register (see section 12.5.11) must be reset to zero to unlock the Power 9100 extended registers using the following procedure:

1. Disable interrupts
2. Write I/O 3C4 with 11 hex
3. Write I/O 3C5
4. Write I/O 3C5
5. Read I/O 3C5 in AL
6. Logical AND AL, DF hex
7. Write to 3C5 with AL
8. Enable interrupts

### 12.1.2. POWER 9100 ADDITIONAL I/O PORT

The bank select register is an additional register that is accessed at the port location shown in figure 155.

| Name | Port (Hex) | Section |
|---|---|---|
| Power 9100 Bank Select | 03CD/03CD | 12.4.10 |

Figure 155. WEITEK-specific register at additional I/O port

### 12.1.3. POWER 9100 EXTENDED BIT DEFINITIONS

The extended bit definitions are unused or reserved bits in existing VGA registers and are always accessible. Figure 156 lists the extended bits, the registers that contain the extended bits, and the register index for those registers.

| VGA Register | Bits | Register Index | Section |
|---|---|---|---|
| Sequencer index | 3,4 | — | 12.5.1 |
| Clocking mode | 6,7 | Sequencer 01 | 12.5.3 |
| CRT underline location | 7 | CRT controller 14 | 12.6.22 |
| CRT cursor start position | 6,7 | CRT controller 0A | 12.6.12 |
| CRT preset row scan | 7 | CRT controller 08 | 12.6.10 |
| Attribute mode control | 1,2 | Attribute 10 | 12.8.3 |
| Attribute color plane enable | 6,7 | Attribute 12 | 12.8.5 |

Figure 156. WEITEK extended bit definitions

## 12.2. Register Groups

### 12.2.1. VGA AND WEITEK REGISTER GROUPS

The VGA and WEITEK register groups are presented in figure 157.

### 12.2.2. REGISTERS ORGANIZED BY GROUP

The VGA and WEITEK registers organized by group are presented in the five-part figure 158.

| Group | Port | Function | Use | Mode | Fields | Section |
|---|---|---|---|---|---|---|
| General | 94 | Enable | Color, monochrome | Read, write | VGA (Motherboard) | 12.4 |
| | 102 | Enable | Color, monochrome | Read, write | VGA (Adaptor) | 12.4 |
| | 3BA | Data | Monochrome only | Read, write | VGA | 12.4 |
| | 3C2 | Data | Color, monochrome | Read, write | VGA | 12.4 |
| | 3C3 | Enable | Color, monochrome | Read, write | VGA (Motherboard) | 12.4 |
| | 3C7 | Data | Color, monochrome | Read, write | VGA | 12.4 |
| | 3CA | Data | Color only | Read only | VGA | 12.4 |
| | 3CC | Data | Color, monochrome | Read only | VGA | 12.4 |
| | 3CD | Data | Color, monochrome | Read, write | Power 9100 | 12.4 |
| | 3DA | Data | Color only | Read, write | VGA | 12.4 |
| | 46E8 | Enable | Color, monochrome | Read, write | VGA (Adaptor) | 12.4 |
| Sequencer | 3C4 | Index | Color, monochrome | Read, write | VGA, Power 9100 | 12.5 |
| | 3C5 | Data | Color, monochrome | Read, write | VGA, Power 9100 | 12.5 |
| CRT controller | 3B4 | Index | Monochrome only | Read, write | VGA | 12.6 |
| | 3B5 | Data | Monochrome only | Read, write | VGA, Power 9100 | 12.6 |
| | 3D4 | Index | Color only | Read, write | VGA | 12.6 |
| | 3D5 | Data | Color only | Read, write | VGA, Power 9100 | 12.6 |
| Graphics controller | 3CE | Index | Color, monochrome | Read, write | VGA, Power 9100 | 12.7 |
| | 3CF | Data | Color, monochrome | Read, write | VGA, Power 9100 | 12.7 |
| Attribute controller | 3C0 | Index and data | Color, monochrome | Write only | VGA, Power 9100 | 12.8 |
| | 3C1 | Index and data | Color, monochrome | Read only | VGA, Power 9100 | 12.8 |

Figure 157. VGA/WEITEK register groups

## 12.2. Register Groups, continued

| Register Group | Port | Index | Access | Use | Register | Fields | Section |
|---|---|---|---|---|---|---|---|
| General | 94 | — | R/W | Both | VGA enable register (Motherboard) | VGA | 12.4.1 |
| | 102 | — | R/W | Both | VGA enable register (Adaptor) | VGA | 12.4.2 |
| | 3BA | — | Read | Mono | Input status 1 register | VGA | 12.4.5 |
| | | — | Write | Mono | Feature control register | VGA | 12.4.6 |
| | 3C2 | — | Read | Both | Input status 0 register | VGA | 12.4.4 |
| | | — | Write | Both | Miscellaneous output register | VGA | 12.4.3 |
| | 3C3 | — | R/W | Both | VGA enable register (Motherboard) | VGA | 12.4.7 |
| | 3C7 | — | R/W | Both | DAC status register | VGA | 12.4.8 |
| | 3CA | — | Read | Color | Feature control register | VGA | 12.4.6 |
| | 3CC | — | Read | Both | Miscellaneous output register | VGA | 12.4.3 |
| | 3CD | — | R/W | Both | Power 9100 bank select register | Power 9100 | 12.4.10 |
| | 3DA | — | Read | Color | Input status 1 register | VGA | 12.4.5 |
| | | — | Write | Color | Feature control register | VGA | 12.4.6 |
| | 46E8 | — | R/W | Both | VGA enable register (AT) | VGA | 12.4.9 |
| Sequencer | 3C4 | — | R/W | Both | Sequencer index register | Both | 12.5.1 |
| | 3C5 | 00 | R/W | Both | Reset register | VGA | 12.5.2 |
| | | 01 | R/W | Both | Clocking mode register | Both | 12.5.3 |
| | | 02 | R/W | Both | Map mask register | VGA | 12.5.4 |
| | | 03 | R/W | Both | Character map select register | VGA | 12.5.5 |
| | | 04 | R/W | Both | Memory mode register | VGA | 12.5.6 |
| | | 05 | R/W | Both | Power 9100 control register 0 | Power 9100 | 12.5.7 |
| | | 06 | R/W | Both | Power 9100 control register 1 | Power 9100 | 12.5.8 |
| | | 07 | R/W | Both | Power 9100 revision register | Power 9100 | 12.5.9 |
| | | 10 | Read | Both | Power 9100 ID | Power 9100 | 12.5.10 |
| | | 11 | R/W | Both | Power 9100 miscellaneous register | Power 9100 | 12.5.11 |
| | | 12 | R/W | Both | Power 9100 output control register | Power 9100 | 12.5.12 |

Figure 158. VGA/WEITEK registers organized by group (part 1 of 6)

## 12.2. Register Groups, continued

| Register Group | Port | Index | Access | Use | Register | Fields | Section |
|---|---|---|---|---|---|---|---|
| CRT controller | 3B4 | — | R/W | Mono | CRT controller index register | VGA | 12.6.1 |
| | 3B5 | 00 | R/W | Mono | Horizontal total register | VGA | 12.6.2 |
| | | 01 | R/W | Mono | Horizontal display enable end register | VGA | 12.6.3 |
| | | 02 | R/W | Mono | Start horizontal blanking register | VGA | 12.6.4 |
| | | 03 | R/W | Mono | End horizontal blanking register | VGA | 12.6.5 |
| | | 04 | R/W | Mono | Start horizontal retrace pulse register | VGA | 12.6.6 |
| | | 05 | R/W | Mono | End horizontal retrace register | VGA | 12.6.7 |
| | | 06 | R/W | Mono | Vertical total register | VGA | 12.6.8 |
| | | 07 | R/W | Mono | Overflow register | VGA | 12.6.9 |
| | | 08 | R/W | Mono | Preset row scan register | Both | 12.6.10 |
| | | 09 | R/W | Mono | Maximum scan line register | VGA | 12.6.11 |
| | | 0A | R/W | Mono | Cursor start register | Both | 12.6.12 |
| | | 0B | R/W | Mono | Cursor end register | VGA | 12.6.13 |
| | | 0C | R/W | Mono | Start address high register | VGA | 12.6.14 |
| | | 0D | R/W | Mono | Start address low register | VGA | 12.6.15 |
| | | 0E | R/W | Mono | Cursor location high register | VGA | 12.6.16 |
| | | 0F | R/W | Mono | Cursor location low register | VGA | 12.6.17 |
| | | 10 | R/W | Mono | Vertical retrace start register | VGA | 12.6.18 |
| | | 11 | R/W | Mono | Vertical retrace end register. | VGA | 12.6.19 |
| | | 12 | R/W | Mono | Vertical display enable end register | VGA | 12.6.20 |
| | | 13 | R/W | Mono | Offset register | VGA | 12.6.21 |
| | | 14 | R/W | Mono | Underline location register | Both | 12.6.22 |
| | | 15 | R/W | Mono | Start vertical blank register | VGA | 12.6.23 |
| | | 16 | R/W | Mono | End vertical blank register | VGA | 12.6.24 |
| | | 17 | R/W | Mono | CRT mode control register | VGA | 12.6.25 |
| | | 18 | R/W | Mono | Line compare register | VGA | 12.6.26 |
| | | 19 | R/W | Mono | Power 9100 interlace register | Power 9100 | 12.6.27 |
| | | 1A | R/W | Mono | Power 9100 serial start address high reg. | Power 9100 | 12.6.28 |
| | | 1B | R/W | Mono | Power 9100 serial start address low reg. | Power 9100 | 12.6.29 |
| | | 1C | R/W | Mono | Power 9100 serial offset register | Power 9100 | 12.6.30 |
| | | 1D | R/W | Mono | Power 9100 total characters per line reg. | Power 9100 | 12.6.31 |

Figure 158. VGA/WEITEK registers organized by group (part 2 of 6)

## 12.2. Register Groups, continued

| Register Group | Port | Index | Access | Use | Register | Fields | Section |
|---|---|---|---|---|---|---|---|
| CRT controller (continued) | 3D4 | — | R/W | Color | CRT controller index register | VGA | 12.6.1 |
| | 3D5 | 00 | R/W | Color | Horizontal total register | VGA | 12.6.2 |
| | | 01 | R/W | Color | Horizontal display enable end register | VGA | 12.6.3 |
| | | 02 | R/W | Color | Start horizontal blanking register | VGA | 12.6.4 |
| | | 03 | R/W | Color | End horizontal blanking register | VGA | 12.6.5 |
| | | 04 | R/W | Color | Start horizontal retrace pulse register | VGA | 12.6.6 |
| | | 05 | R/W | Color | End horizontal retrace register | VGA | 12.6.7 |
| | | 06 | R/W | Color | Vertical total register | VGA | 12.6.8 |
| | | 07 | R/W | Color | Overflow register | VGA | 12.6.9 |
| | | 08 | R/W | Color | Preset row scan register | Both | 12.6.10 |
| | | 09 | R/W | Color | Maximum scan line register | VGA | 12.6.11 |
| | | 0A | R/W | Color | Cursor start register | Both | 12.6.12 |
| | | 0B | R/W | Color | Cursor end register | VGA | 12.6.13 |
| | | 0C | R/W | Color | Start address high register | VGA | 12.6.14 |
| | | 0D | R/W | Color | Start address low register | VGA | 12.6.15 |
| | | 0E | R/W | Color | Cursor location high register | VGA | 12.6.16 |
| | | 0F | R/W | Color | Cursor location low register | VGA | 12.6.17 |
| | | 10 | R/W | Color | Vertical retrace start register | VGA | 12.6.18 |
| | | 11 | R/W | Color | Vertical retrace end register | VGA | 12.6.19 |
| | | 12 | R/W | Color | Vertical display enable end register | VGA | 12.6.20 |
| | | 13 | R/W | Color | Offset register | VGA | 12.6.21 |
| | | 14 | R/W | Color | Underline location register | Both | 12.6.22 |
| | | 15 | R/W | Color | Start vertical blank register | VGA | 12.6.23 |
| | | 16 | R/W | Color | End vertical blank register | VGA | 12.6.24 |
| | | 17 | R/W | Color | CRT mode control register | VGA | 12.6.25 |
| | | 18 | R/W | Color | Line compare register | VGA | 12.6.26 |
| | | 19 | R/W | Color | Power 9100 interlace register | Power 9100 | 12.6.27 |
| | | 1A | R/W | Color | Power 9100 serial start address high reg. | Power 9100 | 12.6.28 |
| | | 1B | R/W | Color | Power 9100 serial start address low reg. | Power 9100 | 12.6.29 |
| | | 1C | R/W | Color | Power 9100 serial offset register | Power 9100 | 12.6.30 |
| | | 1D | R/W | Color | Power 9100 total characters per line reg. | Power 9100 | 12.6.31 |
| | | 24 | R | | Power 9100 attributes state register | Power 9100 | 12.6.32 |

Figure 158. VGA/WEITEK registers organized by group (part 3 of 6)

## 12.2. Register Groups, continued

| Register Group | Port | Index | Access | Use | Register | Fields | Section |
|---|---|---|---|---|---|---|---|
| Graphics controller | 3CE | — | R/W | Both | Graphics index register | Both | 12.7.1 |
| | 3CF | 00 | R/W | Both | Set/reset register | VGA | 12.7.2 |
| | | 01 | R/W | Both | Enable set/reset register | VGA | 12.7.3 |
| | | 02 | R/W | Both | Color compare register | VGA | 12.7.4 |
| | | 03 | R/W | Both | Data rotate register | Both | 12.7.5 |
| | | 04 | R/W | Both | Read map select register | VGA | 12.7.6 |
| | | 05 | R/W | Both | Graphics mode register | Both | 12.7.7 |
| | | 06 | R/W | Both | Miscellaneous register | VGA | 12.7.8 |
| | | 07 | R/W | Both | Color don't care register | VGA | 12.7.9 |
| | | 08 | R/W | Both | Bit mask register | VGA | 12.7.10 |
| | | 09 | | | Reserved | | |
| | | 0A | | | Reserved | | |
| | | 0B | | | Reserved | | |
| | | 0C | | | Reserved | | |
| | | 0D | | | Reserved | | |
| | | 0E | | | Reserved | | |
| | | 0F | | | Reserved | | |

Figure 158. VGA/WEITEK registers organized by group (part 4 of 6)

## 12.2. Register Groups, continued

| Register Group | Port | Index | Access | Use | Register | Fields | Section |
|---|---|---|---|---|---|---|---|
| Attribute controller | 3C0 | — | Write | Both | Attribute index register | VGA | 12.8.1 |
| | | 00 | Write | Both | Palette register 00 | VGA | 12.8.2 |
| | | 01 | Write | Both | Palette register 01 | VGA | 12.8.2 |
| | | 02 | Write | Both | Palette register 02 | VGA | 12.8.2 |
| | | 03 | Write | Both | Palette register 03 | VGA | 12.8.2 |
| | | 04 | Write | Both | Palette register 04 | VGA | 12.8.2 |
| | | 05 | Write | Both | Palette register 05 | VGA | 12.8.2 |
| | | 06 | Write | Both | Palette register 06 | VGA | 12.8.2 |
| | | 07 | Write | Both | Palette register 07 | VGA | 12.8.2 |
| | | 08 | Write | Both | Palette register 08 | VGA | 12.8.2 |
| | | 09 | Write | Both | Palette register 09 | VGA | 12.8.2 |
| | | 0A | Write | Both | Palette register 0A | VGA | 12.8.2 |
| | | 0B | Write | Both | Palette register 0B | VGA | 12.8.2 |
| | | 0C | Write | Both | Palette register 0C | VGA | 12.8.2 |
| | | 0D | Write | Both | Palette register 0D | VGA | 12.8.2 |
| | | 0E | Write | Both | Palette register 0E | VGA | 12.8.2 |
| | | 0F | Write | Both | Palette register 0F | VGA | 12.8.2 |
| | | 10 | Write | Both | Attribute mode control register | Both | 12.8.3 |
| | | 11 | Write | Both | Overscan control register | VGA | 12.8.4 |
| | | 12 | Write | Both | Color plane enable register | Both | 12.8.5 |
| | | 13 | Write | Both | Horizontal pixel panning register | VGA | 12.8.6 |
| | | 14 | Write | Both | Color select register | VGA | 12.8.7 |
| | | 15 | Write | Both | Power 9100 overscan color high register | Power 9100 | 12.8.8 |

Figure 158. VGA/WEITEK registers organized by group (part 5 of 6)

## 12.2. Register Groups, continued

| Register Group | Port | Index | Access | Use | Register | Fields | Section |
|---|---|---|---|---|---|---|---|
| Attribute controller (continued) | 3C1 | — | Read | Both | Attribute index register | VGA | 12.8.1 |
| | | 00 | Read | Both | Palette register 00 | VGA | 12.8.2 |
| | | 01 | Read | Both | Palette register 01 | VGA | 12.8.2 |
| | | 02 | Read | Both | Palette register 02 | VGA | 12.8.2 |
| | | 03 | Read | Both | Palette register 03 | VGA | 12.8.2 |
| | | 04 | Read | Both | Palette register 04 | VGA | 12.8.2 |
| | | 05 | Read | Both | Palette register 05 | VGA | 12.8.2 |
| | | 06 | Read | Both | Palette register 06 | VGA | 12.8.2 |
| | | 07 | Read | Both | Palette register 07 | VGA | 12.8.2 |
| | | 08 | Read | Both | Palette register 08 | VGA | 12.8.2 |
| | | 09 | Read | Both | Palette register 09 | VGA | 12.8.2 |
| | | 0A | Read | Both | Palette register 0A | VGA | 12.8.2 |
| | | 0B | Read | Both | Palette register 0B | VGA | 12.8.2 |
| | | 0C | Read | Both | Palette register 0C | VGA | 12.8.2 |
| | | 0D | Read | Both | Palette register 0D | VGA | 12.8.2 |
| | | 0E | Read | Both | Palette register 0E | VGA | 12.8.2 |
| | | 0F | Read | Both | Palette register 0F | VGA | 12.8.2 |
| | | 10 | Read | Both | Attribute mode control register | Both | 12.8.3 |
| | | 11 | Read | Both | Overscan control register | VGA | 12.8.4 |
| | | 12 | Read | Both | Color plane enable register | Both | 12.8.5 |
| | | 13 | Read | Both | Horizontal pixel panning register | VGA | 12.8.6 |
| | | 14 | Read | Both | Color select register | VGA | 12.8.7 |
| | | 15 | Read | Both | Power 9100 overscan color high register | Power 9100 | 12.8.8 |

Figure 158. VGA/WEITEK registers organized by group (part 6 of 6)

## 12.3. Memory Map

| Port | Index | Group | Access | Use | Register | Fields | Section |
|---|---|---|---|---|---|---|---|
| 94 | — | General | R/W | Both | VGA enable register (Motherboard) | VGA | 12.4.1 |
| 102 | — | General | R/W | Both | VGA enable register (Adaptor) | VGA | 12.4.2 |
| 3B4 | — | CRT controller | R/W | Mono | CRT controller index register | VGA | 12.6.1 |
| 3B5 | 00 | CRT controller | R/W | Mono | Horizontal total register | VGA | 12.6.2 |
| | 01 | CRT controller | R/W | Mono | Horizontal display enable end register | VGA | 12.6.3 |
| | 02 | CRT controller | R/W | Mono | Start horizontal blanking register | VGA | 12.6.4 |
| | 03 | CRT controller | R/W | Mono | End horizontal blanking register | VGA | 12.6.5 |
| | 04 | CRT controller | R/W | Mono | Start horizontal retrace pulse register | VGA | 12.6.6 |
| | 05 | CRT controller | R/W | Mono | End horizontal retrace register | VGA | 12.6.7 |
| | 06 | CRT controller | R/W | Mono | Vertical total register | VGA | 12.6.8 |
| | 07 | CRT controller | R/W | Mono | Overflow register | VGA | 12.6.9 |
| | 08 | CRT controller | R/W | Mono | Preset row scan register | Both | 12.6.10 |
| | 09 | CRT controller | R/W | Mono | Maximum scan line register | VGA | 12.6.11 |
| | 0A | CRT controller | R/W | Mono | Cursor start register | Both | 12.6.12 |
| | 0B | CRT controller | R/W | Mono | Cursor end register | VGA | 12.6.13 |
| | 0C | CRT controller | R/W | Mono | Start address high register | VGA | 12.6.14 |
| | 0D | CRT controller | R/W | Mono | Start address low register | VGA | 12.6.15 |
| | 0E | CRT controller | R/W | Mono | Cursor location high register | VGA | 12.6.16 |
| | 0F | CRT controller | R/W | Mono | Cursor location low register | VGA | 12.6.17 |
| | 10 | CRT controller | R/W | Mono | Vertical retrace start register | VGA | 12.6.18 |
| | 11 | CRT controller | R/W | Mono | Vertical retrace end register | VGA | 12.6.19 |
| | 12 | CRT controller | R/W | Mono | Vertical display enable end register | VGA | 12.6.20 |
| | 13 | CRT controller | R/W | Mono | Offset register | VGA | 12.6.21 |
| | 14 | CRT controller | R/W | Mono | Underline location register | Both | 12.6.22 |
| | 15 | CRT controller | R/W | Mono | Start vertical blank register | VGA | 12.6.23 |
| | 16 | CRT controller | R/W | Mono | End vertical blank register | VGA | 12.6.24 |
| | 17 | CRT controller | R/W | Mono | CRT mode control register | VGA | 12.6.25 |
| | 18 | CRT controller | R/W | Mono | Line compare register | VGA | 12.6.26 |
| | 19 | CRT controller | R/W | Mono | Power 9100 interlace register | Power 9100 | 12.6.27 |
| | 1A | CRT controller | R/W | Mono | Power 9100 serial start address high reg. | Power 9100 | 12.6.28 |
| | 1B | CRT controller | R/W | Mono | Power 9100 serial start address low reg. | Power 9100 | 12.6.29 |
| | 1C | CRT controller | R/W | Mono | Power 9100 serial offset register | Power 9100 | 12.6.30 |
| | 1D | CRT controller | R/W | Mono | Power 9100 total characters per line reg. | Power 9100 | 12.6.31 |

Figure 159. VGA/WEITEK registers organized by I/O address (part 1 of 5)

## 12.3. Memory Map, continued

| Port | Index | Group | Access | Use | Register | Fields | Section |
|---|---|---|---|---|---|---|---|
| 3BA | — | General | Read | Mono | Input status 1 register | VGA | 12.4.5 |
| | — | General | Write | Mono | Feature control register | VGA | 12.4.6 |
| 3C0 | — | Attribute controller | Write | Both | Attribute index register | VGA | 12.8.1 |
| | 00 | Attribute controller | Write | Both | Palette register 00 | VGA | 12.8.2 |
| | 01 | Attribute controller | Write | Both | Palette register 01 | VGA | 12.8.2 |
| | 02 | Attribute controller | Write | Both | Palette register 02 | VGA | 12.8.2 |
| | 03 | Attribute controller | Write | Both | Palette register 03 | VGA | 12.8.2 |
| | 04 | Attribute controller | Write | Both | Palette register 04 | VGA | 12.8.2 |
| | 05 | Attribute controller | Write | Both | Palette register 05 | VGA | 12.8.2 |
| | 06 | Attribute controller | Write | Both | Palette register 06 | VGA | 12.8.2 |
| | 07 | Attribute controller | Write | Both | Palette register 07 | VGA | 12.8.2 |
| | 08 | Attribute controller | Write | Both | Palette register 08 | VGA | 12.8.2 |
| | 09 | Attribute controller | Write | Both | Palette register 09 | VGA | 12.8.2 |
| | 0A | Attribute controller | Write | Both | Palette register 0A | VGA | 12.8.2 |
| | 0B | Attribute controller | Write | Both | Palette register 0B | VGA | 12.8.2 |
| | 0C | Attribute controller | Write | Both | Palette register 0C | VGA | 12.8.2 |
| | 0D | Attribute controller | Write | Both | Palette register 0D | VGA | 12.8.2 |
| | 0E | Attribute controller | Write | Both | Palette register 0E | VGA | 12.8.2 |
| | 0F | Attribute controller | Write | Both | Palette register 0F | VGA | 12.8.2 |
| | 10 | Attribute controller | Write | Both | Attribute mode control register | Both | 12.8.3 |
| | 11 | Attribute controller | Write | Both | Overscan control register | VGA | 12.8.4 |
| | 12 | Attribute controller | Write | Both | Color plane enable register | Both | 12.8.5 |
| | 13 | Attribute controller | Write | Both | Horizontal pixel panning register | VGA | 12.8.6 |
| | 14 | Attribute controller | Write | Both | Color select register | VGA | 12.8.7 |
| | 15 | Attribute controller | Write | Both | Power 9100 overscan color high register | Power 9100 | 12.8.8 |

Figure 159. VGA/WEITEK registers organized by I/O address (part 2 of 5)

## 12.3. Memory Map, continued

| Port | Index | Group | Access | Use | Register | Fields | Section |
|---|---|---|---|---|---|---|---|
| 3C1 | — | Attribute controller | Read | Both | Attribute index register | VGA | 12.8.1 |
| | 00 | Attribute controller | Read | Both | Palette register 00 | VGA | 12.8.2 |
| | 01 | Attribute controller | Read | Both | Palette register 01 | VGA | 12.8.2 |
| | 02 | Attribute controller | Read | Both | Palette register 02 | VGA | 12.8.2 |
| | 03 | Attribute controller | Read | Both | Palette register 03 | VGA | 12.8.2 |
| | 04 | Attribute controller | Read | Both | Palette register 04 | VGA | 12.8.2 |
| | 05 | Attribute controller | Read | Both | Palette register 05 | VGA | 12.8.2 |
| | 06 | Attribute controller | Read | Both | Palette register 06 | VGA | 12.8.2 |
| | 07 | Attribute controller | Read | Both | Palette register 07 | VGA | 12.8.2 |
| | 08 | Attribute controller | Read | Both | Palette register 08 | VGA | 12.8.2 |
| | 09 | Attribute controller | Read | Both | Palette register 09 | VGA | 12.8.2 |
| | 0A | Attribute controller | Read | Both | Palette register 0A | VGA | 12.8.2 |
| | 0B | Attribute controller | Read | Both | Palette register 0B | VGA | 12.8.2 |
| | 0C | Attribute controller | Read | Both | Palette register 0C | VGA | 12.8.2 |
| | 0D | Attribute controller | Read | Both | Palette register 0D | VGA | 12.8.2 |
| | 0E | Attribute controller | Read | Both | Palette register 0E | VGA | 12.8.2 |
| | 0F | Attribute controller | Read | Both | Palette register 0F | VGA | 12.8.2 |
| | 10 | Attribute controller | Read | Both | Attribute mode control register | Both | 12.8.3 |
| | 11 | Attribute controller | Read | Both | Overscan control register | VGA | 12.8.4 |
| | 12 | Attribute controller | Read | Both | Color plane enable register | Both | 12.8.5 |
| | 13 | Attribute controller | Read | Both | Horizontal pixel panning register | VGA | 12.8.6 |
| | 14 | Attribute controller | Read | Both | Color select register | VGA | 12.8.7 |
| | 15 | Attribute controller | Read | Both | Power 9100 overscan color high register | Power 9100 | 12.8.8 |
| 3C2 | — | General | Read | Both | Input status 0 register | VGA | 12.4.4 |
| | — | General | Write | Both | Miscellaneous output register | VGA | 12.4.3 |
| 3C3 | — | General | R/W | Both | VGA enable register (Motherboard) | VGA | 12.4.7 |

Figure 159. VGA/WEITEK registers organized by I/O address (part 3 of 5)

## 12.3. Memory Map, continued

| Port | Index | Group | Access | Use | Register | Fields | Section |
|---|---|---|---|---|---|---|---|
| 3C4 | — | Sequencer | R/W | Both | Sequencer index register | Both | 12.5.1 |
| 3C5 | 00 | Sequencer | R/W | Both | Reset register | VGA | 12.5.2 |
| | 01 | Sequencer | R/W | Both | Clocking mode register | Both | 12.5.3 |
| | 02 | Sequencer | R/W | Both | Map mask register | VGA | 12.5.4 |
| | 03 | Sequencer | R/W | Both | Character map select register | VGA | 12.5.5 |
| | 04 | Sequencer | R/W | Both | Memory mode register | VGA | 12.5.6 |
| | 05 | Sequencer | R/W | Both | Power 9100 control register 0 | Power 9100 | 12.5.7 |
| | 06 | Sequencer | R/W | Both | Power 9100 control register 1 | Power 9100 | 12.5.8 |
| | 07 | Sequencer | R/W | Both | Power 9100 revision register | Power 9100 | 12.5.9 |
| | 10 | Sequencer | Read | Both | Power 9100 ID | Power 9100 | 12.5.10 |
| | 11 | Sequencer | R/W | Both | Power 9100 miscellaneous register | Power 9100 | 12.5.11 |
| | 12 | Sequencer | R/W | Both | Power 9100 output control register | Power 9100 | 12.5.12 |
| 3C7 | — | General | R/W | Both | DAC status register | VGA | 12.4.8 |
| 3CA | — | General | Read | Color | Feature control register | VGA | 12.4.6 |
| 3CC | — | General | Read | Both | Miscellaneous output register | VGA | 12.4.3 |
| 3CD | — | General | R/W | Both | Power 9100 bank select register | Power 9100 | 12.4.10 |
| 3CE | — | Graphics controller | R/W | Both | Graphics index register | Both | 12.7.1 |
| 3CF | 00 | Graphics controller | R/W | Both | Set/reset register | VGA | 12.7.2 |
| | 01 | Graphics controller | R/W | Both | Enable set/reset register | VGA | 12.7.3 |
| | 02 | Graphics controller | R/W | Both | Color compare register | VGA | 12.7.4 |
| | 03 | Graphics controller | R/W | Both | Data rotate register | Both | 12.7.5 |
| | 04 | Graphics controller | R/W | Both | Read map select register | VGA | 12.7.6 |
| | 05 | Graphics controller | R/W | Both | Graphics mode register | Both | 12.7.7 |
| | 06 | Graphics controller | R/W | Both | Miscellaneous register | VGA | 12.7.8 |
| | 07 | Graphics controller | R/W | Both | Color don't care register | VGA | 12.7.9 |
| | 08 | Graphics controller | R/W | Both | Bit mask register | VGA | 12.7.10 |
| | 09 | (Reserved) | | | Reserved | | |
| | 0A | (Reserved) | | | Reserved | | |
| | 0B | (Reserved) | | | Reserved | | |
| | 0C | (Reserved) | | | Reserved | | |
| | 0D | (Reserved) | | | Reserved | | |
| | 0E | (Reserved) | | | Reserved | | |
| | 0F | (Reserved) | | | Reserved | | |

Figure 159. VGA/WEITEK registers organized by I/O address (part 4 of 5)

## 12.3. Memory Map, continued

| Port | Index | Group | Access | Use | Register | Fields | Section |
|---|---|---|---|---|---|---|---|
| 3D4 | — | CRT controller | R/W | Color | CRT controller index register | VGA | 12.6.1 |
| 3D5 | 00 | CRT controller | R/W | Color | Horizontal total register | VGA | 12.6.2 |
| | 01 | CRT controller | R/W | Color | Horizontal display enable end register | VGA | 12.6.3 |
| | 02 | CRT controller | R/W | Color | Start horizontal blanking register | VGA | 12.6.4 |
| | 03 | CRT controller | R/W | Color | End horizontal blanking register | VGA | 12.6.5 |
| | 04 | CRT controller | R/W | Color | Start horizontal retrace pulse register | VGA | 12.6.6 |
| | 05 | CRT controller | R/W | Color | End horizontal retrace register | VGA | 12.6.7 |
| | 06 | CRT controller | R/W | Color | Vertical total register | VGA | 12.6.8 |
| | 07 | CRT controller | R/W | Color | Overflow register | VGA | 12.6.9 |
| | 08 | CRT controller | R/W | Color | Preset row scan register | Both | 12.6.10 |
| | 09 | CRT controller | R/W | Color | Maximum scan line register | VGA | 12.6.11 |
| | 0A | CRT controller | R/W | Color | Cursor start register | Both | 12.6.12 |
| | 0B | CRT controller | R/W | Color | Cursor end register | VGA | 12.6.13 |
| | 0C | CRT controller | R/W | Color | Start address high register | VGA | 12.6.14 |
| | 0D | CRT controller | R/W | Color | Start address low register | VGA | 12.6.15 |
| | 0E | CRT controller | R/W | Color | Cursor location high register | VGA | 12.6.16 |
| | 0F | CRT controller | R/W | Color | Cursor location low register | VGA | 12.6.17 |
| | 10 | CRT controller | R/W | Color | Vertical retrace start register | VGA | 12.6.18 |
| | 11 | CRT controller | R/W | Color | Vertical retrace end register | VGA | 12.6.19 |
| | 12 | CRT controller | R/W | Color | Vertical display enable end register | VGA | 12.6.20 |
| | 13 | CRT controller | R/W | Color | Offset register | VGA | 12.6.21 |
| | 14 | CRT controller | R/W | Color | Underline location register | Both | 12.6.22 |
| | 15 | CRT controller | R/W | Color | Start vertical blank register | VGA | 12.6.23 |
| | 16 | CRT controller | R/W | Color | End vertical blank register | VGA | 12.6.24 |
| | 17 | CRT controller | R/W | Color | CRT mode control register | VGA | 12.6.25 |
| | 18 | CRT controller | R/W | Color | Line compare register | VGA | 12.6.26 |
| | 19 | CRT controller | R/W | Color | Power 9100 interlace register | Power 9100 | 12.6.27 |
| | 1A | CRT controller | R/W | Color | Power 9100 serial start address high reg. | Power 9100 | 12.6.28 |
| | 1B | CRT controller | R/W | Color | Power 9100 serial start address low reg. | Power 9100 | 12.6.29 |
| | 1C | CRT controller | R/W | Color | Power 9100 serial offset register | Power 9100 | 12.6.30 |
| | 1D | CRT controller | R/W | Color | Power 9100 total characters per line reg. | Power 9100 | 12.6.31 |
| | 24 | CRT controller | Read | | Power 9100 attributes states | Power 9100 | 12.6.32 |
| 3DA | — | General | Read | Color | Input status 1 register | VGA | 12.4.5 |
| | — | General | Write | Color | Feature control register | VGA | 12.4.6 |
| 46E8 | — | General | R/W | Both | VGA enable register (Adaptor) | VGA | 12.4.9 |

Figure 159. VGA/WEITEK registers organized by I/O address (part 5 of 5)

## 12.4. General Registers

The general register group controls the timing interface between the VGA hardware and the CPU, system memory, palette DAC, and monitor.

| Register Type | General Register | Port | Section |
|---|---|---|---|
| Standard VGA registers | VGA enable register (Motherboard) | 94 | 12.4.1 |
| | VGA enable register (Adaptor) | 102 | 12.4.2 |
| | Miscellaneous output register | 3CC (read) | 12.4.3 |
| | | 3C2 (write) | 12.4.3 |
| | Input status 0 register | 3C2 (read) | 12.4.4 |
| | Input status 1 register | 3BA (monochrome, read) | 12.4.5 |
| | | 3DA (color, read) | 12.4.5 |
| | Feature control register | 3CA (color, read) | 12.4.6 |
| | | 3BA (monochrome, write) | 12.4.6 |
| | | 3DA (color, write) | 12.4.6 |
| | VGA enable register (Motherboard) | 3C3 | 12.4.7 |
| | DAC status register | 3C7 (read) | 12.4.8 |
| | VGA enable register (Adaptor) | 46E8 | 12.4.9 |

Figure 160. General registers

## 12.4. General Registers, continued

### 12.4.1. VGA ENABLE REGISTER

The *VGA enable register* enables and disables the video I/O and memory address decoding. The register at port 94 is used only in motherboard systems.

*REGISTER FORMAT*

VGA Enable (microChannel) Port 94

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| Reserved | | VSE5 | R | VSE3 | Reserved | | |

| Bit No. | Meaning |
|---|---|
| 7..6 | Reserved |
| 5 | Video Subsystem Enable (VSE5) |
| 4 | Reserved |
| 3 | Video Subsystem Enable (VSE3) |
| 2..0 | Reserved |

Figure 161. VGA enable register format

*FIELD DEFINITION*

| Bit 5 | Bit 3 | Meaning |
|---|---|---|
| 0 | 0 | Disable video I/O and address decoding |
| 0 | 1 | Disable video I/O and address decoding |
| 1 | 0 | Disable video I/O and address decoding |
| 1 | 1 | Enable video I/O and address decoding |

Figure 162. VGA enable register fields

*REGISTER DESCRIPTION*

Port 102 is enabled only when bit 4 of address 46E8 is 1 (for adaptor systems) and when bit 5 of address 94 is 0 (for motherboard systems). Otherwise, port 102 remains locked.

The Power 9100 enable logic for adaptor systems is presented in figure 179 and the enable logic for motherboard systems is presented in figure 180.

For motherboard configurations, the Power 9100 responds to ports 3C3 and 94 only. For adapter configurations, the Power 9100 responds to address 46E8 and disregards any attempt to write to ports 3C3 or 94.

### 12.4.2. VGA ENABLE REGISTER

The *VGA enable register* enables and disables the video I/O and memory address decoding. The register at port 102 is used in both adaptor and motherboard systems.

*REGISTER FORMAT*

VGA Enable Port 102

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | VSE |

| Bit No. | Meaning |
|---|---|
| 7..1 | Reserved |
| 0 | Video Subsystem Enable (VSE) |

Figure 163. VGA enable register format

*FIELD DEFINITION*

| Bit | Value | Meaning |
|---|---|---|
| 0 | 0 | Disable video I/O and address decoding |
| | 1 | Enable video I/O and address decoding |

Figure 164. VGA enable register fields

*REGISTER DESCRIPTION*

Port 102 is enabled only when bit 4 of address 46E8 is 1 (for adaptor systems) and when bit 5 of address 94 is 0 (for motherboard systems). Otherwise, port 102 remains locked.

The Power 9100 enable logic for adaptor systems is presented in figure 179 and the enable logic for motherboard systems is presented in figure 180.

## 12.4. General Registers, continued

### 12.4.3. MISCELLANEOUS OUTPUT REGISTER

The *miscellaneous output register* controls sync pulse polarity, clock frequency, CPU access, and emulation.

*REGISTER FORMAT*

Miscellaneous Output Port 3CC (Read), 3C2 (Write)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| VSP | HSP | PB | R | CS | | ER | IOA |

| Bit No. | Meaning |
|---|---|
| 7 | Vertical Sync Polarity (VSP) |
| 6 | Horizontal Sync Polarity (HSP) |
| 5 | Page Bit for Even/Odd (PB) |
| 4 | Reserved |
| 3..2 | Clock Select (CS) |
| 1 | Enable RAM (ER) |
| 0 | Input/Output Address (IOA) |

Figure 165. Miscellaneous output register format

*REGISTER DESCRIPTION*

All bits of this register are reset to 0 with a hardware reset.

Bits 7 and 6 specify sync polarity to determine the scan rate in multifrequency monitors (see figure 166).

Bits 3 and 2 select the video clock rate to establish the screen resolution and display type (monochrome or color) used. This field also selects one of the four configuration switches on the hardware board to let the software know the type of monitor used via the input status 0 register sense switch (SS) field (see figure 170). The reset register synchronous reset (SR) field of the sequencer registers group (see figure 189) must be 0 to change this field.

Bit 0 determines the address location of the CRT controller index register of the CRT controller registers group (see figure 211).

| Bit 7 | Bit 6 | Vertical Size | Active Lines |
|---|---|---|---|
| 0 | 0 | Reserved | Reserved |
| 0 | 1 | 400 lines | 414 lines |
| 1 | 0 | 350 lines | 362 lines |
| 1 | 1 | 480 lines | 496 lines |

Figure 166. Vertical size and sync polarity

*FIELD DEFINITION*

| Bits | Value | Meaning |
|---|---|---|
| 7 | 0 | Positive vertical retrace sync pulse |
| | 1 | Negative vertical retrace sync pulse |
| 6 | 0 | Positive horizontal retrace pulse |
| | 1 | Negative horizontal retrace pulse |
| 5 | 0 | Low 64K memory page, diagnostic use in Odd/Even modes (0–5) |
| | 1 | High 64K memory page, diagnostic use in Odd/Even modes (0–5) |
| 3..2 | 00 | 25.175 MHz clock (640 horizontal pixels) |
| | 01 | 28.322 MHz clock (720 horizontal pixels) |
| | 10 | External clock from auxiliary video connector (clock between 14.3 and 28.4 MHz) |
| | 11 | Reserved |
| 1 | 0 | Disable video memory access from CPU |
| | 1 | Enable video memory access from CPU |
| 0 | 0 | Monochrome emulation (CRTC addresses set to 3B*x*, input status 1 register address set to 3BA) |
| | 1 | Color emulation (CRTC addresses set to 3D*x*, input status 1 register address set to 3DA) |

Figure 167. Miscellaneous output register fields

## 12.4. General Registers, continued

### 12.4.4. INPUT STATUS 0 REGISTER

The *input status 0 register* monitors the status of the vertical retrace interrupt and senses the setting of the selected configuration switch on the hardware board.

*REGISTER FORMAT*

Input Status 0 Port 3C2 (Read)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| CI | Reserved | | SS | Reserved | | | |

| Bit No. | Meaning |
|---|---|
| 7 | CRT Interrupt (CI) |
| 6..5 | Reserved |
| 4 | Switch Sense (SS) |
| 3..0 | Reserved |

Figure 168. Input status 0 register format

*FIELD DEFINITION*

| Bit | Value | Meaning |
|---|---|---|
| 7 | 0 | Vertical retrace interrupt is pending |
| | 1 | Vertical retrace interrupt is cleared |
| 4 | 0 | Selected sense switch = OFF |
| | 1 | Selected sense switch = ON |

Figure 169. Input status 0 register fields

*REGISTER DESCRIPTION*

Bit 4 reports the status of the sense switch selected by the miscellaneous output register clock select (CS) field (see figure 170). This information is used by the software at power-on self-test to determine the type of display hardware (color or monochrome) being used.

| Clock Select | Bit 4 | Display Type |
|---|---|---|
| 00 | Switch 1 | 320 or 640 columns (color) |
| 01 | Switch 2 | 720 columns (monochrome) |
| 10 | Switch 3 | External clock (color) |
| 11 | Switch 4 | Reserved |

Figure 170. Sense switches

## 12.4. General Registers, continued

### 12.4.5. INPUT STATUS 1 REGISTER

The *input status 1 register* permits the software to examine the color outputs of the attribute controller and to synchronize updates with display retrace periods.

*REGISTER FORMAT*

Input Status 1 Port 3DA/3BA (Read)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| Reserved | | DU | | VR | Reserved | | DE |

| Bit No. | Meaning |
|---|---|
| 7..6 | Reserved |
| 5..4 | Diagnostic use (DU) |
| 3 | Vertical Retrace (VR) |
| 2..1 | Reserved |
| 0 | Display Enable NOT (DE) |

Figure 171. Input status 1 register format

*FIELD DEFINITION*

| Bits | Value | Meaning |
|---|---|---|
| 5..4 | *xx* | Shows color outputs as specified by VSM field of color plane enable register (see figures 173 and 285) |
| 3 | 0 | Video information is being displayed |
| | 1 | Vertical retrace interval |
| 0 | 0 | Display is in display mode |
| | 1 | Horizontal or vertical retrace interval (real-time status of inverted display enable signal) |

Figure 172. Input status 1 register fields

*REGISTER DESCRIPTION*

The video status mux (VSM) field of the color plane enable register in the attribute controller group controls the color output contents of bits 5 and 4 (see figures 173 and 285).

Bits 3 and 0 are used together to synchronize software updates of the display with the horizontal and vertical retrace periods. Bit 3 can be programmed to interrupt the CPU on IRQ2 using the enable vertical interrupt (EVI) and clear vertical interrupt (CVI) fields of the vertical retrace end register in the CRT controllers group (see figure 240).

| VSM | DU | Meaning |
|---|---|---|
| 00 | Bit 5 | Color output P2 |
| | Bit 4 | Color output P0 |
| 01 | Bit 5 | Color output P5 |
| | Bit 4 | Color output P4 |
| 10 | Bit 5 | Color output P3 |
| | Bit 4 | Color output P1 |
| 11 | Bit 5 | Color output P7 |
| | Bit 4 | Color output P6 |

Figure 173. Color output contents of diagnostic bits

| Bit 3 | Bit 0 | Display Status |
|---|---|---|
| 0 | 0 | Display mode |
| 0 | 1 | Horizontal retrace period |
| 1 | 0 | Reserved |
| 1 | 1 | Vertical retrace period |

Figure 174. Retrace periods

## 12.4. General Registers, continued

### 12.4.6. FEATURE CONTROL REGISTER

The *feature control register* selects the vertical sync output signal sent to the monitor.

*REGISTER FORMAT*

Feature Control  Port 3CA (Read), 3DA/3BA (Write)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | VSS | Reserved | | |

| Bit No. | Meaning |
|---|---|
| 7..4 | Reserved |
| 3 | Vertical Sync Select (VSS) |
| 2..0 | Reserved |

Figure 175. Feature control register format

*FIELD DEFINITION*

| Bit | Value | Meaning |
|---|---|---|
| 3 | 0 | Normal vertical sync output |
| | 1 | Not allowed |

Figure 176. Feature control register fields

*REGISTER DESCRIPTION*

Bit 3 must be set to 0 (see figure 176).

### 12.4.7. VGA ENABLE REGISTER

The *VGA enable register* enables and disables the video I/O and memory address decoding. The register at port 3C3 is used only in motherboard systems.

*REGISTER FORMAT*

VGA Enable (motherboard)  Port 3C3

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | VSE |

| Bit No. | Meaning |
|---|---|
| 7..1 | Reserved |
| 0 | Video Subsystem Enable (VSE) |

Figure 177. VGA enable register format

*FIELD DEFINITION*

| Bit | Value | Meaning |
|---|---|---|
| 0 | 0 | Disable video I/O and address decoding |
| | 1 | Enable video I/O and address decoding |

Figure 178. VGA enable register fields

*REGISTER DESCRIPTION*

The Power 9100 enable logic for adaptor systems is presented in figure 179 and the enable logic for motherboard systems is presented in figure 180.

| 46E8 (Hex) bit 4 | 46E8 (Hex) bit 3 | 102 (Hex) bit 0 | Power 9100 |
|---|---|---|---|
| 0 * | 1 * | 1 * | Enable * |
| * Disable = any other combination | | | |

Figure 179. Power 9100 enable logic for adaptor systems

| 3C3 (Hex) bit 0 | 102 (Hex) bit 0 | 94 (Hex) bit 5 | 94 (Hex) bit 3 | Power 9100 |
|---|---|---|---|---|
| 1 * | 1 * | 1 * | (ignored) | Enable * |
| * Disable = any other combination | | | | |

Figure 180. Power 9100 enable logic for motherboard systems

## 12.4. General Registers, continued

### 12.4.8. DAC STATUS REGISTER

The *DAC status register* reflects whether the palette DAC is being read or written.

*REGISTER FORMAT*

DAC Status Port 3C7 (Read)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | STA | |

| Bit No. | Meaning |
|---|---|
| 7..2 | Reserved |
| 1..0 | DAC State Value (STA) |

Figure 181. DAC status register format

*FIELD DEFINITION*

| Bits | Value | Operation |
|---|---|---|
| 1..0 | 00 | Read, PEL address read register accessed last |
| | 01 | Not used |
| | 10 | Not used |
| | 11 | Write, PEL address write register accessed last |

Figure 182. DAC status register fields

*REGISTER DESCRIPTION*

Bits 1 and 0 are located in the palette DAC.

### 12.4.9. VGA ENABLE REGISTER

The *VGA enable register* enables and disables the video I/O and memory address decoding. The register at port 46E8 is used only in adaptor systems.

*REGISTER FORMAT*

VGA Enable (adaptor) Port 46E8

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| Reserved | | | VSE | | Reserved | | |

| Bit No. | Meaning |
|---|---|
| 7..5 | Reserved |
| 4..3 | Video Subsystem Enable (VSE) |
| 2..0 | Reserved |

Figure 183. VGA enable register format

*FIELD DEFINITION*

| Bits | Value | Meaning |
|---|---|---|
| 4..3 | 00 | Disable video I/O and address decoding |
| | 01 | Enable video I/O and address decoding |
| | 10 | Disable video I/O and address decoding |
| | 11 | Disable video I/O and address decoding |

Figure 184. VGA enable register fields

*REGISTER DESCRIPTION*

Port 102 is enabled only when bit 4 of address 46E8 is 1 (for adaptor systems) and when bit 5 of address 94 is 0 (for motherboard systems). Otherwise, port 102 remains locked.

The Power 9100 enable logic for adaptor systems is presented in figure 179 and the enable logic for motherboard systems is presented in figure 180.

## 12.4. General Registers, continued

### 12.4.10. POWER 9100 BANK SELECT REGISTER

The *Power 9100 bank select register* provides the most significant four bits of the display memory address.

*REGISTER FORMAT*

Power 9100 Bank Select  Port 03CD

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| Read Bank/Bank 0 [3..0] | | | | Write Bank/Bank 1 [3..0] | | | |

| Bit No. | Meaning |
|---|---|
| 7..4 | Read Bank/Bank 0 [3..0] (RB0) |
| 3..0 | Write Bank/Bank 1 [3..0] (WB1) |

Figure 185. Power 9100 bank select register format

*FIELD DEFINITION*

Bits [7..4] are the *read bank/bank 0 [3..0]* bits. When the host address maps only the "A" segment to the display memory, if bank switching is enabled, these four bits provide the most significant four bits of the 20-bit display memory address during read operations. When both the "A" and "B" segments are mapped, read or write accesses to the "A" segment will use this register to provide the most significant four bits of the display memory address.

Bits [3..0] are the *write bank/bank 1 [3..0]* bits. When the host address maps only the "A" segment to the display memory, if bank switching is enabled, these four bits provide the most significant four bits of the 20-bit display memory address during write operations. When both the "A" and "B" segments are mapped, read or write accesses to the "B" segment will use this register to provide the most significant four bits of the display memory address.

*REGISTER DESCRIPTION*

Refer to section 12.5.11, the Power 9100 miscellaneous register, for information about the bank switching enable bit. Note that this register is specific to the Power 9100 and is not a standard VGA register.

## 12.5. Sequencer Registers

The sequencer register group controls the update sequence of display functions. These registers are accessed via the sequence index register port at hex address 3C4 and the sequencer data registers port at hex address 3C5.

### 12.5.1. SEQUENCER INDEX REGISTER

The *sequencer index register* provides the address index for the sequencer data registers and enables Power 9100 additional registers in the general registers group.

*REGISTER FORMAT*

Sequencer Index — Port 3C4

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| Reserved | | | IHB | RSVD | IB | | |

| Bit No. | Meaning |
|---|---|
| 7..5 | Reserved |
| 4 | Power 9100 Index High Bit (IHB) |
| 3 | Reserved |
| 2..0 | Standard VGA Index Bits (IB) |

Figure 186. Sequencer index register format

*REGISTER DESCRIPTION*

The index register is a pointer register which is located at address 03C4 hex. The value loaded in this register determines which sequencer register is accessed when I/O operations are performed by the host to address 03C5 hex. This value is referred to as the index. In addition, the sequencer index register contains a bit which enables additional Power 9100 registers.

Bits [7..5] are reserved.

Bit 3 is reserved.

Bits 4 and 2..0 together address the VGA and Power 9100 extended registers in the sequencer registers group (see figure 187). Bit 4 is the *index high* bit and is reserved in the standard VGA. Bits [2..0] are defined as in the standard VGA.

*FIELD DEFINITION*

| Bit 4 | Bits 2..0 | Register | Port | Section |
|---|---|---|---|---|
| 0 | 000 | Reset register | 3C5 | 12.5.2 |
| 0 | 001 | Clocking mode register | 3C5 | 12.5.3 |
| 0 | 010 | Map mask register | 3C5 | 12.5.4 |
| 0 | 011 | Character map select register | 3C5 | 12.5.5 |
| 0 | 100 | Memory mode register | 3C5 | 12.5.6 |
| 0 | 101 | Power 9100 control register 0 | 3C5 | 12.5.7 |
| 0 | 110 | Power 9100 control register 1 | 3C5 | 12.5.8 |
| 0 | 111 | Power 9100 revision register | 3C5 | 12.5.9 |
| 1 | 000 | Power 9100 ID (read) | 3C5 | 12.5.10 |
| 1 | 001 | Power 9100 miscellaneous register | 3C5 | 12.5.11 |
| 1 | 010 | Reserved | | |
| 1 | 011 | Power 9100 output control register | 3C5 | 12.5.12 |
| 1 | 1*xx* | Reserved | | |

Figure 187. Index field definition

## 12.5. Sequencer Registers, continued

### 12.5.2. RESET REGISTER

The *reset register* resets the VGA controller by causing a clear-and-halt condition to occur.

*REGISTER FORMAT*

Reset — Port 3C5, Seq Index 00

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | SR | AR |

| Bit No. | Meaning |
|---|---|
| 7..2 | Reserved |
| 1 | Synchronous Reset (SR) |
| 0 | Asynchronous Reset (AR) |

Figure 188. Reset register format

*FIELD DEFINITION*

| Bit 1 | Bit 0 | Meaning |
|---|---|---|
| 0 | 0 | Generate and hold the system in reset |
| 0 | 1 | |
| 1 | 0 | |
| 1 | 1 | Release the reset |

Figure 189. Reset fields

*REGISTER DESCRIPTION*

Bit 1 must be 0 before changing the dot clock (DC) or 8/9 dot clock (8/9) fields of the clocking mode register in the sequencer registers group (see figure 191), or the clock select (CS) field of the miscellaneous output register in the general registers group (see figure 167).

Using bit 0 to reset the VGA controller can cause a loss of memory contents.

## 12.5. Sequencer Registers, continued

### 12.5.3. CLOCKING MODE REGISTER

The *clocking mode register* configures the sequencer timing circuits.

*REGISTER FORMAT*

Clocking Mode — Port 3C5, Seq Index 01

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| DSP13 | MPMD | SO | S4 | DC | SL | R | 8/9 |

| Bit No. | Meaning |
|---|---|
| 7 | Power 9100 Disable Serial Planes 1 & 3 (DSP13) |
| 6 | Power 9100 Memory Page Mode Disable (MPMD) |
| 5 | Screen Off (SO) |
| 4 | Shift Four (S4) |
| 3 | Dot Clock (DC) |
| 2 | Shift Load (SL) |
| 1 | Reserved |
| 0 | 8/9 Dot Clocks (8/9) |

Figure 190. Clocking mode register format

*FIELD DEFINITION*

| Bit | Value | Meaning |
|---|---|---|
| 7 | 0 | Power 9100 input to planes 1 & 3 enabled |
| | 1 | Power 9100 input to planes 1 & 3 disabled |
| 6 | 0 | Power 9100 page mode access enabled |
| | 1 | Power 9100 page mode access disabled |
| 5 | 0 | Normal screen operation |
| | 1 | Video screen off, maximum memory bandwidth assigned to CPU |
| 4 | 0 | Load serializers every character clock |
| | 1 | Load serializers every fourth clock |
| 3 | 0 | Set dot clock to master clock frequency |
| | 1 | Divide master clock by 2 for dot clock for use with 320 and 360 horizontal pixel modes |
| 2 | 0 | Load serializers every character clock |
| | 1 | Load serializers every other clock |
| 0 | 0 | Character clocks are 8 dots wide |
| | 1 | Character clocks are 9 dots wide |

Figure 191. Clocking mode register fields

*REGISTER DESCRIPTION*

This memory and video control read/write register is accessed through location 03C5 hex when the index field of the sequencer index register is set to 01. In the standard VGA, bits [6..7] are reserved. In the Power 9100, these bits (DSP13 and MPMD) have specific meanings as follows:

1. The DSP13 field should be set in overlap modes when graphics planes are chained together to create two long graphics planes. Bit 7 is the *disable serial planes 1 and 3* bit. When this bit is set to 1, the serial input of planes 1 and 3 is disabled.
2. The MPMD field should be set only for non-page-mode VRAMs and DRAMs. Bit 6 is the *memory page mode disable* bit. When this bit is set to 1, page mode accesses to the frame buffer DRAM/VRAM are disabled. This bit should be set only for non-page-mode VRAM/DRAM.

Bits [5..0] are defined as in the standard VGA.

Bits 4 and 2 together control the loading of the VGA video serializers (see figure 192).

Bit 3 affects all other timings because they are derived from the dot clock.

The horizontal total register in the CRT controller registers group uses bit 0 to specify the character width in pixels for graphics modes (see figure 214).

The reset register synchronous reset (SR) field of the sequencer registers group (see figure 189) must be 0 before changing bits 3 or 0.

| Bit 4 | Bit 2 | Serializer Load | Resolution |
|---|---|---|---|
| 0 | 0 | Every character clock | 720 dots/line |
| 0 | 1 | Every other clock | 360 dots/line |
| 1 | 0 | Every fourth clock | 180 dots/line |
| 1 | 1 | Every fourth clock | 180 dots/line |

Figure 192. Serializer fields

## 12.5. Sequencer Registers, continued

### 12.5.4. MAP MASK REGISTER

The *map mask register* enables CPU writing to the four display memory planes.

*REGISTER FORMAT*

Map Mask — Port 3C5, Seq Index 02

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | EM3 | EM2 | EM1 | EM0 |

| Bit No. | Meaning |
|---|---|
| 7..4 | Reserved |
| 3 | Mask Memory Plane 3 (EM3) |
| 2 | Mask Memory Plane 2 (EM2) |
| 1 | Mask Memory Plane 1 (EM1) |
| 0 | Mask Memory Plane 0 (EM0) |

Figure 193. Map mask register format

*FIELD DEFINITION*

| Bit | Value | CPU Write Operations |
|---|---|---|
| 3 | 0 | Memory plane 3 disabled |
| | 1 | Memory plane 3 enabled |
| 2 | 0 | Memory plane 2 disabled |
| | 1 | Memory plane 2 enabled |
| 1 | 0 | Memory plane 1 disabled |
| | 1 | Memory plane 1 enabled |
| 0 | 0 | Memory plane 0 disabled |
| | 1 | Memory plane 0 enabled |

Figure 194. Map mask register fields

*REGISTER DESCRIPTION*

The chain four (C4) field of the memory mode register (see figure 200) controls display memory bit plane access. In write modes, the display plane is selected normally by bits 3..0. In read modes, the active bit plane is selected normally by the read map select (RMS) field of the read map select register in the graphics controller registers group (see figure 267).

When bits 3..0 are set to a value of all 1's, the CPU can perform a 32-bit write operation in one memory cycle to enhance scrolling operations. In odd/even modes, bits 1 and 0 should have the same map mask value and bits 3 and 2 should have the same map mask value. All maps should be enabled when chain mode 4 is selected.

## 12.5. Sequencer Registers, continued

### 12.5.5. CHARACTER MAP SELECT REGISTER

The *character map select register* permits up to 512 characters to be displayed simultaneously.

*REGISTER FORMAT*

Character Map Select  Port 3C5, Seq Index 03

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| Reserved | | SAH | SBH | SA | | SB | |

| Bit No. | Meaning |
|---|---|
| 7..6 | Reserved |
| 5 | Select Char Generator A, High Order (SAH) |
| 4 | Select Char Generator B, High Order (SBH) |
| 3..2 | Select Character Generator A (SA) |
| 1..0 | Select Character Generator B (SB) |

Figure 195. Character map select register format

*FIELD DEFINITION*

| Bit 5 | Bits 3..2 | Character Table | Address Offset |
|---|---|---|---|
| 0 | 00 | 1 | 0K |
| 0 | 01 | 2 | 16K |
| 0 | 10 | 3 | 32K |
| 0 | 11 | 4 | 48K |
| 1 | 00 | 5 | 8K |
| 1 | 01 | 6 | 24K |
| 1 | 10 | 7 | 40K |
| 1 | 11 | 8 | 56K |

Figure 196. SAH and SA fields

| Bit 4 | Bits 1..0 | Character Table | Address Offset |
|---|---|---|---|
| 0 | 00 | 1 | 0K |
| 0 | 01 | 2 | 16K |
| 0 | 10 | 3 | 32K |
| 0 | 11 | 4 | 48K |
| 1 | 00 | 5 | 8K |
| 1 | 01 | 6 | 24K |
| 1 | 10 | 7 | 40K |
| 1 | 11 | 8 | 56K |

Figure 197. SBH and SB fields

*REGISTER DESCRIPTION*

Bits 5..0 and attribute byte together determine the appearance of the displayed text (see figure 198). When bits 5 and 3..2 and bits 4 and 1..0 superfields have identical contents, the system uses bit 3 of the attribute byte to select the foreground colors. When these two superfields are different, the system uses bit 3 of the attribute byte to select a character generator.

| Condition | Attribute Byte | Selection |
|---|---|---|
| Bits 5 and 3..2 = bits 4 and 1..0 | Bit 3 = 0 | Standard foreground colors |
| | Bit 3 = 1 | Intensified foreground colors |
| Bits 5 and 3..2 ≠ bits 4 and 1..0 | Bit 3 = 0 | Character gen B |
| | Bit 3 = 1 | Character gen A |

Figure 198. Color and character generator selection

## 12.5. Sequencer Registers, continued

### 12.5.6. MEMORY MODE REGISTER

The *memory mode register* controls the way display memory functions.

*REGISTER FORMAT*

Memory Mode — Port 3C5, Seq Index 04

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | C4 | O/E | EM | R |

| Bit No. | Meaning |
|---|---|
| 7..4 | Reserved |
| 3 | Chain Four (C4) |
| 2 | Odd/Even (O/E) |
| 1 | Extended Memory (EM) |
| 0 | Reserved (VGA, Power 9100) |

Figure 199. Memory mode register format

*FIELD DEFINITION*

| Bit | Value | Meaning |
|---|---|---|
| 3 | 0 | Display planes selected by map mask and read map select registers (see figure 194) |
| | 1 | Display planes selected by low-order A1 and A0 address bits (see figure 201) |
| 2 | 0 | Odd (maps 1 and 3) and even (maps 0 and 2) addressing mode enabled |
| | 1 | Sequential addressing mode enabled |
| 1 | 0 | No extended memory present (display memory less than 64 KB) |
| | 1 | Extended memory present (display memory greater than 64 KB) |

Figure 200. Memory mode register fields

| A1 | A0 | Map Selected |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 2 |
| 1 | 1 | 3 |

Figure 201. Memory mode when bit 3 = 1

*REGISTER DESCRIPTION*

Bit 3 controls display memory bit plane access. In write modes, the display plane is selected normally by the EM3..EM0 fields of the map mask register (see figure 194). In read modes, the active bit plane is selected normally by the read map select (RMS) field of the read map select register in the graphics controller registers group (see figure 267).

Bit 2 of the memory mode register must be the complement of the O/E field of the graphics mode register in the graphics controller registers group (see figure 269).

Bit 0 of the memory mode register is reserved, as in the standard VGA.

## 12.5. Sequencer Registers, continued

### 12.5.7. POWER 9100 CONTROL REGISTER 0

The *Power 9100 control register 0* determines character width and the order of memory displays.

The Power 9100 control register 0 must be unlocked before it can be accessed. See section 12.1.1 for more information.

*REGISTER FORMAT*

Power 9100 Control Register 0 — Port 03C5 — Seq Index 05

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| DVSAE | BO | OCWM | CPUD2 | IL | SVS17 | CL | |

| Bit No. | Meaning |
|---|---|
| 7 | Dual Video Start Addr/Offset Enable (DVSAE) |
| 6 | Bit Order (BO) |
| 5 | Odd Character Width Mode (OCWM) |
| 4 | CPU Divide 2 (CPUD2) |
| 3 | Interlace (IL) |
| 2 | Serial Video Start Address Bit 17 (SVS17) |
| 0–1 | Character Length (CL) |

Figure 202. Power 9100 control register 0 format

*FIELD DEFINITION*

Bits [0..1] are the *character width high* bits. These are used in conjunction with the sequencer clocking mode register (SCMR) bit 0 to define the width of a character in pixels according to figure 203.

Bit 2 is *serial video start address* bit 16. This is the most significant bit of the 17-bit serial video start address register. The serial video start address consists of the serial video start address low register (bits [7..0]), the serial video start high register (bits [15..8]) and bit 2 of this register (bit 16). A seventeen bit start address enables the serial video frame buffer to start anywhere within the first 512 K of video memory.

Bit 3 is the *interlace* bit. When this bit is set, interlace mode is used for all modes of operation.

Bit 4 is the *CPU divide 2* bit. When this bit is set to 1, it enables the CPU divide 2 mode. In this mode, the entire 16–bit CPU memory address is shifted down by one bit so that A[16..1] is mapped to MA[15..0]. A[0] is used to select either maps 0 and 1 or maps 2 and 3. In the Power 9100, MA[16] is provided by host A[17].

Bit 5 is the *odd character width mode* bit. When this bit is clear and the Power 9100 is programmed to generate characters of odd width, then the last pixel of the character is determined by the character code and bit 2 of the attribute mode control register (see figure 282) as in standard VGA. When this bit is set, the last pixel of the character always comes from the font data.

Bit 6 indicates *bit order.* When this bit is reset to 0, bit 7 is displayed first; this is the IBM bit order. When this bit is set to 1, bit 0 is displayed first.

Bit 7 is the *dual video start address/offset enable* bit and is used only in applications where sufficient VRAM to store two frame buffers is installed. One of the two frame buffers is an alphanumeric or graphics frame buffer whose start address and offset are specified by the usual CRT controller registers. The second frame buffer is a graphics frame buffer whose start address and offset is specified by the serial video start address and serial video offset registers. This mode is used for the Power 9100 overlapping text and graphics modes of operation and is used in the 8-bit planar graphics mode.

Also, when bit 7 is set, the standard horizontal display enable end register displays the total number of graphics pixels in a scan line, and the Power 9100 characters per line register specifies the total number of characters per scan line.

When this bit is 0, only the standard set of CRT controller registers is used for either VRAM or DRAM applications.

*REGISTER DESCRIPTION*

This video control read/write register is accessed through location 03C5 hex when the index field of the sequencer index register is 05. After a reset, this register is initialized to a value of 01.

| Bit 1 | Bit 0 | SCMR B0 | Character Width (pixels) |
|---|---|---|---|
| 0 | 0 | 1 | 6 |
| 0 | 0 | 0 | 7 |
| 0 | 1 | 1 | 8 |
| 0 | 1 | 0 | 9 |
| 1 | 0 | 1 | 10 |
| 1 | 0 | 0 | 11 |
| 1 | 1 | 1 | 12 |
| 1 | 1 | 0 | 13 |

Figure 203. Character length

## 12.5.8. POWER 9100 CONTROL REGISTER 1

The *Power 9100 control register 1* provides the display mode and memory plane parameters.

The Power 9100 control register 1 must be unlocked before it can be accessed. See section 12.1.1 for more information.

### *REGISTER FORMAT*

Power 9100 Control Register 1 — Port 03C5 — Seq Index 06

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| RTVLI | Res | APME | HBSE | BSE | OPE | Rsvd | DPI |

| Bit No. | Meaning |
|---|---|
| 7 | Real-Time Video Load Inhibit (RTVL1) |
| 6 | Reserved |
| 5 | Alpha Page Mode Enable (APME) |
| 4 | Half Bit Shift Enable (HBSE) |
| 3 | Power 9100 Attributes (BSE) |
| 2 | Overlapping Planes Enable (OPE) |
| 1 | Reserved |
| 0 | DRAM Port Inhibit (DPI) |

Figure 204. Power 9100 control register 1 format

### *FIELD DEFINITION*

Bit 7 is the *real-time video load inhibit* bit. In applications which use the serial output port of VRAM as a source of graphics data, the VRAM internal shift register is loaded during the horizontal blanking period. Also, in modes where the VRAM internal shift register does not hold an integral number of video lines, it is also necessary to reload the shift register during the active line period. When the real-time video load inhibit bit is set to 1, shift register reloading is inhibited during the active line period. This bit should be set only when there is an integral number of lines in the shift register (that is, 1024 pixels per line). Setting this bit provides a small increase in host-to-VRAM bandwidth.

Bit 6 is reserved. This bit should be set to 0.

Bit 5 is the *alpha page mode enable* bit. When this bit is set to 1, the memory address bits [4..0] and [12..8] are swapped during a video refresh font data fetch. This allows page-mode accesses to be used by assuring that all font data fetches occurring during a scan line access the same page of dynamic RAM. The system software must compensate for the memory address bit swapping when loading the fonts. Note that the memory controller is designed to use page mode whenever possible.

Bit 4 is the *half-bit shift enable* bit. When this bit is set to 1, bit 14 of the character font indicates half-bit shift and causes the character pixel row to be shifted half a pixel to the right.

Bit 3 is the *Power 9100 attributes* bit. When this bit is set to 1, the special Power 9100 attributes are used. The attribute byte is defined according to the figure 205.

Bit 2 is the *overlapping planes enable* bit. When this bit is set to 1, the character video stream derived from the parallel VRAM port is superimposed on the graphics video stream derived from the serial VRAM port. This mode applies only to implementations which use VRAM.

Bit 1 is reserved.

Bit 0 is the *DRAM port inhibit* bit. When this bit is reset to 0, the DRAM port is always enabled. When this bit is set to 1, the Power 9100 automatically disables the DRAM port and uses the VRAM port whenever the mode of operation allows for VRAM operation.

### *REGISTER DESCRIPTION*

This video control read/write register is accessed though location 03C5 hex when the index field of the sequencer index register is set to 06. After a reset, this register is initialized to a value of 01. The BSE field enables the Power 9100 struck through position (STP) field of the cursor start register in the CRT controller registers group (see figure 231).

| Bit | Attribute |
|---|---|
| 0 | Color 0 (Half Bright in monochrome) |
| 1 | Underline |
| 2 | Reverse Video |
| 3 | Blinking |
| 4 | Bold |
| 5 | Struck Through |
| 6 | Color 1 |
| 7 | Color 2 |

Figure 205. Power 9100 Attributes

## 12.5. Sequencer Registers, continued

### 12.5.9. POWER 9100 REVISION REGISTER

The *Power 9100 revision register* provides device identification and revision level for the SVGA portion of the Power 9100.

*REGISTER FORMAT*

Power 9100 Revision  Port 03C5  Seq Index 07

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| Device ID | | | Reserved | | Revision Level | | |

| Bit No. | Meaning |
|---|---|
| 7..5 | Device ID (DID) |
| 4..3 | Reserved |
| 2..0 | Revision Level (RL) |

Figure 206. Power 9100 BIOS ROM status register format

*FIELD DEFINITION*

Bits [7..5] are the *device id* bits. These bits identify the device. This field is 010 for the Power 9100

Bits [4..3] are reserved for WEITEK use. This field may contain any combination of bits, as determined by WEITEK.

Bits [2..0] are the *revision level* bits. These bits identify the revision level of the device. This field is reserved for WEITEK use, and may contain any combination of bits, as determined by WEITEK.

*REGISTER DESCRIPTION*

This host I/O read-only register is accessed through location 03C5 hex when the index field of the sequencer index register is 07 hex. Note that this register is specific to the Power 9100 and is not a standard VGA register.

### 12.5.10. POWER 9100 ID REGISTER

The *Power 9100 ID register* is a read-only register.

Power 9100 ID Register  Port 03C5 (Read)  Seq Index 10

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 1 | Reserved | | | |

| Bit No. | Meaning |
|---|---|
| 7..4 | ID Bits |
| 6..0 | Reserved |

Figure 207. ID register

This read-only register is accessed through location 03C5 hex when the index field of the sequencer address register is 10 hex. Note that this register is specific to the Power 9100 and is not a standard VGA register.

Bits [7..4] are always 0101.

Bits [3..0] are reserved.

*REGISTER DESCRIPTION*

This host I/O write-only register is accessed through location 03C5 hex when the index field of the sequencer index register is 10 hex. This register is only accessible at this location on the Power 9100. Note that this register is specific to the Power 9100 and is not a standard VGA register.

## 12.5. Sequencer Registers, continued

### 12.5.11. POWER 9100 MISCELLANEOUS REGISTER

The *Power 9100 miscellaneous register* controls the host I/O read/write interface.

The Power 9100 miscellaneous register must be unlocked before it can be accessed. See section 12.1.1 for more information.

*REGISTER FORMAT*

Power 9100 Miscellaneous — Port 03C5, Seq Index 11

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| 1 | BSE | CRL | 1 | 1 | BRE | 1 | 0 |

| Bit No. | Meaning |
|---|---|
| 7 | This field must always be set to a 1. |
| 6 | Bank Switching Enable (BSE) |
| 5 | Power 9100 Control Register Lock (CRL) |
| 4 | This field must always be set to a 1. |
| 3 | This field must always be set to a 1. |
| 2 | BIOS ROM Enable (BRE) |
| 1 | Reserved, set to a 1. |
| 0 | Reserved, set to a 0. |

Figure 208. Power 9100 miscellaneous register format

*FIELD DEFINITION*

Bit 6 is the *bank switching enable* bit. When this bit is reset to 0 and the address mapping is "A" segment or "A and B" segments, the four most-significant bits of the 20-bit frame buffer address are provided by the bank select register, as defined in the bank select register description. When this bit is set to 1, address mapping is disabled.

Bit 5 is the Power 9100 *control register lock* bit. When set to 1, the Power 9100 extended registers (except for the Power 9100 revision register), as shown in figure 209, are read and write protected. When this bit is 0, these registers can be accessed. Upon reset, this bit is set to 1. To unlock the Power 9100 miscellaneous register, see section 12.1.1.

Bit 2 is the *BIOS ROM enable* bit. When this bit is reset to 0, the external BIOS ROM is disabled. When this bit is set to 1, the external BIOS ROM is enabled and responds to addresses xC0000 hex to xC7FFF hex. On reset, this bit is equal to 1.

*REGISTER DESCRIPTION*

This host I/O read/write register is accessed through location 03C5 hex when the index field of the sequencer index register is 11 hex. This register is only accessible at this location on the Power 9100. Note that this register is specific to the Power 9100 and is not a standard VGA register. It is set to FF hex by RESET.

| Name | Port (Hex) | Index (Hex) |
|---|---|---|
| Power 9100 Control Register 0 | 03C5 | 05 |
| Power 9100 Control Register 1 | 03C5 | 06 |
| BIOS ROM Status | 03C5 (read) | 10 |
| Power 9100 Miscellaneous | 03C5 | 11 |
| Power 9100 Output Control | 03C5 | 13 |

Figure 209. Registers affected by control register lock bit

## 12.5. Sequencer Registers, continued

### 12.5.12. POWER 9100 OUTPUT CONTROL REGISTER

The *Power 9100 output control register* causes the least significant 8-bit AT bus data to be clocked into an off-chip register for the purpose of controlling other peripherals.

The Power 9100 output control register must be unlocked before it can be accessed. See section 12.1.1 for more information.

*REGISTER FORMAT*

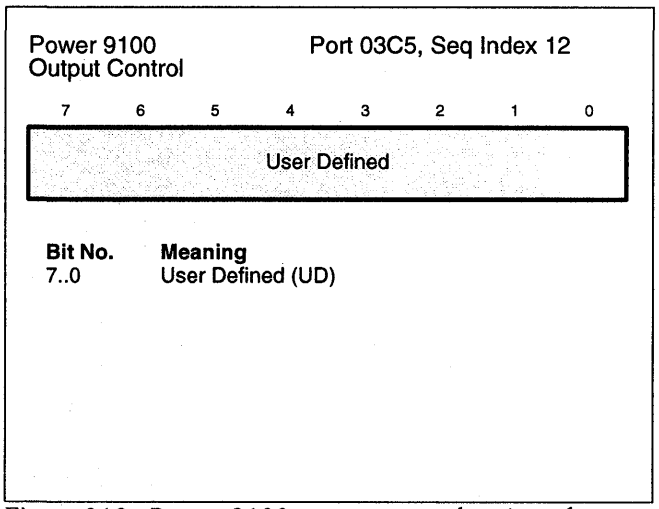



Figure 210. Power 9100 output control register format

*FIELD DEFINITION*

Bits [7..0] are user defined. This register is implemented on-chip for purposes of read back only. Accessing this register causes the Power 9100 output signal MISCWR– to be asserted. This signal should be used to clock the data on the 8 least significant AT bus data lines into an off-chip register for the purpose of controlling of chip peripherals. The Power 9100 asserts the MISCWR– signal when it detects a write to either address 03C5 when the sequencer address register is 12 hex or address 03C2 hex (the standard VGA miscellaneous output register). The WEITEK supplied Power 9100 BIOS accesses this register. When using the Power 9100 BIOS, do not define another use for this register.

*REGISTER DESCRIPTION*

This host I/O read/write register is accessed through location 03C5 hex when the index field of the sequencer index register is 12 hex. Note that this register is specific to the Power 9100 and is not a standard VGA register. This register is only accessible at this location on the Power 9100.

## 12.6. CRT Controller Registers

The CRT controller register group define the raster display. In color systems, these registers are accessed via the CRT controller index register port at hex address 3D4 and the CRT controller data registers port at hex address 3D5. In monochrome systems, these register ports are at hex address 3B4 and hex address 3B5, respectively.

### 12.6.1. CRT CONTROLLER INDEX REGISTER

The *CRT controller index register* provides the address index for the CRT controller registers.

*REGISTER FORMAT*

CRT Controller Index Port 3D4/3B4

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| Reserved | | CRT | | | | | |

| Bit No. | Meaning |
|---|---|
| 7..6 | Reserved |
| 5..0 | CRT Controller Register Index (CRT) |

Figure 211. CRT controller index register format

*REGISTER DESCRIPTION*

The input/output address (IOA) field of the miscellaneous output register of the general registers group (see figure 167) determines whether the CRT controller index register is located at address 3B4 (monochrome emulation) or 3D4 (color emulation).

Note that in the VGA specification, bit 5 is specified as a chip test bit that must be set to zero. In order to access the VGA hidden register at index 24, this bit must be set to one.

*FIELD DEFINITION*

| Field | Register | Section |
|---|---|---|
| 00 | Horizontal total register | 12.6.2 |
| 01 | Horizontal display enable end register | 12.6.3 |
| 02 | Start horizontal blanking register | 12.6.4 |
| 03 | End horizontal blanking register | 12.6.5 |
| 04 | Start horizontal retrace pulse register | 12.6.6 |
| 05 | End horizontal retrace register | 12.6.7 |
| 06 | Vertical total register | 12.6.8 |
| 07 | Overflow register | 12.6.9 |
| 08 | Preset row scan register | 12.6.10 |
| 09 | Maximum scan line register | 12.6.11 |
| 0A | Cursor start register | 12.6.12 |
| 0B | Cursor end register | 12.6.13 |
| 0C | Start address high register | 12.6.14 |
| 0D | Start address low register | 12.6.15 |
| 0E | Cursor location high register | 12.6.16 |
| 0F | Cursor location low register | 12.6.17 |
| 10 | Vertical retrace start register | 12.6.18 |
| 11 | Vertical retrace end register | 12.6.19 |
| 12 | Vertical display enable end register | 12.6.20 |
| 13 | Offset register | 12.6.21 |
| 14 | Underline location register | 12.6.22 |
| 15 | Start vertical blank register | 12.6.23 |
| 16 | End vertical blank register | 12.6.24 |
| 17 | CRT mode control register | 12.6.25 |
| 18 | Line compare register | 12.6.26 |
| 19 | Power 9100 interlace register | 12.6.27 |
| 1A | Power 9100 serial start address high | 12.6.28 |
| 1B | Power 9100 serial start address low register | 12.6.29 |
| 1C | Power 9100 serial offset register | 12.6.30 |
| 1D | Power 9100 total characters per line register | 12.6.31 |
| 24 | Power 9100 attributes state | |

Figure 212. Bits 4..0 field definition

## 12.6. CRT Controller Registers, continued

### 12.6.2. HORIZONTAL TOTAL REGISTER

The *horizontal total register* determines the total horizontal scan time, including active display and retrace.

*REGISTER FORMAT*

Horizontal Total — Port 3D5/3B5, CRT Index 00

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| HT | | | | | | | |

| Bit No. | Meaning |
|---|---|
| 7..0 | Horizontal Total (HT) |

Figure 213. Horizontal total register format

*FIELD DEFINITION*

The value loaded into the HT field is the total horizontal character count per line minus 5.

$HT = horizontal\_character\_total - 5$

*REGISTER DESCRIPTION*

The horizontal total field and the horizontal character counter internal to the CRT controller together determine the total horizontal scan time (see figure 214). All horizontal and vertical timing is based on character clock inputs to this register.

| Mode | Horizontal Character Counter |
|---|---|
| Text | Incremented after each character is output to the screen |
| | Reset after reaching value in horizontal total register |
| Graphics | Incremented every 8 or 9 pixels as specified in 8/9 field of clocking mode register (see figure 191) |
| | Reset after reaching value in horizontal total register |

Figure 214. Determining horizontal scan time

### 12.6.3. HORIZONTAL DISPLAY ENABLE END REGISTER

The *horizontal display enable end register* determines the number of characters on a horizontal line.

*REGISTER FORMAT*

Horizontal Display Enable End — Port 3D5/3B5, CRT Index 01

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| HDE | | | | | | | |

| Bit No. | Meaning |
|---|---|
| 7..0 | Horizontal Display Enable (HDE) |

Figure 215. Horizontal display enable end register format

*FIELD DEFINITION*

The value loaded into the horizontal display enable (HDE) field is the number of displayed characters on a horizontal scan line minus one.

$HDE = number\_of\_displayed\_characters - 1$

*REGISTER DESCRIPTION*

The horizontal display enable field defines the length of the horizontal display enable signal.

## 12.6.4. START HORIZONTAL BLANKING REGISTER

The *start horizontal blanking register* determines the start of the horizontal blanking period.

*REGISTER FORMAT*

Start Horizontal Blanking — Port 3D5/3B5, CRT Index 02

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| SHB | | | | | | | |

| Bit No. | Meaning |
|---|---|
| 7..0 | Start Horizontal Blanking (SHB) |

Figure 216. Start Horizontal Blanking Register

*FIELD DEFINITION*

The value loaded into the start horizontal blanking field is the value of the horizontal character counter at the time the horizontal blanking period should begin.

*REGISTER DESCRIPTION*

The horizontal blanking signal stops the display of data during horizontal CRT refresh. The start horizontal blanking field is used by the end horizontal blanking (EHB) field of the end horizontal blanking register (see figure 218).

## 12.6.5. END HORIZONTAL BLANKING REGISTER

The *end horizontal blanking register* determines the end of the horizontal blanking period.

*REGISTER FORMAT*

End Horizontal Blanking — Port 3D5/3B5, CRT Index 03

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| 1 | DES | | EHB | | | | |

| Bit No. | Meaning |
|---|---|
| 7 | Chip Testing Bit (must be set to 1) |
| 6..5 | Display Enable Skew (DES) |
| 4..0 | End Horizontal Blanking (EHB) |

Figure 217. End Horizontal Blanking Register

*FIELD DEFINITION*

| Bits | Value | Meaning |
|---|---|---|
| 6..5 | 00 | Character clock skew = 0 |
| | 01 | Character clock skew = 1 |
| | 10 | Character clock skew = 2 |
| | 11 | Character clock skew = 3 |
| 4..0 | *xxxxx* | End of horizontal blanking period |

Figure 218. End horizontal blanking register fields

*REGISTER DESCRIPTION*

Bits 6 and 5 contain the six least-significant-bit value of the horizontal character counter when the horizontal blanking interval is to become inactive according to the following formula:

$$EHB = SHB + blanking\_signal\_width$$

The start horizontal blanking (SHB) field is located in the start horizontal blanking register (see figure 216) and the blanking signal width is in character clock units. A sixth end horizontal blanking bit is located in the EHB field of the end horizontal retrace register (see figure 221).

## 12.6. CRT Controller Registers, continued

### 12.6.6. START HORIZONTAL RETRACE PULSE REGISTER

The *start horizontal retrace pulse register* determines the start of the horizontal retrace period.

*REGISTER FORMAT*

Start Horizontal Retrace Pulse — Port 3D5/3B5, CRT Index 04

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| SHR | | | | | | | |

| Bit No. | Meaning |
|---|---|
| 7..0 | Start Horizontal Retrace Pulse (SHR) |

Figure 219. Start horizontal retrace pulse register format

*FIELD DEFINITION*

The value loaded into the start horizontal retrace field is the value of the horizontal character counter at the time the horizontal retrace period should begin.

*REGISTER DESCRIPTION*

The start horizontal retrace field is used to center the screen horizontally and is used by the end horizontal retrace (EHR) field of the end horizontal retrace register (see figure 221).

### 12.6.7. END HORIZONTAL RETRACE REGISTER

The *end horizontal retrace register* determines the end of the horizontal retrace period.

*REGISTER FORMAT*

End Horizontal Retrace register — Port 3D5/3B5, CRT Index 05

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| EHB | HRD | | EHR | | | | |

| Bit No. | Meaning |
|---|---|
| 7 | End Horizontal Blanking (EHB) |
| 6..5 | Horizontal Retrace Delay (HRD) |
| 4..0 | End Horizontal Retrace (EHR) |

Figure 220. End horizontal retrace register format

*FIELD DEFINITION*

| Bits | Value | Meaning |
|---|---|---|
| 7 | *x* | The sixth bit the the EHB field in the end horizontal blanking register (see figure 218) |
| 6..5 | 00 | Horizontal retrace delay = 0 |
| | 01 | Horizontal retrace delay = 1 |
| | 10 | Horizontal retrace delay = 2 |
| | 11 | Horizontal retrace delay = 3 |
| 4..0 | *xxxxx* | End of horizontal retrace period |

Figure 221. End horizontal retrace register fields

*REGISTER DESCRIPTION*

Bits 4..0 contain the five least-significant-bit value of the horizontal character counter when the horizontal retrace signal is to become inactive according to the formula:

$$EHR = SHR + retrace\_signal\_width$$

The start horizontal retrace (SHR) field is located in the start horizontal retrace pulse register (see figure 219) and the retrace signal width is in character clock units. A sixth end horizontal retrace bit is located in the EHB field of the end horizontal retrace register (see figure 221).

## 12.6. CRT Controller Registers, continued

### 12.6.8. VERTICAL TOTAL REGISTER

The *vertical total register* determines the total number of vertical scan lines on the monitor.

*REGISTER FORMAT*

Vertical Total — Port 3D5/3B5, CRT Index 06

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| VT | | | | | | | |

| Bit No. | Meaning |
|---|---|
| 7..0 | Vertical Total (VT) |

Figure 222. Vertical total register format

*FIELD DEFINITION*

The value loaded into the vertical total field is the total horizontal line count per screen minus two.

$$VT = vertical\_retrace + horizontal\_scan\_lines - 2$$

*REGISTER DESCRIPTION*

The ninth and tenth vertical total bits are located in the VT and VT1 fields of the overflow scan register (see figure 224).

### 12.6.9. OVERFLOW REGISTER

The *overflow register* supplies the higher-order bits for several of the CRT controller registers.

*REGISTER FORMAT*

Overflow — Port 3D5/3B5, CRT Index 07

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| VRS | VDE | VT1 | LC | VBS | VRS | VDE | VT |

| Bit No. | Meaning |
|---|---|
| 7 | Vertical Retrace Start Bit 9 (VRS) |
| 6 | Vertical Display Enable End Bit 9 (VDE) |
| 5 | Vertical Total Bit 9 (VT1) |
| 4 | Line Compare Bit 8 (LC) |
| 3 | Start Vertical Blanking Bit 8 (VBS) |
| 2 | Vertical Retrace Start Bit 8 (VRS) |
| 1 | Vertical Display Enable End Bit 8 (VDE) |
| 0 | Vertical Total Bit 8 (VT) |

Figure 223. Overflow register format

*FIELD DEFINITION*

| Bit | Meaning | Section |
|---|---|---|
| 7 | The tenth VRS bit of the vertical retrace start register (the ninth VRS bit is in field VRS, bit 2) | 12.6.18 |
| 6 | The tenth VDE bit of the vertical display end enable register (the ninth bit is in field VDE, bit 1) | 12.6.20 |
| 5 | The tenth VT bit of the vertical total register (the ninth VT bit is in field VT, bit 0) | 12.6.8 |
| 4 | The ninth LC bit of the line compare register | 12.6.26 |
| 3 | The ninth VBS bit of the start vertical blanking register | 12.6.23 |
| 2 | The ninth VRS bit of the vertical retrace start register (the tenth VRS bit is in field VRS, bit 7) | 12.6.18 |
| 1 | The ninth VDE bit of the vertical display end enable register (the tenth VDE bit is in field VDE, bit 6) | 12.6.20 |
| VT | The ninth VT bit of the vertical total register (the tenth VT bit is in field VT1, bit 5) | 12.6.8 |

Figure 224. Overflow register fields

## 12.6. CRT Controller Registers, continued

### 12.6.10. PRESET ROW SCAN REGISTER

The *preset row scan register* controls scrolling and panning operations.

*REGISTER FORMAT*

Preset Row Scan  Port 3D5/3B5, CRT Index 08

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| SVAWC | BP | | PRS | | | | |

| Bit No. | Meaning |
|---|---|
| 7 | Power 9100 Serial Video Address Wrap Control (SVAWC) |
| 6..5 | Byte Panning (BP) |
| 4..0 | Preset Row Scan (PRS) |

Figure 225. Preset row scan register format

*FIELD DEFINITION*

Bit 7 is the *serial video address wrap control* bit. When this bit is reset to 0, the serial video RAM address counter wraps to 0 after a count of 64K. This ensures compatibility with IBM 256K VGA implementation. When this bit is set to 1, the counter wraps after address 128K, enabling access to the maximum 512K memory.

Bits 6 and 5 control the number of bytes to pan, and the horizontal pixel pan (HPP) field in the horizontal pixel panning register of the attribute controller registers group determines the number of pixels to pan (see section 12.8.6). The value of bits 6..5 is normally 0 because there are currently no modes programmed for multiple-shift operation.

The value of bits 4..0 are normally 0 because the entire vertical extent of the top character row is usually displayed.

The preset row scan register and maximum scan line register (see section 12.6.11) together specify the size of the character box. The CE field of the cursor end register (see figure 233) in conjunction with the cursor end (CS) field of the cursor start register (see figure 231) specify the cursor location within this box.

The pixel panning compatibility (PPC) field of the attribute mode control register of the attribute controller registers group (see figure 282) allows line compare (see figure 249) to affect horizontal pixel panning and preset row scan register outputs (see figures 286).

*REGISTER DESCRIPTION*

This video control read/write register is accessed through location 03B5 hex (monochrome mode) or 03D5 hex (color mode) when the index field of the CRT controller index register is 08.

| Bits | Value | Meaning |
|---|---|---|
| 7 | 0 | Serial video RAM address counter wraps to 0 after a count of 64K |
| | 1 | Serial video RAM address counter wraps to 0 after address 128K |
| 6..5 | *xx* | Specifies number of bytes to pan |
| 4..0 | *xxxxx* | The starting row of a character box displayed on the top character row |

Figure 226. Preset row scan register fields

## 12.6. CRT Controller Registers, continued

### 12.6.11. MAXIMUM SCAN LINE REGISTER

The *maximum scan line register* specifies the number of scan lines per character.

*REGISTER FORMAT*

Maximum Scan Line  Port 3D5/3B5, CRT Index 09

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| 2T4 | LC | VBS | MSL | | | | |

| Bit No. | Meaning |
|---|---|
| 7 | 200–to–400–line Conversion (2T4) |
| 6 | Line Compare Bit 9 (LC) |
| 5 | Start Vertical Blanking Bit 9 (VBS) |
| 4..0 | Maximum Line Scan (MSL) |

Figure 227. Maximum scan line register format

*FIELD DEFINITION*

| Bits | Value | Meaning |
|---|---|---|
| 7 | 0 | The row scan counter clock is equal to the horizontal scan rate and line doubling is not enabled |
| | 1 | Allows a 200-line mode to be displayed on 400 display scan lines by dividing the row scan counter clock by 2 to duplicate each line twice |
| 6 | *x* | The tenth LC bit of the line compare register (see section 12.6.26) |
| 5 | *x* | The tenth VBS bit of the start vertical blanking register (see section 12.6.23) |
| 4..0 | *xxxxx* | The number of scan lines in a character minus 1 to specify the number of scan line characters per row |

Figure 228. Maximum scan line register fields

*REGISTER DESCRIPTION*

The preset row scan register (see section 12.6.10) and maximum scan line register together specify the size of the character box.

## 12.6. CRT Controller Registers, continued

### 12.6.12. CURSOR START REGISTER

The *cursor start register* determines the first line of the character box that's part of the cursor.

*REGISTER FORMAT*

Cursor Start — Port 3D5/3B5, CRT Index 0A

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| STP | | COO | CS | | | | |

| Bit No. | Meaning |
|---|---|
| 7..6 | Power 9100 Struck Through Position (STP) |
| 5 | Cursor On/OFF (COO) |
| 4..0 | Row Scan Cursor Start (CS) |

Figure 229. Cursor start register format

*FIELD DEFINITION*

Bits 7..6 designate the *struck through position*. When the Power 9100 attributes are used, these bits specify which character line is struck through according to figure 230.

Bits 5..0 are defined as in the standard VGA.

| Struck through position [1..2] | Line number |
|---|---|
| 0 | 1 |
| 1 | 5 |
| 2 | 9 |
| 3 | 13 |

Figure 230. Struck through position

*REGISTER DESCRIPTION*

This video control read/write register is accessed through location 03B5 hex (monochrome mode) or 03D5 hex (color mode) when the index field of the CRT controller address is 0A hex.

The preset row scan register (see section 12.6.10) and maximum scan line register (see section 12.6.11) together specify the size of the character box.

Bits 4..0 in conjunction with the cursor end (CE) field of the cursor end register (see figure 233) specify the cursor location in the character box. No cursor is generated when the value in bits 4..0 exceeds the value in the CE field.

Bits 7..6 are enabled by the Power 9100 attributes (BSE) field of the Power 9100 control register 1 of the sequencer registers group (see figure 204).

| Bits | Value | Meaning |
|---|---|---|
| 7..6 | 00 | Line 1 struck through (Power 9100 attributes) |
| | 01 | Line 5 struck through (Power 9100 attributes) |
| | 10 | Line 9 struck through (Power 9100 attributes) |
| | 11 | Line 13 struck through (Power 9100 attributes) |
| 5 | 0 | Cursor turned off |
| | 1 | Cursor turned on |
| 4..0 | *xxxxx* | First scan line in character box representing start of cursor |

Figure 231. Cursor start register fields

## 12.6.13. CURSOR END REGISTER

The *cursor end register* determines the last scan line of the character box that's part of the cursor.

*REGISTER FORMAT*

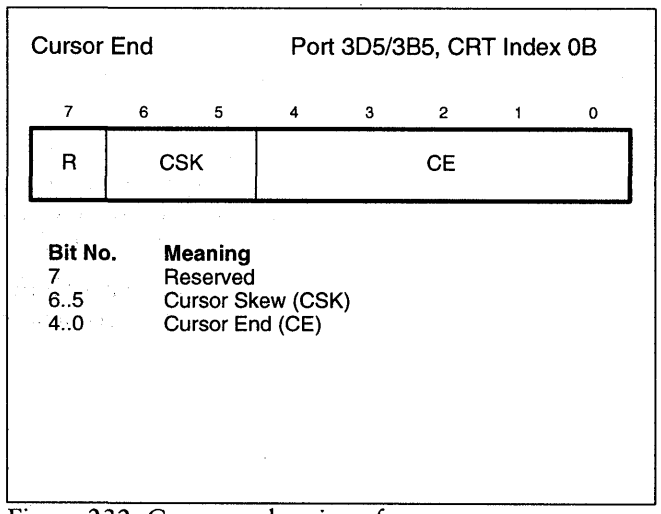



Figure 232. Cursor end register format

*FIELD DEFINITION*

| Bits | Value | Meaning |
|---|---|---|
| 6..5 | 00 | Clock skew = 0 characters |
| | 01 | Clock skew = 1 character |
| | 01 | Clock skew = 2 characters |
| | 11 | Clock skew = 3 characters |
| C4..0 | *xxxxx* | Bottom scan line of character row for cursor display |

Figure 233. Cursor end register fields

*REGISTER DESCRIPTION*

The preset row scan register (see section 12.6.10) and maximum scan line register (see section 12.6.11) together specify the size of the character box.

Bits 6..5 specify the number of characters to delay cursor data for proper synchronization.

Bits 4..0 in conjunction with the cursor end (CS) field of the cursor start register (see figure 231) specify the cursor location in the character box. No cursor is generated when the value in the CS field exceeds the value in bits 4..0.

## 12.6.14. START ADDRESS HIGH REGISTER

The *start address high register*, in conjunction with the start address low register, points to the origin of the display data in the display buffer memory.

*REGISTER FORMAT*

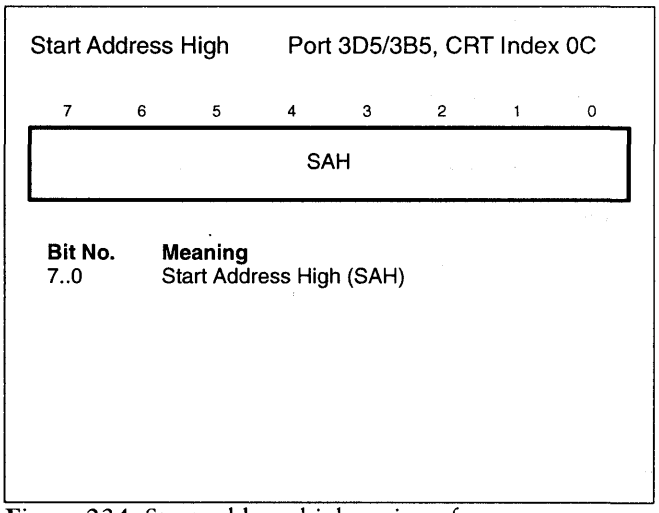



Figure 234. Start address high register format

*FIELD DEFINITION*

The value loaded into the start address high field is the high-order 8 bits of the start address. The low-order 8 bits are located in the start address low (SAL) field of the start address low register (see figure 235).

*REGISTER DESCRIPTION*

The 16-bit value obtained from the start address high and start address low (see figure 235) registers is the first address after the vertical retrace on each screen refresh.

## 12.6. CRT Controller Registers, continued

### 12.6.15. START ADDRESS LOW REGISTER

The *start address low register*, in conjunction with the start address high register, points to the origin of the display data in the display buffer memory.

*REGISTER FORMAT*

Start Address Low  Port 3D5/3B5, CRT Index 0D

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| SAL | | | | | | | |

| Bit No. | Meaning |
|---|---|
| 7..0 | Start Address Low (SAL) |

Figure 235. Start address low register format

*FIELD DEFINITION*

The value loaded into the start address low field is the low-order 8 bits of the start address. The high-order 8 bits are located in the start address high (SAH) field of the start address high register (see figure 234).

*REGISTER DESCRIPTION*

The 16-bit value obtained from the start address high (see figure 234) and start address low registers is the first address after the vertical retrace on each screen refresh.

### 12.6.16. CURSOR LOCATION HIGH REGISTER

The *cursor location high register*, in conjunction with the cursor location low register, points to the cursor position in the display buffer memory.

*REGISTER FORMAT*

Cursor Location High  Port 3D5/3B5, CRT Index 0E

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| CLH | | | | | | | |

| Bit No. | Meaning |
|---|---|
| 7..0 | Cursor Location High (CLH) |

Figure 236. Cursor location high register format

*FIELD DEFINITION*

The value loaded into the cursor location high field is the high-order 8 bits of the cursor location. The low-order 8 bits are located in the cursor location low (CLL) field of the cursor location low register (see figure 237).

## 12.6.17. CURSOR LOCATION LOW REGISTER

The *cursor location low register*, in conjunction with the cursor location high register, points to the cursor position in the display buffer memory.

*REGISTER FORMAT*

Cursor Location Low  Port 3D5/3B5, CRT Index 0F

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| CLL | | | | | | | |

| Bit No. | Meaning |
|---|---|
| 7..0 | Cursor Location Low (CLL) |

Figure 237. Cursor location low register format

*FIELD DEFINITION*

The value loaded into the cursor location low field is the low-order 8 bits of the cursor location. The high-order 8 bits are located in the cursor location high (CLH) field of the cursor location high register (see figure 236).

## 12.6.18. VERTICAL RETRACE START REGISTER

The *vertical retrace start register* determines the start of the vertical retrace period.

*REGISTER FORMAT*

Vertical Retrace Start  Port 3D5/3B5, CRT Index 10

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| VRS | | | | | | | |

| Bit No. | Meaning |
|---|---|
| 7..0 | Vertical Retrace Start (VRS) |

Figure 238. Vertical retrace start register format

*FIELD DEFINITION*

The value loaded into the vertical retrace start field is the value of the scan line counter at the time when the vertical retrace period should begin.

*REGISTER DESCRIPTION*

Access to the vertical retrace start and vertical retrace end (see figure 239) registers is enabled by the end horizontal blanking register compatibility read (CR) field in the CRT controller registers group (see figure 218).

The ninth and tenth vertical retrace start bits are located in the VRS fields of the overflow register (see figure 224).

## 12.6. CRT Controller Registers, continued

### 12.6.19. VERTICAL RETRACE END REGISTER

The *vertical retrace send register* determines the end of the vertical retrace period.

*REGISTER FORMAT*

Vertical Retrace End  Port 3D5/3B5, CRT Index 11

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| PR | BW | EVI | CVI | EVR | | | |

| Bit No. | Meaning |
|---|---|
| 7 | Protect Registers 0–7 (PR) |
| 6 | Select 5 Refresh Cycles (BW) |
| 5 | Enable Vertical Interrupt (EVI) |
| 4 | Clear Vertical Interrupt (CVI) |
| 3..0 | Vertical Retrace End (EVR) |

Figure 239. Vertical retrace end register format

*FIELD DEFINITION*

| Bits | Value | Meaning |
|---|---|---|
| 7 | 0 | Protection disabled (writing enabled) |
| | 1 | Writing from host to CRTC registers disabled |
| 6 | 0 | Select three DRAM cycles |
| | 1 | Select five DRAM cycles for 15.75 MHz sweep-rate displays |
| 5 | 0 | Vertical retrace interrupt on IRQ2 enabled and latched in VR field of input status 0 register (see figure 172) |
| | 1 | Vertical retrace interrupt disabled |
| 4 | 0 | Vertical retrace interrupt cleared by interrupt handler, then set to 1 to avoid holding vertical retrace interrupts inactive |
| | 1 | Enable additional vertical retrace interrupts |
| 3..0 | *xxxx* | End of vertical retrace period |

Figure 240. Vertical retrace end register fields

*REGISTER DESCRIPTION*

Access to the vertical retrace start (see figure 238) and vertical retrace end registers is enabled by the end horizontal blanking register compatibility read (CR) field in the CRT controller registers group (see figure 218).

The vertical retrace (VR) field of the the input status 0 register in the general registers group (see figure 172) can be programmed to interrupt the CPU using bit 5 and bit 4 when the IRQ2 interrupt level is shared.

The value programmed into bits 3..0 is based on the vertical retrace start (VRS) field of the vertical retrace start register (see figure 238) and the width of the vertical retrace signal in horizontal scan units as follows:

$$EVR = VRS + vertical\_retrace\_signal\_width$$

## 12.6.20. VERTICAL DISPLAY ENABLE END REGISTER

The *vertical display enable end register* specifies the last horizontal scan line displayed at the bottom of the screen.

*REGISTER FORMAT*

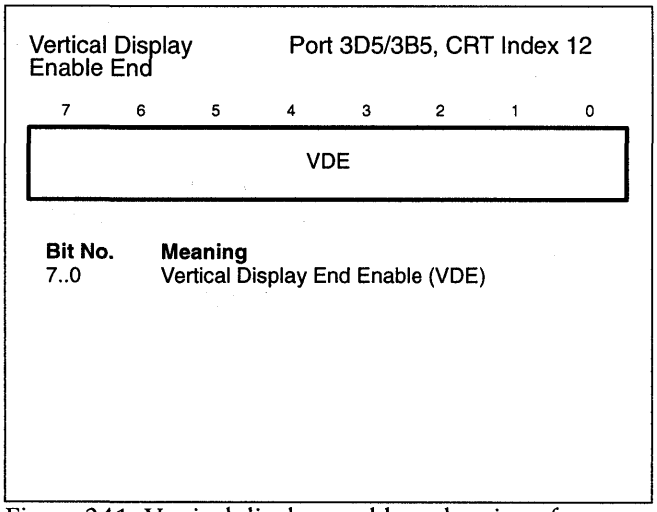



Figure 241. Vertical display enable end register format

*FIELD DEFINITION*

The value loaded into the vertical display end field is the number of the last horizontal scan line at the bottom of the screen.

*REGISTER DESCRIPTION*

Vertical blanking normally occurs before the value in the vertical display enable register is reached.

The ninth and tenth vertical display end bits are located in the VDE fields of the overflow register (see figure 224).

The border area includes the horizontal lines occurring between the value in the start vertical blanking register (see section 12.6.23) and the value in the vertical display enable register. The overscan color register in the attribute controller registers group determines border color (see section 12.8.4).

## 12.6.21. OFFSET REGISTER

The *offset register* specifies the width of the display.

*REGISTER FORMAT*

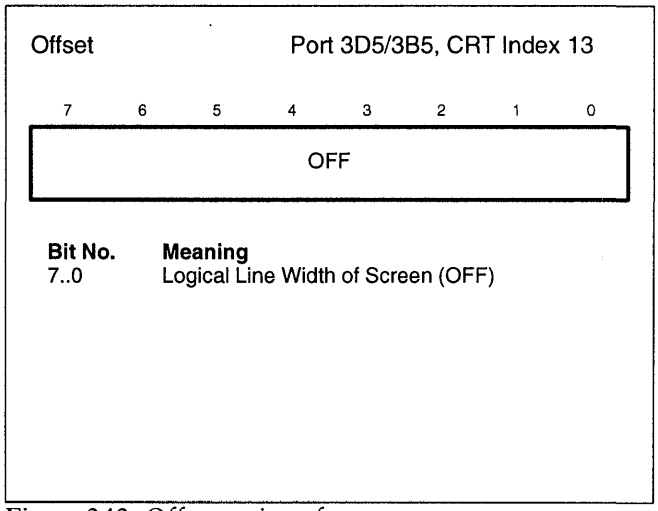



Figure 242. Offset register format

*FIELD DEFINITION*

The value loaded into the logical line width field is the difference in bytes or words between the addresses of vertically adjacent scan lines.

$next\_row\_address = current\_byte\_start\_addr + [(off) * (K$

where:

K = 2, byte addressing

K = 4, word addressing

*REGISTER DESCRIPTION*

The count by two (CBT) field of the CRTC mode control register specifies VGA addressing in either byte mode or word mode (see figure 248).

## 12.6. CRT Controller Registers, continued

### 12.6.22. UNDERLINE LOCATION REGISTER

The *underline location register* determines the horizontal line used for underlining characters.

*REGISTER FORMAT*

Underline Location  Port 3D5/3B5, CRT Index 14

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| SVLD | DW | CB4 | UL | | | | |

| Bit No. | Meaning |
|---|---|
| 7 | Power 9100 Serial Video Line Doubling (SVLD) |
| 6 | Double Word Mode (DW) |
| 5 | Count by Four (CB4) |
| 4..0 | Underline Location (UL) |

Figure 243. Underline location register format

*FIELD DEFINITION*

Bit 7 is the *serial video line doubling* bit. When this bit is set to 1, lines which are accessed through the VRAM serial port are displayed twice (line duplication or double scan).

Bits [6..0] are defined as in the standard VGA.

| Bits | Value | Meaning |
|---|---|---|
| 7 | 0 | Single-scan lines accessed through VRAM serial port |
| | 1 | Double-scan lines accessed through VRAM serial port |
| 6 | 0 | Normal word addressing |
| | 1 | Double word addressing |
| 5 | 0 | Normal clocking |
| | 1 | Divide character clock to memory address counter by 4 |
| 4..0 | *xxxxx* | Underline at character box horizontal scan line minus 1 |

Figure 244. Underline location register fields

*REGISTER DESCRIPTION*

This video control read/write register is accessed through location 03B5 hex (monochrome mode) or 03D5 hex (color mode) when the index field of the CRT controller index register is 14 hex.

The number of scan lines per character is specified by the maximum scan line register (see section 12.6.11).

When bit 6 field is 0, the word/byte (W/B) field of the CRTC mode control register (see figure 248) controls the addressing; and when bit 6 is 1, the addressing is shifted by two bits.

## 12.6.23. START VERTICAL BLANKING REGISTER

The *start vertical blanking register* determines the start of the vertical blanking period.

*REGISTER FORMAT*

Start Vertical Blanking  Port 3D5/3B5, CRT Index 15

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| VBS | | | | | | | |

| Bit No. | Meaning |
|---|---|
| 7..0 | Vertical Blank Start (VBS) |

Figure 245. Start vertical blanking register format

*FIELD DEFINITION*

The value loaded into the vertical blanking start field is the value of the scan line counter at the time when the vertical blanking period should begin.

*REGISTER DESCRIPTION*

A ninth vertical blanking start bit is located in the VBS field of the overflow register (see figure 224), and a tenth vertical blanking start bit is located in the VBS field of the maximum scan line register (see figure 228).

The border area includes the horizontal lines occurring between the value in the start vertical blanking register and the value in the vertical display enable register (see section 12.6.20). The overscan color register in the attribute controller registers group determines border color (see section 12.8.4).

## 12.6.24. END VERTICAL BLANKING REGISTER

The *end vertical blanking register* determines the end of the vertical blanking period.

*REGISTER FORMAT*

End Vertical Blanking  Port 3D5/3B5, CRT Index 16

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| VBE | | | | | | | |

| Bit No. | Meaning |
|---|---|
| 7..0 | End Vertical Blanking (VBE) |

Figure 246. End vertical blanking register format

*FIELD DEFINITION*

The value loaded into the end vertical blanking field is the horizontal scan count value when the vertical output signal becomes inactive, and is determined from the value in the SVB field of the start vertical blanking field (see figure 245) and the width of the vertical blank signal in horizontal scan units as follows:

$$VBE = (SVB - 1) + vertical\_blanking\_signal$$

## 12.6. CRT Controller Registers, continued

### 12.6.25. CRTC MODE CONTROL REGISTER

The *CRTC mode control register* assists in display control.

*REGISTER FORMAT*

CRTC Mode Control  Port 3D5/3B5, CRT Index 17

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| HR | W/B | AW | R | CBT | HRS | SRS | CMS |

| Bit No. | Meaning |
|---|---|
| 7 | Hardware Reset (HR) |
| 6 | Word/Byte Mode (W/B) |
| 5 | Address Wrap (AW) |
| 4 | Reserved |
| 3 | Count by Two (CBT) |
| 2 | Horizontal Retrace Select (HRS) |
| 1 | Select Slow Scan Counter (SRS) |
| 0 | Compatibility Mode Support (CMS) |

Figure 247. CRTC mode control register format

*REGISTER DESCRIPTION*

When the double word mode (WD) field of the underline location register (see figure 244) is 0, bit 6 controls the addressing; and when the WD field of the underline location register is 1, the addressing is shifted by two bits.

Bit 3 creates either a byte or word refresh address for the display buffer in conjunction with the offset register (see section 12.6.21).

*FIELD DEFINITION*

| Bit | Value | Meaning |
|---|---|---|
| 7 | 0 | Horizontal and vertical retrace cleared |
| | 1 | Horizontal and vertical retrace enabled |
| 6 | 0 | Word mode selected (MSB output on LSB address line depends on AW field) |
| | 1 | Byte mode selected |
| 5 | 0 | Address bit 13 sent as LSB to display memory in byte address mode and address bit 0 sent as LSB to display memory in word address mode |
| | 1 | Address bit 15 sent as LSB to display memory |
| 3 | 0 | Memory address counter clocked with character clock input |
| | 1 | Memory address counter clocked every other character clock input |
| 2 | 0 | Scan line counter clocked every horizontal retrace |
| | 1 | Scan line counter clocked every other horizontal retrace |
| 1 | 0 | Row scan counter bit 1 placed on memory address bus bit 14 during active display time |
| | 1 | Sequential output of memory addresses |
| 0 | 0 | Substitute row scan address bit 0 for memory address bit 13 |
| | 1 | No substitution (memory address output bit 13 signal of CRT controller is memory address bit 13) |

Figure 248. CRTC mode control register fields

## 12.6.26. LINE COMPARE REGISTER

The *line compare register* allows a split-screen display.

*REGISTER FORMAT*

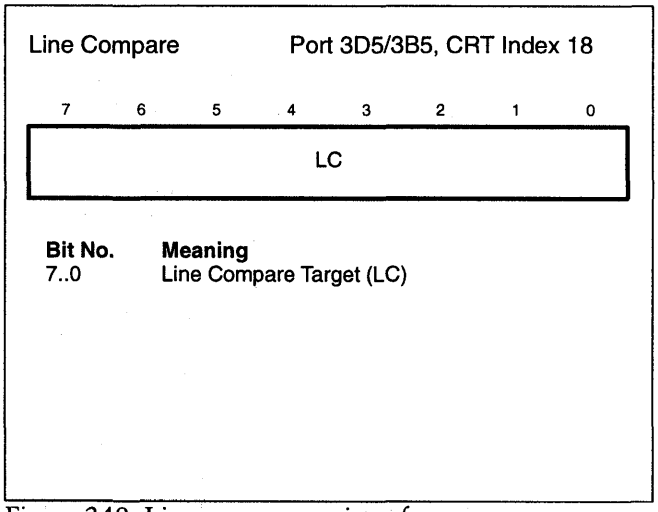



Figure 249. Line compare register format

*FIELD DEFINITION*

The value loaded into the line compare field is the low-order 8 bits of the horizontal scan line counter at the time when the horizontal line counter is to be cleared.

*REGISTER DESCRIPTION*

A ninth line compare bit is located in the LC field of the overflow register (see figure 224), and a tenth line compare bit is located in the LC field of the maximum scan line register (see figure 228).

## 12.6.27. POWER 9100 INTERLACE REGISTER

The *Power 9100 interlace register* determines when the vertical counter is clocked for the second time during interlace modes of operation.

The Power 9100 interlace register must be unlocked before it can be accessed. See section 12.1.1 for more information.

*REGISTER FORMAT*

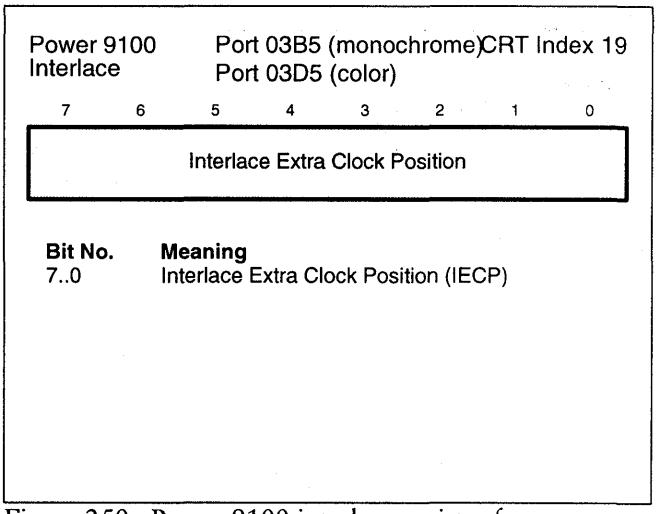



Figure 250. Power 9100 interlace register format

*FIELD DEFINITION*

Bits [7..0] define the *horizontal counter value* at which the vertical counter is clocked for the second time during interlaced modes of operation.

*REGISTER DESCRIPTION*

This video control read/write register is accessed through location 03B5 hex in monochrome mode and location 03D5 hex in color mode when the index field of the CRT controller index register is 19 hex.

## 12.6. CRT Controller Registers, continued

### 12.6.28. POWER 9100 SERIAL START ADDRESS HIGH REGISTER

The *Power 9100 serial start address high register* specifies the high-order bits of the start address of the frame buffer.

The Power 9100 serial start address high register must be unlocked before it can be accessed. See section 12.1.1 for more information.

*REGISTER FORMAT*

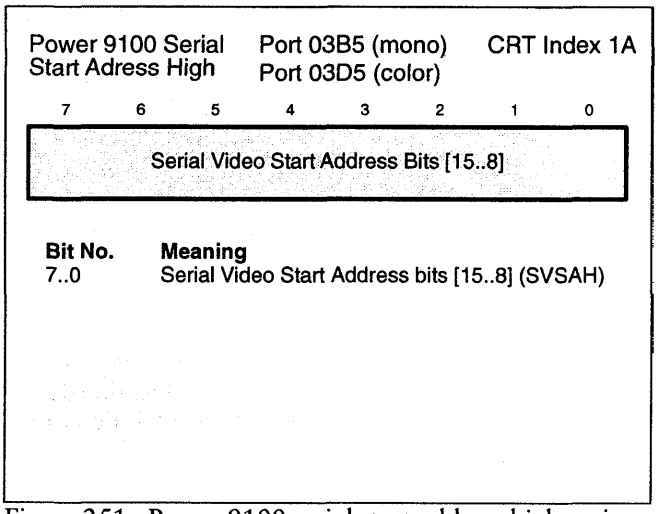



Figure 251. Power 9100 serial start address high register format

*FIELD DEFINITION*

Bits [15..8] are the high-order bits of the 17-bit serial video start address that consists of the serial video start address low register (bits [7..0]), the serial video start high register (bits [15..8]) and bit 2 of the Power 9100 control register 0 (bit 16).The Power 9100 has two sets of start address registers. When bit 7 of the Power 9100 control register 0 is reset to 0, only the set or start address registers that occupies the standard VGA addresses is enabled. When bit 7 of the Power 9100 control register 0 is set to 1, both sets of start address registers are enabled.When both sets of registers are enabled, the serial video start address specifies the location of the first pixel displayed for the frame buffer which is accessed through the VRAM serial port. The remaining set of registers, the standard VGA start address low and high registers, specify the start address of the frame buffer which is accessed through the VRAM parallel port.

*REGISTER DESCRIPTION*

This video control read/write register is accessed through location 03B5 hex in monochrome mode or location 03D5 hex in color mode when the index field of the CRT controller index register is 1A hex.

## 12.6.29. POWER 9100 SERIAL START ADDRESS LOW REGISTER

The *Power 9100 serial start address low register* specifies the high-order bits of the start address of the frame buffer.

The Power 9100 serial start address low register must be unlocked before it can be accessed. See section 12.1.1 for more information.

### *REGISTER FORMAT*

| Power 9100 Serial Start Address Low | Port 03B5 (mono) Port 03D5 (color) | CRT Index 1B |
|---|---|---|

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| Serial Video Start Address Bits [7..0] | | | | | | | |

| Bit No. | Meaning |
|---|---|
| 7..0 | Serial Video Start Address bits [7..0] (SVSAL) |

Figure 252. Power 9100 serial start address low register format

### *FIELD DEFINITION*

Bits [7..0] are the low-order bits of the 17-bit serial video start address that consists of the serial video start address low register (bits [7..0]), the serial video start high register (bits [15..8]) and bit 2 of the Power 9100 control register 0 (bit 16). The Power 9100 has two sets of start address registers. When bit 7 of the Power 9100 control register 0 is reset to 0, only the set that occupies the standard VGA addresses is enabled. When bit 7 of the Power 9100 control register 0 is set to 1, both sets of start address registers are enabled. When both sets of registers are enabled, the serial video start address specifies the location of the first pixel displayed for the frame buffer which is accessed through the VRAM serial port. The remaining set of registers, the standard VGA start address low and high registers, specify the start address of the frame buffer which is accessed through the VRAM parallel port.

### *REGISTER DESCRIPTION*

This video control read/write register is accessed through location 03B5 hex in monochrome mode or location 03D5 hex in color mode when the index field of the CRT controller index register is 1B hex.

## 12.6. CRT Controller Registers, continued

### 12.6.30. POWER 9100 SERIAL OFFSET REGISTER

The *Power 9100 serial offset register* specifies the logical line width of the frame buffer that is accessed through the VRAM serial port.

The Power 9100 serial offset register must be unlocked before it can be accessed. See section 12.1.1 for more information.

*REGISTER FORMAT*

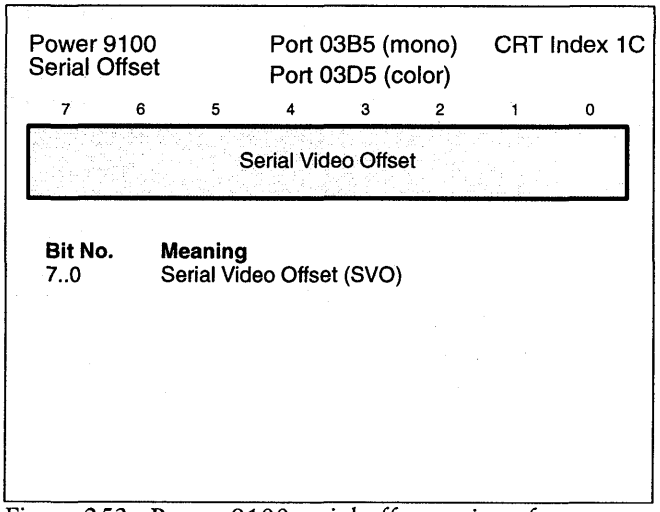



Figure 253. Power 9100 serial offset register format

*FIELD DEFINITION*

Bits [7..0] are the *serial video offset.* This value specifies the logical line width of the frame buffer which is accessed through the VRAM serial port. The starting memory address for the next character row is two or four times this amount. The serial video offset register is programmed with a word address. Depending on the clocking mode of the CRT controller, this is either a word address or a double-word address. The Power 9100 has two offset registers. When bit 7 of the Power 9100 control register 0 is clear, only one is enabled and it occupies the standard VGA addresses. When bit 7 of the Power 9100 control register 0 is set, both offset registers are enabled. In this case, the standard VGA offset register defines the offset for the frame buffer which is accessed through the VRAM parallel port; the other, the serial video offset register, defines the offset for the frame buffer which is accessed through the VRAM serial port.

*REGISTER DESCRIPTION*

This video control read/write register is accessed through location 03B5 hex in monochrome mode and location 03D5 hex in color mode when the index field of the CRT controller index register is 1C hex.

## 12.6.31. POWER 9100 TOTAL CHARACTERS PER LINE REGISTER

The *Power 9100 total characters per line register* specifies the total number of characters per line when the frame buffer is accessed through the parallel VRAM port.

The Power 9100 total characters per line register must be unlocked before it can be accessed. See section 12.1.1 for more information.

### *REGISTER FORMAT*

Power 9100 Total Characters/Line — Port 03B5 (monochrome) Port 03D5 (color) — CRT Index 1D

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| Total Characters Per Line | | | | | | | |

| Bit No. | Meaning |
|---|---|
| 7..0 | Total Characters Per Line (TCPL) |

Figure 254. Power 9100 total characters/line register format

### *FIELD DEFINITION*

Bits [7..0] specify the *total characters per line*. The Power 9100 has two registers which can specify the number of characters in a line. When bit 7 of the Power 9100 control register 0 is clear, only the horizontal display enable end register is enabled and it occupies the standard VGA addresses. When bit 7 of the Power 9100 control register 0 is set, both registers are enabled. In this case, the standard VGA horizontal display end register defines the number of graphics bytes per line for the frame buffer which is accessed through the VRAM serial port; the other, the total characters per line register, defines the total number of characters per line for the frame buffer which is accessed through the VRAM parallel port.

### *REGISTER DESCRIPTION*

This video control read/write register is accessed through location 03B5 hex in monochrome mode or 03D5 hex in color mode when the index field of the CRT controller index register is 1D hex. When DRAM is used in the packed pixel modes, this register must be programmed with a value equal to the total number of bytes in a line divided by 8.

## 12.6. CRT Controller Registers, continued

### 12.6.32. POWER 9100 ATTRIBUTES STATE REGISTER

The *Power 9100 attributes state register* is a read only register which returns the state of the Attributes toggles flip-flop.

Power 9100 Attributes State — Port 03B5 (monochrome)CRT Index 24
Port 03D5 (color)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| ATFF | Reserved | | | | | | |

| Bit No. | Meaning |
|---|---|
| 7 | Returns state of Attributes toggle flip-flop<br>0 = index mode<br>1 = data mode |
| 6..0 | Reserved |

Figure 255. Power 9100 attribute state register

*FIELD DEFINITION*

Bit 7 of this read–only register returns the state of the attributes toggle flip-flop. 0 indicates index mode and 1 indicates data mode.

Bits [6..0] are reserved.

*REGISTER DESCRIPTION*

This video control read register is accessed through location 03B5 hex in monochrome mode and location 03D5 hex in color mode when the index field of the CRT controller index register is 24 hex.

## 12.7. Graphics Controller Registers

The graphics controller register group provide hardware assistance to graphics drawing operations. These registers are accessed via the graphics controller index register port at hex address 3CE and the graphics controller data registers port at hex address 3CF.

### 12.7.1. GRAPHICS INDEX REGISTER

The *graphics index register* provides the address index for the graphics controller registers.

*REGISTER FORMAT*

Graphics Index Port 3CE

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| Res | Reserved | | | GRA | | | |

| Bit No. | Meaning |
|---|---|
| 7 | Reserved |
| 6..4 | Reserved |
| 3..0 | Graphics Controller Register Index (GRA) |

Figure 256. Graphics index register format

*FIELD DEFINITION*

Bit 7 is reserved.

Bits [6..0] are defined as in the standard VGA.

| Field | Register | Section |
|---|---|---|
| 0 | Set/reset register | 12.7.2 |
| 1 | Enable set/reset register | 12.7.3 |
| 2 | Color compare register | 12.7.4 |
| 3 | Data rotate register | 12.7.5 |
| 4 | Read map select register | 12.7.6 |
| 5 | Graphics mode register | 12.7.7 |
| 6 | Miscellaneous register | 12.7.8 |
| 7 | Color don't care register | 12.7.9 |
| 8 | Bit mask register | 12.7.10 |
| 9 | Reserved | |
| A | Reserved | |
| B | Reserved | |
| C | Reserved | |
| D | Reserved | |
| E | Reserved | |
| F | Reserved | |

Figure 257. Bit 3..0 field

*REGISTER DESCRIPTION*

The *graphics index register* is a pointer register which is located at address 03CE hex. The value loaded in this register determines which sequencer register is accessed when I/O operations are performed to address 03CF hex. This value is referred to as the index.

## 12.7. Graphics Controller Registers, continued

### 12.7.2. SET/RESET REGISTER

The *set/reset register* provides color fill data to the display memory maps.

*REGISTER FORMAT*

Set/Reset — Port 3CF, Gra Index 00

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | S/R | | | |

| Bit No. | Meaning |
|---|---|
| 7..4 | Reserved |
| 3..0 | Set/Reset (S/R) |

Figure 258. Set/reset register format

*FIELD DEFINITION*

| Bit | Value | Meaning |
|---|---|---|
| 3 | x | Fill data for memory map 3 |
| 2 | x | Fill data for memory map 2 |
| 1 | x | Fill data for memory map 1 |
| 0 | x | Fill data for memory map 0 |

Figure 259. S/R field

*REGISTER DESCRIPTION*

Bits 3..0 are enabled by the enable set/reset (ESR) field of the enable set/reset register (see figure 261). Memory maps that are disabled by the ESR field receive regular data from the CPU.

The bit mask (BM) field of the bit mask register (see figure 276) write-protects individual memory bits from set/reset fill operations.

### 12.7.3. ENABLE SET/RESET REGISTER

The *enable set/reset register* determines the memory maps that receive fill data from the set/reset register.

*REGISTER FORMAT*

Enable Set/Reset — Port 3CF, Gra Index 01

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | ESR | | | |

| Bit No. | Meaning |
|---|---|
| 7..4 | Reserved |
| 3..0 | Enable Set reset (ESR) |

Figure 260. Enable set/reset register format

*FIELD DEFINITION*

| Bit | Value | Meaning |
|---|---|---|
| 3 | 0 | Disable set/reset for Map 3 |
| | 1 | Enable set/reset for Map 3 |
| 2 | 0 | Disable set/reset for Map 2 |
| | 1 | Enable set/reset for Map 2 |
| 1 | 0 | Disable set/reset for Map 1 |
| | 1 | Enable set/reset for Map 1 |
| 0 | 0 | Disable set/reset for Map 0 |
| | 1 | Enable set/reset for Map 0 |

Figure 261. ESR field

*REGISTER DESCRIPTION*

Bits 3..0 enable the memory maps that receive fill data from the set/reset (S/R) field of the set/reset register (see figure 259). Memory maps that are disabled by bits 3..0 receive regular data from the CPU.

The bit mask (BM) field of the bit mask register (see figure 276) write-protects individual memory bits from set/reset fill operations.

## 12.7. Graphics Controller Registers, continued

### 12.7.4. COLOR COMPARE REGISTER

The *color compare register* permits the comparison of data on all four color maps to a reference color and reports whether a match was found for each pixel position.

*REGISTER FORMAT*

Color Compare — Port 3CF, Gra Index 02

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | CC | | | |

| Bit No. | Meaning |
|---|---|
| 7..4 | Reserved |
| 3..0 | Color Compare (CC) |

Figure 262. Color compare register format

*FIELD DEFINITION*

| Bit | Value | Meaning |
|---|---|---|
| 3 | *x* | Color compare value for Map 3 |
| 2 | *x* | Color compare value for Map 2 |
| 1 | *x* | Color compare value for Map 1 |
| 0 | *x* | Color compare value for Map 0 |

Figure 263. Bits 3..0 field

## 12.7. Graphics Controller Registers, continued

### 12.7.5. DATA ROTATE REGISTER

The *data rotate register* modifies data as it is being transferred from the CPU to the display memory.

*REGISTER FORMAT*

Data Rotate — Port 3CF, Gra Index 03

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| Res | Res | Res | FS | | DR | | |

| Bit No. | Meaning |
|---|---|
| 7 | Reserved |
| 6 | Reserved |
| 5 | Reserved |
| 4..34 | Function Select (FS) |
| 2..0 | Data Rotate (DR) |

Figure 264. Data rotate register format

*REGISTER DESCRIPTION*

This graphics read/write register is accessed through location 03CF hex when the index field of the graphics address register is 03.

The data rotate operation occurs before other operations such as:

set/reset controlled by the set/reset (S/R) field of the set/reset register (see figure 259) and the enable set/reset (ESR) field of the enable set/reset register see (figure 261)

write mode selected by the write mode (WM) field of the mode register (see figure 269)

logical read before write controlled by the FS field

bit mask selected by the bit mask (BM) field of the bit mask register (see figure 276)

*FIELD DEFINITION*

Bits [7..5] are reserved.

Bits [4..0] are defined as in the standard VGA.

| Bits | Value | Meaning |
|---|---|---|
| 7 | | Reserved |
| 6 | | Reserved |
| 5 | | Reserved |
| 4..3 | 00 | Data written unmodified |
| | 01 | Data ANDed with latched data |
| | 10 | Data ORed with latched data |
| | 11 | Data XORed with latched data |
| D2..0 | 000 | Rotate right shift 0 bits |
| | 001 | Rotate right shift 1 bit |
| | 010 | Rotate right shift 2 bits |
| | 011 | Rotate right shift 3 bits |
| | 100 | Rotate right shift 4 bits |
| | 101 | Rotate right shift 5 bits |
| | 110 | Rotate right shift 6 bits |
| | 111 | Rotate right shift 7 bits |

Figure 265. Data rotate register fields

## 12.7. Graphics Controller Registers, continued

### 12.7.6. READ MAP SELECT REGISTER

The *read map select register* enable the memory map to be read by the CPU.

*REGISTER FORMAT*

Read Map Select  Port 3CF, Gra Index 04

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | RMS | |

| Bit No. | Meaning |
|---|---|
| 7..2 | Reserved |
| 1..0 | Read Map Select (RMS) |

Figure 266. Read map select register format

*FIELD DEFINITION*

| Bits | Value | Meaning |
|---|---|---|
| 1..0 | 00 | Select Map 0 |
| | 01 | Select Map 1 |
| | 10 | Select Map 2 |
| | 11 | Select Map 3 |

Figure 267. Read map select register fields

*REGISTER DESCRIPTION*

Bits 1..0 are not active in color compare mode (see figures 263).

The chain four (C4) field of the memory mode register of the sequencer registers group (see figure 200) controls display memory bit plane access. In write modes, the display plane is selected normally by the EM3..EM0 fields of the map mask register of the sequencer registers group (see figure 194). In read modes, the active bit plane is selected normally by bits 1..0.

## 12.7. Graphics Controller Registers, continued

### 12.7.7. GRAPHICS MODE REGISTER

The *graphics mode register* controls read and write modes.

*REGISTER FORMAT*

Graphics Mode — Port 3CF, Gra Index 05

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| 0 | 256CM | SRM | O/E | RM | R | WM | |

| Bit No. | Meaning |
|---|---|
| 7 | Reserved (must be 0) |
| 6 | 256 Color Mode (256CM) |
| 5 | Shift Register Mode (SRM) |
| 4 | Odd/Even (O/E) |
| 3 | Read Mode (RM) |
| 2 | Reserved |
| 1..0 | Write Mode (WM) |

Figure 268. Graphics mode register format

*REGISTER DESCRIPTION*

This graphics read/write register is accessed through location 03CF hex when the index field of the graphics address register is 05.

Bit 4 of the graphics mode register must be the complement of the O/E field of the memory mode register in the sequencer registers group (see figure 200).

Operation of bit 3 depends on read map select (RMS) field of read map select register (see figure 267), chain four (C4) field of sequencer memory mode register (see figure 200), and color compare (CC) field of color compare register (see figure 263).

Operation of bits 1..0 depend on data rotate (DR) field of rotate register (see figure 265), set/reset register (see figure 259), enable set/reset register (see figure 261), and bit mask register (see figure 276).

*FIELD DEFINITION*

Bits [6..0] are defined as in the standard VGA.

| Bits | Value | Meaning |
|---|---|---|
| 6 | 0 | SRM field controls shift register loading |
| | 1 | Shift register load supports 256-color mode |
| 5 | 0 | Normal serial data stream formatting |
| | 1 | Serial data stream formatted with even-numbered bits from both maps on even-numbered maps and odd-numbered bits from both maps on odd-numbered maps |
| 4 | 0 | Normal VGA operating mode |
| | 1 | Even host addresses even display planes 0 and 2, odd host address odd display planes 1 and 3 |
| 3 | 0 | Read mode 0: CPU reads memory map selected by read map select register (see figure 267), but has no effect when C4 field of sequencer memory mode register (see figure 200) = 1 |
| | 1 | Read mode 1: CPU reads results of comparison of the four memory maps with CC field of color compare register (see figure 263) |
| 1..0 | 00 | Direct CPU write (write mode 0) either rotated by DR field of rotate register (see figure 265), or by the contents of the set/reset register (see figure 259) when enabled (see figure 261) |
| | 01 | Use latch content as write data (write mode 1) |
| | 10 | Color plane *n* filled with bit *n* value in processor write data (write mode 2) |
| | 11 | Write each plane with 8 identical bits (write mode 4) from set/reset register (see figure 259), with rotated CPU data ANDed with bit mask register data (see figure 276) |

Figure 269. Graphics mode register fields

## 12.7. Graphics Controller Registers, continued

### 12.7.8. MISCELLANEOUS REGISTER

The *miscellaneous register* controls the display mode, monochrome graphics emulation, and memory mapping.

*REGISTER FORMAT*

Miscellaneous Port 3CF, Gra Index 06

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | MM | | COE | G/A |

| Bit No. | Meaning |
|---|---|
| 7..4 | Reserved |
| 3..2 | Memory Map (MM) |
| 1 | Chain Odd-Even (COE) |
| 0 | Graphics/Alphanumeric Mode (G/A) |

Figure 270. Miscellaneous register format

*FIELD DEFINITION*

| Bits | Value | Meaning |
|---|---|---|
| 3..2 | 00 | Memory location A0000–BFFFF hex, 128K memory length |
| | 01 | Memory location A0000–AFFFF hex, 64K memory length |
| | 10 | Memory location B0000–B7FFF hex, 32K memory length |
| | 11 | Memory location B8000–B7FFF hex, 32K memory length |
| 1 | 0 | Standard addressing |
| | 1 | Higher-order address bit replaces host address bit (even and odd addresses access even and odd planes, respectively) |
| 0 | 0 | Select alphanumeric mode |
| | 1 | Select graphics mode |

Figure 271. Miscellaneous register fields

*REGISTER DESCRIPTION*

Bit 0 should have the same contents as the G/A field of the attribute mode control register in the attribute controller registers group (see figure 282).

| MM bit 1 | MM bit 0 | Display Buffer Starts at | Length (bytes) |
|---|---|---|---|
| 0 | 0 | A0000H | 128K |
| 0 | 1 | A0000H | 64K |
| 1 | 0 | B0000H | 32K |

Figure 272. Location of frame buffer in memory page

## 12.7. Graphics Controller Registers, continued

### 12.7.9. COLOR DON'T CARE REGISTER

The *color don't care register* masks particular memory maps from being tested during color compares.

*REGISTER FORMAT*

Color Don't Care — Port 3CF, Gra Index 07

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | CDC | | | |

| Bit No. | Meaning |
|---|---|
| 7..4 | Reserved |
| 3..0 | Color Don't Care (CDC) |

Figure 273. Color don't care register format

*FIELD DEFINITION*

| Bit | Value | Meaning |
|---|---|---|
| 3 | 0 | Participate in color compare cycle, Map 3 |
| | 1 | Ignore color compare cycle, Map 3 |
| 2 | 0 | Participate in color compare cycle, Map 2 |
| | 1 | Ignore color compare cycle, Map 2 |
| 1 | 0 | Participate in color compare cycle, Map 1 |
| | 1 | Ignore color compare cycle, Map 1 |
| 0 | 0 | Participate in color compare cycle, Map 0 |
| | 1 | Ignore color compare cycle, Map 0 |

Figure 274. Color don't care field

### 12.7.10. BIT MASK REGISTER

The *bit mask register* prevents certain bit positions from being modified during memory read-modify-write cycles.

*REGISTER FORMAT*

Bit Mask — Port 3CF, Gra Index 08

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| BM | | | | | | | |

| Bit No. | Meaning |
|---|---|
| 7..0 | Bit Mask (BM) |

Figure 275. Bit mask register format

*FIELD DEFINITION*

| Bit | Value | Meaning |
|---|---|---|
| *x* | 0 | Bit *x* immune to change |
| | 1 | Bit *x* writes enabled |

Figure 276. Bit mask field

*REGISTER DESCRIPTION*

Bit *x* is immune to change when the bit mask field is 0 only if the location being written is the last location read.

The bit mask write-protects individual memory bits from set/reset fill operations specified by the set/reset (S/R) field of the set/reset register (see figure 259) and enabled by the enable set/reset (ESR) field of the enable set/reset register (see figure 261).

## 12.8. Attribute Controller Registers

The attribute controller register group controls display attributes such as color, blinking, and underlining. These registers are written at hex address 3C0, with access alternating between the attribute controller index register and the selected attribute controller data register. These registers are read at hex address 3C1, with access alternating between the address and data registers.

### 12.8.1. ATTRIBUTE INDEX REGISTER

The *attribute index register* provides the address index for the attribute controller registers.

*REGISTER FORMAT*

Attribute Index — Port 3C1 (Read), 3C0 (Write)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| Reserved | | PAS | ACR | | | | |

| Bit No. | Meaning |
|---|---|
| 7..6 | Reserved |
| 5 | Palette Address Source (PAS) |
| 4..0 | Attribute Address (ACR) |

Figure 277. Attribute index register format

*FIELD DEFINITION*

| Field | Registers | Section |
|---|---|---|
| 00000 | Palette register 00 | 12.8.2 |
| 00001 | Palette register 01 | 12.8.2 |
| 00010 | Palette register 02 | 12.8.2 |
| 00011 | Palette register 03 | 12.8.2 |
| 00100 | Palette register 04 | 12.8.2 |
| 00101 | Palette register 05 | 12.8.2 |
| 00110 | Palette register 06 | 12.8.2 |
| 00111 | Palette register 07 | 12.8.2 |
| 01000 | Palette register 08 | 12.8.2 |
| 01001 | Palette register 09 | 12.8.2 |
| 01010 | Palette register 0A | 12.8.2 |
| 01011 | Palette register 0B | 12.8.2 |
| 01100 | Palette register 0C | 12.8.2 |
| 01101 | Palette register 0D | 12.8.2 |
| 01110 | Palette register 0E | 12.8.2 |
| 01111 | Palette register 0F | 12.8.2 |
| 10000 | Attribute mode control register | 12.8.3 |
| 10001 | Overscan control register | 12.8.4 |
| 10010 | Color plane enable register | 12.8.5 |
| 10011 | Horizontal pixel panning register | 12.8.6 |
| 10100 | Color select register | 12.8.7 |
| 10101 | Power 9100 overscan color high register | 12.8.8 |
| 10110..11111 | Reserved | |

Figure 278. Attribute index field definition

## 12.8. Attribute Controller Registers, continued

### 12.8.2. PALETTE REGISTERS

The 16 *palette registers* allow the CPU to determine which colors are displayed at any time.

*REGISTER FORMAT*

Palette — Port 3C1 (Read), 3C0 (Write) ACR Indices 00–0F

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| Reserved | | SR | SG | SB | R | G | B |

| Bit No. | Meaning |
|---|---|
| 7..6 | Reserved |
| 5 | Secondary Red (SR) |
| 4 | Secondary Green (SG) |
| 3 | Secondary Blue (SB) |
| 2 | Red (R) |
| 1 | Green (G) |
| 0 | Blue (B) |

Figure 279. Palette registers format

*FIELD DEFINITION*

| Bit | Value | Meaning |
|---|---|---|
| 5 | 0 | Secondary red not present |
| | 1 | Secondary red present |
| 4 | 0 | Secondary green not present |
| | 1 | Secondary green present |
| 3 | 0 | Secondary blue not present |
| | 1 | Secondary blue present |
| 2 | 0 | Red not present |
| | 1 | Red present |
| 1 | 0 | Green not present |
| | 1 | Green present |
| 0 | 0 | Blue not present |
| | 1 | Blue present |

Figure 280. Palette register fields

*REGISTER DESCRIPTION*

The palette registers should be modified only during the vertical retrace interval.

## 12.8. Attribute Controller Registers, continued

### 12.8.3. ATTRIBUTE MODE CONTROL REGISTER

The *attribute mode control register* controls VGA attributes.

*REGISTER FORMAT*

Attribute Mode Control — Port 3C1 (Read), 3C0 (Write) ACR Index 10

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| P54S | PELW | PELPC | R | EB | ELG | MEBB | EGM |

| Bit No. | Meaning |
|---|---|
| 7 | P5–P4 Select (Pr4S) |
| 6 | Pixel Width (PELW) |
| 5 | Pixel Panning Compatibility (PELPC) |
| 4 | Reserved |
| 3 | Enable Blink or Intensity (EB) |
| 2 | Enable Line Graphics Char Codes/Clip Disable (ELG) |
| 1 | Mono Mode/Black Background (MEBB) |
| 0 | Enable Graphics Mode (EGM) |

Figure 281. Attribute mode control register format

*FIELD DEFINITION*

Bits [7..5] are defined as in the standard VGA.

Bit 4 is reserved, as defined in the standard VGA.

Bit 3 is defined as in the standard VGA.

Bit 2 is the *enable line graphics character code/clip disable* bit. In normal VGA modes, when this bit is is reset to 0, the ninth dot of nine-dot-wide characters is the background color. When this bit is set to 1, the eighth bit of the font data is duplicated into the ninth bit for character codes C0 through DF. When the Power 9100 character attributes are used, this bit defines whether a character is allowed to stretch over the edges of a character cell, as can be invoked by the bold and half-bit-shift attributes. When this bit is clear, the character is clipped at the edges of the character cell; when this bit is set, the character is not clipped.

Bit 1 is the *mono mode/black background* bit. (Note that this is also a standard VGA bit.) In normal VGA modes, when this bit is set to 1, monochrome mode is enabled. When this bit is reset to 0, a color mode is enabled. When the Power 9100 overlapping text and graphics mode is invoked, if this bit is set to 1, the character background color is the first entry in the attribute color palette.

Bit 0 is defined as in the standard VGA.

| Bit | Value | Meaning |
|---|---|---|
| 7 | 0 | Palette register bits 4 and 5 provide address bits 4 and 5 to color registers (see figure 280) |
| | 1 | Color select register C45 field provides address bits 4 and 5 to color registers (see figure 289). |
| 6 | 0 | Pixel data changed each dot clock cycle |
| | 1 | Pixel data changed every other dot clock |
| 5 | 0 | Prevent line compare from affecting pixel panning register output |
| | 1 | Allow line compare to affect horizontal pixel panning and preset row scan register outputs (see figures 286 and 226) |
| 3 | 0 | Character attribute code bit 7 selects background color. inhibit blinking |
| | 1 | Character attribute code bit 7 enables or disables blinking |
| 2 | 0 | Set ninth character dot to background color |
| | 1 | Set ninth character dot to eighth character dot for all graphics characters |
| 1 | 0 | Select color display attributes |
| | 1 | Select IBM Monochrome Display Adapter attributes |
| 0 | 0 | Select alphanumeric mode |
| | 1 | Select graphics mode |

Figure 282. Attribute mode control register fields

*REGISTER DESCRIPTION*

This video control read/write register is accessed through location 03C0 hex (write) or 03C1 hex (read) when the index field of the attribute address register is 10 hex.

Bit 5 allows line compare (see figure 249) to affect horizontal pixel panning and preset row scan register outputs (see figure 226 and 286).

Bit 0 should have the same contents as the G/A field of the miscellaneous register in the graphics controller registers group (see figure 271).

## 12.8. Attribute Controller Registers, continued

### 12.8.4. OVERSCAN COLOR REGISTER

The *overscan color register* determines the color of the border area displayed on the CRT.

*REGISTER FORMAT*

Overscan Color — Port 3C1 (Read), 3C0 (Write) ACR Index 11

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| P7 | P6 | P5 | P4 | P3 | P2 | P1 | P0 |

| Bit No. | Meaning |
|---|---|
| 7 | Pixel Address 7 (P7) |
| 6 | Pixel Address 6 (P6) |
| 5 | Pixel Address 5 (P5) |
| 4 | Pixel Address 4 (P4) |
| 3 | Pixel Address 3 (P3) |
| 2 | Pixel Address 2 (P2) |
| 1 | Pixel Address 1 (P1) |
| 0 | Pixel Address 0 (P0) |

Figure 283. Overscan color register format

*REGISTER DESCRIPTION*

The border area includes the horizontal lines occurring between the value in the start vertical blanking register (see section 12.6.23) and the value in the vertical display enable register of the CRT controller registers group (see section 12.6.20).

The overscan color high (OCH) field of the Power 9100 overscan color high register of the attribute controller registers group (see figure 290) provide 8 additional high-order border-color bits for Power 9100 modes.

## 12.8. Attribute Controller Registers, continued

### 12.8.5. COLOR PLANE ENABLE REGISTER

The *color plane enable register* controls which color planes will be enabled during the display process.

*REGISTER FORMAT*

Color Plane Enable — Port 3C1 (Read), 3C0 (Write) ACR Index 12

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| SBE | HBM | VSM | | CPE | | | |

| Bit No. | Meaning |
|---|---|
| 7 | Power 9100 Slow Blink Enable (SBE) |
| 6 | Power 9100 Half Bright Mode (HBM) |
| 5..4 | Video Status MUX (VSM) |
| 3..0 | Color Planes Enable (CPE) |

Figure 284. Color plane enable register format

*FIELD DEFINITION*

Bit 7 is the *slow-blink enable* bit. The character/line blinking rate on the standard VGA is determined by a counter which counts a fixed number of vertical frame periods. On modes with higher frame rates, this results in faster blinking. When this bit is set to 1, the blinking rate is halved.

Bit 6 is the *half-bright mode* bit. This mode is intended for use with monitors which have only two levels of grey. When this bit is set to 1, and the half-bright attribute is used, alternate lines of the character are blanked, simulating half brightness.

Bits [5..0] are defined as in the standard VGA.

| Bits | Value | Meaning |
|---|---|---|
| 7 | 0 | Normal blinking rate |
| | 1 | Cut blinking rate in half |
| 6 | 0 | Normal brightness control |
| | 1 | Blank alternate lines of the character when the half bright attribute is being used |
| 5..4 | 00 | DU field shows color outputs P2/P0 |
| | 01 | DU field shows color outputs P5/P4 |
| | 10 | DU field shows color outputs P3/P1 |
| | 11 | DU field shows color outputs P7/P6 |
| 3..0 | *xxx*0 | Do not select display plane 0 |
| | *xxx*1 | Select display plane 0 |
| | *xx*0*x* | Do not select display plane 1 |
| | *xx*1*x* | Select display plane 1 |
| | *x*0*xx* | Do not select display plane 2 |
| | *x*1*xx* | Select display plane 2 |
| | 0*xxx* | Do not select display plane 3 |
| | 1*xxx* | Select display plane 3 |

Figure 285. Color plane enable register fields

*REGISTER DESCRIPTION*

This video control read/write register is accessed through location 03C0 hex (write) or 03C1 hex (read) when the index field of the attribute address register is 12 hex.

Bits 5..4 specify the color output contents of the diagnostic use (DU) field of the input status 1 register in the general registers group (see figure 173).

## 12.8. Attribute Controller Registers, continued

### 12.8.6. HORIZONTAL PIXEL PANNING REGISTER

The *horizontal pixel panning register* selects the number of pixels by which to left-shift the video data horizontally.

*REGISTER FORMAT*

Horizontal Pixel Panning — Port 3C1 (Read), 3C0 (Write) ACR Index 13

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | HPP | | | |

| Bit No. | Meaning |
|---|---|
| 7..4 | Reserved |
| 3..0 | Horizontal Pixel Pan (HPP) |

Figure 286. Horizontal pixel panning register format

*FIELD DEFINITION*

| Bits 3..0 | Number of Pixels Shifted to Left | | |
|---|---|---|---|
| | 0+, 1+, 2+, 3+, 7, 7+ | All Other Modes | Mode 13 |
| 0000 | 1 | 0 | 0 |
| 0001 | 2 | 1 | — |
| 0010 | 3 | 2 | 1 |
| 0011 | 4 | 3 | — |
| 0100 | 5 | 4 | 2 |
| 0101 | 6 | 5 | — |
| 0110 | 7 | 6 | 3 |
| 0111 | 8 | 7 | — |
| 1000 | 0 | — | — |
| 1001–1111 | — | — | — |

Figure 287. Bits 3..0 field

*REGISTER DESCRIPTION*

Bits 3..0 determine the number of pixels to pan, and the preset row scan register byte panning (BP) field in the CRT controller registers group controls the number of bytes to pan (see figure 226). The pixel panning compatibility (PPC) field of the attribute mode control register of the attribute controller registers group (see figure 282) allows line compare (see figure 249) to affect horizontal pixel panning and preset row scan register outputs (see figure 226).

## 12.8.7. COLOR SELECT REGISTER

The *color select register* combines with the palette registers to address the video DAC registers.

*REGISTER FORMAT*

Color Select — Port 3C1 (Read), 3C0 (Write) ACR Index 14

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | C67 | | C45 | |

| Bit No. | Meaning |
|---|---|
| 7..4 | Reserved |
| 3..2 | Color Register Address Bits 6 and 7 (C67) |
| 1..0 | Color Register Address Bits 4 and 5 (C45) |

Figure 288. Color select register format

*FIELD DEFINITION*

| Bits | Meaning |
|---|---|
| 3..2 | Combined with 6 color palette output bits to form 8-bit address to color registers |
| 1..0 | Combined with C67 field and 4 low-order color palette output bits for form 8-bit color register address (see figure 280) |

Figure 289. Color select register fields

*REGISTER DESCRIPTION*

Bits 1..0 are used only when the internal palette size (IPS) field of the attribute mode control register is set to 1 (see figure 282).

## 12.8.8. POWER 9100 OVERSCAN COLOR HIGH REGISTER

The *Power 9100 overscan color high register*, in conjunction with the standard VGA overscan color register, determines the border color.

*REGISTER FORMAT*

Power 9100 Overscan Color High — Port 03C0 (write) Port 03C1 (read) — Att Index 15

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| Overscan Color High | | | | | | | |

| Bit No. | Meaning |
|---|---|
| 7..0 | Overscan Color High (OCH) |

Figure 290. Power 9100 overscan color high register format

*FIELD DEFINITION*

Bits [7..0] are the *overscan color high* bits. This 8-bit value determines the 8 high-order bits of the overscan or border color. The border is a band of color, one 80-column character wide, around the edges of the display area. Borders are not supported in 40-column alphanumeric modes or in 320 PEL graphics modes, except mode 13 (256 color). This register is usable in 16-bit direct color modes.

*REGISTER DESCRIPTION*

This video control read/write register is accessed through location 03C0 hex (write) or location 03C1 hex (read) when the index field of the attribute address register is 15 hex. This Power 9100 register is not a standard VGA register.

The 8 low-order bits specifying border colors for either standard VGA modes or Power 9100 modes is provided by the overscan color register of the attribute controller registers group (see figure 283).

# Chapter 13. Specifications

## 13.1. DC Specifications

### 13.1.1. ABSOLUTE MAXIMUM RATINGS

| Parameter | Value | Unit |
|---|---|---|
| Supply Voltage | –0.5 to +7.0 | Volts |
| Input Voltage | –0.5 to Vcc + 0.5 | Volts |
| Output Voltage | –0.5 to Vcc + 0.5 | Volts |
| Storage Temperature Range | –40 to +125 | °C |
| Lead Temperature (~10 seconds) | 250 | ° C |

Figure 291. Absolute maximum ratings

### 13.1.2. RECOMMENDED OPERATING CONDITIONS

| Parameter | Minimum Value | Nominal Value | Maximum Value |
|---|---|---|---|
| Supply Voltage ($V_{CC}$) | 4.75 VDC | 5.0 VDC | 5.25 VDC |
| Operating Case Temperature ($T_{CASE}$) | 0° C | | 85° C |

Figure 292. Recommended operating conditions

### 13.1.3. PIN CAPACITANCE

| Parameter | Description | Value |
|---|---|---|
| $C_{IN}$ | Input capacitance | 12 pF (typical) |
| $C_{OUT}$ | Output capacitance | 12 pF (typical) |
| $C_{IO}$ | Bidirectional capacitance | 15 pF (typical) |
| $C_{CLK}$ | Clock capacitance | 12 pF (typical) |

Figure 293. Pin capacitance (Capacitance not tested, TA = 25° C, f = 1 MHz, VCC = 5 VDC)

### 13.1.4. DC CHARACTERISTICS

| Parameter | Description | Test Conditions | Minimum | Maximum |
|---|---|---|---|---|
| $V_{IH}$ | High-level input voltage | $V_{CC}$ = MAX | 2.0 VDC | |
| $V_{IHC}$ | High-level input voltage for clock signals | $V_{CC}$ = MAX | 2.4 VDC | |
| $V_{IL}/V_{ILC}$ | Low-level input voltage | $V_{CC}$ = MIN | | 0.8 VDC |
| $V_{OH}$ | High-level output voltage | $V_{CC}$ = MIN, $I_{OH}$ = – 1.0 mA | 2.8 VDC | |
| $V_{OL}$ | Low-level output voltage | $V_{CC}$ = MIN, $I_{OL}$ = 4.0 mA | | 0.4 VDC |
| $I_{LO}$ | Output leakage current | $V_{CC}$ = MAX, $V_{OUT}$= 0 to $V_{CC}$ | –10 uA | +10 uA |
| $I_{LI}$ | Input leakage current | $V_{CC}$ = MAX, $V_{IN}$ = 0 to $V_{CC}$ | –10 uA | +10 uA |
| $I_{CC}$ | Standby current | $V_{CC}$ = MAX, f = 1 MHz | | 150 mA |
| $I_{DD}$ | Switching current | $V_{CC}$ = MAX, f = 50 Mhz | | TBD |
| $I_{OHD}$ | HD[31..0] drive current | | –8 mA | +8 mA |
| $I_{OLD}$ | HD[31..0] drive current | | –8 mA | +8 mA |

Figure 294. DC characteristics over the operating range

## 13.2. Supported Components

WEITEK has analyzed the AC specifications of a number of VRAM, RAMDAC, and clock generator parts. The products listed below have published AC specifications that are compatible with the Power 9100.

*[Editor's Note: Results of the component analysis effort were not available at press time.]*

### 13.2.1. COMPATIBLE VRAMS

The following VRAMs work with the Power 9100. Some are not fast enough for 50 MHz operation, but will run at 40 or 45 MHz.

| VRAM Type | 40 MHz | 45 MHz | 50 MHz |
|---|---|---|---|
| Micron MT42C8256–6 | ✓ | ✓ | ✓ |
| Micron MT42C8256–7 | ✓ | ✓ | |
| | ✓ | ✓ | ✓ |
| | ✓ | ✓ | ✓ |
| | ✓ | ✓ | ✓ |

Figure 295. VRAMs known to work with the Power 9100

### 13.2.2. COMPATIBLE RAMDACS

The following RAMDACs are electrically compatible with the Power 9100. See the "Comments" field for information about software driver support for individual RAMDACs.

| Type | Comments |
|---|---|
| Brooktree Bt485 | |
| AT&T 20C505 | Bt485 compatible |
| Brooktree Bt885 | |
| TI TVP3010 | |
| TI TVP3020 | |

Figure 296. Power 9100-compatible RAMDACs

### 13.2.3. COMPATIBLE CLOCK GENERATORS

The following clock generators are compatible with the Power 9100.

| Type | Comments |
|---|---|
| IC Designs ICD2061 | |
| IC Designs ICD2062 | |

Figure 297. Power 9100-compatible clock generators

## 13.3. AC Specifications

| Param. | Description | Reference Signal | 50 MHz | | Unit | Loading |
|---|---|---|---|---|---|---|
| | | | MIN | MAX | | |
| *VESA Local Bus Interface* | | | | | | |
| $T_{LCT}$ | LCLK cycle time | | 20 | | ns | |
| $T_{LCHL}$ | LCLK high/low time | | 8 | | ns | |
| $T_{G1S}$ | ADR[31..2], BE[3..0]–, M/IO–, RESET–, RDYRNT– input setup time | LCLK | | 7 | ns | |
| $T_{G1H}$ | BE[3..0]–, M/IO–, RESET–, RDYRNT– input hold time | | 0 | | ns | |
| $T_{LRDD}$ | LRDY– output delay time | LCLK | | 10 | ns | 100 pF |
| $T_{LRDV}$ | LRDY– output valid time | LCLK | 3 | | ns | 100 pF |
| $T_{DS}$ | DATA[31..0] input setup time | LCLK | | 7 | ns | |
| $T_{DH}$ | DATA[31..0] input hold time | | 0 | | ns | |
| $T_{DD}$ | DATA[31..0] output delay time | LCLK | | 15 | ns | 100 pF |
| $T_{DTO}$ | DATA[31..0] output turn-off time | LCLK | | 3 | ns | 100 pF |
| $T_{LDD}$ | LDEV– output delay time | ADR[31..2], BE– | | 20 | ns | 33 pF |
| $T_{LRDV}$ | LDEV– output valid time | | 0 | | ns | 33 pF |
| *PCI Bus Interface* | | | | | | |
| $T_{BCT}$ | BCLK cycle time | | 20 | | ns | |
| $T_{BCHL}$ | BCLK high/low time | | 8 | | ns | |
| $T_{INS}$ | C/BE[3..0]–, FRAME–, IRDY–, IDSL input setup times | BCLK | | 7 | ns | |
| $T_{INH}$ | C/BE[3..0]–, FRAME–, IRDY–, IDSL input hold times | | 0 | | ns | |
| $T_{IOS}$ | AD[31..0], PAR input setup times | BCLK | | 7 | ns | |
| $T_{IOS}$ | AD[31..0], PAR input hold times | | 0 | | ns | |
| $T_{IOD}$ | AD[31..0], PAR output delay times | BCLK | | 11 | ns | 50 pF |
| $T_{IOV}$ | AD[31..0], PAR output valid times | | 2 | | ns | 50 pF |
| $T_{IOTO}$ | AD[31..0], PAR turn-off times | | 2 | 15 | ns | 50 pF |

Figure 298. AC Specifications for VL and PCI bus interfaces

## 13.3. AC Specifications, continued

| Param. | Description | Reference Signal | MIN | MAX | Unit | Loading |
|---|---|---|---|---|---|---|
| *Video Interface* | | | | | | |
| $T_{DCPY}$ | DIVPIXCLK period | | 11.76 | | ns | |
| $T_{DPCH}$ | DIVPIXCLK high | | 6.12 | | ns | |
| $T_{DPCL}$ | DIVPIXCLK low | | 4.23 | | ns | |
| $T_{PCY}$ | PIXCLK period | | 12.5 | | ns | |
| $T_{PCH}$ | PIXCLK high | | 6.5 | | ns | |
| $T_{PCL}$ | PIXCLK low | | 4.5 | | NS | |
| $T_{HSD}$ | HSYNC– output delay | VIDOUTCLK+ | | 5 | ns | |
| $T_{VSD}$ | VSYNC– output delay | VIDOUTCLK+ | | 5 | ns | |
| $T_{BD}$ | BLANK– output delay | VIDOUTCLK+ | | 5 | ns | |
| $T_{HSS}$ | HSYNC– setup time | | TBD | | ns | |
| $T_{HSH}$ | HSYNC– hold time | | TBD | | ns | |
| $T_{VSS}$ | VSYNC– setup time | | TBD | | ns | |
| $T_{VSH}$ | VSYNC– hold time | | TBD | | ns | |
| $T_{SCD}$ | SC output delay | VIDOUTCLK | 1 | TBD | ns | |
| $T_{SED}$ | SE output delay | VIDOUTCLK | | TBD | | |
| $T_{OH}$ | | | 0 | | ns | |
| $T_{OE}$ | | | 6M | | | |
| $T_{ACC}$ | | | 6M | | | |
| *Video Coprocessor Interface* | | | | | | |
| *[Editor's Note: These specs are not available yet.]* | | | | | | |
| | | | | | | |

Figure 299. Switching characteristics over the operating range

## 13.3. AC Specifications, continued

M = one memclk period [ns].

Timings for 50MHz mode depend on the following settings:

mem_config.vram_miss_adj=1

mem_config.vram_read_adj=1

mem_config.vram_write_adj=1

mem_config.vram_sample_adj=1

| Param. | Description | Reference Signal | MIN | MAX | Unit | Loading |
|---|---|---|---|---|---|---|
| *VRAM Interface* | | | | | | |
| $T_{PLL}$ | PLL clock lock time at 25 MHz after reset for 50 MHz part[1] | | | | clocks | |
| $T_{AA}$ | Access time from column address | | | 2.25M–15 | ns | |
| $T_{AR}$ | Column address hold time | RAS– | 3M–10 | | ns | |
| $T_{ASC}$ | Column address setup time | CAS– | 0.25M–5 | | ns | |
| $T_{ASR}$ | Row address setup time[1] | RAS– | 2M–9 | | ns | |
| $T_{CAC}$ | Access time from CAS–[1] | | | 1M–12 | ns | |
| $T_{CAH}$ | Column address hold time | CAS– | 0.75M–5 | | ns | |
| $T_{CAS}$ | CAS– pulse width | | 1M–5 | | ns | |
| $T_{CHR}$ | CAS– hold time during refresh[1] | | 3M–9 | | ns | |
| $T_{CP}$ | CAS– precharge time during page mode | | 0.75M–5 | | ns | |
| $T_{CPA}$ | Access time from CAS– precharge | | | 2.25M–13 | ns | |
| $T_{CPN}$ | CAS– precharge time[1] | | 0.75M–5 | | ns | |
| $T_{CRP}$ | CAS– to RAS– precharge time[1] | | 3M–9 | | ns | |
| $T_{CSH}$ | CAS– hold time | RAS– | 3M | | ns | |
| $T_{CSR}$ | CAS– setup time during refresh[1] | | 2M–9 | | ns | |
| $T_{CWL}$ | Write command to CAS– lead time[1] | | 2M–9 | | ns | |
| $T_{DH}$ | Data input hold time | CAS– | 0.75M–5 | | ns | |
| $T_{DHR}$ | Data input hold time | RAS– | 3.25M–9 | | ns | |
| $T_{DS}$ | Data input setup time | CAS– | 0.25M–5 | | ns | |
| $T_{FSR}$ | DSF setup time[1] | RAS– | 3M–9 | | ns | |
| $T_{MCH}$ | MEMCLK high | | 8 | | ns | |
| $T_{MCL}$ | MEMCLK low | | 8 | | ns | |
| $T_{MCY}$, $T_M$ | MEMCLK period | | 20 | | ns | |
| $T_{MH}$ | Mask data to RAS– hold time[1] | | 2M–9 | | ns | |
| $T_{MS}$ | Mask data to RAS– setup time[1] | | 1.75M–9 | | ns | |
| $T_{OEA}$, $T_{OE}$ | Access time from OE–[1] | | | 1M–5 | ns | |
| $T_{OEZ}$ | Output disable time | OE– | 0 | 15 | ns | |
| $T_{OFF}$ | Output buffer turn-off delay from CAS–[1] | | 0 | 15 | ns | |
| $T_{PC}$ | CAS– page mode cycle time | | 2M | | ns | |

1. Not tested, but guaranteed by design.

Figure 299, continued. Switching characteristics over the operating range

## 13.3. AC Specifications, continued

| Param. | Description | Reference Signal | MIN | MAX | Unit | Loading |
|---|---|---|---|---|---|---|
| *VRAM Interface, continued* | | | | | | |
| $T_{RAC}$ | Access time from RAS–[1] | | | 3M | ns | |
| $T_{RAH}$ | Row address hold time[1] | | 1M–10 | | ns | |
| $T_{RAL}$ | Column address to RAS– lead time[1] | | 2M–5 | | ns | |
| $T_{RAS}$ | RAS– pulse width[1] | | 3M | | ns | |
| $T_{RASP}$ | RAS– pulse width during page mode[1] | | 4M–6 | | ns | |
| $T_{RC}$ | RAS– random cycle time | | 6M–10 | | ns | |
| $T_{RCD}$ | RAS– to CAS– delay time[1] | | 2M–9 | | ns | |
| $T_{RCH}$ | Read command hold time[1] | CAS– | 1M–9 | | ns | |
| $T_{RCS}$ | Read command setup time[1] | CAS– | 0.75M–6 | | ns | |
| $T_{RFH}$ | DSF hold time[1] | RAS– | 1M–10 | | ns | |
| $T_{ROH}$ | RAS– hold time | OE– | 1.5M–9 | | ns | |
| $T_{RP}$ | RAS– precharge time | | 2M–6 | | ns | |
| $T_{RPC}$ | RAS– to CAS– precharge time[1] | | 1M–9 | | ns | |
| $T_{RRH}$ | Read command hold time[1] | RAS– | 1M–9 | | ns | |
| $T_{RSH}$ | RAS– hold time[1] | | 1M–5 | | ns | |
| $T_{RWH}$ | WE– to RAS– hold time[1] | | 2M–9 | | ns | |
| $T_{RWL}$ | Write command to RAS– lead time[1] | | 2M–9 | | ns | |
| $T_{THH}$, $T_{YH}$ | OE– high hold time[1] | | 2.5M–9 | | ns | |
| $T_{THS}$, $T_{YS}$ | OE– high setup time[1] | | 3M–9 | | ns | |
| $T_{TLH}$ | OE– low hold time from RAS–[1] | | 1M–10 | | ns | |
| $T_{TLS}$ | OE– low setup time to RAS–[1] | | 2M–9 | | ns | |
| $T_{TRW}$,$T_{TP}$ | TR(OE–) precharge time | | 8.5M–9 | | | |
| $T_{TRP}$ | OE– to RAS– precharge time[1] | | 6M–9 | | ns | |
| $T_{WCH}$ | Write command hold time[1] | CAS– | 1.5M–9 | | ns | |
| $T_{WCR}$ | Write command hold time[1] | RAS– | 4M–9 | | ns | |
| $T_{WCS}$ | Write command setup time[1] | CAS– | 0.5M–9 | | ns | |
| $T_{WP}$ | Write command pulse width[1] | | 2M–9 | | ns | |
| $T_{WSR}$ | WE– to RAS– setup time[1] | | 2M–9 | | ns | |
| 1. Not tested, but guaranteed by design. | | | | | | |

Figure 299, continued. Switching characteristics over the operating range

## 13.4. VL Bus Pin Configuration

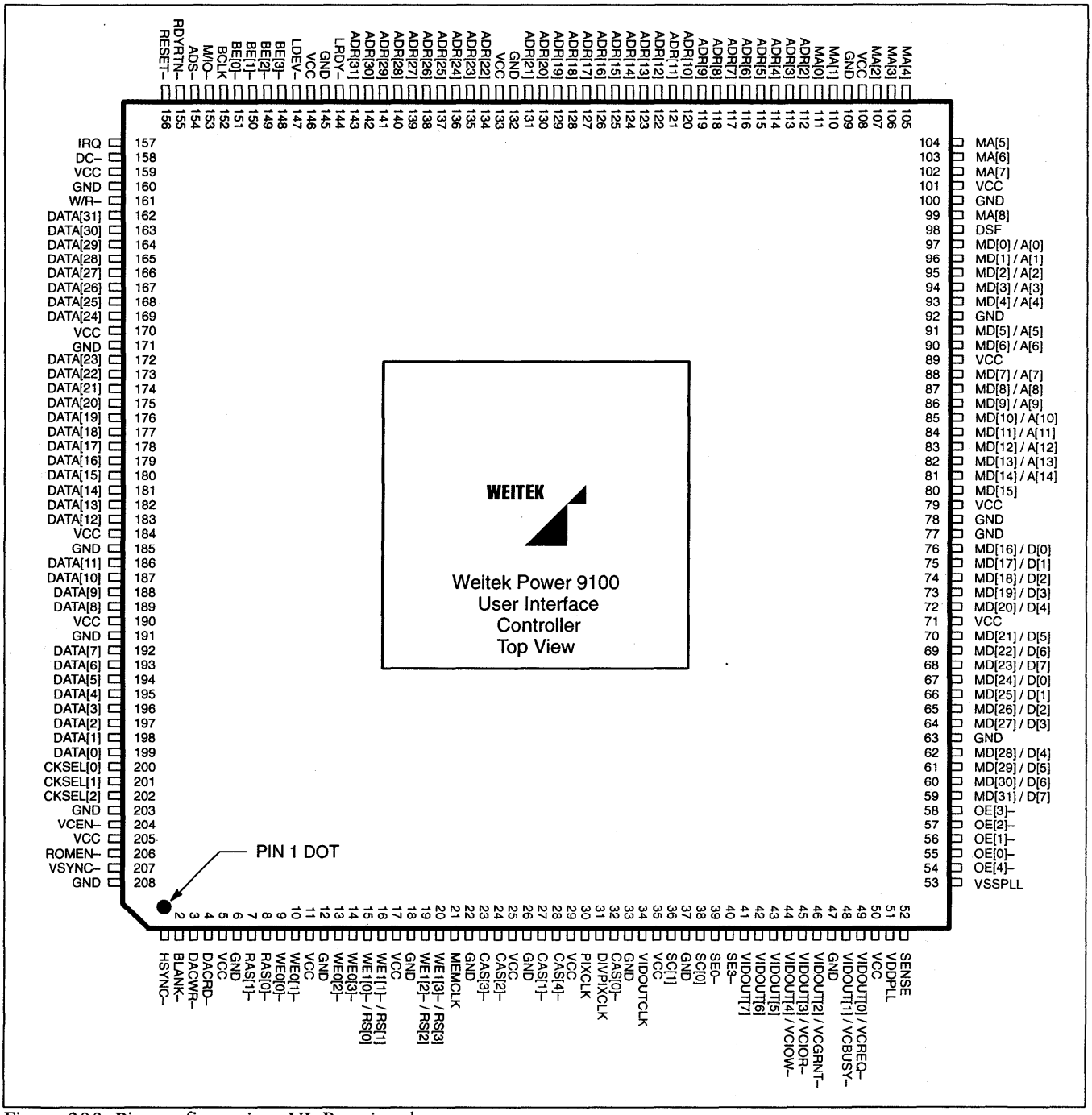



Figure 300. Pin configuration: VL Bus signals

## 13.5. PCI Bus Pin Configuration

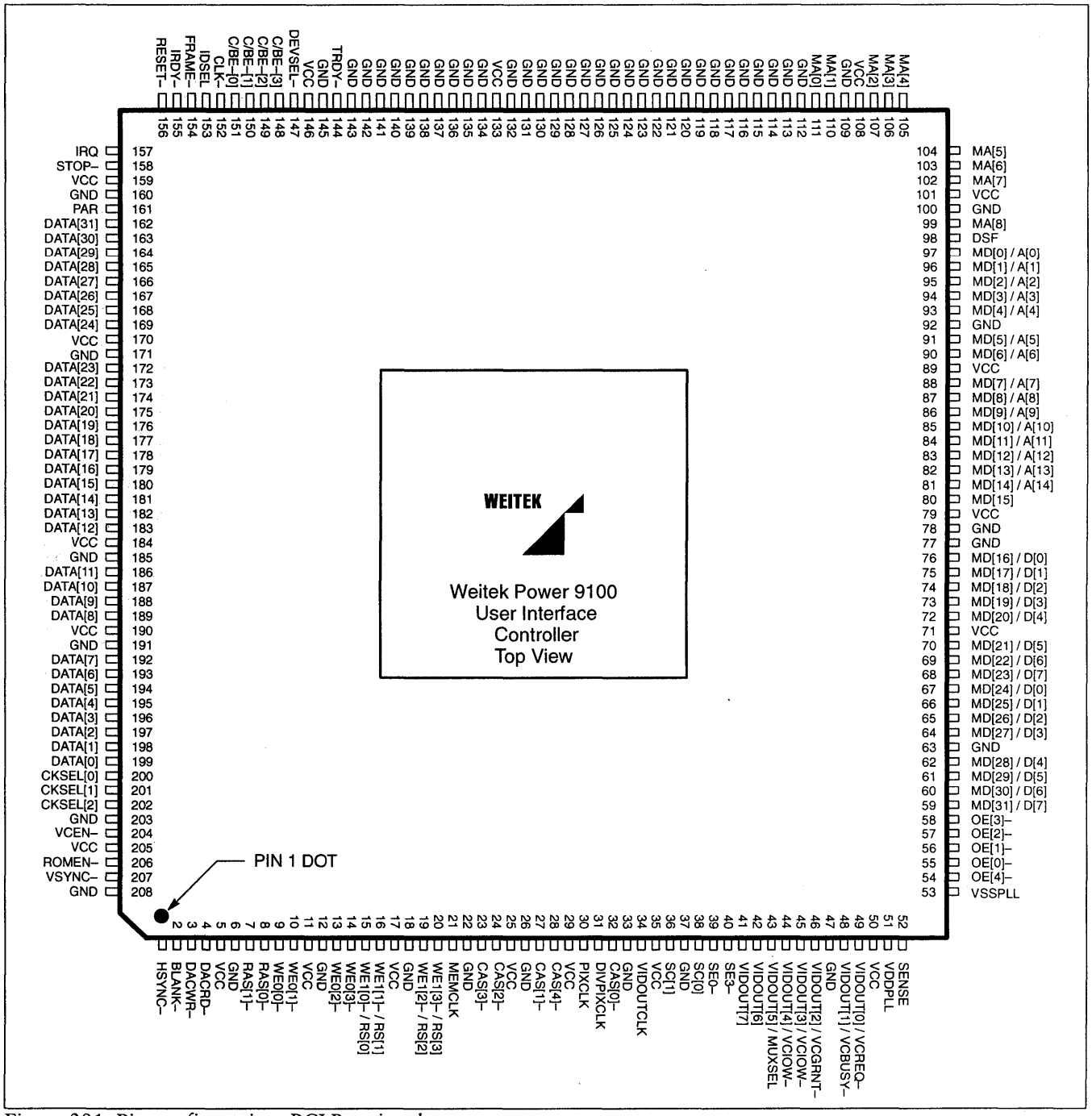



Figure 301. Pin configuration: PCI Bus signals

## 13.6. Pin Assignments

| Pin | Signal / VL Bus | | PCI Bus | | RAMDAC | | BIOS ROM | | Video Coprocessor | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Signal | Type | Signal | Type | Signal | Type | Signal | Type | Signal | Type |
| 1 | HSYNC– | Input/ Output | | | | | | | | |
| 2 | BLANK– | Output | | | | | | | | |
| 3 | DACWR– | Output | | | | | | | | |
| 4 | DACRD– | Output | | | | | | | | |
| 5 | VCC | | | | | | | | | |
| 6 | GND | | | | | | | | | |
| 7 | RAS[1]– | Tri-stated/ Output | | | | | | | | |
| 8 | RAS[0]– | Tri-stated/ Output | | | | | | | | |
| 9 | WE0[0]– | Tri-stated/ Output | | | | | | | | |
| 10 | WE0[1]– | Tri-stated/ Output | | | | | | | | |
| 11 | VCC | | | | | | | | | |
| 12 | GND | | | | | | | | | |
| 13 | WE0[2]– | Tri-stated/ Output | | | | | | | | |
| 14 | WE0[3]– | Tri-stated/ Output | | | | | | | | |
| 15 | WE1[0]– | Tri-stated/ Output | | | RS[0] | Output | | | | |
| 16 | WE1[1]– | Tri-stated/ Output | | | RS[1] | Output | | | | |
| 17 | VCC | | | | | | | | | |
| 18 | GND | | | | | | | | | |
| 19 | WE1[2]– | Tri-stated/ Output | | | RS[2] | Output | | | | |
| 20 | WE1[3]– | Tri-stated/ Output | | | RS[3] | Output | | | | |
| 21 | MEMCLK | Input | | | | | | | | |
| 22 | GND | | | | | | | | | |
| 23 | CAS[3]– | Tri-stated/ Output | | | | | | | | |
| 24 | CAS[2]– | Tri-stated/ Output | | | | | | | | |
| 25 | VCC | | | | | | | | | |
| 26 | GND | | | | | | | | | |

Figure 302. Pin assignments (1 of 8)

## 13.6. Pin Assingments, continued

| Pin | Signal / VL Bus | | PCI Bus | | RAMDAC | | BIOS ROM | | Video Coprocessor | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Signal | Type | Signal | Type | Signal | Type | Signal | Type | Signal | Type |
| 27 | CAS[1]– | Tri-stated/ Output | | | | | | | | |
| 28 | CAS[0]B– | Tri-stated/ Output | | | | | | | | |
| 29 | VCC | | | | | | | | | |
| 30 | PIXCLK | Input | | | | | | | | |
| 31 | DIV-PIXCLK | Input | | | | | | | | |
| 32 | CAS[0]A– | Tri-stated/ Output | | | | | | | | |
| 33 | GND | | | | | | | | | |
| 34 | VID-OUTCLK | Output | | | | | | | | |
| 35 | VCC | | | | | | | | | |
| 36 | SC[1] | Output | | | | | | | | |
| 37 | GND | | | | | | | | | |
| 38 | SC[0] | Output | | | | | | | | |
| 39 | SE[0]– | Output | | | | | | | | |
| 40 | SE[3]– | Output | | | | | | | | |
| 41 | VIDOUT[7] | Tri-stated/ Output | | | | | | | | |
| 42 | VIDOUT[6] | Tri-stated/ Output | | | | | | | | |
| 43 | VIDOUT[5] | Tri-stated/ Output | | | | | | | | |
| 44 | VIDOUT[4] | Tri-stated/ Output | | | | | | | VCIOW– | Output |
| 45 | VIDOUT[3] | Tri-stated/ Output | | | | | | | VCIOR– | Output |
| 46 | VIDOUT[2] | Tri-stated/ Output | | | | | | | VCGRNT– | Output |
| 47 | GND | | | | | | | | | |
| 48 | VIDOUT[1] | Tri-stated/ Output | | | | | | | VCBUSY– | Input |
| 49 | VIDOUT[0] | Input/ Output | | | | | | | VCREQ– | Input |
| 50 | VCC | | | | | | | | | |
| 51 | PLLVDD | | | | | | | | | |
| 52 | SENSE | Input | | | | | | | | |
| 53 | PLLGND | | | | | | | | | |

Figure 302. Pin assignments (2 of 8)

## 13.6. Pin Assingments, continued

| Pin | Signal / VL Bus | | PCI Bus | | RAMDAC | | BIOS ROM | | Video Coprocessor | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Signal | Type | Signal | Type | Signal | Type | Signal | Type | Signal | Type |
| 54 | OE[0]B– | Tri-stated/ Output | | | | | | | | |
| 55 | OE[0]A– | Tri-stated/ Output | | | | | | | | |
| 56 | OE[1]– | Tri-stated/ Output | | | | | | | | |
| 57 | OE[2]– | Tri-stated/ Output | | | | | | | | |
| 58 | OE[3]– | Tri-stated/ Output | | | | | | | | |
| 59 | MD[31] | Input/ Output | | | | | D[7] | Input | VDDAT[31] | Input/ Output |
| 60 | MD[30] | Input/ Output | | | | | D[6] | Input | VDDAT[30] | Input/ Output |
| 61 | MD[29] | Input/ Output | | | | | D[5] | Input | VDDAT[29] | Input/ Output |
| 62 | MD[28] | Input/ Output | | | | | D[4] | Input | VDDAT[28] | Input/ Output |
| 63 | GND | | | | | | | | | |
| 64 | MD[27] | Input/ Output | | | | | D[3] | Input | VDDAT[27] | Input/ Output |
| 65 | MD[26] | Input/ Output | | | | | D[2] | Input | VDDAT[26] | Input/ Output |
| 66 | MD[25] | Input/ Output | | | | | D[1] | Input | VDDAT[25] | Input/ Output |
| 67 | MD[24] | Input/ Output | | | | | D[0] | Input | VDDAT[24] | Input/ Output |
| 68 | MD[23] | Input/ Output | | | D[7] | Input/ Output | | | VDDAT[23] | Input/ Output |
| 69 | MD[22] | Input/ Output | | | D[6] | Input/ Output | | | VDDAT[22] | Input/ Output |
| 70 | MD[21] | Input/ Output | | | D[5] | Input/ Output | | | VDDAT[21] | Input/ Output |
| 71 | VCC | | | | | | | | | |
| 72 | MD[20] | Input/ Output | | | D[4] | Input/ Output | | | VDDAT[20] | Input/ Output |
| 73 | MD[19] | Input/ Output | | | D[3] | Input/ Output | | | VDDAT[19] | Input/ Output |
| 74 | MD[18] | Input/ Output | | | D[2] | Input/ Output | | | VDDAT[18] | Input/ Output |
| 75 | MD[17] | Input/ Output | | | D[1] | Input/ Output | | | VDDAT[17] | Input/ Output |
| 76 | MD[16] | Input/ Output | | | D[0] | Input/ Output | | | VDDAT[16] | Input/ Output |

Figure 302. Pin assignments (3 of 8)

## 13.6. Pin Assingments, continued

| Pin | Signal / VL Bus | | PCI Bus | | RAMDAC | | BIOS ROM | | Video Coprocessor | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Signal | Type | Signal | Type | Signal | Type | Signal | Type | Signal | Type |
| 77 | GND | | | | | | | | | |
| 78 | GND | | | | | | | | | |
| 79 | VCC | | | | | | | | | |
| 80 | MD[15] | Input/ Output | | | | | | | VDDAT[15] | Input/ Output |
| 81 | MD[14] | Input/ Output | | | | | A[14] | Output | VDDAT[14] | Input/ Output |
| 82 | MD[13] | Input/ Output | | | | | A[13] | Output | VDDAT[13] | Input/ Output |
| 83 | MD[12] | Input/ Output | | | | | A[12] | Output | VDDAT[12] | Input/ Output |
| 84 | MD[11] | Input/ Output | | | | | A[11] | Output | VDDAT[11] | Input/ Output |
| 85 | MD[10] | Input/ Output | | | | | A[10] | Output | VDDAT[10] | Input/ Output |
| 86 | MD[9] | Input/ Output | | | | | A[9] | Output | VDDAT[9] | Input/ Output |
| 87 | MD[8] | Input/ Output | | | | | A[8] | Output | VDDAT[8] | Input/ Output |
| 88 | MD[7] | Input/ Output | | | | | A[7] | Output | VDDAT[7] | Input/ Output |
| 89 | VCC | | | | | | | | | |
| 90 | MD[6] | Input/ Output | | | | | A[6] | Output | VDDAT[6] | Input/ Output |
| 91 | MD[5] | Input/ Output | | | | | A[5] | Output | VDDAT[5] | Input/ Output |
| 92 | GND | | | | | | | | | |
| 93 | MD[4] | Input/ Output | | | | | A[4] | Output | VDDAT[4] | Input/ Output |
| 94 | MD[3] | Input/ Output | | | | | A[3] | Output | VDDAT[3] | Input/ Output |
| 95 | MD[2] | Input/ Output | | | | | A[2] | Output | VDDAT[2] | Input/ Output |
| 96 | MD[1] | Input/ Output | | | | | A[1] | Output | VDDAT[1] | Input/ Output |
| 97 | MD[0] | Input/ Output | | | | | A[0] | Output | VDDAT[0] | Input/ Output |
| 98 | DSF | Output | | | | | | | | |
| 99 | MA[8] | Tri-stated/ Output | | | | | | | VDADR[8] | Tri-stated/ Output |

Figure 302. Pin assignments (4 of 8)

## 13.6. Pin Assingments, continued

| Pin | Signal / VL Bus | | PCI Bus | | RAMDAC | | BIOS ROM | | Video Coprocessor | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Signal | Type | Signal | Type | Signal | Type | Signal | Type | Signal | Type |
| 100 | GND | | | | | | | | | |
| 101 | VCC | | | | | | | | | |
| 102 | MA[7] | Tri-stated/ Output | | | | | | | VDADR[7] | Tri-stated/ Output |
| 103 | MA[6] | Tri-stated/ Output | | | | | | | VDADR[6] | Tri-stated/ Output |
| 104 | MA[5] | Tri-stated/ Output | | | | | | | VDADR[5] | Tri-stated/ Output |
| 105 | MA[4] | Tri-stated/ Output | | | | | | | VDADR[4] | Tri-stated/ Output |
| 106 | MA[3] | Tri-stated/ Output | | | | | | | VDADR[3] | Tri-stated/ Output |
| 107 | MA[2] | Tri-stated/ Output | | | | | | | VDADR[2] | Tri-stated/ Output |
| 108 | VCC | | | | | | | | | |
| 109 | GND | | | | | | | | | |
| 110 | MA[1] | Tri-stated/ Output | | | | | | | VDADR[1] | Tri-stated/ Output |
| 111 | MA[0] | Tri-stated/ Output | | | | | | | VDADR[0] | Tri-stated/ Output |
| 112 | ADR[2] | Input | GND | | | | | | | |
| 113 | ADR[3] | Input | GND | | | | | | | |
| 114 | ADR[4] | Input | GND | | | | | | | |
| 115 | ADR[5] | Input | GND | | | | | | | |
| 116 | ADR[6] | Input | GND | | | | | | | |
| 117 | ADR[7] | Input | GND | | | | | | | |
| 118 | ADR[8] | Input | GND | | | | | | | |
| 119 | ADR[9] | Input | GND | | | | | | | |
| 120 | ADR[10] | Input | GND | | | | | | | |
| 121 | ADR[11] | Input | GND | | | | | | | |
| 122 | ADR[12] | Input | GND | | | | | | | |
| 123 | ADR[13] | Input | GND | | | | | | | |
| 124 | ADR[14] | Input | GND | | | | | | | |
| 125 | ADR[15] | Input | GND | | | | | | | |
| 126 | ADR[16] | Input | GND | | | | | | | |
| 127 | ADR[17] | Input | GND | | | | | | | |
| 128 | ADR[18] | Input | GND | | | | | | | |

Figure 302. Pin assignments (5 of 8)

## 13.6. Pin Assingments, continued

| Pin | Signal / VL Bus | | PCI Bus | | RAMDAC | | BIOS ROM | | Video Coprocessor | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Signal | Type | Signal | Type | Signal | Type | Signal | Type | Signal | Type |
| 129 | ADR[19] | Input | GND | | | | | | | |
| 130 | ADR[20] | Input | GND | | | | | | | |
| 131 | ADR[21] | Input | GND | | | | | | | |
| 132 | GND | | | | | | | | | |
| 133 | VCC | | | | | | | | | |
| 134 | ADR[22] | Input | GND | | | | | | | |
| 135 | ADR[23] | Input | GND | | | | | | | |
| 136 | ADR[24] | Input | GND | | | | | | | |
| 137 | ADR[25] | Input | GND | | | | | | | |
| 138 | ADR[26] | Input | GND | | | | | | | |
| 139 | ADR[27] | Input | GND | | | | | | | |
| 140 | ADR[28] | Input | GND | | | | | | | |
| 141 | ADR[29] | Input | GND | | | | | | | |
| 142 | ADR[30] | Input | GND | | | | | | | |
| 143 | ADR[31] | Input | GND | | | | | | | |
| 144 | LRDY– | Tri-stated | TRDY– | Tri-stated | | | | | | |
| 145 | GND | | | | | | | | | |
| 146 | VCC | | | | | | | | | |
| 147 | LDEV– | Output | DEVSEL– | Tri-stated | | | | | | |
| 148 | BE[3]– | Input | C/BE[3]– | Input | | | | | | |
| 149 | BE[2]– | Input | C/BE[2]– | Input | | | | | | |
| 150 | BE[1]– | Input | C/BE[1]– | Input | | | | | | |
| 151 | BE[0]– | Input | C/BE[0]– | Input | | | | | | |
| 152 | BCLK | Input | CLK– | Input | | | | | | |
| 153 | M/IO– | Input | IDSEL | Input | | | | | | |
| 154 | ADS– | Input | FRAME– | Input | | | | | | |
| 155 | RDYRTN– | Input | IRDY– | Input | | | | | | |
| 156 | RESET– | Input | | | | | | | | |
| 157 | IRQ | Output | IRQ– | Open Collector | | | | | | |
| 158 | D/C– | Input | STOP– | Input/ Output | | | | | | |
| 159 | VCC | | | | | | | | | |
| 160 | GND | | | | | | | | | |

Figure 302. Pin assignments (6 of 8)

## 13.6. Pin Assingments, continued

| Pin | Signal / VL Bus | | PCI Bus | | RAMDAC | | BIOS ROM | | Video Coprocessor | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Signal | Type | Signal | Type | Signal | Type | Signal | Type | Signal | Type |
| 161 | W/R– | Input | PAR | Input/ Output | | | | | | |
| 162 | DATA[31] | Input/ Output | | | | | | | | |
| 163 | DATA[30] | Input/ Output | | | | | | | | |
| 164 | DATA[29] | Input/ Output | | | | | | | | |
| 165 | DATA[28] | Input/ Output | | | | | | | | |
| 166 | DATA[27] | Input/ Output | | | | | | | | |
| 167 | DATA[26] | Input/ Output | | | | | | | | |
| 168 | DATA[25] | Input/ Output | | | | | | | | |
| 169 | DATA[24] | Input/ Output | | | | | | | | |
| 170 | VCC | | | | | | | | | |
| 171 | GND | | | | | | | | | |
| 172 | DATA[23] | Input/ Output | | | | | | | | |
| 173 | DATA[22] | Input/ Output | | | | | | | | |
| 174 | DATA[21] | Input/ Output | | | | | | | | |
| 175 | DATA[20] | Input/ Output | | | | | | | | |
| 176 | DATA[19] | Input/ Output | | | | | | | | |
| 177 | DATA[18] | Input/ Output | | | | | | | | |
| 178 | DATA[17] | Input/ Output | | | | | | | | |
| 179 | DATA[16] | Input/ Output | | | | | | | | |
| 180 | DATA[15] | Input/ Output | | | | | | | | |
| 181 | DATA[14] | Input/ Output | | | | | | | | |
| 182 | DATA[13] | Input/ Output | | | | | | | | |

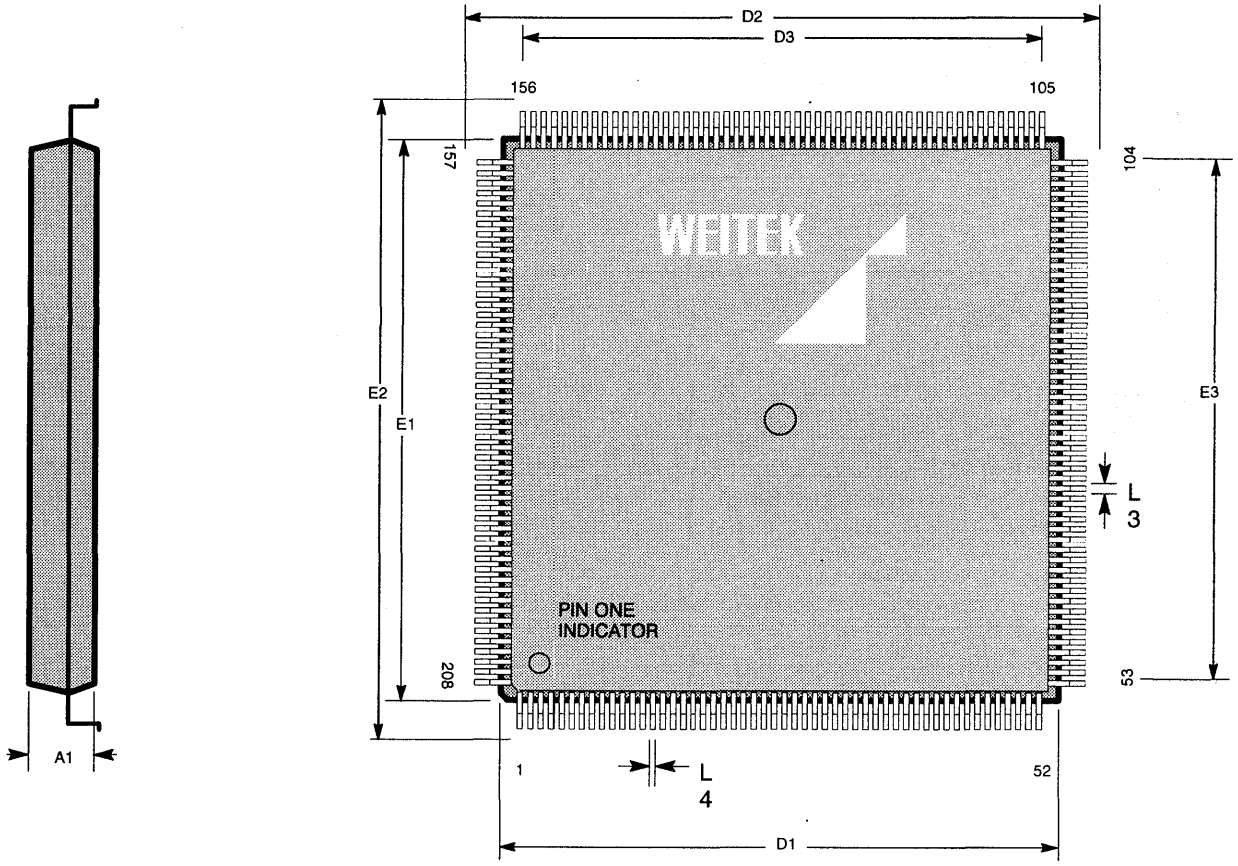
Figure 302. Pin assignments (7 of 8)

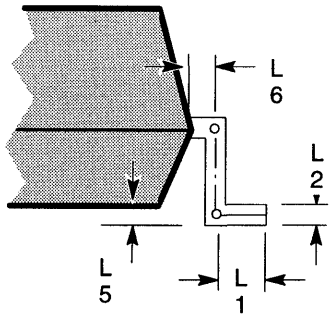
## 13.6. Pin Assingments, continued

| Pin | Signal / VL Bus | | PCI Bus | | RAMDAC | | BIOS ROM | | Video Coprocessor | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Signal | Type | Signal | Type | Signal | Type | Signal | Type | Signal | Type |
| 183 | DATA[12] | Input/ Output | | | | | | | | |
| 184 | VCC | | | | | | | | | |
| 185 | GND | | | | | | | | | |
| 186 | DATA[11] | Input/ Output | | | | | | | | |
| 187 | DATA[10] | Input/ Output | | | | | | | | |
| 188 | DATA[9] | Input/ Output | | | | | | | | |
| 189 | DATA[8] | Input/ Output | | | | | | | | |
| 190 | VCC | | | | | | | | | |
| 191 | GND | | | | | | | | | |
| 192 | DATA[7] | Input/ Output | | | | | | | | |
| 193 | DATA[6] | Input/ Output | | | | | | | | |
| 194 | DATA[5] | Input/ Output | | | | | | | | |
| 195 | DATA[4] | Input/ Output | | | | | | | | |
| 196 | DATA[3] | Input/ Output | | | | | | | | |
| 197 | DATA[2] | Input/ Output | | | | | | | | |
| 198 | DATA[1] | Input/ Output | | | | | | | | |
| 199 | DATA[0] | Input/ Output | | | | | | | | |
| 200 | CKSEL[0] | Output | | | | | | | | |
| 201 | CKSEL[1] | Output | | | | | | | | |
| 202 | CKSEL[2] | Output | | | | | | | | |
| 203 | GND | | | | | | | | | |
| 204 | VCEN– | Output | | | | | | | | |
| 205 | VCC | | | | | | | | | |
| 206 | ROMEN– | Output | | | | | | | | |
| 207 | VSYNC– | Input/ Output | | | | | | | | |
| 208 | GND | | | | | | | | | |

Figure 302. Pin assignments (8 of 8)

## 13.7. Mechanical Specifications




| SYMBOL | INCHES | | MM | |
|---|---|---|---|---|
| | MAX | MIN | MAX | MIN |
| A1 | 0.130 TYP | | 3.49 TYP | |
| D1 | 1.106 | 1.098 | 28.10 | 27.90 |
| D2 | 1.215 | 1.195 | 30.85 | 30.35 |
| D 3 | 1.004 REF | | 25.5 REF | |
| E1 | 1.106 | 1.098 | 28.10 | 27.90 |
| E2 | 1.215 | 1.195 | 30.85 | 30.35 |
| E3 | 1.004 REF | | 25.5 REF | |
| L1 | 0.026 | 0.016 | 0.650 | 0.400 |
| L2 | 0.008 | 0.004 | 0.2 | 0.1 |
| L3 | 0.0197 TYP | | 0.50 TYP | |
| L4 | 0.009 TYP | | 0.22 TYP | |
| L5 | 0.020 | 0.010 | 0.5 | 0.25 |
| L6 | — | 0.016 | — | 0.40 |

Figure 303. Physical dimensions (208-pin QFP)

## 13.8. Test Conditions

| Signal | Description | Pin Loading |
|---|---|---|
| MD[31..0] | Memory data bus | 50 pF |
| RAS[1..0]– | Row address strobe | 80 pF |
| CAS[1..0]– | Column address strobe | 80 pF |
| WE0[3..0]–<br>WE1[3..0]– | Write enable | 50 pF |
| MA[8..0] | Memory address bus | 150 pF |
| OE– | Output enable | 150 pF |
| DSF | Special function control | 150 pF |
| | RAMDAC chip enable | 50 pF |
| A 33Ω damping resistor is used in test load for the VRAM and RAMDAC controller signals. | | |

Figure 304. VRAM and RAMDAC controllers test loading

## 13.9. I/O Characteristics

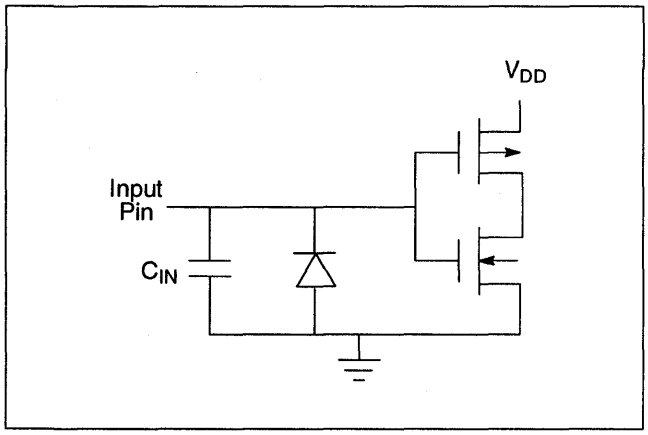



Figure 305. Input equivalent circuits

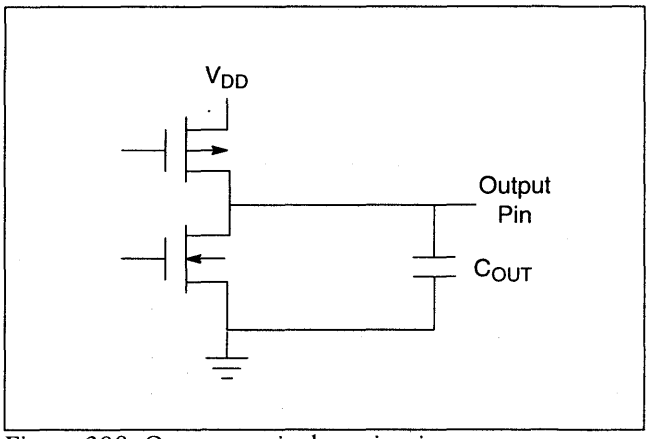



Figure 308. Output equivalent circuits

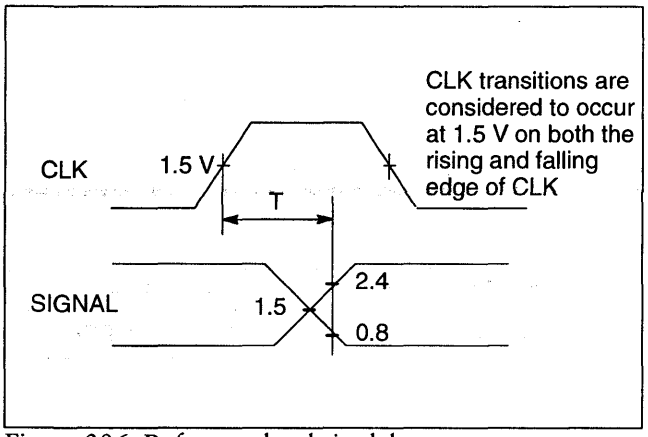



Figure 306. Reference levels in delay measurements on frame buffer controller pins

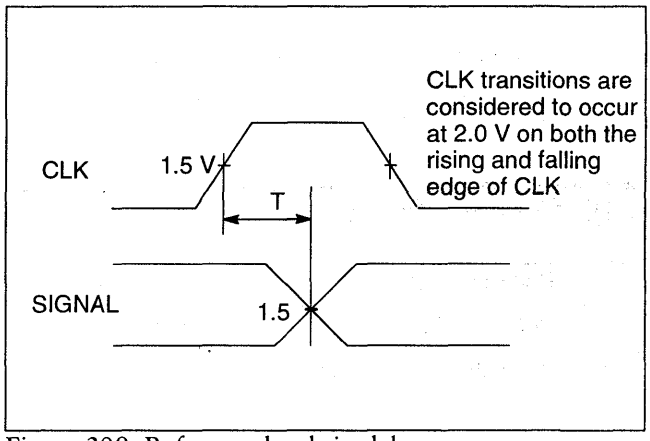



Figure 309. Reference levels in delay measurements on all pins other than frame buffer controller pins

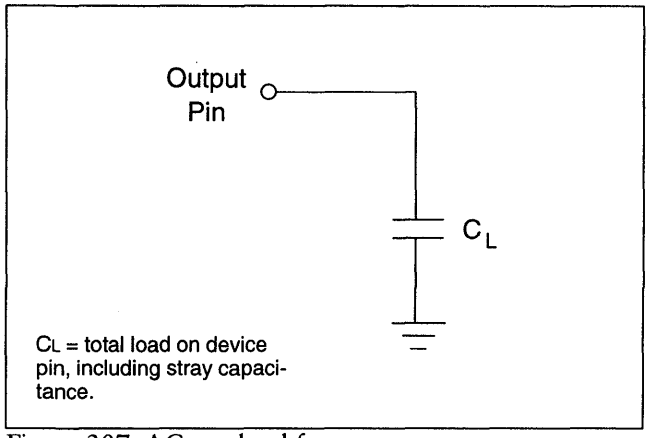



Figure 307. AC test load for test measurement

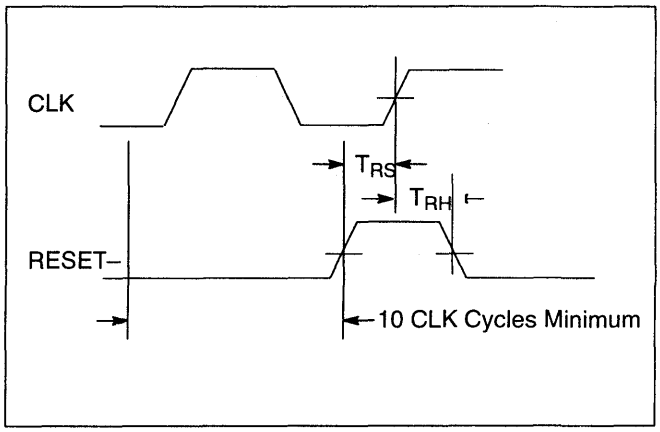
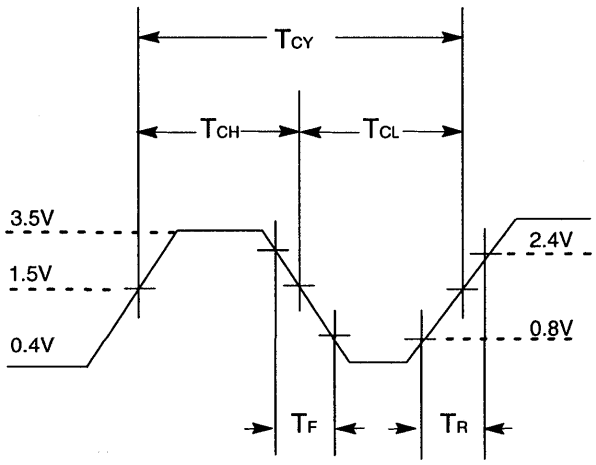



Figure 310. Reset timing

## 13.9. I/O Characteristics , continued




TR and TF are not measured but are guaranteed by design.

Figure 311. Clock timing

## 13.10. Ordering Information

| Package Type | Speed Grade | Temperature Range (Case) | Order Number |
|---|---|---|---|
| 208-pin PQFP | 50 MHz | 0 – 85°C | P9100–050–PFP |

Figure 312. Ordering information

For additional information on WEITEK products, please fill out the form below and mail.

Name _____ Title _____

Company _____ Phone _____

Address _____

Comments _____

I am currently involved in a design with the following WEITEK products _____ and wish to be added to your design data base to insure that I receive status updates.

**APPLICATION:**

- ☐ PERSONAL COMPUTERS
- ☐ ENGINEERING WORKSTATIONS
- ☐ SCIENTIFIC COMPUTERS
- ☐ OTHER
- ☐ LASER PRINTERS
- ☐ TYPESETTERS
- ☐ OTHER PAGE PRINTERS

Check the products on which you would like to receive data books:

| POWER 9x00 SERIES | WORKSTATION | PRINTER |
|---|---|---|
| ☐ Power 9100 Eval. Board App. Note | ☐ W8701 | ☐ XL-8220 Technical Overview |
| ☐ Power 9100 SW Designer's Guide | ☐ W8720 | ☐ |
| ☐ Power 9100 Technical Specification | ☐ W3171 | ☐ |
| ☐ Power 9000 Data Book | | |
| ☐ Power 9000 PCI Application Note | | |

WEITEK use: Red'd Out TPT Source:DS

Status:

## POWER 9100 DATA BOOK (PRELIMINARY DATA)
## (November 1993 Preliminary Data Edition)

Please comment on the quality of this document, and on any corrections or clarifications you would like to see in a future edition.

WEITEK

Fold, Staple and Mail to Weitek Corp.

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

**BUSINESS REPLY MAIL**

FIRST CLASS PERMIT NO. 1374 SUNNYVALE, CA

POSTAGE WILL BE PAID BY ADDRESSEE

**WEITEK Corporation**
**1060 E. Arques Ave.**
**Sunnyvale, CA 94088–9861**

**ATTN: Tech Pubs**

WEITEK

## WEITEK'S CUSTOMER COMMITMENT:

Weitek's mission is simple: to provide you with VLSI solutions to solve your compute-intensive problems. We translate that mission into the following corporate objectives:

**1.** To be first to market with performance breakthroughs, allowing you to develop and market systems at the edge of your art.

**2.** To understand your product, technology, and market needs, so that we can develop Weitek products and corporate plans that will help you succeed.

**3.** To price our products based on the fair value they represent to you, our customers.

**4.** To invest far in excess of the industry average in Research and Development, giving you the latest products through technological innovation.

**5.** To invest far in excess of the industry average in Selling, Marketing, and Technical Applications Support, in order to provide you with service and support unmatched in the industry.

**6.** To serve as a reliable, resourceful, and quality business partner to our customers.

These are our objectives. We're committed to making them happen. If you have comments or suggestions on how we can do more for you, please don't hesitate to contact us.

**Art Collmeyer**
President


**Headquarters**
Weitek Corporation
1060 E. Arques Avenue
Sunnyvale, CA 94086
TEL (408) 738-8400
TWX 910-339-9545
WEITEK SVL
FAX (408) 738-1185

**Domestic Sales Office**
Weitek Corporation
1060 E. Arques Avenue
Sunnyvale, CA 94086
TEL (408) 738-8400
TWX 910-339-9545
WEITEK SVL
FAX (408) 738-1185

Weitek Corporation
1500 West Park Drive, Building 5
Westborough Office Park
Westborough, MA 01581
TEL (508) 366-9030

**European Sales Headquarters**
Hertfordshire Business Centre
Unit 15B
Alexander Road
London Colney
Herts AL2 1JG
United Kingdom
44-72-782-6973

**Japanese Representative**
4-8-1 Tsuchihashi
Miyamae-Ku
Kawasaki, Kanagawa-Pre
213 Japan
TEL (011) 81-44-852-1135
FAX (011) 81-44-888-3158

WEITEK

# POWER 9100 PROGRAMMER'S MANUAL

## PRELIMINARY DATA
July 11, 1994

The Power 9100 Programmer's Manual describes the architecture, addressing modes, initialization, and basic programming approaches for the Power 9100. A large reference section is divided into four parts: device, register, board, and VGA chapters. The Appendix includes BIOS revision history, and a description of the driver configuration file.

The Power 9100 Programmer's Manual is used along side the Power 9100 Data Book and Board Application Note. It is not inclusive, but does provide significant programming information for the Power 9100.

**Contents**

# Power 9100 Programmer's Manual

Version: 0.1 *Preliminary Data*
Revision: 0.03
Revision Date: July 11, 1994

IMPORTANT NOTICE: WEITEK Corporation assumes no responsibility for errors in this document, and retains the right to make changes at any time, without notice. Please contact your sales office to obtain the latest specifications.


1060 East Arques Avenue
Sunnyvale, California 94086
Telephone (408) 738-8400


COVER711.DOC
INSID711.DOC
PMAN0711.DOC

## Trademarks

All trademarks used in this document are property of their respective owners.

## Revision History

July 7, 1994 Version 0.1, Revision 0.01 - Reformatted for ***Preliminary Data*** release
July 11, 1994 Version 0.1, Revision 0.03 - minor changes, updates

# WEITEK'S GRAPHICS CONTROLLERS

Power 9000 Workstation Graphics Accelerator

VRAM based memory controller, High Performance Drawing Engine, and Non-interlaced displays describe the 9000. This technology has been enhanced and integrated with our SVGA logic to create the POWER 9100.

W5286 SVGA with Accelerator

This SVGA chip supports all VGA modes for DOS compatibility.

POWER 9100 Integrated Workstation Graphics and SVGA

The Power 9100 has two operating modes, Emulation and Native Mode. Emulation Mode is VGA mode using 5286 SVGA logic with minor modifications. Native Mode is the Power 9100 operating in a workstation graphics mode similar to the 9000, but with many enhancements:

Enhanced 9000 + SVGA
Integrated PCI and VLBus Interfaces
16, 24, and 32-bit pixel formats
4 MB VRAM (twice the 9000)
1280x1024 x24 bpp
Video Coprocessor Interface (shared display buffer with Video Power & 3D)
50 MHz MEMCLK (up from 33 MHz)

This Programmer's Manual describes the Power 9100 in a Board Design.

# TABLE OF CONTENTS

# PREFIX

This is an early release of the Power 9100 Programmer's Reference Manual. I welcome your suggestions, corrections, and general comments. Please fax, or e-mail:

Robert Embry @ roberte!weitek.com
Weitek Corporation, 1060 E. Arques Ave, Sunnyvale, CA 94086
Tel # (408) 522-7551, Fax # (408) 522-7504

Use this document with the data book. A board application note and misc. tech notes are also available for hardware things.

## DOCUMENTS COMBINED TO CREATE THE PROGRAMMER'S MANUAL

SOFTxxxx.DOC Tech Note (new errata will create a new one)
Power 9100 Initialization Note (new errata in the Windows release notes?)
P9x00RES.DAT specification (new errata in the Windows release notes?)
PU_CONFIG Register Definition (new errata in the BIOS release notes?)

## MISSING ITEMS

Things already in the data book (required reading) or board app note (extra credit)
Compiler information for the DDAKs, Install, etc. (I'm a hardware guy)
Code fragments (look in a DDAK, the board test, or kdebug software)

[complete? accurate?] Caution, this means the information is in DRAFT form. Do not expect it to be complete or accurate.

## RELATED DOCUMENTS

Power 9100 Graphics Controller Data Book -- March 8, 1994
Power 9100/Video Power Board Application Note -- March 1994
Power 9100 Board Test Manual and Programmer's Reference -- (soon)
Power 9100 Graphics Controller Technical Notes -- (hardware stuff)

We expect an updated data book to be back from the press by August.

The Board Test and Diagnostic Software Guide is describes how to configure and operate the Power 9100 Board Test program.

The technical notes are created to describe stuff that may or may not be in the data book, app note, or this programmer's manual. List of some of the tech notes:

| | |
|---|---|
| INTxxxx.DOC | Interrupt connections to PCI and VLBus |
| PHILxxxx.DOC | Errata on the video channel board |
| HARDxxxx.DOC | Board design review & help |
| | new ones created to satisfy important short term issues |

## APPENDICES

The Appendix has a number of important items.

- BIOS Status Report
- P9x00RES.DAT File Format
- Display Problems

## NAMING AND WRITING CONVENTIONS

| | |
|---|---|
| offset 2208h | Native Mode register: the prefix 31..15 is not specified, bits 14..0 are defined only. |
| 0000 0000 | even boundary digits (hex or binary) |
| 000 00 | bits 6..2 are zero |
| [32h]config | same as 32hCONFIG[50] in the data book |
| sysconfig.pixel_buf_read.10..4<br>register_name.field_name.bits | |
| | means bits 10..4 in the pixel_buf_read field of sysconfig |
| N4E-A4 | silicon revision identification. The example refers to an N4E marking (ignoring the L, if it exists) on the top of the device and a 4h read from sysconfig.rev.2..0 {see Device Revision for a special note on reading the sysconfig register.} |
| VIDOUTCLK | is referred to as LCLK in this document. |
| shadow_dac | is also called palette_snoop |
| vga_present | is also called vga_absent_bios_disable |

Emulation Mode and VGA mode terms can generally be used interchangeable. Strictly speaking, Emulation mode is the mode of the chip and VGA mode is Emulation Mode running with the VGA BIOS.

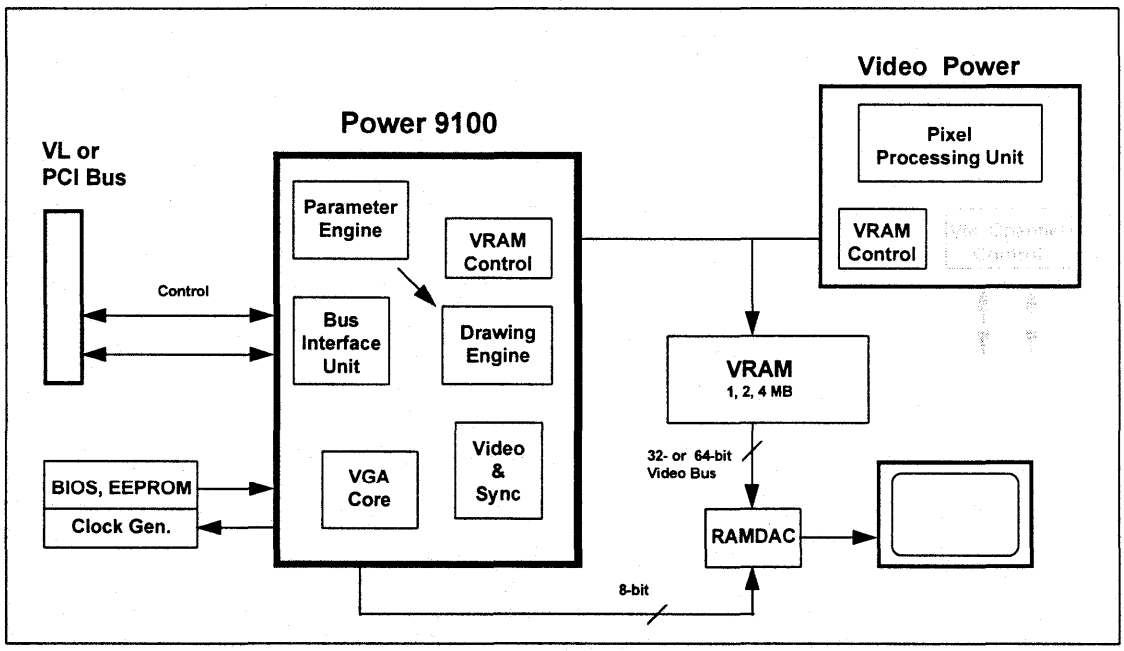
# 1. INTRODUCTION

The Power 9100 is a high performance graphics accelerator. The architecture takes advantage of the speed of VRAMs and high-speed local bus interfaces. It's hardwired controllers write at memory bandwidth and provide a host of features.

The Architecture is based on a memory mapped approach with a parameter engine to setup attributes for the commands and a drawing engine to execute the commands to the display buffer.

The Power 9100 is divided into VGA (or Emulation) and Native Mode. VGA Mode is provided for DOS compatibility and its function is limited to supporting these types of applications. Native Mode is desired operating condition to support accelerated graphics.

This Programmer's Manual describes many Native Mode situations and is a supplement to the Power 9100 Graphics Controller data book.




POWER 9100 BOARD ARCHITECTURE
(see Device & Software Architecture sections for additional diagrams)

# 2. SOFTWARE PRODUCTS

Weitek provides reference designs and software for Windows, NT, OS/2, AutoCAD, & Microstation.

The software for a Power 9100 board can come from many sources. Weitek provides base-level drivers that can be configured in a run-time environment or they can be modified to work with slightly different board devices. We also have working board diagnostic programs.

The Device Driver Adaptation Kit (DDAK) and the hardware diagnostic programs provide initialization and programming examples for creating Power 9100 graphics drivers.

This section identifies which parts of our base-level drivers can be made to accommodate hardware components or to change the look, feel, and operation of the graphics board.

Various levels of modifications can be made, beginning with the simplest and working toward the more complex: It begins with configuration changes that are easily done, progresses to suggest source code modifications to simpler initialization and installation software, and then on to the source code of the main driver.

## 2.1. LIST OF SOFTWARE

Windows 3.1 Driver
Windows NT 3.1 Driver
OS/2 Driver
AutoCAD Driver
MicroStation Driver
DOS Drivers
Board Test Program
Diagnostic Debug Program
BIOS
VESA TSR

## 2.2. BOARD CONFIGURATIONS SUPPORTED

Weitek supports the Power 9100 and Video Power 9130 on the PCI and VL Buses. Our reference designs use the IBM RGB525 and BT485A DACs. We use the ICD 2061A clock generator in all designs except a few RGB525 cases where a clock oscillator provides the reference clock. All boards are either 2 or 4 MB. We use 256kbx8 and 256kbx16 VRAMs with full and half size SAM ports.

We plan to support the AT&T 511 and BT489 DACs this summer. Modifications for these kinds of peripheral chips are usually done at the DDAK level. We also have plans to provide example code for our 3D Power Coprocessor.

## 2.3. CONFIGURABILITY USING P9X00RES.DAT FILE & PU_CONFIG

The Power 9100 graphics drivers and BIOS initialize the board using information from a power up configuration register call pu_config and a text file called "res dot dat" or P9x00RES.DAT. The P9x00RES.DAT file is normally customized by the OEM using a text editor to create a driver diskette that ships with the board. The pu_config register is configured with external pull up resistors. A utility

program called EDITBCT can change the video boot banner and the OEM string, both embedded in the BIOS binary image. Refer to the BIOS release notes for information on EDITBCT.

The Appendix includes a review of the P9x00RES.DAT file. The pu_config register is explained in the General Reference Section of this document. The EDITBCT program is described in Appendix D.

The Power 9100 driver development has moved toward auto detection by reading the pu_config register, reading other device registers (or trying to), and by performing tests. The notable exception to this is the Board Test which depends on an older version of the P9x00RES.DAT file. The Board Test uses table driven initialization techniques where most of the other programs and drivers use an algorithm method.

## 2.4. DLL SOFTWARE

The Power 9100 drivers are modularized. A Dynamically Linked Library (DLL) of functions initialize and control the board hardware. The DLL is usually written to work with a set of boards.

The DLL initializes, configures, and controls peripheral chips such as the clock synthesizer and DAC. The source code for a DLL is included in the Device Driver Adaptation Kit (DDAK).

For boards that contain unsupported hardware components, a programmer will need to modify and compile a new version of the DLL. Simple DAC or clock synthesizer changes can be handled by modifying the DLL code.

## 2.5. WINDOWS DRIVER INSTALLATION SOFTWARE

The Windows Driver installation program copies files, accepts user options, and modifies INI files. The install program reads the P9x00RES.DAT file, the pu_config register, and performs some simple tests on the system. It then displays a menu for user input. After completion of this, the installation program modifies or writes to the P9x00RES.INI and SYSTEM.INI files.

The installation software can be modified to change the look, feel, and operation of the installation program. The installation software is distributed in source form in the DDAK.

## 2.6. DEVICE DRIVER ADAPTATION KIT (DDAK)

A DDAK includes source and object code to enable the driver to be adapted to a specific board and to allow for customization of the install program. A DDAK is available for each operating system. The Windows code is written in ix86 assembly. The Windows NT and OS/2 code is mostly written in C, but also contain some assembly.

The DDAK includes the following software:

INSTALL -- SOURCE
The source to the install program allows an OEM to customize the User Interface and create or restrict the use of options in the P9x00RES.DAT file.

DLL -- SOURCE
The Dynamic Linked Library (DLL) supports initialization and configuration code for the controllers in the Power 9100, the pixel clock synthesizer (external), the DAC (external), and any other external hardware. Board specific device code is part of the DLL.

DRIVER -- OBJECT
The driver supports the GUI by calling the DLL, controlling the Power 9100 accelerator, and accessing the display buffer directly. The drivers are available in object code version and can be linked to recompiled DLL files.

VDD -- OBJECT
The Virtual Device Driver supports DOS boxes (virtual DOS display) and must switch the Power 9100 between Native and VGA Mode to support DOS full screen.

## 2.7. DRIVER SOURCE CODE

In some situations, it is necessary to modify the source code to a driver. Additional features and/or hardware is added by board manufactures to improve the look, feel, and operation of their products. These changes sometimes need appropriate support within the driver.

Consider working with the driver source code only after determining that the standard driver cannot be configured using the P9x00RES.DAT file or the DDAK software. Under special consideration, Weitek provides source code to our Windows 3.1 and Windows NT 3.1 drivers.

## 2.8. BOARD DIAGNOSTIC SOFTWARE

There are two diagnostic or test programs available in run-only and source code form.

KDEBUG is the most up to date. It is an engineering development tool.

BOARD TEST uses older initialization techniques, but has interactive and run only modes. it also includes a users guide and a listing of the procedures.

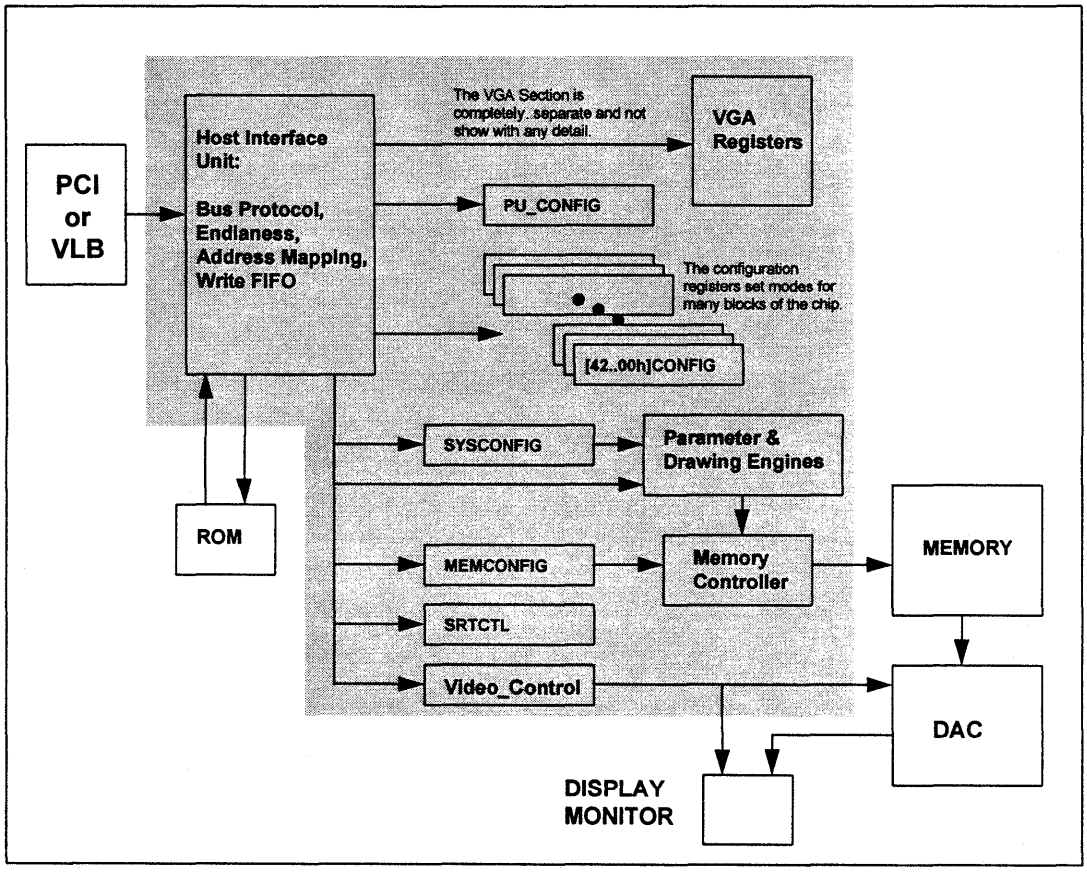
## 2.9. WRITING DRIVERS

New drivers, from scratch, are required for embedded applications like X Servers or for programs running on non-ix86 microprocessors like the Power PC or MIPS. In this case, begin with the OS/2 DDAK or KDEBUG and thoroughly review the data book and this programmer's guide. The Board Test Program is another program distributed in source format.

# 3. POWER 9100

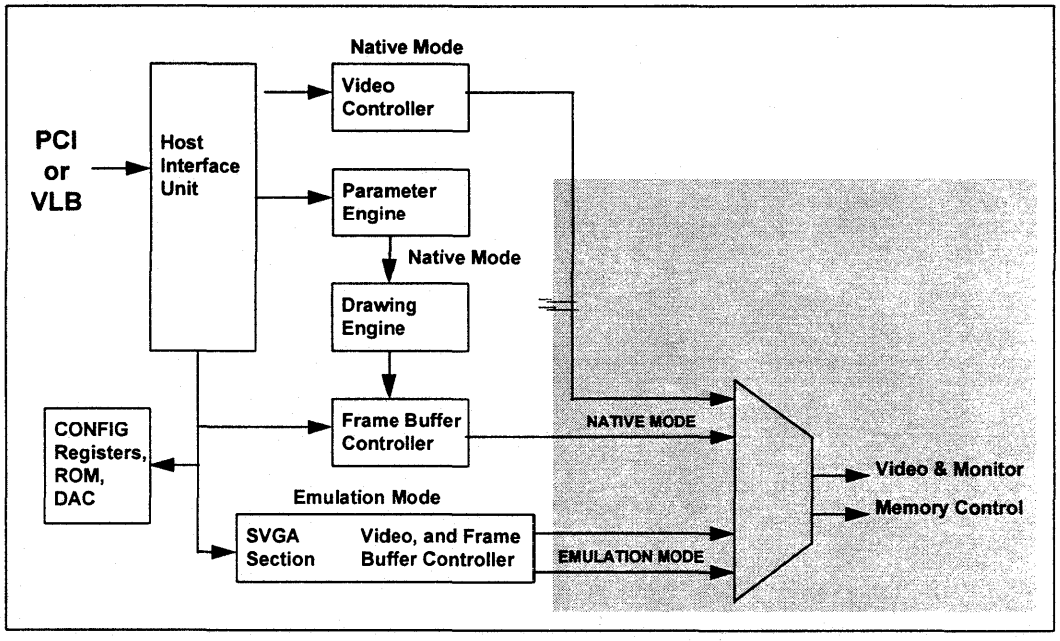
## 3.1. DEVICE ARCHITECTURE

The data book describes the chip architecture in detail. Here is a Control Flow diagram for the Power 9100.




SOFTWARE CONTROL FLOW

The Power 9100 is made up of two chips, a VGA core and a workstation graphics accelerator. Integrated muxing logic switches between the two. Here is simple diagram showing the two sources for monitor and memory control.



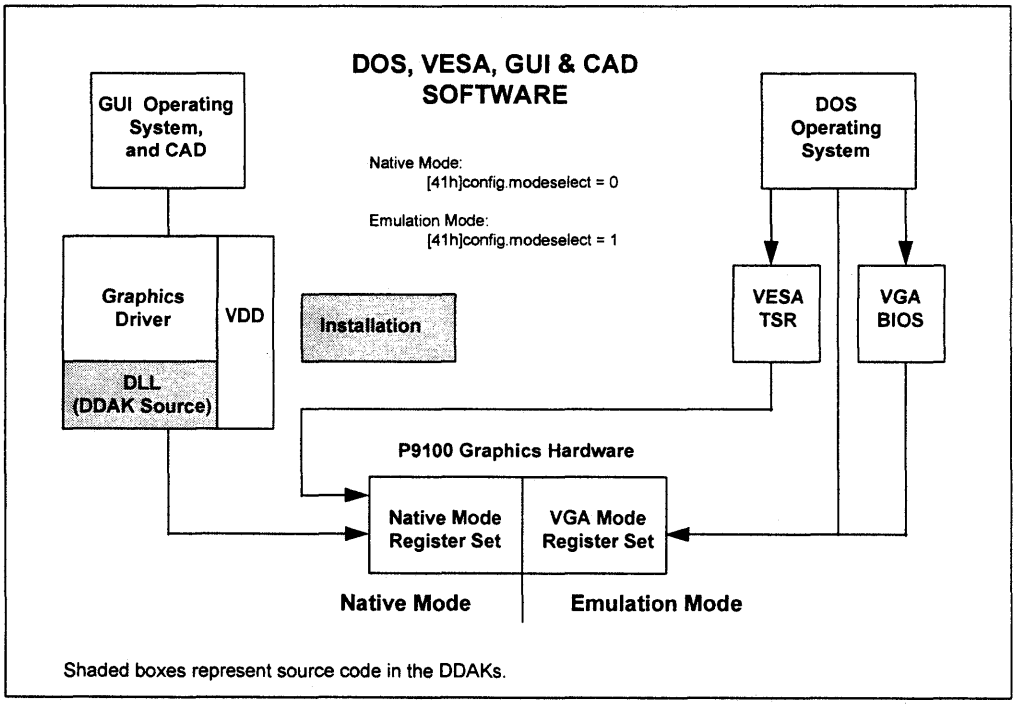

DUAL MONITOR AND MEMORY CONTROL

## 3.2. SOFTWARE ARCHITECTURE

The Power 9100 is operated in Emulation Mode to support VGA compatibility. The VGA BIOS code is embedded in ROM.

The VESA modes are supported in a Terminate and Stay Resident (TSR) program and operates the Power 9100 in Native Mode.

The GUI operating systems and CAD applications also use the Power 9100 in Native Mode. The drivers for these include the code to initialization the Power 9100 for Native Mode and to switch between Native Mode and VGA Mode.




SOFTWARE ARCHITECTURE

## 3.3. IDENTIFYING SILICON REVISION

This programmer's guide discusses two silicon revisions as described below. A summary of the silicon anomalies are listed in Appendix B.

| Reference | Mask | Label(s) | sysconfig.id.2..0 |
|---|---|---|---|
| N4C-A2 | A2 | N4C or N4CL | 2h |
| N4E-A4 | A4 | N4E or N4EL | 4h |

Before you start initializing the Power 9100 in Native Mode you can read the revision level as follows:

```
// You must write something into SYSCONFIG before you can
// read the revision level
//
*P9100_SYSCONFIG = 0x00000000;
revision = *P9100_SYSCONFIG & 0x00000007;
switch(revision)
    {
    case 0x02:
            printf("You have N4C-A2 silicon\n");
            break;
    case 0x04:
            printf("You have N4E-A4 silicon\n");
            break;
    }
```

The *SYSCONFIG* register must be written before the revision level can be read. If you always read back revision level zero, then please inspect your code. If you read back something besides a 0, 2 or 4 then contact Weitek.

# 4. SYSTEM ISSUES

## 4.1. POWER UP

The Power 9100 typically powers up in Emulation mode in anticipation of booting in to a VGA compliant state using BIOS code. A pull up resistor is required on MD[26] to set pu_config.init_modeselect.26 =1 to enable this.

Power up state of registers that effect VGA Mode

| Registers | PCI | VLB | |
|---|---|---|---|
| [04h]config.palette_snoop.5 | 1 | 1 | N4C-A2 silicon |
| | 0 | 1 | N4E-A4 silicon |
| [04h]config.mem_enable.1 | 0 | 1 | |
| [04h]config.io_enable.0 | 0 | 1 | |
| [33..31h]config.rom_base | 000C | 000C | (bit 15 = 0, also) |

[complete? accurate?]

There are no known differences between the power on state and cycling RESET.

## 4.2. DPMS

There are two hardware mechanism to support DPMS, one for Native Mode, the other for VGA Mode. These are discussed separately in the reference sections of this manual.

## 4.3. INTERRUPTS

### 4.3.1. Use of Interrupts

The P9100 and the P9130 each have an interrupt output signal. Weitek software drivers only use the interrupt capability of the P9130. The interrupt pin on the P9100 is provided for those that want to use it with their own software.

The P9130 interrupt line is connected to IRQ9 (VLB design ) for our software to work. In a PCI system, it is connected to INTA#, but in some silicon revisions it is not recognized by the System PCI BIOS as described below. In a development environment, it is possible to enable INTA# for the P9130 by programming the system chip set.

The Video Power software maintains a command queue that is interrupt driven for playback and for frame capture in applications that include the Philips 7196 device.

### 4.3.2. PCI Bus Interrupts

The PCI specification requires two registers in the Configuration Space Header to support interrupts. They are called the "Interrupt Pin" and "Interrupt Line" registers, offsets 3Ch and 3Dh. The interrupt line register specifies connection to INTA#.

**Product Status Report**

| |
|---|
| N4C-A2<br>The P9100 N4C-A2 silicon mask does not included these registers so it is impossible to support PCI interrupts per the PCI specification. |

| |
|---|
| N4E-A4<br>The P9100 N4E-A4 silicon includes the interrupt pin and line registers so the P9100 or P9130 get their INTA# interrupts assigned to an IRQ line. |

## 4.4. DUAL BOARD SYSTEMS

Hardware jumpers on MD[25] and MD[26] control the initial mode select and vga_absent_bios_disable state for the PCI configuration (or config) register.

To enable a board NOT to boot VGA, and not be defined as VGA device, pull MD[25] high, but let MD[26] settle low.

| | VGA Board | Non-VGA Board | |
|---|---|---|---|
| pu_config.init_modeselect.26 | 1 (VGA mode) | 0 (native Mode) | |
| pu_config.vga_absent_bios_disable.250 | (VGA device) | 1 (no VGA) | see Note, below. |

Note: The pu_config.vga.vga_absent_BIOS_disable field is for the PCI bus mode only. It has the BIOS disable mode for BIOS shadowing VL Bus mode (for N4E-A4.)

# 5. PROGRAMMING NOTES

(1) Emulation (VGA) Mode and Native Mode are mutually exclusive. Switching to one mode will reset the other.

(2) Do not read from undefined registers. This may hang the Power 9100. An access to offset 0000h is reported to hang the system because the Power 9100 asserts DEVSEL#, but not LRDY#.

(3) The data book states that some register bits are reserved and must be set to 0. In some cases, these bits are don't cares. In all cases, it is best to write a 0 and don't assume anything on read back.

# 6. ADDRESSING MODES

The addressing modes that are available at any given time is dependent on the mode of the Power 9100 and the setting of various bits. Some addressing modes are only available in Native Mode, others only available in VGA Mode, and others that are available in both modes, but may or may not be available due to the bus type or disabling by mode bits.

| Logic | Accessibility |
|---|---|
| Config Registers | PCI: Always Configuration Space |
| | VLB: Always I/O 9000h & 9004h, etc. |
| Native Registers | Native Mode Only |
| Display buffer | Native Mode Only |
| ROM | Native or VGA Mode |
| DAC | Native Mode |
| DAC | VGA Mode |
| VGA Registers ** | VGA Mode Only |
| VGA Memory ** | VGA Mode Only |

** VGA Registers and Memory make up a complete compatibility specification that is in not included in this document.

## 6.1. CONFIG REGISTERS

The Power 9100 supports the required PCI Configuration Space Header registers. These registers can be read as 8, 16, or 32-bit quantities [complete? accurate?]. Bytes appear on the byte lane defined by the lowest 2 address bits.

The index pointer and data registers for VLBus mode, are always accessed as 32-bit registers.

The Power 9100 extends this register set to include [40h]config, [41h]config, and [42h]config Registers. All these registers are in PCI Configuration Space when PCI Bus Mode is in effect. All these registers are indexed at 9000h when VLBus Mode is in effect. The location of the I/O mapped index/data registers in VLBus Mode can be offset slightly by pu_config.cfgba.29..27.

## 6.2. NATIVE REGISTERS

The Native Registers include many types of registers. Native Registers must always be accessed as 32-bit words. Byte and 16-bit word accessing is not supported.

The following are the groups of Native Mode registers and some specific registers included in the group:

| Group | Specific Registers in the Group |
|---|---|
| System Control | sysconfig, interrupt, alt_read_bank |
| Video Control | hrzt, vrtt, srtctl |
| VRAM Control | mem_config, rfperiod, pu_config |
| DAC Control | DAC registers |
| Video Coprocessor | Video Power, 3D Power registers |
| Drawing Engine | color, plane_mask, raster, pattern, window |
| Parameter Engine | device_coordinate, window parameters, status |
| Command | quad, pixel1, blit |

Not all memory locations in the Native Register memory space are used. Unused address locations must be avoided for reliable system operation.

## 6.3. DISPLAY BUFFER

The display buffer is either 1, 2, or 4 MB in size and is a contiguous block of memory. The display buffer is map-able in a number of ways and is supported with 8, 16, and 32-bit accesses. This is described in the Native Mode Addressing section.

## 6.4. DAC

In all designs, the DAC is physically connected to the third byte lane of the VRAM data bus, MD[23..16]. DAC Control is a Native Mode register and must be accessed as a 32-bit quanity. Only the 8-bits of data on the third byte lane is meaningful.

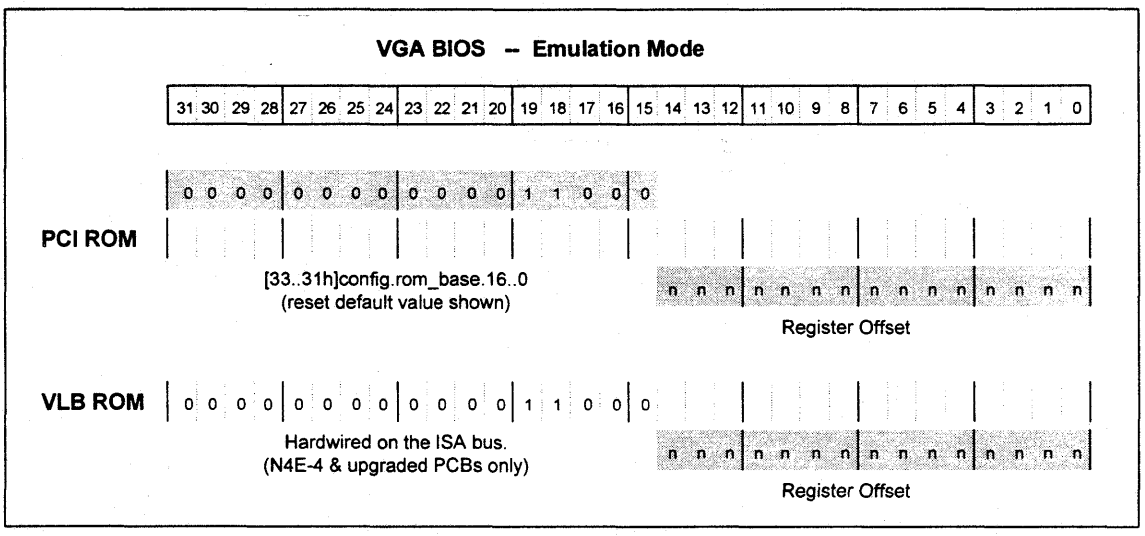
## 6.5. CLOCK GENERATOR

The clock synthesizer is programmed via a [42h]config.CKSEL.1..0. The config register is accessed like all the other config registers.

## 6.6. ROM

In PCI designs, the ROM is physically connected to the upper byte lane of the VRAM data bus, MD[31..24]. []config.rom_base16..0 register holds the base address.

In VLB designs, the ROM response is dependent on the board design and the silicon revision. The diagram below shows the ROM on ISA with N4D-4 silicon.




ROM ADDRESSING

[30H]config.rom_enable. = 1
[04h]config.mem_enable. = 1

### Product Status Report

N4C-A2

| |
|---|
| The N4C-A2 silicon interfaces the ROM onto the VLBus. This revision of the chip nor the board design support BIOS shadowing. |

N4E-A4

| |
|---|
| New board designs that use N4E-A4 have the ROM on the ISA bus. The ROM is hardwired at 000C 0000 to 000C 7FFF. Old board designs, with the ROM on the VLBus always operate like N4C-A2 silicon. |

## 6.7. VIDEO AND 3D POWER COPROCESSOR

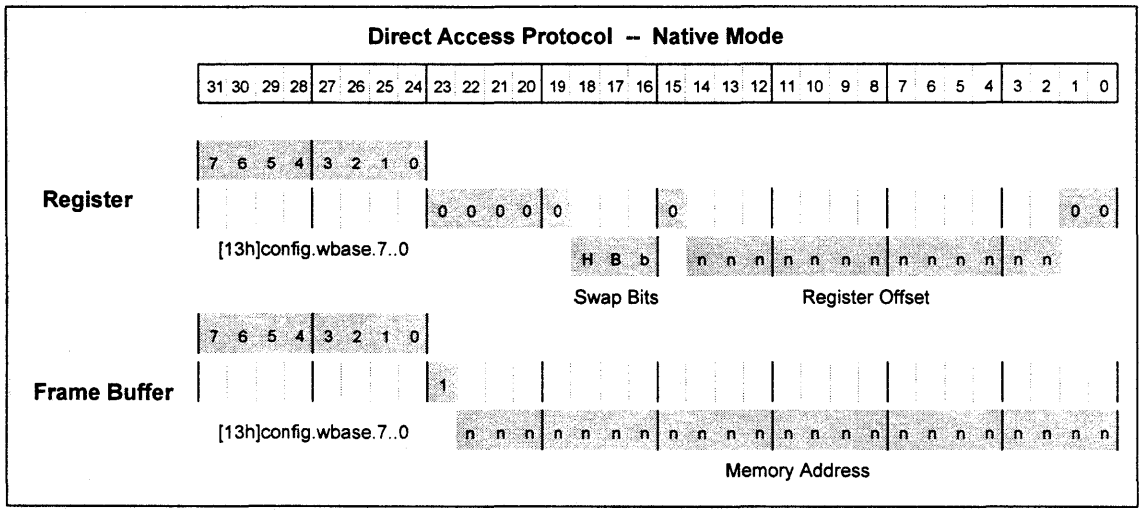
These devices are addressed as a Native Mode register.

# 7. NATIVE MODE ADDRESSING

| | |
|---|---|
| Config Registers | PCI: Always Configuration Space<br>VLB: Always i/o 9000h & 9004h, etc |
| Native Register | Direct or In-direct Access Protocol |
| Display buffer | Direct or In-direct Access Protocol |
| ROM | Base address set to 000C 0000h on reset. |

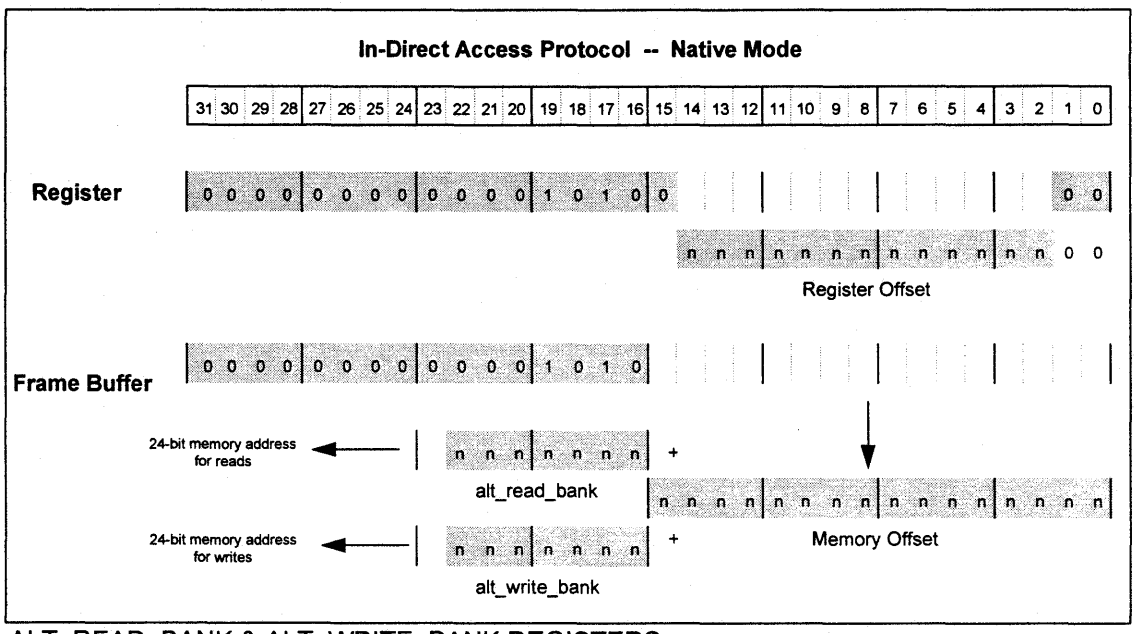
## 7.1. DIRECT ACCESS PROTOCOL

Direct Access Protocol maps all the Native Mode Registers and memory into a linear address.

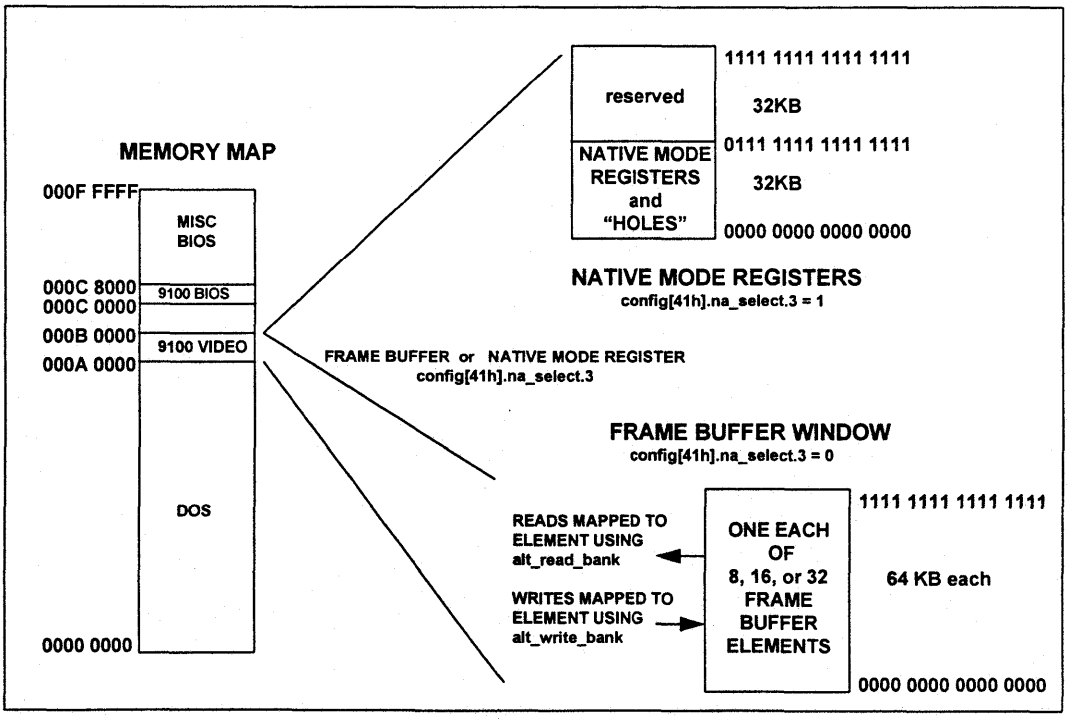


## 7.2. INDIRECT ACCESS PROTOCOL

Indirect Address Protocol maps the Native Mode registers to 000A 0000 to 000A FFFF.




ALT_READ_BANK & ALT_WRITE_BANK REGISTERS




MEMORY MAPPING WITH ALT_READ_BANK & ALT_WRITE_BANK

## 7.3. ENDIANESS

The Power 9100 supports bit, byte, and 16-bit word swapping. All software programmers must be aware of swapping. Swap byte and 16-bit word allows the Power 9100 to work easily with Little and Big Endian microprocessors and system buses.

Setting a swapping bit to a 1 causes the swap to occur, a 0 does not swap.

If H & B are set to 1s, the bytes in a 32-bit quanity are swapped in the classic Big Endian/Little Endian fashion.

### 7.3.1. REGISTER ENDIANESS

The data book describes the internal bit assignments as they appear on the Host Interface with no swapping. Since the display buffer is Big Endian (see next section), the Native Registers associated with the display buffer data must be written in Big Endian mode.

Big Endian Registers (all others are Little Endian)
Color[3:0]
Pixel1 data
Pixel8 data

### 7.3.2. DISPLAY BUFFER ENDIANESS

The display buffer operates in Big Endian Mode. The Power 9100 displays MD[31..24] first, followed by MD[23..16], MD[15..8], and MD[7..0]. The accelerator always displays pixels in this format when operating from left to right.

In 64-bit SAM bank designs, the data is used in controller memory bank order: 0, 1, 2, and 3.

### 7.3.3. ENDIAN CONTROL BITS

There are 3 places to control endianess, depending on the operation.

Direct Access Protocol
- Native Registers -- upper address bits 18..16
- Display buffer -- sysconfig13..11

Indirect Access Protocol
- Native Registers -- [41h]config.H,B,b
- Display buffer -- sysconfig13..11

# 8. INITIALIZATION

## 8.1. PU_CONFIG

Initialization begins with the deassertion of RESET#. This action causes the pu_config register to be initialized to the state of the MD bus. The Power 9100 has weak internal pull down resistors on this bus. External pull up resistors (with or without jumpers) are attached to individual MD signals to create a 1 in the pu_config register. The definition of these bits are in the Reference Section of this document.

If the Power 9100 boots VGA then the chip is in Emulation Mode. The BIOS programs the clock, DAC, and the Power 9100 to be VGA compliant.

For Native Mode, the software sets up more configuration registers. If not already ticking, the software must obtain a stable MEMCLK and pixel clock selected by mem_config.video_clk_sel.20 before accessing the Video Control Registers. In some cases nothing must be done, in other cases a clock generator and the video_clk_sel bit must be programmed.

The memory and video controllers are initialized to get a display. The Drawing and Parameter Engines are initialized to display graphics.

## 8.2. MEMORY CONTROLLER

The Memory Controller reads and writes data for the accelerator and host CPU. It also is responsible for refreshing the memory, servicing Video Controller requests for a split shift register transfer, and granting the bus to the Video Coprocessor.

The memory controller is programmed using the mem_config register. Details are described in the Native Mode Register Reference section of this document and in the data book.

Simple memory tests determin the existance and size of the memory.

```
// Code to size memory using Direct Access Protocol.
//
```

## 8.3. VIDEO CONTROLLER

The Video Controller maintains the CRT display by controlling the shift registers in the VRAMs, the clocking of pixels into the DAC, and the monitor timing signals.

Do not read any of the Video Control Registers if there is not a stable clock on the selected pixel clock input (pixclk/divpixclk). This will hang the Power 9100.

## 8.4. VRAM CONTROLLER

to be done.

## 8.5. OTHER INITIALIZATION

The mem_config.soe_mode should be set to 10b whenever the Power 9100 is in VGA Mode. If this is not done, contending serial enables may be asserted at the same time. This results in high power consumption and may shorten the life of the VRAMs.

# 9. ACCELERATOR OPERATION

The Graphics Accelerator is split in to two sections, the Parameter Engine and the Drawing Engine. These two work in parallel to provide a two stage level of graphic command pipelining.

The operation of the Accelerator is described in the data book.

The Power 9100 accelerator can be used once the chip has been initialized and the display buffer is accessible and displayable.

The Parameter Engine is set with attributes for each command. The Drawing Engine contains attributes that are generally common to a command stream.

The data book describes the architecture of the Drawing and Parameter Engines. A few clarifications are in order.

The Status Register is read at offset 2000h. The Status Register is also returned when reading command registers.

The values in the Status Register are updated quickly after a register write. There have been no reported delays of an update to an immediate Status Register read following a parameter or command register write.

The Power 9100 has a powerful color expansion feature between the host and the display buffer. Monochrome images should be cached in main memory & blitted to the Power 9100 using the PIXEL1 command.

## 9.1. PARAMETER ENGINE

Refer to the data book

PIXEL/BYTE WINDOWS
Load the pixel_window_min and the byte_windows_min appropriately:

```
byte_window_min = pixel_window_min * Bpp
byte_window_max = [(pixel_window_max + 1) * Bpp] -1

        Bpp = Bytes per pixel (1, 2, 3, or 4)
```

## 9.2. DRAWING ENGINE

DEVICE_COORDINATE
STATUS
CONTROL & CONDITION

Refer to the data book.

## 9.3. ISSUEING ACCELERATOR COMMANDS

Refer to the data book on how to use the status register and command registers.

The status register is very responsive. No known delays in status updates have been recorded.

---

TIP: The Blit Command is faster than the Quad command for read-modify-write operations. This is because the Blit Command uses the 8 element queue on the VRAM interface to reduce bus turn around and row misses.

---

---

TIP: A second PIXEL8 command was added to the Power 9100 so the CPU can use the move string instruction for faster CPU execution.

---

# 10. DEVICE REFERENCE

## 10.1. I/O MEMORY MAP

| | | |
|---|---|---|
| VGA Registers | 3C3h, 3C4h, etc | io mode enable |
| Weitek-Specific VGA Regs | 3C5h[12..10h], etc | lock/unlock |
| CONFIG | 91xxh | CFGBA: always and only VLB |
| CONFIG | Config Cycle | PCI Compliant |

## 10.2. MEMORY MAP

| | | |
|---|---|---|
| VGA display memory space | | |
| Emulation Mode: | A0000-BFFFF | VGA compliant |
| Native Mode: | A0000-AFFFF | 41h.Config[65] control<br>Mappable to FB & Native Registers |
| BIOS ROM | C0000-C7FFF | Reset condition: VGA compliant |
| | Any 128 KB boundary | Native or VGA Mode: PCI or VLB, [33..31h]config.rom_base |
| Linear Display buffer & Native Mode Registers | | |
| | Any 16 MB boundary | |

## 10.3. VGA & WEITEK-SPECIFIC VGA REGISTER

| | Accessiblity | | | Comments |
|---|---|---|---|---|
| | io mode en | enable | unlock | |
| 94h | yes | x* | x | *motherbrd, pu_config.add_in_brd.0 = 0 |
| 102h | yes | x | x | |
| 3BAh | yes | yes | x | |
| 3B5h[1D..19] | yes | yes | yes | monochrome mode only |
| 3C0h, 3C1h | yes | yes | yes | |
| 3C2h | yes | yes | x | |
| 3C3h | yes | x | x | |
| 3C4h | yes | yes | x | |
| 3C5h[4..0] | yes | yes | x | |
| 3C5h[12..10,7..5] | yes | yes | yes | |
| 3C7h | yes | | | |
| 3CAh | yes | | | |
| 3CCh | yes | | | |
| 3CD | yes | | | |
| 3DAh | yes | | | |
| 3D5h[24,1D..19] | yes | yes | yes | |
| 46E8h | yes | x | x | |

[complete? accurate?]

io mode enable — means the register is accessible if the master VGA register enables are set. None of the VGA Registers are accessible until otherwise:
41h.config[65].modeselect.1 = 1
04h.config[4].io_enable.0 = 1

enable — means the io mode must be enabled and the following bits must be set according to the application

| Add-in Board | MotherBoard |
|---|---|
| 46E8h, bit 4 = 0 | 3C3h, bit 0 = 1 |
| 46E8h, bit 3 = 1 | 102h, bit 0 = 1 |
| 102h, bit 0 = 1 | 94h, bit 5 = 1 |

unlock — means access to the register controlled by 3C5h, Index 11, bit 5. See Unlocking and Locking Weitek Specific VGA Registers.

## 10.4. ROM

The ROM base address can be set on any 32KB boundary using:

Config[33..31h].rom_base[16..0].

## 10.5. CONFIG REGISTERS

CONFIG[42..0h]

These are PCI compliant registers that are also mapped to I/O space for VLB Bus mode. The three Weitek-Specific PCI registers, Config[42..40h] provide status on bus type, CKSEL[2] pin level, Native/VGA Mode select, mapping of A0000-AFFFF in VGA Mode, and control of VCEN# and CKSEL[2..0] pins.

### 10.5.1. VLB Bus

The I/O location for the Data & Index registers used to access the config registers is specified by pu_config.cfgba[29..27]. Set the Index Register (usually at I/O location 9100h) to the hex value (not decimal value) of the desired config register. Use the Data Register (usually at I/O location 9104h) to read and write the config register data.

### 10.5.2. PCI Bus

Configuration Space Header accesses use IDSEL# signal pin. Enable configuration bus cycles and append the offset register value to the PCI device #. Example: if the Power 9100 is PCI device 0Eh, then the device ID register is 0E03h and OE02h in Configuration Space.

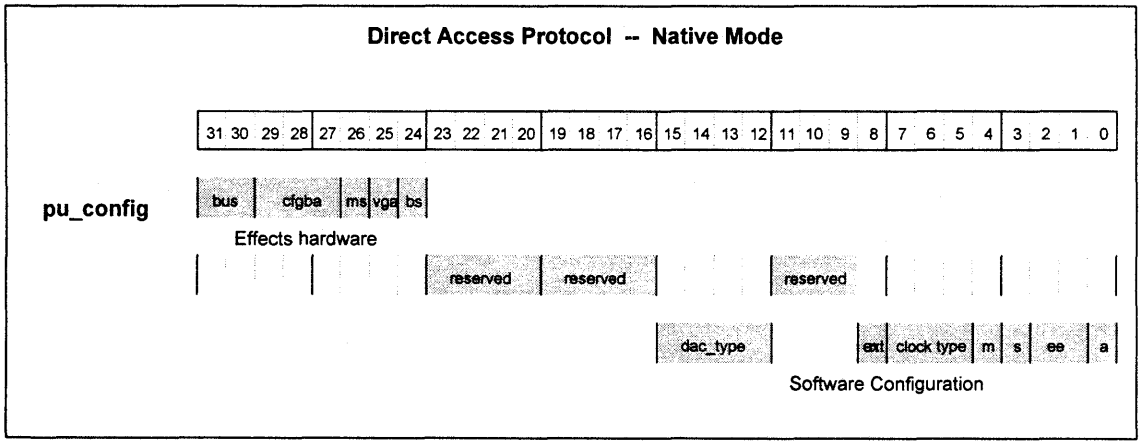
## 10.6. PU_CONFIG REGISTER

This read only register is initialized by the state of the MD[31..0] bus at the desassertion of the RESET# signal pin. The bus has weak pull downs. Resistors with or without jumpers are added to create a 1.

The pu_config register is read only. Some of the fields are copied to other registers for easy access since the pu_config register itself is only accessible in Native Mode at offset 198h (VRAM Control Group.)

Changing any of these fields will change the operation of the (hardware) or the (software).

CONTACT WEITEK IF CHANGES ARE NEEDED.
CONTACT WEITEK FOR THE LATEST DEFINITIONS.




PU_CONFIG REGISTER

| | | |
|---|---|---|
| ***pu_config.bus.31..30*** | 00 | reserved |
| (hardware) | 01 | PCI Bus |
| [40h]config.7..6, read only | 10 | VL Bus |
| | 11 | reserved |

***pu_config.cfgba.29..27***
(hardware)
[40h]config.5..3, read only

I/O Addresses for index/data to config[42..00h] registers in VLB mode only

| | | | | |
|---|---|---|---|---|
| 000 | Index @ | 9100h | Data @ | 9104h (most common) |
| 001 | | 9108h | | 910Ch |
| 010 | | 9110h | | 9114h |
| 011 | | 9118h | | 911Ch |
| 100 | | 9120h | | 9124h |
| 101 | | 9128h | | 912Ch |
| 110 | | 9130h | | 9134h |
| 111 | | 9138h | | 913Ch |

| | | |
|---|---|---|
| ***pu_config.init_modeselect.26*** | 0 | Native Mode |
| (hardware) | 1 | VGA Mode |
| [41h]config.1, read/write | | |

***pu_config.init_vga_absent_bios_disable.25***
(hardware)
[0Ah]config.7, read/write

In PCI Mode, this bit affects the PCI Class Code. VGA will still be available if this bit is set, but it will not be the PCI boot device.

In VLB Mode, this bit has another definition that is dependent on the revision of silicon. The BIOS must be physically connected to the desired bus. N4C-A2 silicon only supports the BIOS on VLB where BIOS shadowing is not supported. Designs with N4E-A4 can support BIOS shadowing if the BIOS is connected to the ISA bus and pu_config.init_vga_absent_bios_disable.25 is pulled high.

| | | |
|---|---|---|
| PCI Mode | 0 | VGA is present in VGA mode |
| | 1 | VGA is absent in VGA mode |
| VLB Mode, N4C-A2 | 0 | BIOS enabled on VLB |
| | 1 | BIOS enabled on VLB |
| VLB Mode, N4E-A4 | 0 | BIOS enabled on VLB |
| | 1 | BIOS disabled on VLB, ISA enabled |
| VLB Mode, future | 0/1 | not used |

***pu_config.init_bios_shadow.24***
(hardware)

| | | |
|---|---|---|
| PCI mode | 0/1 | not used |
| VLB mode | 0 | BIOS enabled on VLB |
| | 1 | BIOS disabled on VLB, ISA enabled |
| | | This is required for compatibility with future Weitek controllers. |
| ***pu_config.reserved.23..16*** (software) | | Reserved, do no use these bits. They cannot be used with certain types of DACs and are now reserved for future use. If we can resolve the DAC issue, then these will become available in the future. [The RGB525 drives its data bus during RESET so pu_config.reserved.23..16 get loaded with 1s.] |
| ***pu_config.dac_type.15..12*** (software) | 0000 | DACs supporting ID in dac_status_register.7..4. Not all DACs are supported. |

| | |
|---|---|
| 0000 | PX2080 Pixel Semi MediaDAC |
| 0010 | BT485A |
| 0011 | reserved for unannounced product |
| 0100 | BT484 |
| 10xx | BT485 |
| 1100 | ATT20C504 |
| 1101 | ATT20C505 |

*NOTE: Software does not necessary look at the DAC status register. If you are considering using a BT485 compatible DAC with additional features, please contact us first so we can determine fi the DAC status register alone will be enough to determine your DAC's unique requirements.*

0001 BT489 (Brooktree's 64-bit BT485 with extensions)
*Call for support status.*

0010 ATT20C510/511 (ATT's 64-bit)
*Call for support status.*

1000 RGB525 (64-bit, w/1 PLL)
*Supported in our drivers.*

| | | |
|---|---|---|
| ***pu_config.reserved.11..9*** (software) | 000 | Set to zero, Do not put pull ups on these. |

*NOTE: Is there anything we missed that you need? If so, please contact Weitek and we can work something out!*

| | | |
|---|---|---|
| ***pu_config.external_io.8*** (software) | 0 | External I/O registers not connected to DAC interface. |
| | 1 | External I/O registers are connected to the DAC interface, accessible by setting CKSEL[2] and accessing 3C9..6h. Read 3C6h.3..0 with CKSEL[2] set to a 1 to identify board:<br>0000 reserved, no id<br>0001 assigned<br>0010 assigned<br>0011 open, call Weitek<br>. . . .<br>1101 open, call Weitek<br>1110 assigned<br>1111 reserved, no id |
| ***pu_config.clock.7..5*** (software) | 000 | ICD2061A, ICD9161 or compatible<br>DAC generates all clocks |
| | 001 | Fixed 50 MHz MEMCLK & RGB525 Ref clock<br>DAC generates pixclk |
| ***pu_config.mem_depth.4*** (hardware) | 0 | 256kbx8 or 256kbx16 VRAMs |
| | 1 | 128kbx8 VRAMs |
| | | This is used in VGA mode only to scale the memory size. |
| ***pu_config.sam_size.3*** (software) | 0 | Full-size shift registers |
| | 1 | Half-size shift registers |
| ***pu_config.eeprom.2..1*** (software) | 00 | AT24MEL 128x8 or compatible |
| ***pu_config.add_in_board.0*** (hardware) | | VGA I/O port Enable (see VGA & Weitek Specific VGA Register section) |
| | 0 | Motherboard application. |
| | 1 | Add-In Board application |

# 11. NATIVE MODE REGISTER REFERENCE

Always use 32-bit wide data words when reading or writing native mode registers.

## 11.1. SYSTEM CONTROL GROUP

**offset 0000h**

sysconfig
interrupt (see data book)
interupt_en (see data book)
alt_read_bank (see Indirect Access Protocol)
alt_write_bank (see Indirect Access Protocol)

**Product Status Report (ref # 028)**

N4C-A2 & N4E-A4

Due to a problem in the silicon, all System Control Group Register writes must be preceeded by a Display buffer read with address bits 14..7 matching the System Control Group address.

Read the Display buffer at 1000 0000 0000 0000 0000 0000, then
Write to the System Control Register.
Additional System Control Register writes permissible immediately afterward.

Care must be taken so an interrupt does not write to a CRTC register or the display buffer between the read of the frame buffer and the write(s) to a System Control Register.
Reads of the System Control Registers are unrestricted.

**SYSCONFIG -- SYSTEM_CONTROL GROUP**
**offset 004h**

| | | |
|---|---|---|
| sysconfig.reserved.31 | = 0 | |
| sysconfig.shift3.30..29 | | shift control 3 |
| sysconfig.pixel_size.28..26 | | bits per pixel |
| sysconfig.disable_selftime.25 | = 0 | |
| sysconfig.driveload2.24 | = 0 | |
| sysconfig.pllbackup.23 | = 0 | |
| sysconfig.shift0.22..20 | | shift control 0 |
| sysconfig.shift1.19..17 | | shift control 1 |
| sysconfig.shift2.16..14 | | shift control 2 |
| sysconfig.pixel_swap_half.13 | | halfword swap for display buffer accesses by CPU |
| sysconfig.pixel_swap_byte.12 | | byte swap for display buffer accesses by CPU |
| sysconfig.pixel_swap_bits.11 | = 0 | |
| sysconfig.pixel_buf_read.10 | | buffer selection for display buffer reads by CPU |
| sysconfig.pixel_buf_write.9 | | buffer selection for display buffer writes by CPU |
| sysconfig.reserved.8..3 | = 0 | |
| sysconfig.id2..0 | | silicon revision |

SYSCONFIG.ID2..0
See "Identifying Silicon Revision"

SYSCONFIG.PIXEL_BUF_READ.10
SYSCONFIG.PIXEL_BUF_WRITE.9
The host CPU can program the Power 9100 for individual control of which buffer is accessed for CPU reads and writes. These are for CPU access the display buffer only. The memory to memory blitter does not have this ability.

SYSCONFIG.PIXEL_SWAP_HALF.13
SYSCONFIG.PIXEL_SWAP_BYTE.12
SYSCONFIG.PIXEL_SWAP_BIT.11
These bits affect the endianess between the host bus and the display buffer. Refer to "Endianess" section.

SYSCONFIG.SHIFT3..0
To calculate the Shift Control counts, total up the number of displayed horizontal pixels and multiply by the pixel size. Set the shift control fields so they add up to the total. Many combination of shift control values will work for a given resolution. Remember, this is a byte count of the horizontal line of active display.

Examples

| | Total Bytes | Shift0 | Shift1 | Shift2 | Shift3 |
|---|---|---|---|---|---|
| 1280x1024 x8-bits | 1280 | 110 | 100 | 000 | 00 |
| 1280x1024 x24-bits | 3840 | 110 | 101 | 100 | 10 |

SYSCONFIG.PIXEL_SIZE.28..26
Refer to the data sheet. 24 bpp mode packs 4 pixels into three 32-bit words.

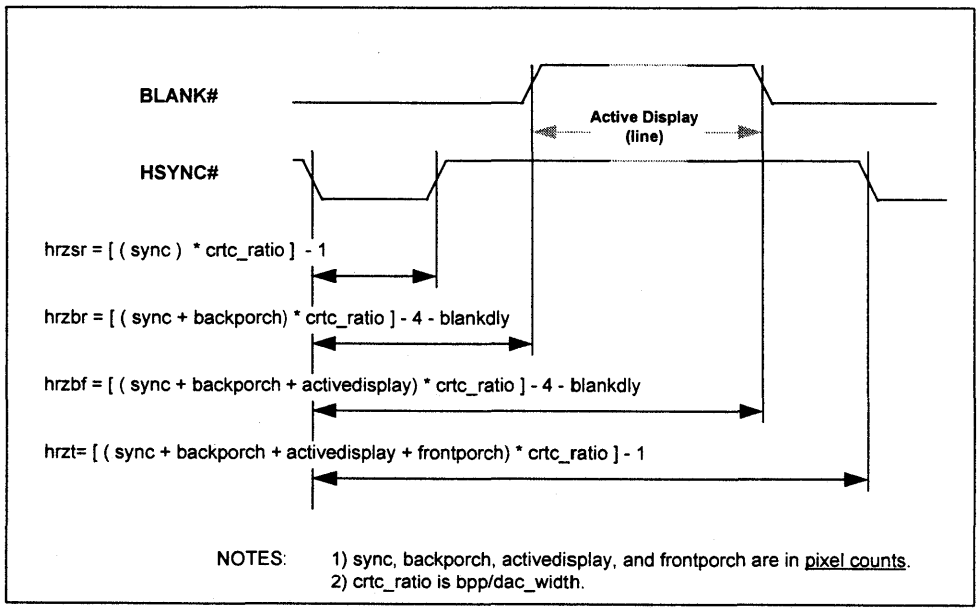
## 11.2. VIDEO CONTROL GROUP
**offset 0100h**

hrzt, hrzsr, hrzbr, hrzbf, prehrzc, vrtt, vrtsr, vrtbr, vrtbf, prevrtc, srtctl2, srtctl

### HORIZONTAL BLANK AND SYNC TIMING

The horizontal blank and sync timing are controlled by the CRTC_CLK (known as LCLK or VIDOUTCLK externally.) Add the timing, in pixel counts, multiply by the fractional crtc_ratio, and make a final adjustment. mem_config.blnkdly.28..27 only effects hrzbr and hrzbf.

$$\text{crtc\_ratio} = \frac{bpp}{RAMDACWIDTH}.$$




HSYNC REGISTER WAVEFORMS

**Documentation Correction**

Horizontal Monitor Timing Registers --
The P9100 interprets the native mode horizontal CRT timing registers slightly differently than the P9000. When the software does not compensate for this difference then when the monitor timings are taken from the P9000, they produce different timings on the P9100 board.

The Nov 93 P9100 data book is incorrect, it still documents the operation of the P9000 for hrzbr and hrzbf. Referring to Figure 132, the rising edge of hblnk- occurs hrzbr + 4 + mem_config.blnkdly CRTC clocks after the falling edge of hsync (not hrzbr +1 as the figure states.) Similarly, the falling edge of hblnk- occurs hrzbf + 4 + mem_config.blnkdly CRTC clocks after the falling edge of hsync- (not hrzbf + 1 as the figure states.)

**Product Status Report** (ref # 029)

N4C-A2

This problem does not occur in N4C-A2 designs because MEMCLK must be lowered, as described in the Memory Controller Initialization section, to work around a N4C-A2 problem.

N4E-A4

In designs using 2 controller banks and a 64-bit DAC, the horizontal front porch may have to be increased at low resolutions such as 640x480x8, but MEMCLK can still be 50 MHz. Please contact your sales office for more information.

**SRTCTL.QSFSELECT.**
**SRTCTL.SRC_INCS.**
**MEM_CONFIG.VAD.SHIFT**

These registers control the timers for the VRAM serial shift registers. You need to know the number of VRAM SAM banks of memory and the size of the shift registers.

To determin information about the VRAMs, read the pu_config register.

| VRAM | Row Size | Full SAM * | Half SAM |
|---|---|---|---|
| 128kbx? | 256 | 256 | 128 |
| 256kbx? | 512 | 512 | 256 ** |

* typically Micron
** typically IBM, TI

**SRTCTL.QSFSELSECT**
The srtctl.qsfselect controls how frequently the shift registers have to be reloaded. First, compute effective depth of the shift registers:

$$effective\_shift\_register\_depth = depth\_of\_one\_shift\_register * effective\_sam\_banks$$

Where: $$effective\_sam\_banks = \frac{32[bits] * controller\_banks}{DACWIDTH\ [bits]}$$

Now, compute srtctl.qsfselect:

$$srtctl.qsfselect = \log_2(effective\_shift\_register\_depth) - 5$$

Example: Full size, 256kx16 VRAM
4 MB, 64-bit DAC

effective_sam_depth = 1024; qsfselect = 5.

**SRTCTL.SRC_INCS**

The srtctl.src_incs controls the address generation for the shift register reloads. The effective row size is determined by the number of memory banks and the depth of a memory row. The width of the DAC does not matter. It is the effective row size that determines the value for srtctl.src_incs:

$$effective\_row\_size = row\_size\_of\_one\_bank * controller\_banks$$

$$srtctl.src\_incs = \log_2(effective\_row\_size) - 9$$

**VAD_SHIFT**

For full sized VRAMs, use:

$mem\_config.vad\_shft = 0$ This is true in most cases. Exception below.

HALF SIZE SAM VRAMs

The simplistic approach for half-size SAM VRAM is to compute the values as if the VRAM were full-size SAM VRAM and then adjust the values as follows:

```
if (half_size_SAM)
    {
    srtctl.qsfselect--;
    if (srtctl.src_incs != 0)
            srtctl.src_incs--;
    else
            mem_config.vad_shft = 1;
    }
```

For a half size sam VRAM, decrement qsfselect by 1. If srtctl.src is greater than 0, then decrement it too, otherwise set vad_shft to a 1.

**SRTCTL2 -- VIDEO CONTROL GROUP**

DPMS

| | srtctl2.vsync_plt.3..2 | srtctl2.hsync_plt.1..0 |
|---|---|---|
| OFF | 1 0 | 1 0 |
| STANDBY | 0x | 10 |
| SUSPEND | 10 | 0x |
| ON | 0x | 0x |

Note: The x value is dependent on the desired sync polarity.
The mechanism to assert BLANK has not be determined at this time. The srtctl.enable_video bit also disables syncs. [complete? accurate?]

**Documentation Correction**

Sync Polarities --
The databook and errata do not correctly document HSYNC and VSYNC. The correct specification follows:

SRTCTL2 register --

hsync_plt 1,0 HSYNC Polarity control
- 00= High true polarity
- 01= Low true polarity
- 10= Forced low
- 11= Forced high

vsync_plt 3,2 VSYNC Polarity control
- 00= High true polarity
- 01= Low true polarity
- 10= Forced low
- 11= Forced high

unused 11..4 Read back as 1s.
unused 31..12 Read back as 0s.

## 11.3. VRAM CONTROL GROUP

MEM_CONFIG -- VRAM CONTROL GROUP
The following fields in the mem_config register are often programmed with the same values, others are programmed according to the memory type, size and organization. These are descibed in more detail:

| | | |
|---|---|---|
| mem_config.vram_read_sample.31 | | = 1 |
| mem_config.slow_host_hifc.30 | = 1 | |
| mem_config.config[3].29 | | high order bit of memory configuration |
| mem_config.blnkdly.28..27 | | blank delay |
| mem_config.reserved.26 | = 0 | |
| mem_config.soe_mode.25..24 | | serial output enables on VRAMs |
| mem_config.shiftclk_mode.23..22 | | serial shift clock on VRAMs |
| mem_config.vad_sht.21 | | additional divide for Video Transfer address |
| mem_config.video_clk_sel.20 | | select video source |
| mem_config.blank_edge.19 | | LCLK clock edge for H/VSYNC |
| mem_config.reserved.18..16 | = 0 | |
| mem_config.shiftclk_freq.15..13 | | divide rate for LCLK |
| mem_config.crtc_freq.12..10 | | divide rate for CRTC |
| mem_config.hold_reset.9 | = 0 | |
| mem_config.dac_mode.8 | = 0 | |
| mem_config.dac_access_adj.7 | = 0 | |
| mem_config.priority_select.6 | = 1 | |
| mem_config.vram_write_adj.5 | = 1 | |
| mem_config.vram_read_adj.4 | = 1 | |
| mem_config.vram_miss_adj.3 | = 1 | |
| mem_config.config[2..0] | | 3 or 4 bits of memory configuration number |

MEM_CONFIG.CONFIG[3..0]29,2..0
Separate the fields into three sub fields:

```
config[3] = 0;                                  // assume no double buffering

if (VRAM_256K)                              // depth of the VRAM page
    config[2] = 1;
else
    config[2] = 0;

config[1..0] = number_of_banks - 1;         // number of 32-bit banks
```

Some common configurations are shown in the databook. There is no difference between a 256kbx8 and a 256kbx16 implementation. The 256kbx16 VRAMs must have individual byte WEs, however.

MEM_CONFIG.VRAM_MISS_ADJ.3
The *vram_miss_adj.3* field may be programmed to zero to improve performance. This will affect the VRAM timings and should not be done without an understanding of potential negative implications. (Most likely several timing parameters will be violated, but the board will still work.)

MEM_CONFIG.CRTC_FREQ.12..10
MEM_CONFIG.SHIFTCLK_FREQ.15..13
MEM_CONFIG.VIDEO_CLK_SEL.20
The video back-end runs from a load clock (LCLK) that controls the generation of the blanking and synchronization signals and controls the clocking of groups of pixels into the DAC. The relationship between the pixel clock, which controls the rate at which pixels are sent to the display, and the load clock is given by the following simple formula:

$$LCLK[\frac{clocks}{\sec}] = \frac{bpp[\frac{bits}{pixel}]}{DACWIDTH\ [bits]} * pixclk[\frac{pixelclocks}{\sec}]$$

The pixel clock is multiplied by the factor $\frac{bpp}{DACWIDTH}$ to generate the LCLK.

The DAC generates SCLK which is connected to DIVPIXCLK on the Power 9100. The SCLK frequency is based on the pixel bus width, the color depth, and the pixel clock. SCLK is fed though the Power 9100 (via DIVPIXCLK) to become LCLK that is connected back to LCLK on the DAC.

LCLK is a delayed version of the SCLK (DIVPIXCLK), the frequency is not affected with the following settings:

mem_config.crtc_freq.12..10 = 000b
mem_config.shiftclk_freq.15..13 = 000b

Select DIVPIXCLK as the input for SCLK:

mem_config.video_clk_sel.20 = 0b

MEM_CONFIG.BLANK_EDGE.19
The mem_config.blank_edge.19 bit determins the edge of LCLK used to generate BLANK#, HSYNC, and VSYNC. In general the falling edge is preferred and should be used whenever possible. Some older boards will fail if the rising edge is used.

0 means the rising edge of LCLK
1 means the falling edge of LCLK

**Silicon Product Status (ref # 014)**

**N4C-A2**

N4C-A2 cannot use the falling edge when: $f_{LCLK} > 33MHz$.

When mem_config.blank_edge.19 = 1 and the LCLK frequency is too high the screen may display a horizontal ripple. When mem_config.blank_edge = 0 <u>and</u> there is no series damping on the BLANK# signal, then noise may be seen on the left and right edges of the screen.

**N4E-A4**

The N4E-A4 silicon uses the falling edge at any frequency.

MEM_CONFIG.VAD_SHT.21
Refer to the discussion about programming the srtctl registers, Video Control Group.

MEM_CONFIG.SHIFTCLK_MODE.23..22
MEM_CONFIG.SOE_MODE.25..24
First, compute the number of effective SAM banks in the current configuration. A SAM bank has the width of the bus connecting the VRAMs to the DAC. The controller banks refer the number of memory banks that the controller interleaves. The controller banks are always 32 bits wide:

$$sam\_banks = \frac{32[bits]*controller\_banks}{DACWIDTH\ [bits]}$$

NOTE: If this number is ever less than one, then you must reduce the effective width of the DAC. For example, if you had one bank of memory connected to a 64-bit wide DAC, then you would have to configure the DAC as a 32-bit DAC. In this case, generate an intelligible error message and wait for customer requests before supporting this type of configuration.

Apply the following formula:

$$\text{mem\_config.shiftclk\_mode} = \log_2(sam\_banks)$$

$$\text{mem\_config.soe\_mode} = \log_2(sam\_banks)$$

MEM_CONFIG.BLNKDLY.28..27

## Silicon Product Status (ref # 014)

**N4C-A2**

```
if (mem_config.blank_edge == 0)
    // positive edge LCLK generates blank and syncs
    mem_config.blnkdly = 1;
else
    // negative edge LCLK generates blank and syncs
    mem_config.blnkdly = 2;

// The following configuration requires special casing because
// an external PAL is used to generate the shift clocks. This
// requires an additional cycle of blank delay to align blank
// properly with the the serial clocks.
//
if ((actual_banks == 4) && (dacwidth == 32))
    mem_config.blnk_dly++;
```

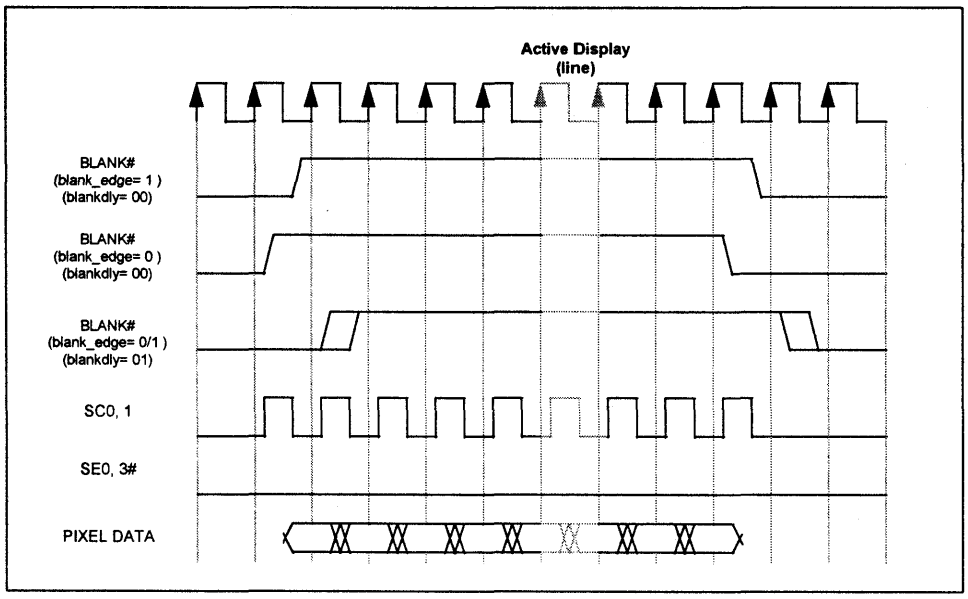
**N4E-A4**

```
mem_config.blnkdly = 1;
// The following configuration requires special casing because
// an external PAL is used to generate the shift clocks. This
// requires an additional cycle of blank delay to align blank
// properly with the the serial clocks.
//
if ((actual_banks == 4) && (dacwidth == 32))
    mem_config.blnkdly++;
```

## 1 SAM BANK DESIGN

MEMCONFIG.SHIFTCLK_MODE.23..22 = 00
MEMCONFIG.SOE_MODE.25..24 = 00
MEMCONFIG.BLANK_EDGE.19 = see below
MEMCONFIG.BLNKDLY.28..27 = see below

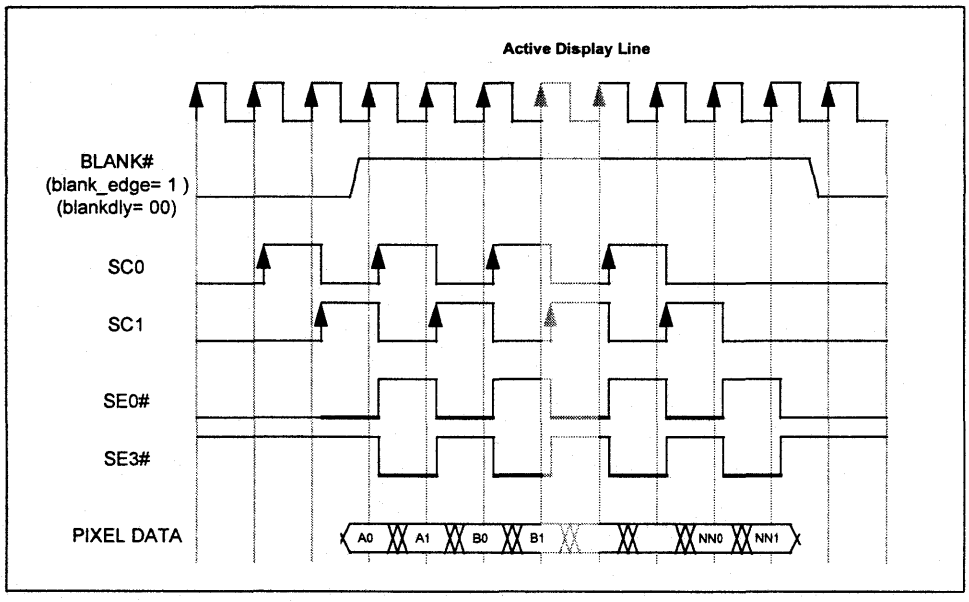
[complete? accurate?]



**2 SAM BANK DESIGN**

MEMCONFIG.SHIFTCLK_MODE.23..22 = 01
MEMCONFIG.SOE_MODE.25..24 = 01
MEMCONFIG.BLANK_EDGE.19 = 1
MEMCONFIG.BLNKDLY.28..27 = 00

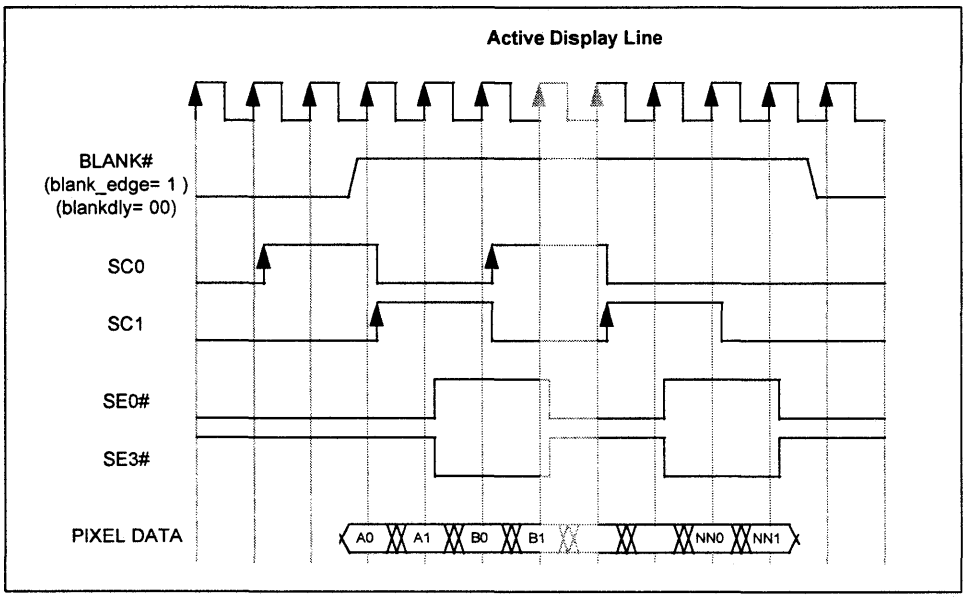
[complete? accurate?]



**4 SAM BANK DESIGN**
MEMCONFIG.SHIFTCLK_MODE.23..22 = 10
MEMCONFIG.SOE_MODE.25..24 = 10
MEMCONFIG.BLANK_EDGE.19 = 1
MEMCONFIG.BLNKDLY.28..27 = 00

[complete? accurate?]

An external PAL is required to generate SE1# and SE2#. This PAL is described in the Application Note. This is an unusual implementation, but is fully supported.



## 11.4. DAC CONTROL GROUP

DAC REGISTER
offset 200h

Address bit 5..3 map to the 4 address lines for the DAC. Accesses must be 32-bit words, but only 1 byte of data is transferred. This byte is transferred on the MD[23..16] byte lance and is driven on to the host interface bus with the effects of data swapping (endianess) for Native Registers in full effect. Use 32-bit long words to read and write the DAC. Only one byte lane has significance. [complete? accurate?]

For writes, it may be simpler to duplicate the 8-bit data in all byte lanes. For reads, the correct byte must be extracted.

**Product Status Report (ref # 028)**

N4C-A2 & N4E-A4

Due to a problem in the silicon, all DAC writes must be preceeded by a Display buffer read with address bits 14..7 matching the DAC address.

Read the Display buffer at 1000 0000 0000 0010 0000 0000, then
Write to the DAC.
Additional DAC writes permissible immediately afterward.

Care must be taken so an interrupt does not write to a CRTC register or the display buffer between the read of the frame buffer and the write(s) to a DAC.

Reads of the DAC are unrestricted.

IBM RGB525 Notes:

Get a copy of IBM's document that explains how to minimize the clock jitter of the PLL inside the IBM RGB525:

*Programming the RGB525 Clock Generator*
*March 10, 1994, David Warfield, Revision 1.0*

DAC Recovery Time Notes:

Most of the DACs have restrictions and cautions associated with their programming. Below is a partial list of know limitations.

The BT485 DAC requires a recovery time between palette accesses. The P9100 native mode has a programming mode to provide this recovery time, but it is not completely effective. It is advisable to extend the recover time by reading a native mode CRTC register or the PU_CONFIG register between palette writes. In VGA mode, the P9100 has the required delay between palette accesses for the DAC to function properly.

## 11.5. VIDEO POWER & 3D COPROCESSOR GROUP

refer to the source software and the Video Power programmer's and 3D programmer's guides, when available.

## 11.6. PARAMETER ENGINE GROUP

refer to the data book

## 11.7. DRAWING ENGINE GROUP

The drawing engine group is covered in the data book. A few clarifications are included in this section.

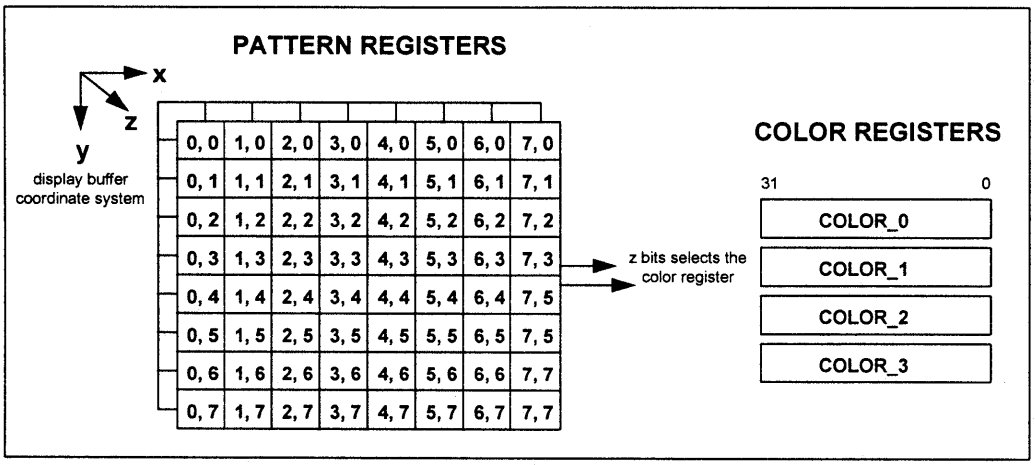
### 11.7.1. PATTERN REGISTER

DRAWING_ENGINE GROUP
offset 2280h

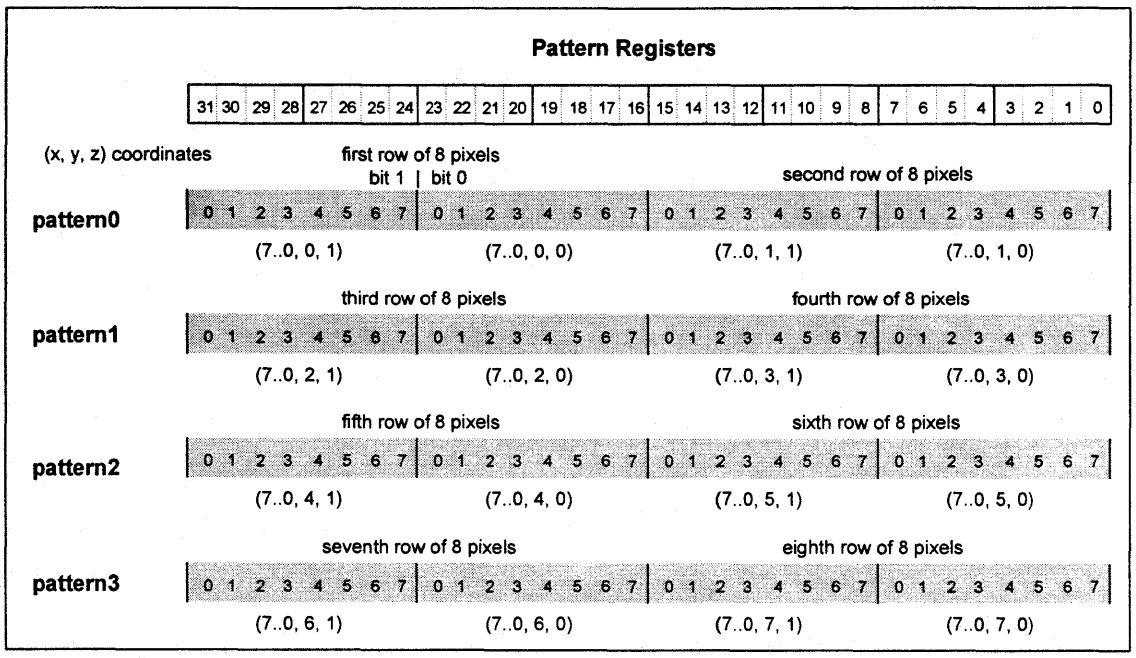
[complete? accurate?] -- Due to time considerations, this section may not be accurate, but it does convey the concept used by the pattern register. An accurate depiction is planned for the next revision.

The Pattern Registers consist of four 32-bit read/write registers. As a group they specify a bit pattern of pointers to the color registers for use by the Quad command.

This group of pattern registers make up an 8x8 pointer map. Its 2 bit deepness selects one of 4 color registers. The 8x8 pattern is repeated over the entire display area in all directions. The origin (0,0) is relocatable using the Pattern Origin XY Register. The 2 bit value at the origin is in pattern register 0, bit locations 23 and 31, least and most sifgnifcant values respectively.




PATTERN REGISTER BIT MAP




PATTERN REGISTER

### 11.7.2. COLOR REGISTERS

[3..0]COLOR -- DRAWING_ENGINE GROUP
offset 22xxh

The color registers are read/write registers that are selected in the Pixel1 Command, and all situations using the Minterms.

| | | |
|---|---|---|
| 2200h | COLOR[0] | Background (for PIXEL1 command) |
| 2204h | COLOR[1] | Foreground (for PIXEL1 command) |
| 2238h | COLOR[2] | (raster.pattern_depth[14] = 1) |
| 223Ch | COLOR[3] | (raster.pattern_depth[14] = 1) |

Color data must be replicated in all four bytes of a register when operating in 8- or 16-bit per pixel mode. In 8-bit mode, write the same color value in all four bytes. In 16 bpp mode, replicate the color value twice. The upper byte in 24 bpp mode must be loaded with the Blue element (same as the first byte.) [rde IC or HW ENGINEERING or SW ENGINEERING: Confirm the upper byte is a care.]

### 11.7.3. RASTER REGISTER

RASTER REG -- DRAWING_ENGINE GROUP
offset 2218h

PATTERN MODES -- RASTER REG
There are 3 individual bits in the Raster Register to control the mode of the Pattern Register.

[complete? accurate?]

PATTERN DEPTH
raster.pattern_depth[14] must be set to 0 in 16, 24 & 32 bpp modes. Select either in 8 bpp mode with no impact on performance.
0 = 2 color pattern (color[0] and color[1] only, color[2] and color[3] are ignored)
[rde ENGINEERING Confirm this 2 color mode uses color[0] and color[1]]
1 = 4 color pattern (8 bpp only) [rde ENGINEERING: Why only 8 bbp?]

PATTERN TRANSPARENT
raster.transparent_enable[17] works in all bbp modes.
0 = the raster.midterms operate normally
1 = Color[] Reg = 00h the pixel not written (uses write-per-pixel mode of VRAM)
Color[] Reg NOT= 00h the raster.midterms operate normally

[rde IC or HW ENGINEERING or SW ENGINEERING: these definitions need reviewing.]

PATTERN SOLID COLOR
0 = solid color enabled, the pattern register appears as if it has all 1s.
1 = solid color disabled, pattern enabled (pattern value selected color register)

**MINTERMS -- RASTER REG**
**"LOGICAL OPERATIONS ON SOURCE, DESTINATION, AND PATTERN COLOR "**
The drawing engine uses raster or logical operations for drawing quads and blitting areas.
Minterms are used by the raster ops engine to perform Boolean operations on pixel color. Pixel widths of 8, 16, 24, or 32-bits are supported. The raster.minterms[7..0] operate on each bit of the pixel in parallel.

There are 3 pixel color sources for the Boolean operation:

SOURCE COLOR
The source pixel color depends on the operation.

| Command | Source |
|---|---|
| Quad | color register selected by the pattern and mode bits |
| Blit | display buffer pixel color value |
| Pixel8 | data from the host bus |
| Pixel1 | color[0] or color[1], using individual bits from the host bus data to select |
| Next_Pixels | same as the preceeding Pixel Command |

DESTINATION COLOR
The destination color is the pixel color in the Display buffer where the pixel is being written.

PATTERN COLOR
One of the four color registers (selected by the pattern register)

DETERMINING MINTERMS

Taking each bit in the pixel individually, the Minterms operate according to how the bits are set in raster.minterms[7..0]. Set one or more bits to enable one or more cases to generate a 1 in the resulting pixel color value.

On a bit by bit basis, determin the conditions when a 1 is desired for the color result.

3 Input Logical Operation: Examples:

| pattern | source | destination | | Pixel Drawing | Pattern Drawing | Source XOR Destination | Source AND Destination |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | minterms[0] = 1 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | minterms[1] = 1 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | minterms[2] = 1 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | minterms[3] = 1 | 1 | 0 | 0 | 1 |
| 1 | 0 | 0 | minterms[4] = 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | minterms[5] = 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | minterms[6] = 1 | 1 | 1 | 1 | 0 |
| 1 | 1 | 1 | minterms[7] = 1 | 1 | 1 | 0 | 1 |
| | | | raster.minterms[7..0] | CCh | F0h | 66h | 88h |

---

TIP: When the destination color becomes a don't care, this is special case in the hardware that eliminates a read cycle to improve performance. The special case occurs when the value in raster.minterms[7..0] has only adjoining bit value pairs the same. Destination Color becoming a Don't Care is a Special Case (improved performance)

High Performance Minterms:

raster.minterms[7..0] = 03h, 0Ch, 30h, C0h & OR'ings: 0Fh, CFh, CCh. etc.

[rde ENGINEERING: is this true for all these cases?]

[complete? accurate?]

---

# 12. BOARD REFERENCE

## 12.1. CLOCK SYNTHESIZER PROGRAMMING

The clock synthesizer is a serially programmed device that uses two data bits (pins Serial Clock and Serial Data.) These two pins are controlled by either the [42h]config.clock_select.3..2 field (Native Mode) or I/O ports 3C2/3CC bits 3..2 (VGA Mode.)

[42h]config.clock_select.2 is CKSEL0, and [42h]config.clock_select.3 is CKSEL1. Always program [42h]config.4 (CKSEL2) to a 0 unless External I/O Register Mode is desired.

The ICD 2061A clock synthesizer has two outputs that are controlled by writing a series of values to cause the Serial Clock and Serial Data lines to toggle in order to serially program information into the chip. Refer to the initialization code examples. The ICD 2061A takes a 10 mS to time out and adjust itself. Wait for this to occur before accessing Power 9100 registers or registers it controls.

The ICD 2061A has a watch-dog timer on the Serial Clock and Data lines. After 5 mS of inactivity, these bits select which of 3 internal registers will be used to set the frequency of VCLK. It takes another 5 mS for the frequency to adjust.

| Bits 3..2 | Selection after 5 mS of In-Activity |
|---|---|
| 00 | Reg0, default freq = 25.175 MHz |
| 01 | Reg1, default freq = 28.322 MHz |
| 10 | |
| 11 | Reg2, Programmed by the BIOS or Driver |

The ICD 2061A is used in at least 2 configurations.

Configuration #1:
MCLK is the MEMCLK
VCLK is the Pixel Clock

Configuration #2:
MCLK is the MEMCLK
VCLK is the Reference clock for another PLL (IBM RGB525 for instance.)

MEMCLK FREQUENCY

**Product Status Report (ref # 017)**

N4C-A2

| |
|---|
| If the number of controller banks = 2 and the DAC width = 64 then the MEMCLK frequency must be less than 7 times theLCLK frequency, but not greater than 50 MHz, the Power 9100 spec. |

N4E-A4

| |
|---|
| MEMCLK can be set as high as 50 MHz, the Power 9100 spec. |

CLOCK OSCILLATORS
Some designs do not contain a descrete clock synthesizer. In this case, a clock oscillator (tin can) supplies the same clock for MEMCLK and the reference clock for another PLL. In this case there is no ICD2061A and the N4C-A2 silicon does not work at low resolutions in a 2 MB design using a 64-bit DAC.

ICD 2061A Notes

(1) Do not program the two phase locked loops to the same frequency. Drivers should contain code which insures that the ICD 2061A pixclk and memclk PLLs are always operating at different frequencies.

Here are two examples of how this can happen:

You're using the ICD 2061A with a Brooktree DAC and are running with a 50 MHz memclk and select a display resolution of 800x600@72Hz refresh rate which requires a 50 MHz pixclk. If the driver programs the same control values into both PLLs then visible screen jitter will occur.

You're using the ICD 2061A to generate a reference clock for the IBM RGB525

(2) After selecting a new pixel frequency you must wait at least 10 ms for the frequency to stabilize. The host interface of some DACs is not usable until the frequency is stable. For instance is you are using the ICD2061A to generate a reference clock to the IBMRGB525 RAMDAC and you toggle the clock select bits in VGA to select a new frequency, then a delay should be inserted before the DAC host interface is used. This is currently handled by the BIOS.

(3) There are often several ways to obtain the same output frequency which will produce different levels of clock jitter. Some customers like to pick these control values carefully to produce the most stable display.

# 13. VGA MODE REFERENCE

In VGA Mode the Power 9100 operates like a VGA chip. VGA Mode is selected by setting [41h]config.modeselect.1 = 1.

## 13.1. ENABLING VGA COMPATIBLE REGISTERS AND MEMORY

The controller must be in VGA Mode with I/O and memory enabled to support VGA register and memory compatibility. Power up values to VGA Mode:

Config[41h].modeselect.1 = 1
Config[04h].mem_enable.1 = 1
Config[04h].io_enable.0 = 1
[complete? accurate?]

## 13.2. VGA PALETTE SNOOPING

VGA palette snooping works when the Power 9100 is in VGA mode.

Palette Snooping is enabled to allow palette writes to migrate to all buses, including the ISA bus, if present, so all VGA compatible DACs get updated color information. This is accomplished by having the Power 9100 controller perform the write, but not claim the bus cycle.

[04h]config.shadow_dac.5 = 1 enables Palette Snooping

Palette reads proceed normally, claiming the bus cycle with the assertion of LDEV#/DEVSEL#.

**Silicon Product Status (ref # 026)**

**N4C-A2**

| |
|---|
| The N4C-A2 silicon does not support Palette Snooping on the PCI Bus due to unacceptable responses to aborted cycles. This problem does not effect VLBus systems. The Video BIOS should disable Palette Snooping on PCI boards using N4C-A2 silicon. |

**N4E-A4**

| |
|---|
| Palette snooping works on PCI and VLBus. |

## 13.3. DPMS

CRTC registers

OFF
STANDBY
SUSPEND
ON

Contact Weitek for DPMS support.

## 13.4. SAVE & RESTORE VGA

When switching to Native mode, the contents of all VGA registers and memory locations are not preserved so a complete save operation is required before switching in order to restore the state at a later time.

SAVING VGA REGISTERS & MEMORY

The VGA registers can be saved by reading all of them. The display buffer data must also be read before changing state. After changing state, immediately store the contents of the extended DAC registers. The device is now in Native Mode.

Save VGA registers
Save VGA memory
Switch to Native Mode
Save Extended DAC registers

```
// Code to save VGA.
//
```

RESTORING VGA REGISTERS & MEMORY

Before switching to VGA Mode, restore the extended DAC registers and set mem_config.soe_mode= 01h. Switch mode and restore the VGA registers and memory.

```
// Code to restore VGA
//
```

### Silicon Product Status (ref # 030)

N4C-A2 & N4E-A4

The Power 9100 loads pu_config.mem_depth.4 at the deassertion of RESET# and on the transition from Native to VGA Mode. The value loaded is taken from the MD[4] at the time of the transition. To clock the MD[4] bit properly, ROM code fetching (or pre-fetching) cannot be occuring at the moment the mode switch is made.

The BIOS code pushes instructions on to the stack before execution. This calls off prefetching of code from the ROM so the MD[31..0] bus settles to a 0.

# 14. APPENDICES

## 14.1. BIOS REVISION HISTORY

Not all BIOS versions were formally released. Released versions include a release note & date (as shown below). Here is a summary of BIOS revisions.

**Release 1.10**
Basis -- Production Release.

**Release 1.20 (see release notes, April 18, 1994)**
DPMS support added.
File names changed to show version.
EDITBCT.EXE included to allow modification of sign-on banner & OEM string.

**Release 1.21**
Palette snooping OFF for all PCI designs.
Alternate Font Select fixed (??)

**Release 1.22**
AutoMem sizing stored in 3C4 Index 12 (SEQ 12), bits 5,4:

| | | |
|---|---|---|
| 00 | 1 MB | |
| 01 | 2 MB | This must change because bit 5 |
| 10 | 4 MB | is CKSEL2. |

Palette snooping OFF for PCI N4C-A2 designs.
No change in palette snooping policy for PCI N4E-A4 designs.
Palette snooping ON for all VLB designs.

**Release 1.23**
Amount of memory available in native and emulation mode are stored in the Output Control Register (3C5h INDEX 12h).

| | | |
|---|---|---|
| 3..0 | SVGA clock frequency | |
| 4 | 0 = | 512 KB Emulation memory |
| | 1 = | 1 MB Emulation memory |
| 5 | | CLSEL[2], always set to zero. |
| 7..6 | 00 = | 1 MB Native memory |
| | 01 = | 2 MB Native memory |
| | 10 = | 4 MB Native memory |

Fixed memory detection problem in emulation mode. Modes 1024x768 x6 and 1280x1024 x4 work properly. Assumes 256k x memory types, always.

Palette Snooping policy --
VLB: Enabled all the time.
PCI: Disabled for A0 - A3, no change to control bit when A4 is present.

VESA DPMS does not trash video memory. Refresh cycles occur during DPMS now.

VESA function 0 (Return Super VGA Information) does not list modes 110, 111, 112, 113, and 114 as supported modes.

**Release 1.24 (see release notes)**
to be done

## 14.2. P9X00RES.DAT FILE

[complete? accurate?] -- additional discussions are planned for the next release.

P9X00RES.DAT FILE USAGE
The P9x00RES.DAT file contains two types of information that are used by the install program: system parameters & monitor configuration.

P9X00RES.DAT FILE CONTENT
The P9x00RES.DAT file contains four types of information:
1) variables
2) monitor blocks
3) timing blocks
4) overrides, and
5) comments

VARIABLES

MONITOR BLOCKS

TIMING BLOCKS

OPTIONAL OVERRIDES -- Windows Driver

NOTE: The optional override parameters are used on a display timing basis. This is independent of color depth. The memcfgset & memcfgclr overrides are design to allow special ability to change bits 19, 27, & 28. To clear these bits, regardless of what the driver wants to do, set memcfgclr to E7F7 FFFF. To set these bits, set memcfgset to 1808 0000. The software actually performs anding and oring on all 32-bits, but the usefulness of this is only experimental.

COMMENTS

## 14.3. DISPLAY PROBLEMS

**WAVY DISPLAY -- Native or VGA Mode**
ICD2061A is incorrectly programmed. Set the clock frequencies so they are always slightly different (Windows Driver 1.00.06 and beyond do this).

**UNDULATING LEFT AND RIGHT EDGES -- Native Mode**
The blank signal is not clocked correctly in to the DAC. On N4C-A2 silicon, memconfig.blank_edge.19 needs to be set to zero (rising edge) to satisfy internal timing at high frequency LCLK. Hold time to the DAC can be satisfied by placing a 33 to 47 ohm series damping resistor near the Power 9100 on the BLANK# signal.

see mem_config.blank_edge.19 for details.

**JITTERY DISPLAY -- Native or VGA Mode**
If the PLL on the IMB RGB525 is used, see DAC programming for details.

**BLANK SCREEN DURING OPERATION -- Native Mode**
The sync timings become corrupted because a DAC or System Control register was no programmed correctly. All software code must read the display buffer before writing to the DAC or System Control registers. See DAC programming or System Control Register programming for details.

# 15. INDEX

WEITEK

WEITEK

# POWER 9100 GRAPHICS CONTROLLER

## ERRATA

March 8, 1994

The WEITEK Power 9100 is a high-performance display controller for use with graphical user interfaces such as Microsoft Windows. It combines an extremely high-speed frame buffer controller with a workstation-style accelerated display controller and a local-bus host interface for maximum performance.

## Contents

Power 9100 Data Book (Errata)
March 8, 1994



WEITEK Corporation
1060 East Arques Avenue
Sunnyvale, California 94086
Telephone (408) 738–8400



Written by Claire-Marie Costanza
Edited by Robert Plamondon
Additional writing by Robert Plamondon, D.R. Sevedge, and Allen Samuels.
Illustrated by Claire-Marie Costanza, Robert Plamondon, Allen Samuels, and D.R. Sevedge

Printed in the United States of America
94 95 6 5 4 3 2 1

4710–9404–00 Rev. A

# Foreword

This Power 9100 Graphics Controller Data Book Errata release consists of 31 pages. Each page has changed information marked with change bars, like this paragraph.

The information in this document applies to the A2 silicon only. (A2 chips are marked N4C.)

The changes can generally be classified as either *corrections* or *updates*. A correction changes erroneous information; an update provides additional information.

The most important changes are:

1. The system block diagram has been updated.
2. The clock signal names (pin 152) have been corrected. The correct clock signal name for VL is LCLK (was BCLK). The correct clock signal name for PCI is CLK (was CLK–). These names now correspond with the names in the interface specification. The operation of these pins has not changed.
3. The serial enable signals SE1 and SE2, shown in some memory configurations in section 7.1, must be generated using external logic. This was not previously explained. Refer to the *Power9100/Video Power Board Application Note* for more information.
4. The power-up configuration bits register (PU_CONFIG) has been updated. The updated register definition explains in greater detail how this register is used. Some fields have been altered and some new fields have been added.
5. The Supported Components section (13.2) has been updated.
6. These changes have altered pagination slightly, as shown in the table on the right.

| Errata Page | Data Book Page |
|---|---|
| 7 | 7 |
| 8 | 8 |
| 9 | 9 |
| 17 | 17 |
| 18 | 17 |
| 21 | 20 |
| 24 | 23 |
| 27 | 26 |
| 53 | 53 |
| 54 | 54 |
| 57 | 57 |
| 58 | 58 |
| 69 | 69 |
| 77 | 75 |
| 78 | 76 |
| 80 | 78 |
| 81 | 79 |
| 83 | 81 |
| 93 | 91 |
| 95 | 93 |
| 97 | 95 |
| 109 | 107 |
| 147 | 145 |
| 186 | 184 |
| 187 | 185 |
| 189 | 187 |
| 190 | 188 |
| 191 | 189 |
| 192 | 190 |
| 198 | 196 |
| 200 | 198 |

## 1.4. Host Bus Interface

The Power 9100 supports the VL and PCI buses directly, with no glue logic. See figures 6 through 9. Other buses can be accommodated with a small amount of external glue logic.

Some pins on the Power 9100 change function depending on which bus is selected. Rather than introduce a confusing third signal nomenclature to that of PCI and VL, we have provided two pin configuration diagrams: one for each bus, each using that bus' signal naming conventions. See section 13.4 for the pin configuration.

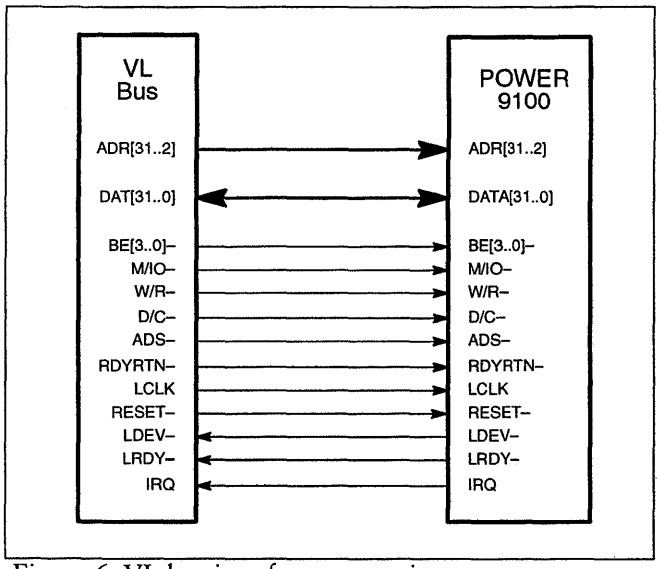



Figure 6. VL bus interface connection

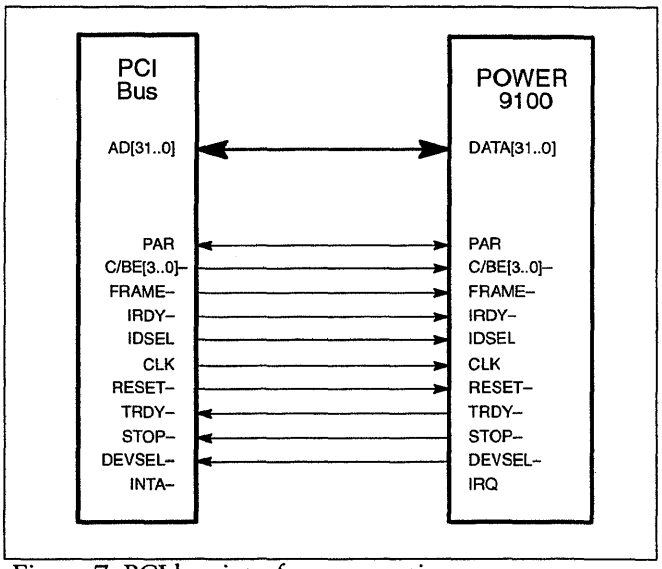



Figure 7. PCI bus interface connection

| VL Bus | | |
|---|---|---|
| **Signal** | **I/O** | **Description** |
| DATA[31..0] | I/O | Data bus |
| ADR[31..2] | Input | Address bus |
| BE[3..0]– | Input | Byte enable |
| W/R– | Input | Write or read status |
| M/IO– | Input | Memory or I/O status |
| ADS– | Input | Address data strobe |
| LDEV– | Output | Local device |
| LRDY– | Output | Local ready |
| RDYRTN– | Input | Ready return |
| IRQ | Output | Interrupt request |
| LCLK | Input | VL clock |
| RESET– | Input | Reset |
| D/C– | Input | Data or code status |

Figure 8. VL bus interface signals

| PCI Bus | | |
|---|---|---|
| **Signal** | **I/O** | **Description** |
| DATA[31..0] | I/O | Address and data bus |
| C/BE[3..0]– | Input | Bus command/byte enable |
| PAR | I/O | Parity |
| IDSEL | Input | Initialization device select |
| STOP– | Output | Stop |
| FRAME– | Input | Cycle frame |
| DEVSEL– | Output | Device select |
| TRDY– | Output | Target ready |
| IRDY– | Input | Initiator ready |
| IRQ | Output | Interrupt request |
| CLK | Input | PCI clock |
| RESET– | Input | Reset |

Figure 9. PCI bus interface signals

## 1.5. System Block Diagram

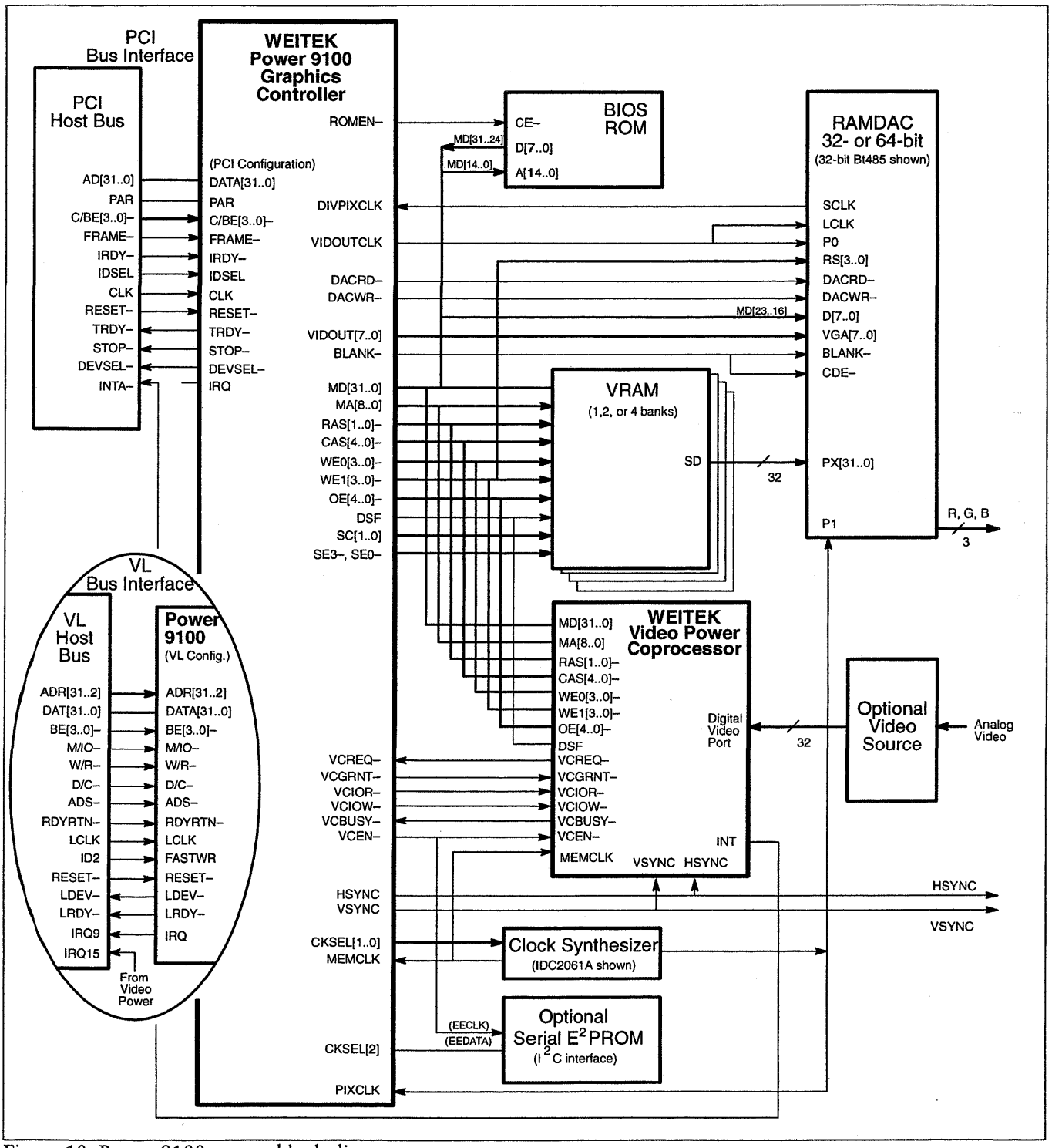



Figure 10. Power 9100 system block diagram

## 1.5. Frame Buffer and Video Interfaces, continued

### 1.5.1. SIGNAL DESCRIPTIONS

| Signal | Type | Description |
|---|---|---|
| MD[31..0] | I/O | Memory data bus |
| MA[8..0] | Output | Memory address bus |
| RAS[1..0]– | Output | Row address strobes |
| CAS[4..0]– | Output | Column address strobes. CAS[0]– controls the least-significant 16 bits of bank 0; CAS[4]– controls the most-significant bits of bank 0 (necessary for VGA transfers). Note that in two-bank systems, bank 1 is controlled by CAS[3]–, not CAS[1]– |
| WE0[3..0]– | Output | Write-enable for the individual bytes in bank 0 (2-bank systems), or banks 0–1 (4-bank systems) |
| WE1[3..0]– | Output | Write-enable for the individual bytes in bank 1 (2-bank systems), or banks 2–3 (4-bank systems) |
| OE[4..0]– | Output | Output Enables. OE[0]– controls the least-significant 16 bits of bank 0; OE[4]– controls the most-significant bits of bank 0 (necessary for VGA transfers) |
| DSF | Output | VRAM special function pin |
| DACRD– | Output | RAMDAC control line |
| DACWR– | Output | RAMDAC write control signal |
| MEMCLK | Input | Main chip clock |
| ROMEN– | Output | BIOS ROM enable |
| EECLK | Output | EEPROM clock control signal |
| EEDATA | I/O | EEPROM data control signal |
| VDDPLL | Input | Supply voltage for on-chip clock generator |
| VSSPLL | Output | Ground for on-chip clock generator |

Figure 11. Memory control signals

| Signal | Type | Description |
|---|---|---|
| SE[3]–, SE[0]– | Output | VRAM serial shift enable |
| SC[1..0] | Output | VRAM serial shift clock |
| VIDOUT[7..0] | Output | Video data out (SVGA modes) |
| VIDOUTCLK | Output | Video data clock out |
| HSYNC– | I/O | Horizontal synchronization. Normally an output; can also be used as an input for external sync |
| VSYNC– | I/O | Vertical synchronization. Normally an output; can also be used as an input for external sync |
| BLANK– | Output | Blanking interval |
| PIXCLK | Input | Pixel clock |
| DIVPIXCLK | Input | Divided pixel clock |
| CKSEL[2..0] | Output | Frequency synthesizer control |

Figure 12. Video control signals

## 3.3. Configuration Registers, continued

### 3.3.1. POWER-UP CONFIGURATION

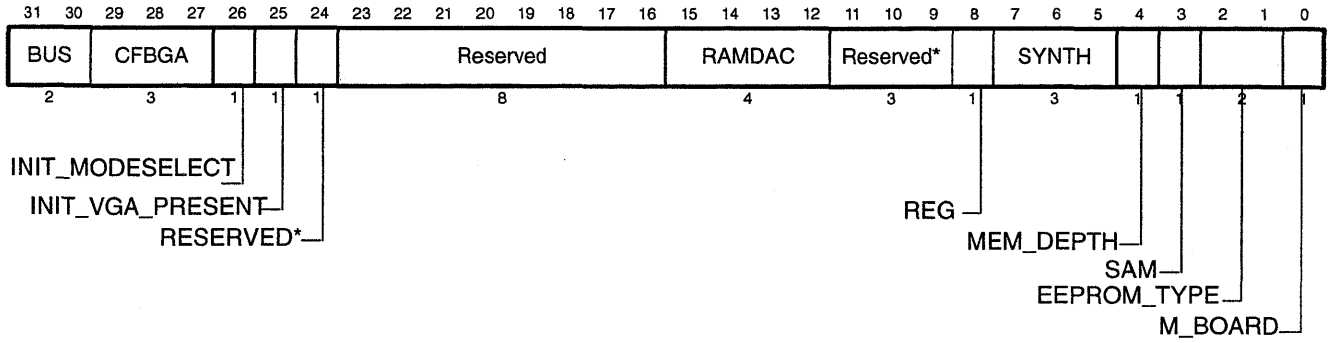
The deassertion of RESET– forces the system to sample the frame buffer data bus to determine its power-up configuration. Data bus drivers have built-in pull-down resistors causing the bus to float to a low state. By placing large (~10KΩ) pull-up resistors on the data bus, selected bits can be forced into the high state. The initial settings of the data bus are preserved in the read-only PU_CONFIG register.

The Power9100's power-up configuration register is used to configure the controller and inform software about a board's configuration, and to identify a board's resources without referring to any additional external information, such as a file on a disk or an EEPROM. (The number of banks of memory, not addressed in this register, can easily be determined through software.) The register, as currently defined, supports Brooktree RAMDACs and the IBM RGB525 RAMDAC.

The power-up configuration register can and will be further redefined to support other RAMDACs as the need arises. Please contact WEITEK if you want to make additional bit assignments to accommodate a different RAMDAC. We will make bit assignments as needed. We strongly suggest that you do not build a board requiring an additional encoding without contacting WEITEK unless you plan to write your own drivers.

Figure 19 illustrates and defines the power-up configuration register.

**Power-up Configuration Bits (PU_CONFIG register)**




*Must be set to zero. Do not use these bits or WEITEK software will not work.

| Field | Bits | Contents |
|---|---|---|
| BUS | 31–30 | The bus signalling protocol.<br>00 = Reserved (internal Testing mode)<br>01 = PCI Bus<br>10 = VESA Local Bus<br>11 = Reserved |
| CFGBA | 29–27 | I/O address for configuration register index and data registers.<br>000 = 9100h, 9104h 001 =9108h, 910Ch<br>010 = 9110h, 9114h 011 = 9118h, 911Ch<br>100 = 9120h, 9124h 101 =9128h, 912Ch<br>110 = 9130h, 9134h 111 = 9138h, 913Ch |

Figure 19. Power-up Configuration bits (PU_CONFIG register)

## 3.3. Configuration Registers, continued

| Field | Bits | Contents |
|---|---|---|
| INIT_MODESELECT | 26 | The initial value for CONFIG[65].MODESELECT<br>0 = Native mode 1 = Emulation mode (VGA) |
| INIT_VGA_PRESENT | 25 | The initial value for CONFIG[10].VGA_PRESENT<br>0 = VGA present in emulation mode<br>1 = VGA absent in emulation mode<br>(This setting changes the PCI sub-class code; see figure 27.) |
| Reserved | 24–16 | Bit 24 must be set to zero or the WEITEK software will not work. Bits 23–16 cannot be used with certain types of RAMDACs and are reserved for future use. (Note: One customer is using bits [19..16] as a board identification field, having determined that the RAMDAC in their design is compatible with this usage.) |
| RAMDAC | 15–12 | RAMDAC type.<br>0000 = Bt 485-style RAMDAC, as well as variations identifiable through the RAMDAC status register (identify variations through the upper four bits of the status register*)<br>1000 = IBM525 (IBM 64-bit RAMDAC) |
| Reserved | 11–9 | Must be set to zero. Do not use these bits or WEITEK software will not work |
| REG | 8 | External registers available.<br>0 = External registers are not implemented on the board<br>1 = External registers are implemented on the board and can be accessed through the RAMDAC address space when CKSEL[2] is set to one |
| SYNTH | 7–5 | Frequency synthesizer.<br>000 = ICD2061A, ICS9161, or compatible, or RAMDAC generates all clocks<br>001 = Fixed MEMCLK; RAMDAC generates PIXCLK. (This selection is currently intended to support IBM RGB525 designs wherein the MEMCLK is used as a reference clock for the RAMDAC PLL. It is assumed that the memory clock is running at 50 MHz, and, thus, the RAMDAC reference clock is also assumed to be 50 MHz.) |
| MEM_DEPTH | 4 | Depth of memory chips.<br>0 = 256K 1 = 128K |
| SAM | 3 | VRAM SAM size.<br>0 = Full-sized shift registers<br>1 = Half-sized shift registers |
| EEPROM_TYPE | 2–1 | 00 = AT24C01 (ATMEL 128x8) or compatible (or not installed) |
| M_BOARD | 0 | Motherboard VGA address decode control.<br>0 = On motherboard 1 = On add–in board |

*At this time, WEITEK software does not check the RAMDAC status register. If you are considering using a Bt 485-compatible RAMDAC with additional features, contact WEITEK first so that we can determine whether the status register alone will be enough to determine your RAMDAC's unique requirements.

Figure 19, continued. Power-up Configuration bits (PU_CONFIG register)

## 3.3. Configuration Registers, continued

### 3.3.8. CONFIG[8] REGISTER

**08h CONFIG[8] Register (Read Only)**

| 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Field | Bits | Definition |
|---|---|---|
| | 7–0 | Revision ID. Current revision ID. |

Figure 26. CONFIG[8] register (Read Only)

### 3.3.9. CONFIG[10] REGISTER

**0Ah CONFIG[10] Register (Read Only)**

| 7 | 6 | | | | | | 0 |
|---|---|---|---|---|---|---|---|
| vga_present | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Field | Bits | Definition |
|---|---|---|
| VGA_PRESENT | 0 | 0 = VGA present in emulation mode (sub-class code = 00h, VGA compatible controller)<br>1 = VGA absent in emulation mode (sub-class code = 80h, other display controller)<br>This bit is set by PU_CONFIG[25] on reset (see figure 19). |

Figure 27. CONFIG[10] register (read only)

### 3.3.10. CONFIG[11] REGISTER

**0Bh CONFIG[11] Register (Read Only)**

| 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |

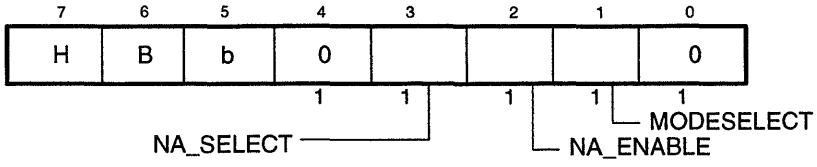
| Field | Bits | Definition |
|---|---|---|
| | 1–0 | Display controller (base class code = 03h, display controller)<br>These bits set to 11 in accordance with PCI 2.0. |

Figure 28. CONFIG[11] register (read only

## 3.3. Configuration Registers, continued

### 3.3.17. CONFIG[65] REGISTER

**41h CONFIG[65] Register**

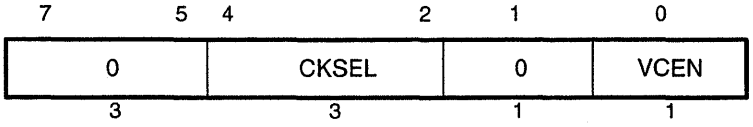



| Field | Bits | Contents |
|---|---|---|
| MODESELECT | 1 | Native/emulation mode.<br>This bit is set by PU_CONFIG[26] on reset (see figure 19).<br>0 = Power 9100 is in native mode.<br>1 = Power 9100 is in emulation (VGA) mode. |
| NA_ENABLE | 2 | Native mode. Respond to address range 000A0000–000AFFFF.<br>This bit is set to 0 (default) on reset.<br>0 = disabled.<br>1 = enabled. |
| NA_SELECT | 3 | Native mode. Select Coprocessor or frame buffer to respond to address range 000A0000–000AFFFF<br>This bit is set to 0 (default) on reset.<br>0 = Frame Buffer<br>1 = Coprocessor |
| b | 5 | Bit swap for commands through 000A0000–000AFFFF |
| B | 6 | Byte swap for commands through 000A0000–000AFFFF |
| H | 7 | Halfword swap for commands through 000A0000–000AFFFF |

Figure 35. CONFIG[65] register

### 3.3.18. CONFIG[66] REGISTER

**42h CONFIG[66] Register**




| Field | Bits | Contents |
|---|---|---|
| CKSEL | 4–2 | Sets the current state of pin CKSEL[2..0] in native mode. (default 0)<br>(CKSEL[2] also used as EEDATA; for writing, must be read through CONFIG[64]) |
| VCEN | 0 | The inverse of the level of the VCEN– pin.<br>This bit is set to 0 (default) on reset.<br>(Also used as EECLK) |

Figure 36. CONFIG[66] register

## 3.4. General Address Formats, continued

### 3.4.5. RAMDAC ADDRESS FORMAT

The RAMDAC address format, illustrated in figure 43, allows direct access to the RAMDAC. RAMDAC reads and writes are initiated immediately and completed within a few clock cycles; the longest possible wait time for a direct RAMDAC access is the time required to complete a VRAM refresh, plus the time required to complete a shift register reload, plus the time required for the access. The 8-bit data transfer to or from the RAMDAC occurs across the eight MD bus bits that are connected to the RAMDAC data bus (see chapter 8).

### 3.4.6. VIDEO COPROCESSOR INTERFACE

The video coprocessor address format, illustrated in figure 44, allows direct access to the video coprocessor. Video coprocessor reads and writes are initiated immediately and completed within a few clock cycles. The video coprocessor interface is only enabled when operating in native mode. Many of the emulation mode pins are shared.

**RAMDAC Address Format**

| 31–15 | 14–6 | 5–2 | 1–0 |
|---|---|---|---|
| Prefix | 0 0 0 0 0 1 0 0 0 | RAMDAC ADDR | 0 0 |
| 17 | 9 | 4 | 2 |

| Field | Bits | Definition |
|---|---|---|
| | 31–15 | Prefix. See figures 41 and 42. |
| RAMDAC ADDR | 5–2 | Adresses RAMDAC registers; each RAMDAC is product specific<br>Bit 5 = RS[3]<br>Bit 4 = RS[2]<br>Bit 3 = RS[1]<br>Bit 2 = RS[0] |

Figure 43. RAMDAC address format

**Video Coprocessor Interface Format**

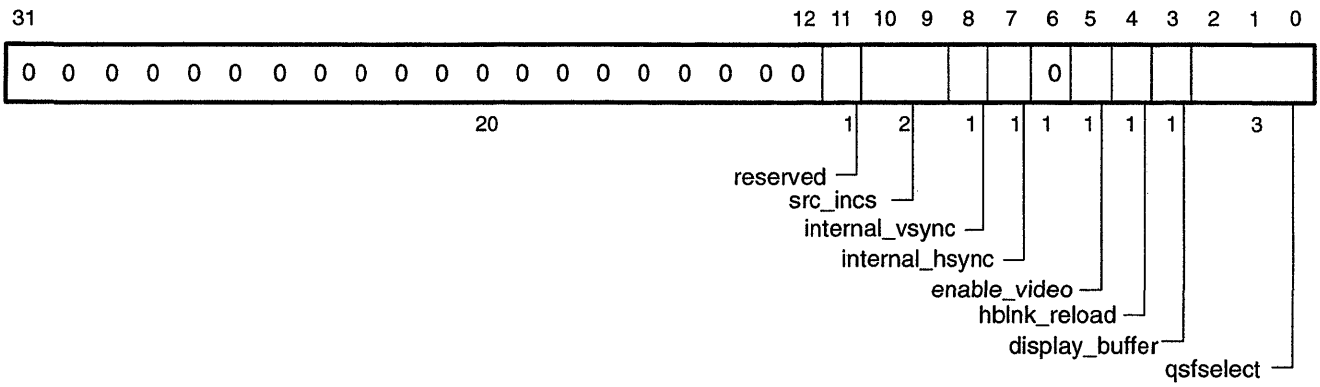
| 31–15 | 14–10 | 9–2 | 1–0 |
|---|---|---|---|
| Prefix | 0 0 0 0 1 | REG | 0 0 |
| 17 | 5 | 8 | 2 |

| Field | Bits | Definition |
|---|---|---|
| | 31–15 | Prefix. See figures 41 and 42. |
| | 9–2 | The register of the video coprocessor |

Figure 44. Video coprocessor interface format

## 4.6. Video Control Registers, continued

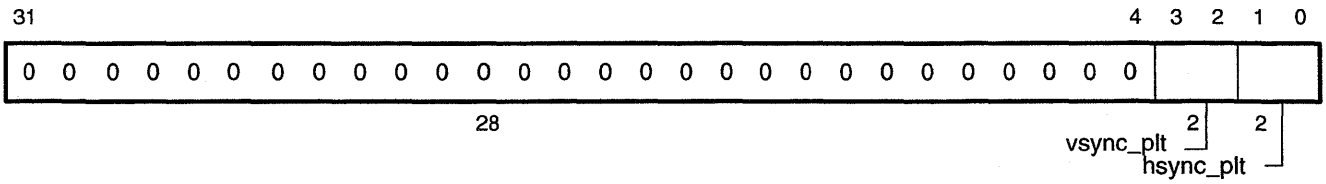
**Screen Repaint Timing control (srtctl) Register**




| Field | Bits | Contents |
|---|---|---|
| reserved | 11 | Reserved, set to 0 |
| src_incs | 10–9 | Amount to increment DRADDR register (see section 8.1.4)<br>00 = 256<br>01 = 512<br>10 = 1024 |
| internal_vsync | 8 | Internal vertical synchronization:<br>0 = Select external VSYNC–<br>1 = Select internal VSYNC– |
| internal_hsync | 7 | Internal horizontal synchronization:<br>0 = Select external HSYNC–<br>1 = Select internal HSYNC– |
| enable_video | 5 | Enable video:<br>0 = Blanks asserted<br>1 = Normal operation |
| hblnk_reload | 4 | Screen repaint mode (see section 8.1.9):<br>0 = Normal (split shift mode)<br>1 = Restricted (HBLNK– reload mode) |
| display_buffer | 3 | Buffer for display,:Any change to this field takes effect only after the next VBLNK–.<br>0 = Buffer 0<br>1 = Buffer 1 |
| qsfselect | 2–0 | QSF counter position. See section 4.6 |

Figure 74. Screen repaint timing control (srtctl) register

**Screen Repaint Timing Control (srtctl2) Register**

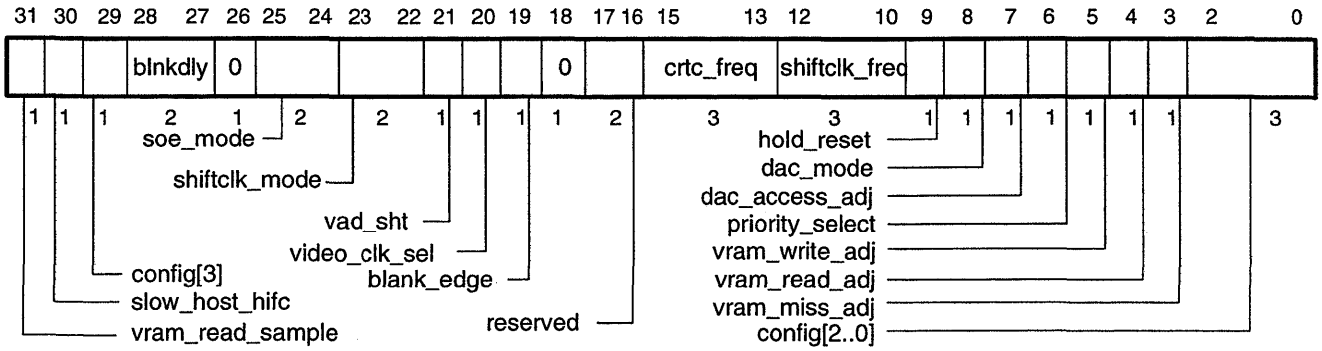



| Field | Bits | Contents |
|---|---|---|
| hsync_plt | 3–2 | External HSYNC polarity control.<br>00 = High true polarity<br>01 = Low true polarity<br>10 = Forced low<br>11 = Forced high |
| vsync_plt | 1–0 | External HVYNC polarity control.<br>00 = High true polarity<br>01 = Low true polarity<br>10 = Forced low<br>11 = Forced high |

Figure 75. Screen repaint timing control (srtctl2) register

## 4.7. VRAM Control Registers, continued

**Memory Configuration (mem_config) Register**




| Field | Bits | Contents |
|---|---|---|
| config[2..0] | 2–0 | Low-order 3 bits of memory configuration. See Figures 105 through 112 |
| vram_miss_adj | 3 | Adjustment for row miss timing template, see Figures 113 and 114 |
| vram_read_adj | 4 | Adjustment for read timing template, see Figures 115 and 116 |
| vram_write_adj | 5 | Adjustment for VRAM write timing template, see Figures 119 and 120 |
| priority_select | 6 | 0 = Video coprocessor has lower priority than drawing engine, 1 = higher |
| dac_access_adj | 7 | Enable RAMDAC back to back transfer checking, see Figures 139 and 140<br>Note: This feature does not work reliably in current silicon. Bact-to-back RAMDAC accesses should be separated with NOPS. |
| dac_mode | 8 | Controls dac read/write signalling mode, see Figures 139 and 140 |
| hold_reset | 9 | 1 Forces memory and video into reset state, 0 for normal operation |
| shiftclk_freq | 12–10 | Sets the frequency of the RAMDAC latch clock state machine.<br>Normally set to 000 because the RAMDAC generates the divided clock.<br>000 = pixel_clk / 1 001 = pixel_clk / 2 010 = pixel_clk / 4<br>011 = pixel_clk / 8 100 = pixel_clk / 16 |
| crtc_freq | 15–13 | Sets the frequency of the internal CRTC divided dot clk.<br>Normally set to 000 because the RAMDAC generates the divided clock.<br>000 = pixel_clk / 1 001 = pixel_clk / 2 010 = pixel_clk / 4<br>011 = pixel_clk / 8 100 = pixel_clk / 16 |

Figure 78. Memory configuration (mem_config) register (1 of 2)

| Field | Bits | Contents |
|---|---|---|
| reserved | 17–16 | Reserved, set to 0. |
| | 18 | Not used; set to 0. |
| blank_edge | 19 | Clock edge that HSYNC and VSYNC are synchronized to:<br>1 = Falling edge of video_clk 0 = Rising edge of video_clk |
| video_clk_sel | 20 | Selects the video_clk source for the video section:<br>0 = PIXCLK 1 = DIVPIXCLK |
| vad_sht | 21 | Additional divide for Video Transfer address (see section 8.1.4):<br>0 = Don't divide by two 1 = divide by two |
| shiftclk_mode | 23–22 | Sets timing patterns for shift clocks (see figure 126):<br>00 = 1 bank style 01 = 2 bank style 10 = 4 bank style |
| soe_mode | 25–24 | Sets timing patterns for serial output enables:<br>00 = 1 bank style 01 = 2 bank style 10 = 4 bank style |
| blnkdly | 28–27 | Controls delay for blank generation in units of shiftclk_freq:<br>00 = 0 shiftclk_freq delay 01 = 1 shiftclk_freq delay<br>10 = 2 shiftclk_freq delay 11 = 3 shiftclk_freq delay |
| config[3] | 29 | High order bit of memory configuration. See Figures 105 through 112 |
| slow_host_hifc | 30 | Slow host interface adjust. |
| vram_read_sample | 31 | Adjustment for read timing template, see Figures 115 through 118. |

Figure 78. Memory configuration (mem_config) register (2 of 2)

## 6.3. PCI Bus Operation, continued

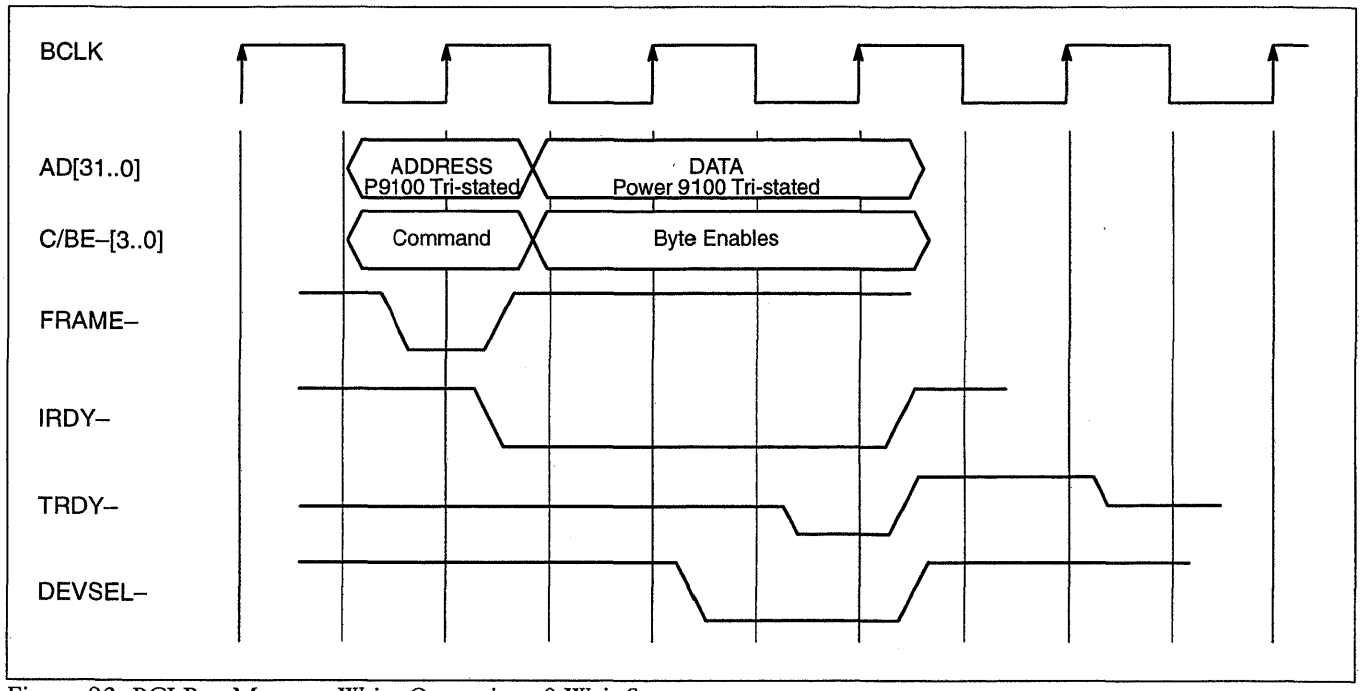



Figure 93. PCI Bus Memory Write Operation, 0 Wait States

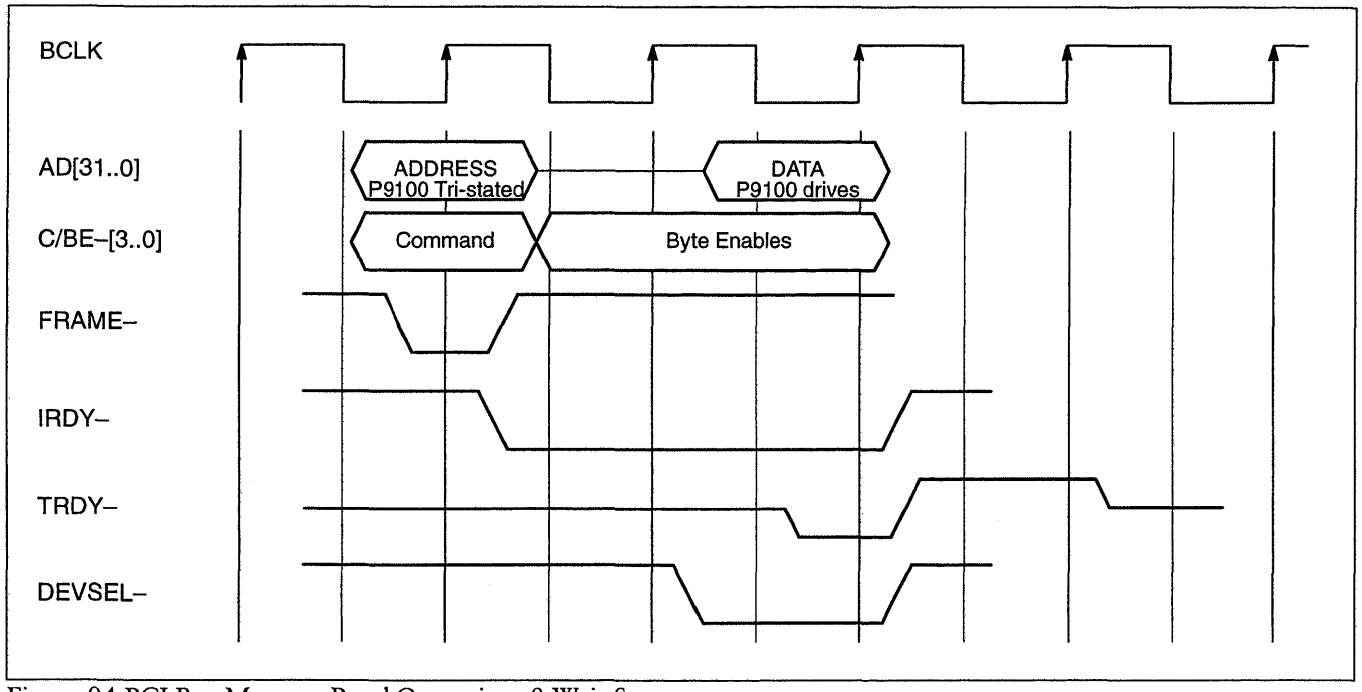



Figure 94.PCI Bus Memory Read Operation, 0 Wait States

# Chapter 7. Frame Buffer Interface

## 7.1. Frame Buffer Design Notes

This section presents information necessary to select, configure, and connect VRAMs. The first set of figures (105 through 112) shows how to wire together the memory chips for 1, 2 and 4 bank memory configurations.

See figure 51, system configuration register bits [9..10] and figure 74, screen repaint timing control (srtctl) register bit 3 for more information about defining the buffer.

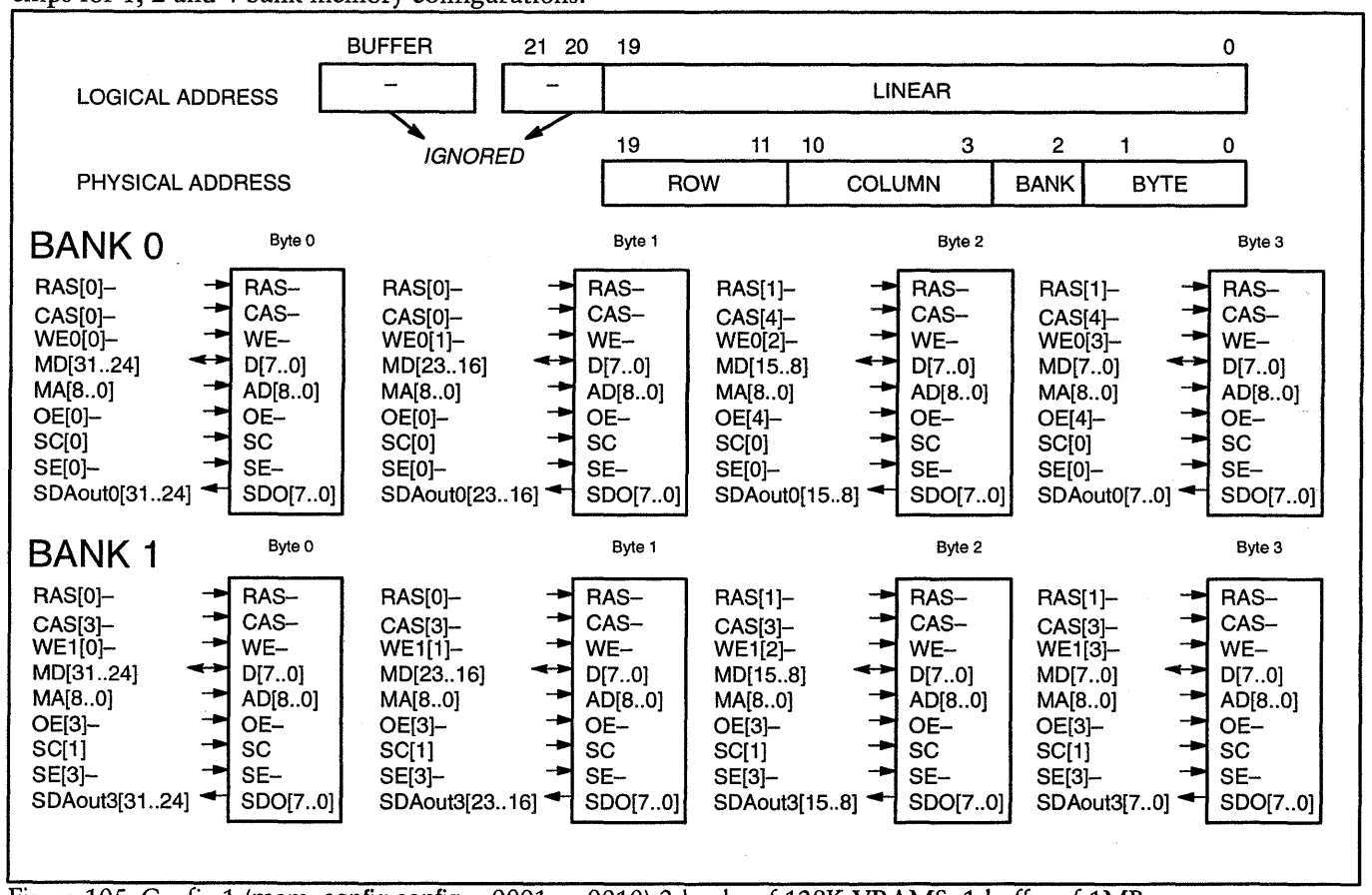



Figure 105. Config 1 (mem_config.config = 0001 or 0010) 2 banks of 128K VRAMS, 1 buffer of 1MB

Figure 109 assumes that you are using a 64-bit wide RAMDAC. If you are using a 32-bit wide RAMDAC, use external logic to generate additional serial enable (SE) signals to address banks 1 and 2. Refer to the *Power 9100/Video Power Board Application Note* for more information.

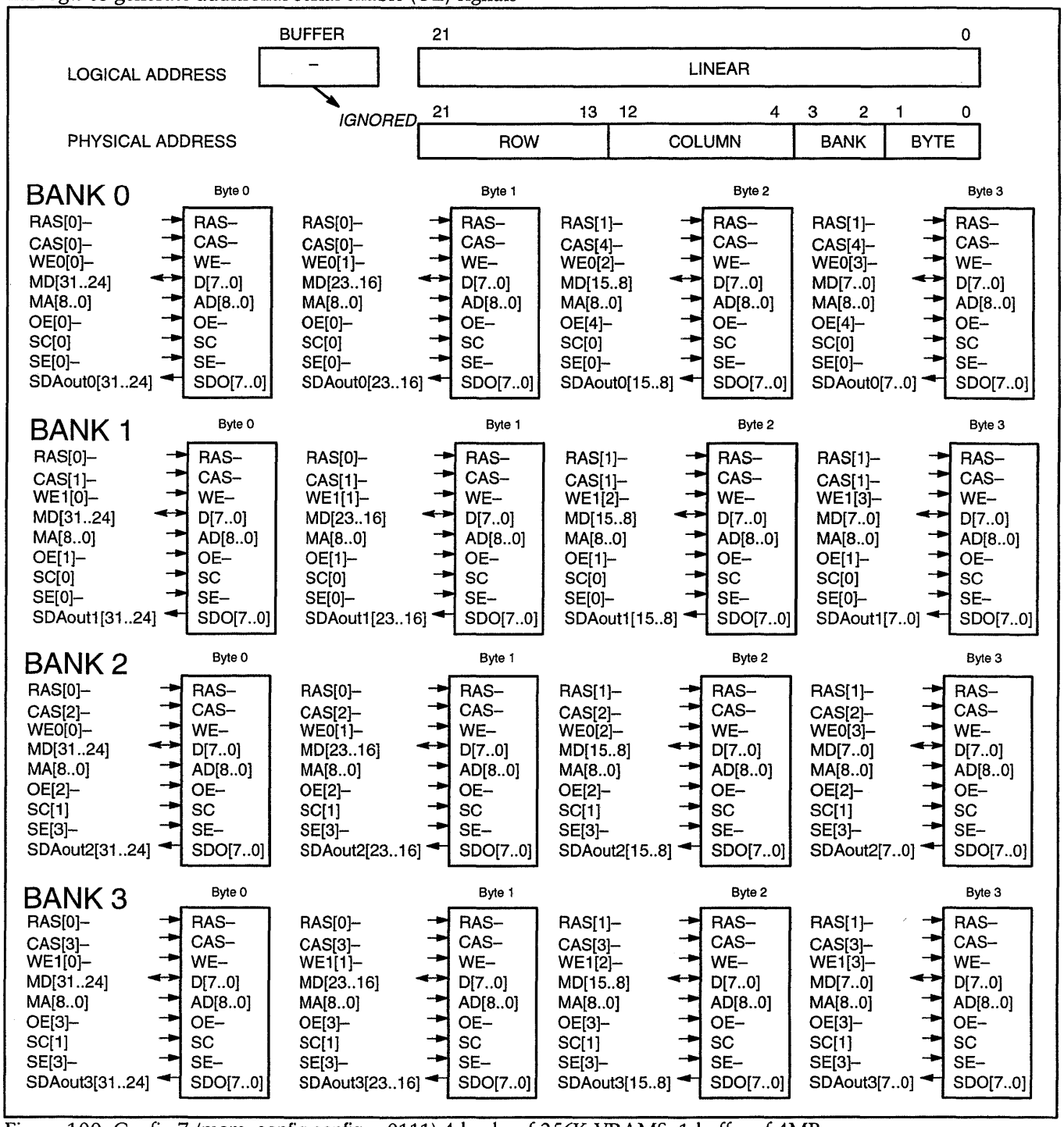



Figure 109. Config 7 (mem_config.config = 0111) 4 banks of 256K VRAMS, 1 buffer of 4MB

## 7.1. Frame Buffer Design Notes, continued

Figure 106 assumes that you are using a 64-bit wide RAMDAC. If you are using a 32-bit wide RAMDAC, use external logic to generate additional serial enable (SE) signals to address banks 1 and 2. Refer to the *Power 9100/Video Power Board Application Note* for more information.

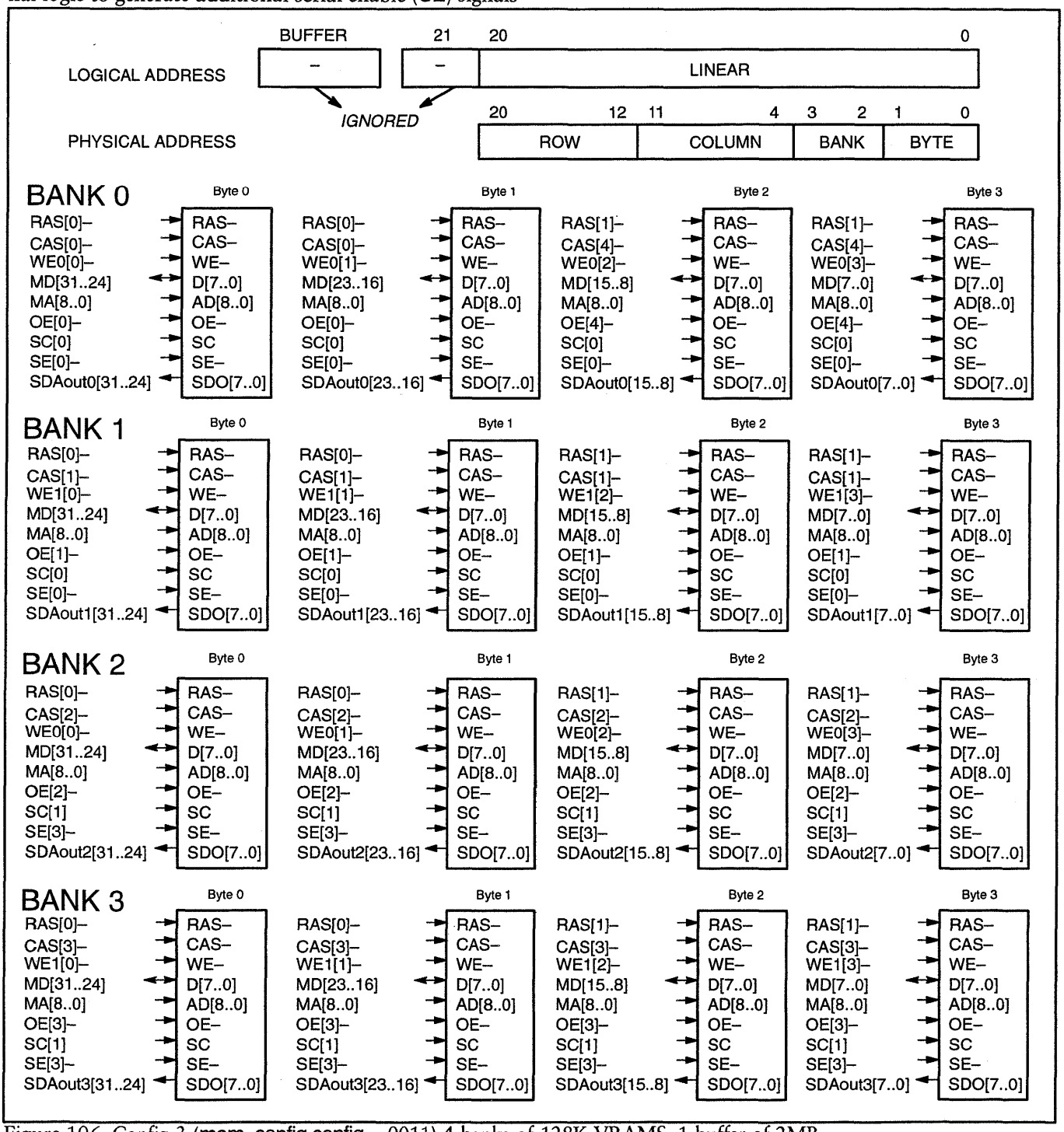



Figure 106. Config 3 (mem_config.config = 0011) 4 banks of 128K VRAMS, 1 buffer of 2MB

## 7.1. Frame Buffer Design Notes, continued

Figure 110 assumes that you are using a 64-bit wide RAMDAC. If you are using a 32-bit wide RAMDAC, use external logic to generate additional serial enable (SE) signals to address banks 1 and 2. Refer to the *Power 9100/Video Power Board Application Note* for more information.

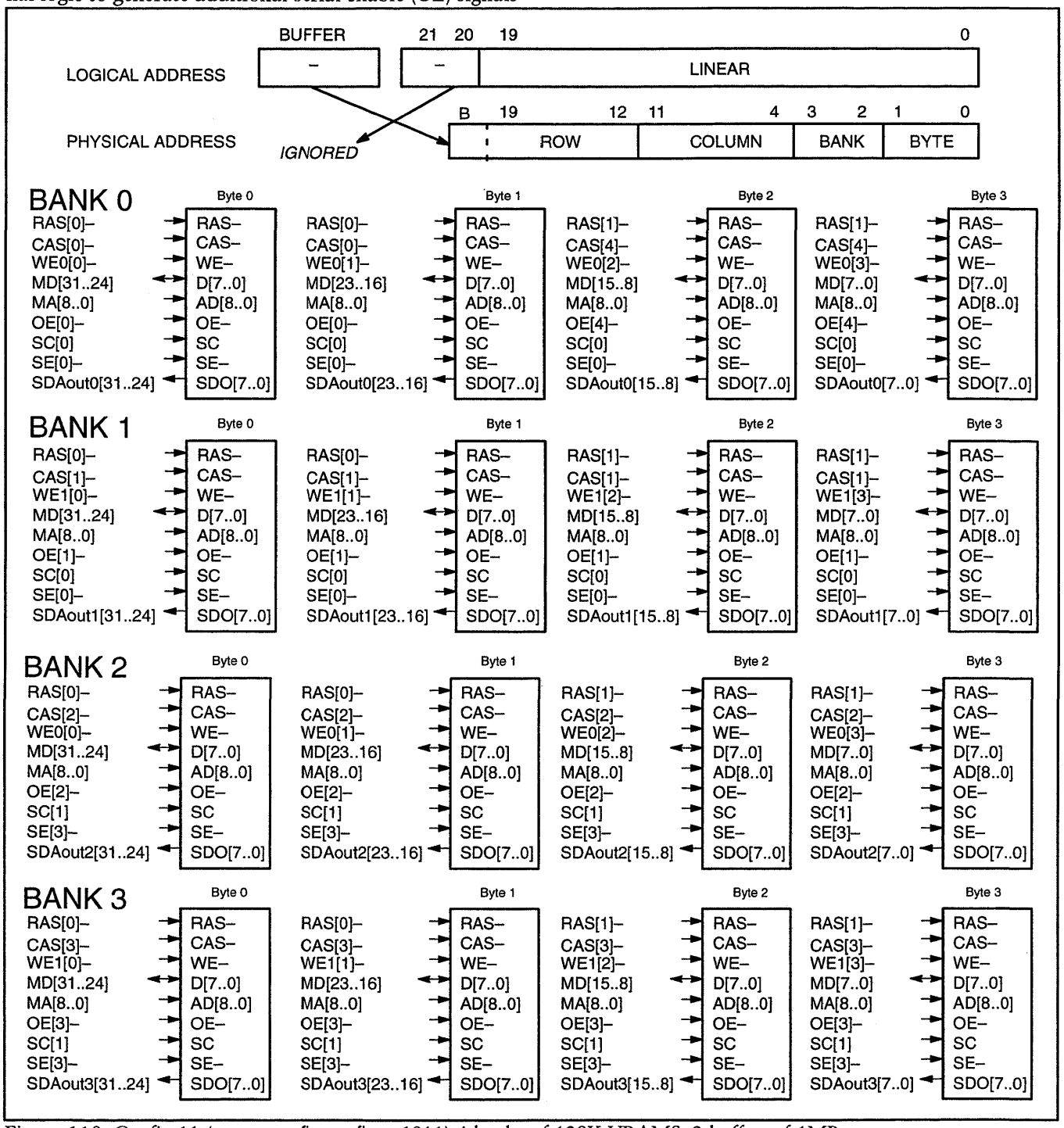



Figure 110. Config 11 (mem_config.config = 1011) 4 banks of 128K VRAMS, 2 buffers of 1MB

## 7.1. Frame Buffer Design Notes, continued

Figure 112 assumes that you are using a 64-bit wide RAMDAC. If you are using a 32-bit wide RAMDAC, use external logic to generate additional serial enable (SE) signals to address banks 1 and 2. Refer to the *Power 9100/Video Power Board Application Note* for more information.

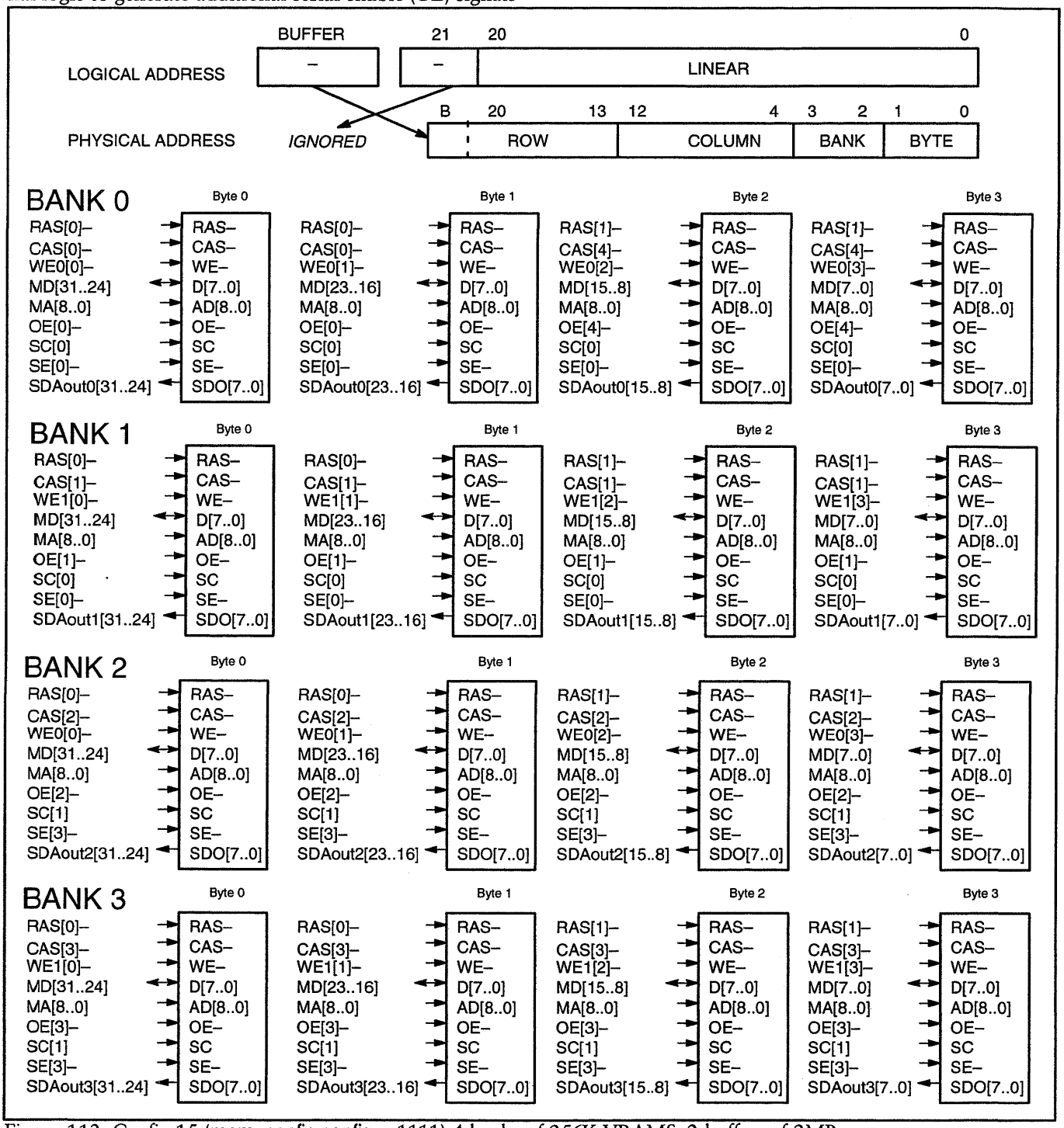



Figure 112. Config 15 (mem_config.config = 1111) 4 banks of 256K VRAMS, 2 buffers of 2MB

## 8.1. Video Control, continued

### 8.1.3. VIDEO SERIAL ENABLE GENERATION

Figure 127 shows the different serial enable generation patterns for different values of mem_config.soe_mode and mem_config.vad_sht.

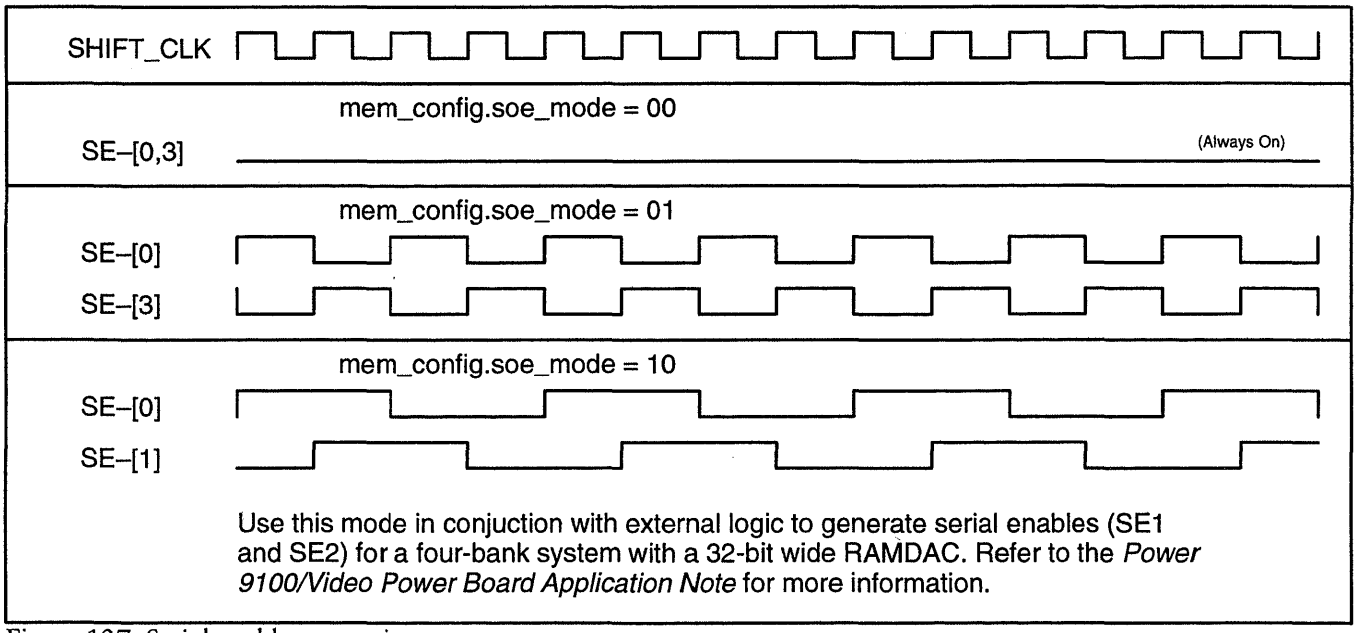



Figure 127. Serial enable generation

## 8.1. Video Control, continued

### 8.1.5. VIDEO SIGNALS

The Power 9100 provides separate HSYNC–, VSYNC–, and BLANK– video signals.

The srtctl2.hsync_plt and srtctl2.vsync_plt bits control the polarity of HSYNC– and VSYNC– and can also force them high or low to support VESA standard monitor power management. See figure 75.

### 8.1.6. VIDEO CONTROL REGISTERS

A complete video control section resides on the Power 9100 chip. This section provides the synchronization, blanking, and timing signals for the graphics subsystem, as well as the control sections for screen refresh timing and VRAM control.

The video control section of the Power 9100 is fully programmable and is defined by the host system. For example: The video control section can be defined to generate an internal sync signal or accept an external signal to control screen refresh, thereby simplifying the task of merging Power 9100-created graphics with other video images.

This document talks consistently about VSYNC– and HSYNC–. This refers to the internal signals, not the external pins. The external pins are controlled by srtctl2.vsync_plt and srtctl2.hsync_plt. These controls allow the selection of external signal polarity and the support of VESA monitor power down modes.

For easy reference, figure 129 repeats the video control register summary presented in chapter 4.

## 8.1. Video Control, continued

### 8.1.7. VIDEO TIMING

The video signals and registers work together to drive the signals that control the monitor. Remember, all counts are in units of CRTC_CLK (see figure 125).

*HORIZONTAL VIDEO TIMING*

Figure 132 presents the timing sequence for horizontal video control . All of the horizontal timing registers are loaded with counts derived from CRTC_CLK which may represent multiple pixels. Each of the values stored in the horizontal timing registers represents the total count minus one to allow for zero in the count sequence.

Figure 131 describes the functions of the horizontal timing registers.

The hrzt register must be a multiple of 8 for all modes except 8-bit-per-pixel with a 64-bit wide RAMDAC, when it must be a multiple of 16. For simplicity, hrzt may be a multiple of 16 for all modes.

The programmer must satisfy the following condition:

hrzsr < hrzbr < hrzbf < hrzt

*VERTICAL VIDEO TIMING*

Figure 133 presents the timing diagram for vertical video timing control. All of the vertical timing registers are loaded with counts that represent horizontal scan lines. Also, each number loaded into a vertical timing register represents the total count in the count sequence.

Figure 130 describes the functions of the vertical timing registers.

The programmer must satisfy the following condition:

vrtsr < vrtbr < vrtbf < vrtt

| Register | Description/Function |
|---|---|
| vrtt | In the vertical timing sequence, vrtt defines the number of horizontal lines from the falling edge of VSYNC– to the falling edge of the next VSYNC– (the number of horizontal lines in a complete vertical scan cycle). The vrtc register counts from zero to vrtt–1 and then repeats. |
| vrtsr | Specifies the VSYNC– active low count. The rising and falling transitions of VSYNC– should match those of HSYNC–. |
| vrtbr | Specifies the distance from VSYNC– low to the rising edge of VBLNK–. |
| vrtbf | Specifies the distance from VSYNC– low to the falling edge of VBLNK–. |

Figure 130. Vertical timing registers

| Register | Description/Function |
|---|---|
| hrzt | Defines the number of CRTC_CLKs from the falling edge of HSYNC– to the falling edge of the next HSYNC–. |
| hrzc | Counts from zero to hrzt and then repeats. |
| hrzsr | Specifies the HSYNC– active low count. |
| hrzbr | Specifies the distance from HSYNC– low to the rising edge of HBLINK–. |
| hrzbf | Specifies the distance from HSYNC– low to the falling edge of HBLINK–. |

Figure 131. Horizontal timing registers

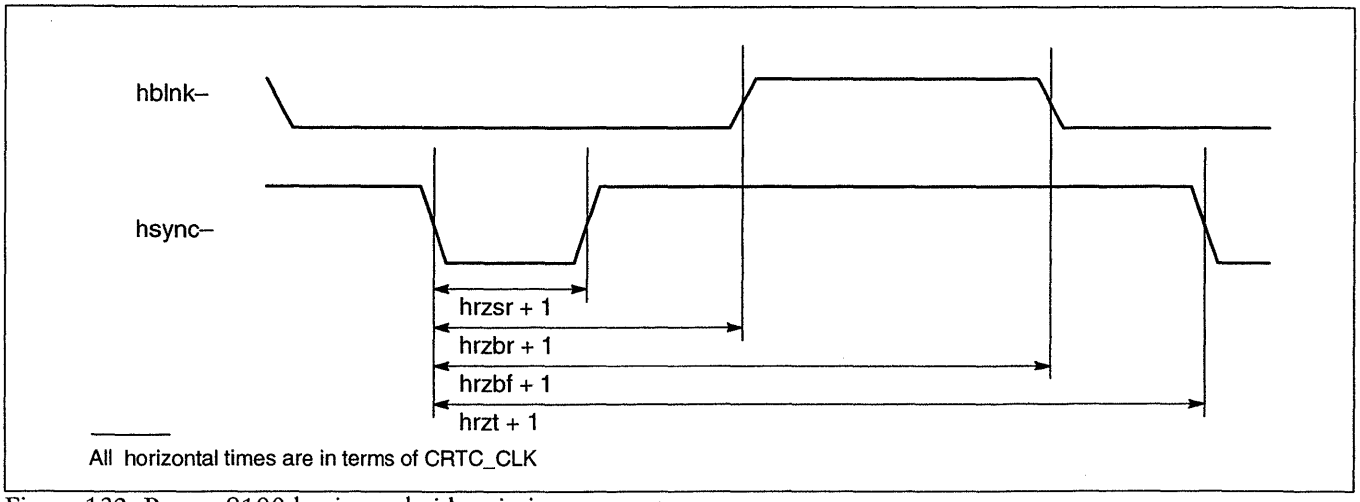



Figure 132. Power 9100 horizontal video timing parameters

# Chapter 10. Auxiliary Chip Control

## 10.1. EEPROM Control

The optional EEPROM is connected via two shared pins: CKSEL[2] and VCEN–. We use the $I^2C$ slave protocol. The software sees three bits: CONFIG[66].VCEN, CONFIG[66].CKSEL[2], and CONFIG[64].EEDAIN. The circuit schematic for connecting them is shown in figure 148

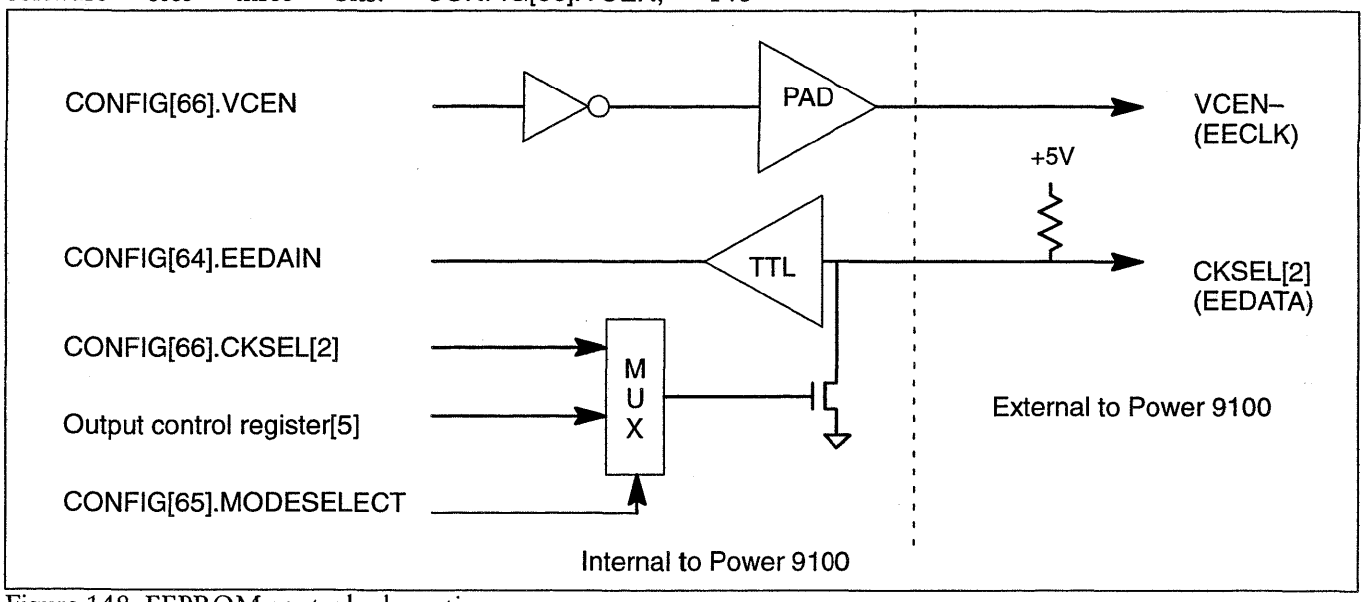



Figure 148. EEPROM control schematic.

## 10.2. Clock Synthesizer Control

The Power 9100 directly supports programmable clock synthesizers such as the ICD2016A from IC Designs. This requires 3 output pins to drive CKSEL[2..0].In emulation mode, the pins are driven directly by the output control register[5] (CKSEL[2]) and miscellaneous output register[3..2] (CKSEL1..0]), I/O port 3C2h write/3CCh read. See sections 12.4.3 and 12.5.12. In native mode, the pins are driven by CONFIG[66].CKSEL.

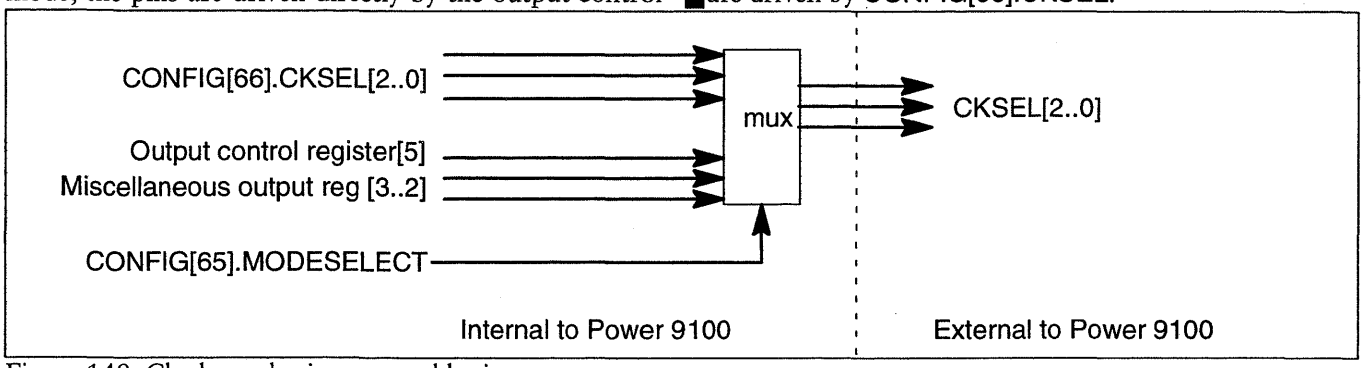



Figure 149. Clock synthesizer control logic

## 12.5. Sequencer Registers, continued

### 12.5.12. POWER 9100 OUTPUT CONTROL REGISTER

The Power 9100 output control register must be unlocked before it can be accessed. See section 12.1.1 for more information.

*REGISTER FORMAT*

Power 9100 Output Control — Port 03C5, Seq Index 12

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| User Defined | | CKSEL [2] | User Defined | | | | |

| Bit No. | Meaning |
|---|---|
| 7..6 | User Defined (UD) |
| 5 | CKSEL[2] in emulation mode |
| 4..0 | User defined (UD) |

Figure 210. Power 9100 output control register format

*FIELD DEFINITION*

This register is implemented on-chip for purposes of read back only.

Bits[7..6] are user defined.

Bit 5 is the most significant bit of CLSEL[2..0] in emulation mode and is used to support programmable clock synthesizers. See section 10.2.

Bits [4..0] are user defined.

*REGISTER DESCRIPTION*

This host I/O read/write register is accessed through location 03C5 hex when the index field of the sequencer index register is 12 hex. Note that this register is specific to the Power 9100 and is not a standard VGA register. This register is only accessible at this location on the Power 9100.

## 13.2. Supported Components

WEITEK has analyzed the AC specifications of a number of VRAM, RAMDAC, and clock generator parts. The products listed below have published AC specifications that are compatible with the Power 9100.

### 13.2.1. COMPATIBLE VRAMS

The following VRAMs work with the Power 9100. Some are not fast enough for 50 MHz operation, but will run at 40 or 45 MHz.

| VRAM Type | 40 MHz | 45 MHz | 50 MHz |
|---|---|---|---|
| Micron MT42C8256–6 | ✓ | ✓ | ✓ |
| Micron MT42C8256–7 | ✓ | ✓ | |
| Micron MT42C8255–7 | ✓ | | |
| Samsung KM428C256–6 | ✓ | ✓ | ✓ |
| IBM IBM025170–60 | ✓ | ✓ | ✓ |
| IBM IBM025170–70 | ✓ | ✓ | |
| IBM IBM025171–60 | ✓ | ✓ | ✓ |
| IBM IBM025171–70 | ✓ | ✓ | |
| NEC µPD482234–60 | ✓ | ✓ | ✓ |
| NEC µPD482234–70 | ✓ | | |
| NEC µPD482235–60 | ✓ | ✓ | ✓ |
| NEC µPD482235–70 | ✓ | | |
| Mitsubishi M5M482256–70 | ✓ | | |
| Mitsubishi M5M482257–70 | ✓ | | |
| Hitachi HM538253–70 | ✓ | | |
| Hitachi HM538254–70 | ✓ | | |
| Toshiba TC528257–70 | ✓ | | |
| OKI M5M548263–70 | ✓ | | |

Figure 295. VRAMs known to work with the Power 9100

### 13.2.2. COMPATIBLE RAMDACS

The following RAMDACs are electrically compatible with the Power 9100. See the "Comments" field for information about software driver support for individual RAMDACs.

| Type | Comments |
|---|---|
| Brooktree Bt 485 | 32-bit RAMDAC |
| AT&T 20C505 | Bt485 compatible, 32-bit |
| Brooktree Bt 885 | 32-bit RAMDAC |
| IBM RGB525 | 64-bit RAMDAC |

Figure 296. Power 9100-compatible RAMDACs

### 13.2.3. COMPATIBLE CLOCK GENERATORS

The following clock generators are compatible with the Power 9100.

| Type | Comments |
|---|---|
| IC Designs ICD2061A | |
| IC Designs ICD2062 | |
| ICS9161 | ICD2061A compatible |

Figure 297. Power 9100-compatible clock generators

## 13.3. AC Specifications

| Param. | Description | Reference Signal | 50 MHz | | Unit | Loading |
|---|---|---|---|---|---|---|
| | | | MIN | MAX | | |
| *VESA Local Bus Interface* | | | | | | |
| $T_{LCT}$ | LCLK cycle time | | 20 | | ns | |
| $T_{LCHL}$ | LCLK high/low time | | 8 | | ns | |
| $T_{G1S}$ | ADR[31..2], BE[3..0]–, M/IO–, RESET–, RDYRNT– input setup time | LCLK | | 7 | ns | |
| $T_{G1H}$ | BE[3..0]–, M/IO–, RESET–, RDYRNT– input hold time | | 0 | | ns | |
| $T_{LRDD}$ | LRDY– output delay time | LCLK | | 10 | ns | 100 pF |
| $T_{LRDV}$ | LRDY– output valid time | LCLK | 3 | | ns | 100 pF |
| $T_{DS}$ | DATA[31..0] input setup time | LCLK | | 7 | ns | |
| $T_{DH}$ | DATA[31..0] input hold time | | 0 | | ns | |
| $T_{DD}$ | DATA[31..0] output delay time | LCLK | | 15+1 LCLK | ns | 100 pF |
| $T_{DTO}$ | DATA[31..0] output turn-off time | LCLK | | 3 | ns | 100 pF |
| $T_{LDD}$ | LDEV– output delay time | ADR[31..2], BE– | | 20 | ns | 33 pF |
| $T_{LRDV}$ | LDEV– output valid time | | 0 | | ns | 33 pF |
| *PCI Bus Interface* | | | | | | |
| $T_{BCT}$ | CLK cycle time | | 20 | | ns | |
| $T_{BCHL}$ | CLK high/low time | | 8 | | ns | |
| $T_{INS}$ | C/BE[3..0]–, FRAME–, IRDY–, IDSL input setup times | CLK | | 7 | ns | |
| $T_{INH}$ | C/BE[3..0]–, FRAME–, IRDY–, IDSL input hold times | | 0 | | ns | |
| $T_{IOS}$ | AD[31..0], PAR input setup times | CLK | | 7 | ns | |
| $T_{IOS}$ | AD[31..0], PAR input hold times | | 0 | | ns | |
| $T_{IOD}$ | AD[31..0], PAR output delay times | CLK | | 11+1 CLK | ns | 50 pF |
| $T_{IOV}$ | AD[31..0], PAR output valid times | | 2 | | ns | 50 pF |
| $T_{IOTO}$ | AD[31..0], PAR turn-off times | | 2 | 15 | ns | 50 pF |

Figure 298. AC Specifications for VL and PCI bus interfaces

## 13.3. AC Specifications, continued

M = one memclk period [ns].

Timings for 50MHz mode depend on the following settings:

mem_config.vram_miss_adj=1

mem_config.vram_read_adj=1

mem_config.vram_write_adj=1

mem_config.vram_sample_adj=1

| Param. | Description | Reference Signal | MIN | MAX | Unit | Loading |
|---|---|---|---|---|---|---|
| *VRAM Interface* | | | | | | |
| $T_{PLL}$ | PLL clock lock time at 50 MHz after reset[1] | | 250 | | clocks | |
| $T_{AA}$ | Access time from column address | | | 2.25M–15 | ns | |
| $T_{AR}$ | Column address hold time[1] | RAS– | 3M–10 | | ns | |
| $T_{ASC}$ | Column address setup time | CAS– | 0.25M–5 | | ns | |
| $T_{ASR}$ | Row address setup time[1] | RAS– | 2M–9 | | ns | |
| $T_{CAC}$ | Access time from CAS– | | | 1.5M–12 | ns | |
| $T_{CAH}$ | Column address hold time | CAS– | 0.75M–5 | | ns | |
| $T_{CAS}$ | CAS– pulse width | | 1M–5 | | ns | |
| $T_{CHR}$ | CAS– hold time during refresh[1] | | 3M–9 | | ns | |
| $T_{CP}$ | CAS– precharge time during page mode | | 0.75M–5 | | ns | |
| $T_{CPA}$ | Access time from CAS– precharge | | | 2.25M–13 | ns | |
| $T_{CPN}$ | CAS– precharge time[1] | | 0.75M–5 | | ns | |
| $T_{CRP}$ | CAS– to RAS– precharge time[1] | | 3M–9 | | ns | |
| $T_{CSH}$ | CAS– hold time | RAS– | 3M | | ns | |
| $T_{CSR}$ | CAS– setup time during refresh[1] | | 2M–9 | | ns | |
| $T_{CWL}$ | Write command to CAS– lead time[1] | | 2M–9 | | ns | |
| $T_{DH}$ | Data input hold time | CAS– | 0.75M–5 | | ns | |
| $T_{DHR}$ | Data input hold time[1] | RAS– | 3.25M–9 | | ns | |
| $T_{DS}$ | Data input setup time | CAS– | 0.25M–5 | | ns | |
| $T_{FSR}$ | DSF setup time[1] | RAS– | 3M–9 | | ns | |
| $T_{MCH}$ | MEMCLK high | | 8 | | ns | |
| $T_{MCL}$ | MEMCLK low | | 8 | | ns | |
| $T_{MCY}$,$T_M$ | MEMCLK period | | 20 | | ns | |
| $T_{MH}$ | Mask data to RAS– hold time[1] | | 2M–9 | | ns | |
| $T_{MS}$ | Mask data to RAS– setup time[1] | | 1.75M–9 | | ns | |
| $T_{OEA}$, $T_{OE}$ | Access time from OE– | | | 1M–5 | ns | |
| $T_{OEZ}$ | Output disable time[1] | OE– | 0 | 15 | ns | |
| $T_{OFF}$ | Output buffer turn-off delay from CAS–[1] | | 0 | 15 | ns | |
| $T_{PC}$ | CAS– page mode cycle time[1] | | 2M | | ns | |
| 1. Not tested, but guaranteed by design. | | | | | | |

Figure 299, continued. Switching characteristics over the operating range

## 13.3. AC Specifications, continued

| Param. | Description | Reference Signal | MIN | MAX | Unit | Loading |
|---|---|---|---|---|---|---|
| *VRAM Interface, continued* | | | | | | |
| $T_{RAC}$ | Access time from RAS– | | | 3M | ns | |
| $T_{RAH}$ | Row address hold time | | 1M–10 | | ns | |
| $T_{RAL}$ | Column address to RAS– lead time[1] | | 2M–5 | | ns | |
| $T_{RAS}$ | RAS– pulse width | | 3M | | ns | |
| $T_{RASP}$ | RAS– pulse width during page mode[1] | | 4M–6 | | ns | |
| $T_{RC}$ | RAS– random cycle time | | 7M–10 | | ns | |
| $T_{RCD}$ | RAS– to CAS– delay time[1] | | 2M–9 | | ns | |
| $T_{RCH}$ | Read command hold time[1] | CAS– | 1M–9 | | ns | |
| $T_{RCS}$ | Read command setup time[1] | CAS– | 0.75M–6 | | ns | |
| $T_{RFH}$ | DSF hold time | RAS– | 1M–10 | | ns | |
| $T_{ROH}$ | RAS– hold time[1] | OE– | 1.5M–9 | | ns | |
| $T_{RP}$ | RAS– precharge time | | 3M–6 | | ns | |
| $T_{RPC}$ | RAS– to CAS– precharge time[1] | | 1M–9 | | ns | |
| $T_{RRH}$ | Read command hold time[1] | RAS– | 1M–9 | | ns | |
| $T_{RSH}$ | RAS– hold time[1] | | 1M–5 | | ns | |
| $T_{RWH}$ | WE– to RAS– hold time[1] | | 2M–9 | | ns | |
| $T_{RWL}$ | Write command to RAS– lead time[1] | | 2M–9 | | ns | |
| $T_{THH}$, $T_{YH}$ | OE– high hold time[1] | | 2.5M–9 | | ns | |
| $T_{THS}$, $T_{YS}$ | OE– high setup time[1] | | 3M–9 | | ns | |
| $T_{TLH}$ | OE– low hold time from RAS– | | 1M–10 | | ns | |
| $T_{TLS}$ | OE– low setup time to RAS–[1] | | 2M–9 | | ns | |
| $T_{TRW}$,$T_{TP}$ | TR(OE–) precharge time | | 8.5M–9 | | | |
| $T_{TRP}$ | OE– to RAS– precharge time[1] | | 6M–9 | | ns | |
| $T_{WCH}$ | Write command hold time[1] | CAS– | 1.5M–9 | | ns | |
| $T_{WCR}$ | Write command hold time[1] | RAS– | 4M–9 | | ns | |
| $T_{WCS}$ | Write command setup time[1] | CAS– | 0.5M–9 | | ns | |
| $T_{WP}$ | Write command pulse width[1] | | 2M–9 | | ns | |
| $T_{WSR}$ | WE– to RAS– setup time[1] | | 2M–9 | | ns | |
| 1. Not tested, but guaranteed by design. | | | | | | |

Figure 299, continued. Switching characteristics over the operating range

## 13.4. VL Bus Pin Configuration

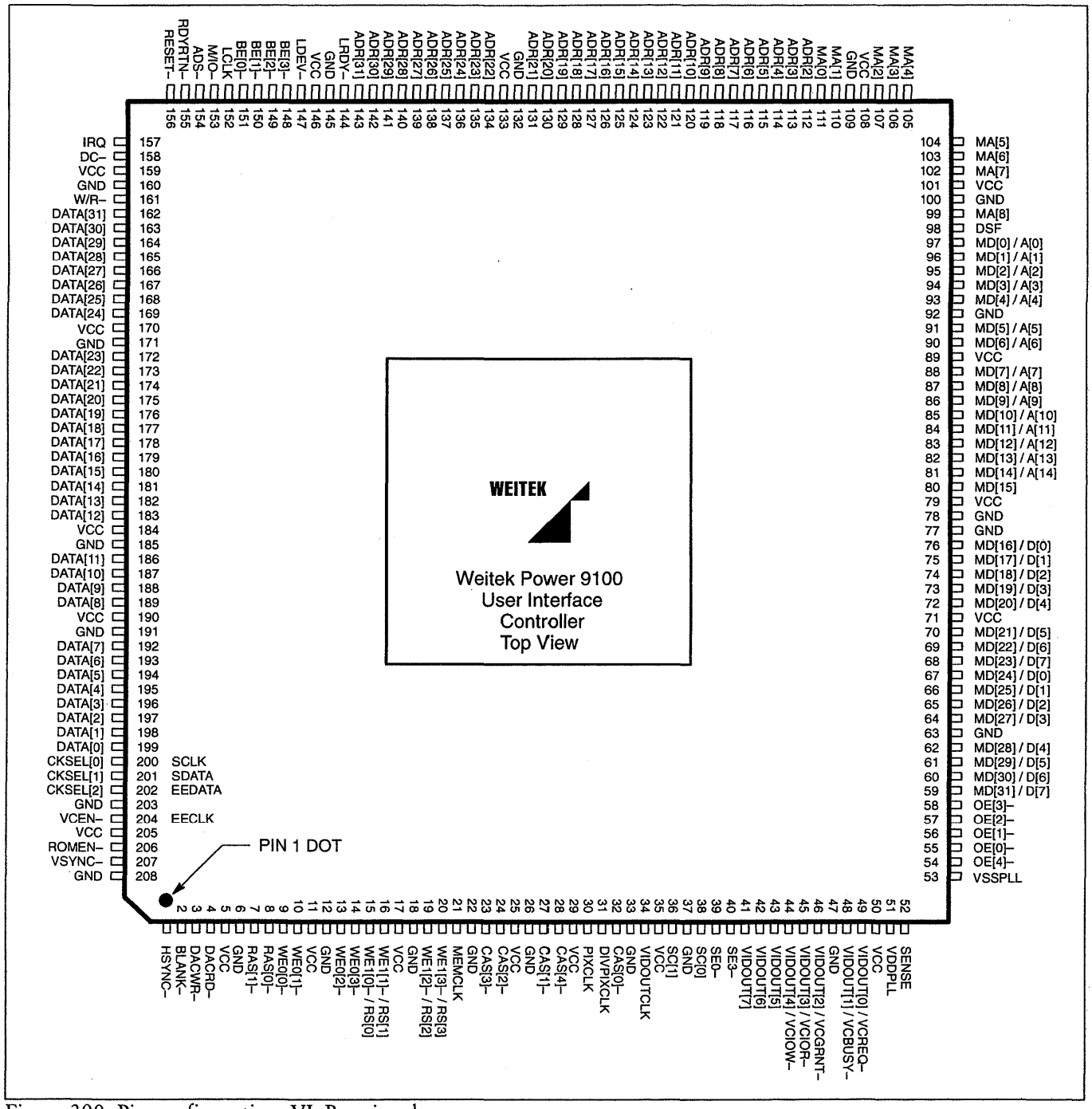



Figure 300. Pin configuration: VL Bus signals

## 13.5. PCI Bus Pin Configuration

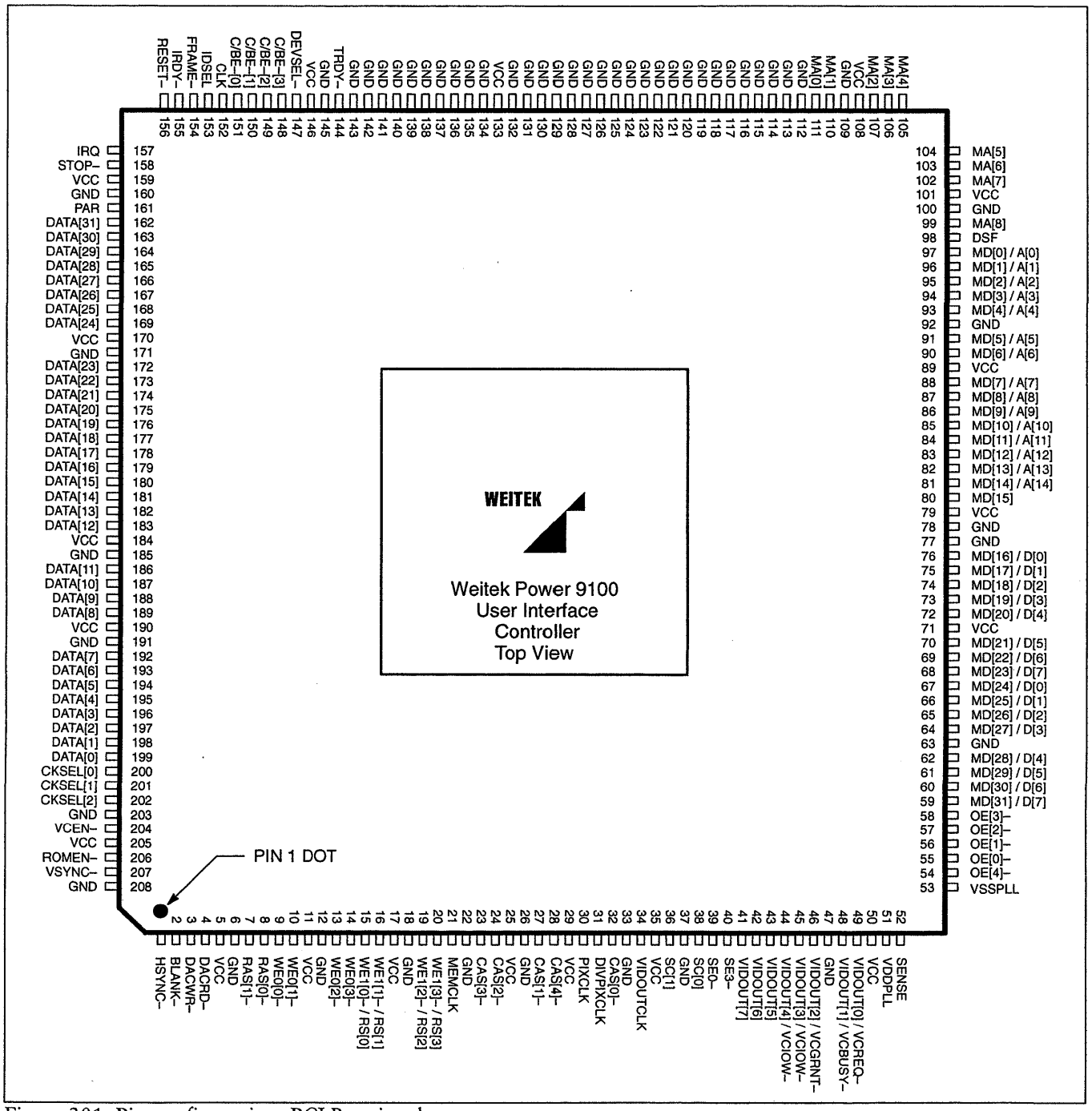



Figure 301. Pin configuration: PCI Bus signals

## 13.6. Pin Assignments, continued

| Pin | Signal / VL Bus | | PCI Bus | | RAMDAC | | BIOS ROM | | Video Coprocessor | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Signal | Type | Signal | Type | Signal | Type | Signal | Type | Signal | Type |
| 129 | ADR[19] | Input | GND | | | | | | | |
| 130 | ADR[20] | Input | GND | | | | | | | |
| 131 | ADR[21] | Input | GND | | | | | | | |
| 132 | GND | | | | | | | | | |
| 133 | VCC | | | | | | | | | |
| 134 | ADR[22] | Input | GND | | | | | | | |
| 135 | ADR[23] | Input | GND | | | | | | | |
| 136 | ADR[24] | Input | GND | | | | | | | |
| 137 | ADR[25] | Input | GND | | | | | | | |
| 138 | ADR[26] | Input | GND | | | | | | | |
| 139 | ADR[27] | Input | GND | | | | | | | |
| 140 | ADR[28] | Input | GND | | | | | | | |
| 141 | ADR[29] | Input | GND | | | | | | | |
| 142 | ADR[30] | Input | GND | | | | | | | |
| 143 | ADR[31] | Input | GND | | | | | | | |
| 144 | LRDY– | Tri-stated | TRDY– | Tri-stated | | | | | | |
| 145 | GND | | | | | | | | | |
| 146 | VCC | | | | | | | | | |
| 147 | LDEV– | Output | DEVSEL– | Tri-stated | | | | | | |
| 148 | BE[3]– | Input | C/BE[3]– | Input | | | | | | |
| 149 | BE[2]– | Input | C/BE[2]– | Input | | | | | | |
| 150 | BE[1]– | Input | C/BE[1]– | Input | | | | | | |
| 151 | BE[0]– | Input | C/BE[0]– | Input | | | | | | |
| 152 | LCLK | Input | CLK | Input | | | | | | |
| 153 | M/IO– | Input | IDSEL | Input | | | | | | |
| 154 | ADS– | Input | FRAME– | Input | | | | | | |
| 155 | RDYRTN– | Input | IRDY– | Input | | | | | | |
| 156 | RESET– | Input | | | | | | | | |
| 157 | IRQ | Output | IRQ– | Open Collector | | | | | | |
| 158 | D/C– | Input | STOP– | Input/ Output | | | | | | |
| 159 | VCC | | | | | | | | | |
| 160 | GND | | | | | | | | | |

Figure 302. Pin assignments (6 of 8)

## 13.6. Pin Assignments, continued

| Pin | Signal / VL Bus | | PCI Bus | | RAMDAC | | BIOS ROM | | Video Coprocessor | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Signal | Type | Signal | Type | Signal | Type | Signal | Type | Signal | Type |
| 183 | DATA[12] | Input/ Output | | | | | | | | |
| 184 | VCC | | | | | | | | | |
| 185 | GND | | | | | | | | | |
| 186 | DATA[11] | Input/ Output | | | | | | | | |
| 187 | DATA[10] | Input/ Output | | | | | | | | |
| 188 | DATA[9] | Input/ Output | | | | | | | | |
| 189 | DATA[8] | Input/ Output | | | | | | | | |
| 190 | VCC | | | | | | | | | |
| 191 | GND | | | | | | | | | |
| 192 | DATA[7] | Input/ Output | | | | | | | | |
| 193 | DATA[6] | Input/ Output | | | | | | | | |
| 194 | DATA[5] | Input/ Output | | | | | | | | |
| 195 | DATA[4] | Input/ Output | | | | | | | | |
| 196 | DATA[3] | Input/ Output | | | | | | | | |
| 197 | DATA[2] | Input/ Output | | | | | | | | |
| 198 | DATA[1] | Input/ Output | | | | | | | | |
| 199 | DATA[0] | Input/ Output | | | | | | | | |
| 200 | CKSEL[0] | Output | | | | | | | | |
| 201 | CKSEL[1] | Output | | | | | | | | |
| 202 | CKSEL[2] | Input/ Output | | | | | | | | |
| 203 | GND | | | | | | | | | |
| 204 | VCEN– | Output | | | | | | | | |
| 205 | VCC | | | | | | | | | |
| 206 | ROMEN– | Output | | | | | | | | |
| 207 | VSYNC– | Input/ Output | | | | | | | | |
| 208 | GND | | | | | | | | | |

Figure 302. Pin assignments (8 of 8)

WEITEK

## WEITEK'S CUSTOMER COMMITMENT:

Weitek's mission is simple: to provide you with VLSI solutions to solve your compute-intensive problems. We translate that mission into the following corporate objectives:

**1.** To be first to market with performance breakthroughs, allowing you to develop and market systems at the edge of your art.

**2.** To understand your product, technology, and market needs, so that we can develop Weitek products and corporate plans that will help you succeed.

**3.** To price our products based on the fair value they represent to you, our customers.

**4.** To invest far in excess of the industry average in Research and Development, giving you the latest products through technological innovation.

**5.** To invest far in excess of the industry average in Selling, Marketing, and Technical Applications Support, in order to provide you with service and support unmatched in the industry.

**6.** To serve as a reliable, resourceful, and quality business partner to our customers.

These are our objectives. We're committed to making them happen. If you have comments or suggestions on how we can do more for you, please don't hesitate to contact us.

**Art Collmeyer**
President


**Headquarters**
Weitek Corporation
1060 E. Arques Avenue
Sunnyvale, CA 94086
TEL (408) 738-8400
TWX 910-339-9545
WEITEK SVL
FAX (408) 738-1185

**Domestic Sales Office**
Weitek Corporation
1060 E. Arques Avenue
Sunnyvale, CA 94086
TEL (408) 738-8400
TWX 910-339-9545
WEITEK SVL
FAX (408) 738-1185

Weitek Corporation
1500 West Park Drive, Building 5
Westborough Office Park
Westborough, MA 01581
TEL (508) 366-9030

**European Sales Headquarters**
Hertfordshire Business Centre
Unit 15B
Alexander Road
London Colney
Herts AL2 1JG
United Kingdom
44-72-782-6973

**Japanese Representative**
4-8-1 Tsuchihashi
Miyamae-Ku
Kawasaki, Kanagawa-Pre
213 Japan
TEL (011) 81-44-852-1135
FAX (011) 81-44-888-3158

# WEITEK

1060 East Arques Avenue
Sunnyvale, CA 94086

Phone: (408) 738-8400
FAX: (408) 739-4374

# P9100 Initialization Notes

**Version: 1.0**
**Revision: 0.01**
**Revision Date: May 24, 1994**

**IMPORTANT NOTICE: This has not been tested. WEITEK Corporation assumes no responsibility for errors in this document, and retains the right to make changes at any time, without notice. Please contact your sales office to obtain the latest specifications.**

**Purpose**
text.

## Intellectual Property



## Trademarks



## Related Documents

Power 9100 Graphics Controller -- Preliminary Data, March 8, 1994
Power 9100 Graphics Controller -- Errata, March 1994
Power 9100/Video Power Board Application Note -- Preliminary, March 1994

## Revision History

| | | |
|---|---|---|
| May 24, 1994 | Version 1.0, Revision 0.01 | - Added A1/A2 pixel clock divides routines |
| May 11, 1994 | Version 0.1α, Revision 0.02 | - Reformatted. |
| April 21, 1994 | Version 0.1α, Revision 0.01 | - First version. |

## Author

Written by Keith Evans.
Edited by Dave Marshall (v1.0, r0.01)

# Table of contents

## Preface

Many people cannot figure out how to program the P9100 video back-end. I have written this document to demonstrate a simple approach for generating correct values for the P9100 MEMCONFIG and SRTCTL registers.

Revisions A0, A1, and A2 can be programmed identically. (NOTE: Rev A0 is not in circulation). Revision A3 will require a minor change in the mem_config.blnkdly field. There are some other positive aspects of A3 that include that we will discuss later.

I have a version of *kdebug* that uses this method. I have tested it on a few board configurations and it seems to work. It took me less than one day to write this document and apply this method to *kdebug*. Source code is available.

## 1. What revision of silicon do I have?

Before you start initializing the P9100 in native mode you can read the revision level as follows:

```
// You must write something into SYSCONFIG before you can
// read the revision level
//
*P9100_SYSCONFIG = 0x00000000;
revision = *P9100_SYSCONFIG & 0x00000007;
switch(revision)
   {
   case 0x00:
         printf("You have A0 or A1 silicon\n");
         break;
   case 0x02:
         printf("You have A2 silicon\n");
         break;
   case 0x03:
         printf("You have A3 silicon\n");
         break;
   }
```

## 2. Invariant fields

The following fields are always programmed with the same values:

```
mem_config.vram_read_sample = 1
mem_config.vram_write_adj = 1
mem_config.vram_read_adj = 1
mem_config.vram_miss_adj = 1
mem_config.slow_host_intf = 1
mem_config.reserved = 0
mem_config.hold_reset = 0
mem_config.dac_mode = 0
mem_config.dac_access_adj = 0 (This feature is broken so just set it to
0)
mem_config.priority_select = 1
```

## 3. mem_config.config[3..0]

This is easy if you separate the field into three sub fields:

```
// NOTE: assume that we will NOT use double buffering
config[3] = 0;

if (VRAM_256K)
   config[2] = 1;
else
   config[2] = 0;

config[1..0] = number_of_banks - 1;
```

# 4. Computing mem_config.{shiftclk_freq, crtc_freq, video_clk_sel}

The P9100 video back-end runs from a load clock (LCLK) that controls the generation of the blanking and synchronization signals and controls the clocking of groups of pixels into the RAMDAC. The relationship between the pixel clock, which controls the rate at which pixels are sent to the display, and the load clock is given by the following simple formula:

$$LCLK[\frac{clocks}{\sec}] = \frac{bpp[\frac{bits}{pixel}]}{RAMDACWIDTH[bits]} * pixclk[\frac{pixelclocks}{\sec}]$$

You can see that the pixel clock is multiplied by the factor $\frac{bpp}{RAMDACWIDTH}$ to generate the LCLK.

The RAMDAC generates a load clock for the P9100 automatically based on its pixel bus width, the current color depth, and the pixel clock. We call this clock that the RAMDAC generates SCLK. This SCLK is fed though the P9100 and then sent back to the RAMDAC where it is called LCLK. This is the preferred mode of operation. (The LCLK is merely a delayed version of the SCLK -- the frequency is not affected.)

## 4.1. RAMDAC divides the pixel clock

When the RAMDAC is dividing the pixel clock just program the registers as follows:

mem_config.shiftclk_freq = mem_config.crtc_freq = 0
mem_config.video_clk_sel =

## 4.2. P9100 divides the pixel clock

Things get slightly more complicated because of a bug in the P9100. Under certain conditions it is necessary for the P9100 Revs A0-A2 to generate the LCLK itself from the pixel clock. In this case the SCLK from the RAMDAC is ignored. I can't give an exact formula for predicting when this bug will occur, here is a WAG (wild ass guess) which should serve for the short term:

$$LCLK * 8 < MEMCLK$$

*NOTE: This should probably only happen at 640x480x8, 640x480x15/16, and 800x600x8. Try running these resolutions on a two bank card with a 64-bit wide RAMDAC and a Rev A0-A2 P9100.*

Alternatively switches in a ".ini" file can be used to determine when the work around is required.

It is important to realize that there are three cases where you do not want to have the P9100 attempt to generate the LCLK from the pixel clock.

(1) You are operating at a color depth of 24[bits/pixel]
(2) The multiplication factor $\frac{bpp}{RAMDACWIDTH}$ is equal to one. (LCLK = pixclk)
(3) You are using a P9100 Rev A3 -- No need with these parts!

In case (1) the P9100 cannot generate the LCLK due to the special clocking requirements for 24bpp

In case (2) it doesn't make sense for the P9100 to divide the clock because the divisor is '1'. It will not work around the bug, and it will make things more confusing when the RAMDAC clock doubler feature is used.

In all of these cases go immediately to the *"RAMDAC divides the pixel clock"* section.

Once you have determined that you want the P9100 to divide the pixel clock it is easy to compute the register values. All that you are trying to do is to match the multiplication factor that the RAMDAC would have used.

The following is the simple formula when the full frequency pixel clock is available to the P9100.

$$\text{mem\_config.shiftclk\_freq} = \text{mem\_config.crtc\_freq} = \log_2\left(\frac{RAMDACWIDTH}{bpp}\right)$$

$$\text{mem\_config.video\_clk\_sel} = 0$$

Sometimes the full frequency pixel clock will not be available to the P9100. Usually this is because the RAMDAC contains some clock multiplication circuitry that must be enabled above a certain pixel clock frequency. Currently the only time that this happens is when a Brooktree Bt485 compatible RAMDAC is used. In this case a half frequency pixel clock is sent to the P9100 and to the RAMDAC. Since the pixel clock has already been divided by two before entering the P9100 we must adjust the formula as follows:

$$\text{mem\_config.shiftclk\_freq} = \text{mem\_config.crtc\_freq} = \log_2\left(\frac{RAMDACWIDTH}{bpp}\right) - 1$$

It's really not very complicated...

## 5. Programming the blank and sync timing

Jacques sent out some very detailed information over EMail. I have not had time to look at it yet. It gives the details of how timing if affected by some of the mem_config fields.

The big picture is this: The blank and sync timing are controlled by the LCLK so you must convert pixel values into pixel group values by multiplying by the multiplication factor $\frac{bpp}{RAMDACWIDTH}$ when you're computing the blank/sync register values.

## 6. Programming mem_config.{shiftclk_mode, soe_mode}

These are very easy to program. First you must compute the number of effective back-end banks in the current configuration. This is easy:

$$effective\_backend\_banks = \frac{32[bits] * actual\_banks}{RAMDACWIDTH[bits]}$$

If this number is ever less than one, then you must reduce the effective width of the RAMDAC. For example if you had one bank of memory connected to a 64-bit wide RAMDAC, then you would have to

configure the RAMDAC as a 32-bit RAMDAC. This is a silly thing to do, so just generate an intelligible error message and wait for customer complaints before supporting this type of configuration.

Now just apply the following simple formula:

$$\text{mem\_config.shiftclk\_mode} = \text{mem\_config.soe\_mode} = \log_2(\textit{effective\_backend\_banks})$$

## 7. Programming srtctl.{qsfselect, src_incs}, and mem_config.vad_shft

These fields have caused many problems in the Windows drivers with the introduction of the half-size shift register (SAM) VRAM. The P9100 supports two basic types of VRAM 128k deep and 256k deep.

### 7.1. Full-size SAM VRAM

For full SAM VRAM the depth of the shift register is the same size as a row of memory:

128k deep VRAM has a row size of 256 => depth of one shift register is 256
256k deep VRAM has a row size of 512 => depth of one shift register is 512

The srtctl.qsfselect controls how frequently the shift registers have to be reloaded. To compute this we have to use the effective depth of the shift registers. The effective depth of all the shift registers in all the banks of memory is given by the following formula:

$$\textit{effective\_shift\_register\_depth} = \textit{depth\_of\_one\_shift\_register} * \textit{effective\_backend\_banks}$$

$$\textit{srtctl.qsfselect} = \log_2(\textit{effective\_shift\_register\_depth}) - 5$$

The srtctl.src_incs controls the address generation for the shift register reloads. The effective row size is determined by the number of memory banks and the depth of a memory row. The width of the RAMDAC does not come into play. It is the effective row size that determines the value loaded into srtctl.src_incs:

$$\textit{effective\_row\_size} = \textit{row\_size\_of\_one\_bank} * \textit{actual\_banks}$$
$$\text{srtctl.src\_incs} = \log_2(\textit{effective\_row\_size}) - 9$$

The remaining field requires no considerations just do it:

$$\text{mem\_config.vad\_shft} = 0$$

### 7.2. Half-size SAM VRAM

The simplistic approach for half-size SAM VRAM is to compute the values as if the VRAM were full-size SAM VRAM and then adjust the values as follows:

```
if (half_size_SAM)
   {
   srtctl.qsfselect--;
   if (srtctl.src_incs != 0)
         srtctl.src_incs--;
   else
```

```
        mem_config.vad_shft = 1;
    }
```

## 8. Programming mem_config.blank_edge

Program this field with:

0 if you want the blank and sync signals to be generated on the rising edge of vidoutclk (LCLK)
1 if you want the blank and sync signals to be generated on the falling edge of vidoutclk (LCLK)

In general the negative edge is preferred and should be used whenever possible. (Some cards in the field WILL break if the positive edge is used.) However the P9100 revisions A1 and A2 cannot use the negative edge when: $f_{LCLK} > 33MHz$. The A3 silicon should be able to use the negative edge at any frequency.

When (blank_edge == 1) and the LCLK frequency is too high the screen may display a horizontal ripple.

When (blank_edge == 0) and there is no series damping on the blank line, then you may see noise on the left and right edges of the screen.

## 9. Programming mem_config.blnkdly

### 9.1. RAMDAC divides the pixel clock

This case is relatively straightforward:

#### 9.1.1. P9100 Rev A0-A2

```
if (mem_config.blank_edge == 0)
    // positive edge vidoutclk generates blank and syncs
    mem_config.blnkdly = 1;
else
    // negative edge vidoutclk generates blank and syncs
    mem_config.blnkdly = 2;

// The following configuration requires special casing because
// an external PAL is used to generate the shift clocks. This
// requires an additional cycle of blank delay to align blank
// properly with the the serial clocks.
//
if ((actual_banks == 4) && (ramdacwidth == 32))
    mem_config.blnk_dly++;
```

#### 9.1.2. P9100 Rev A3

```
mem_config.blnkdly = 1;
// The following configuration requires special casing because
// an external PAL is used to generate the shift clocks. This
// requires an additional cycle of blank delay to align blank
// properly with the the serial clocks.
//
if ((actual_banks == 4) && (ramdacwidth == 32))
    mem_config.blnkdly++;
```

## 9.2. P9100 divides the pixel clock

This section sort of ruins this whole document for me. I can't see any obvious pattern here. This may be because some of the boards are in bad shape. Please use a table look-up and special casing for now. The following tables represent the sum total of my knowledge in this area:

### 9.2.1. P9100 Rev A0-A2

```
// Index these tables with:
// vram type     0 = 128k, 1 = 256k
// ramdacwidth   0 = 32 bits, 1 = 64 bits
// #banks        0 = 1 bank, 1 = 2 banks, 3 = 4 banks
//
// 0xff values indicate illegal indices
// 0x00 values indicate unknowns
//

// Use this table when(mem_config.blank_edge == 0)
positive_blnkdly_table[2][2][3] =
   //   1b    2b    4b
   {{{0xff, 0x00, 0x00},            // 128k VRAM, 32 bit RAMDAC
     {0xff, 0x00, 0x00}},           // 128k VRAM, 64-bit RAMDAC
    {{0x01, 0x02, 0x00},            // 256k VRAM, 32 bit RAMDAC
     {0xff, 0x01, 0x02}}};    // 256k VRAM, 64-bit RAMDAC

// Use this table when(mem_config.blank_edge == 1)
negative_blnkdly_table[2][2][3] =
   //   1b    2b    4b
   {{{0xff, 0x02, 0x03},            // 128k VRAM, 32 bit RAMDAC
     {0xff, 0x00, 0x00}},           // 128k VRAM, 64-bit RAMDAC
    {{0x01, 0x02, 0x03},            // 256k VRAM, 32 bit RAMDAC
     {0xff, 0x01, 0x02}}};    // 256k VRAM, 64-bit RAMDAC

// ***NOTE*** There are two special cases that must be handled!
// These do not make any sense to me, and should never actually
// encountered in normal operation, because at 32bpp you should
// never need the P9100 to divide the pixel clock.
//
if ((bpp == 32) &&(#banks == 1) && (vram type == 256k) && (ramdac width
== 32))
   blnkdly = 2;
else if ((bpp == 32) &&(#banks == 2) && (vram type == 256k) &&
         (ramdac width == 32) && (mem_config.blank_edge == 0))
   blnkdly = 1;
```

### 9.2.2. P9100 Rev A3

Let the RAMDAC divide the clock in all cases.

## 10. Programming the RAMDAC and Frequency Synthesizer

I don't have time to get into the details here. I have attached some pictures showing how the clocks are routed in our most popular designs.

## 11. Miscellaneous Notes

Here are some random programming notes that I have included them for your edification. Weitek driver programmers are probably familiar with all of these points.

## 11.1. ICD2061A Notes

(1) Do not ever program the two phase locked loops to the same frequency. Drivers should contain code which insures that the ICD2061A pixclk and memclk PLLs are always operating at different frequencies.

Here are two examples of how this can happen:

You're using the ICD2061 with a Brooktree RAMDAC and are running with a 50 MHz memclk and select a display resolution of 800x600@72Hz refresh rate which requires a 50 MHz pixclk. If the driver programs the same control values into both PLLs then visible screen jitter will occur.

You're using the ICD2061A to generate a reference clock for the IBM RGB525

(2) After selecting a new pixel frequency you must wait at least ? ms for the frequency to stablize. The host interface of some RAMDACs is not usable until the frequency is stable so this is an important fact. For instance is you are using the ICD2061A to generate a reference clock to the IBMRGB525 RAMDAC and you toggle the clock select bits in VGA to select a new frequency, then a delay should be inserted before the RAMDAC host interface is used. This is currently handled by the BIOS.

(3) There are often several ways to obtain the same output frequency which will produce different levels of clock jitter. Some customers like to pick these control values carefully to produce the most stable display.

## 11.2. IBM RGB525 Notes

(1) Get a copy of IBM's document:

*Programming the RGB525 Clock Generator*
*March 10, 1994*
*David Warfield*
*Revision 1.0*

This explains how the to minimize the clock jitter of the PLL inside their RAMDAC.

(2) Take the DAC access workaround seriously when using this RAMDAC. (See below)

## 11.3. General P9100 Notes

(1) Do not ever try to read any of the screen controller registers if there is not a stable clock on the selected pixel clock input (pixclk/divpixclk). This will hang the P9100.

(2) Emulation (VGA) mode and native mode are mutually exclusive. Switching to one mode will reset the other.

*< < <NOTES ABOUT SAVE AND RESTORE> > >*

(3) Do not read from undefined registers. This may hang the P9100.

(4) Be aware that there is a workaround which must be performed in native mode when accessing the RAMDAC. Every RAMDAC access must be followed by a read from one of the P9100 display controller registers to insure that the minimum RAMDAC access time parameter is not violated.

(5) The mem_config.soe_mode should be set to ? whenver the P9100 is in emulation mode. If this is not done then all of the serial enables may be enabled at the same time. This may result high power consumption and may shorten the life of the VRAM.

## 11.4. VideoPower Notes

Get a copy of my *Video Power Programmer's Guide*.

R. Embry.

SuperMAC / Apple Confidential

WEITEK

POWER 9100 GRAPHICS CONTROLLER

## Lab Report

**Customer Specific**

SMAC0609.DOC

I worked with SuperMAC in their lab yesterday to characterize the incorrect data writes they have been experiencing. We do not understand the problem, but we have two test programs and evidence that the problem maybe related to the revision of the planer board. The PROTO1 planer board did not work, but the EVT1 board had no errors. We are now prepared to continue the lab work with Apple, the designer of the planer board, to learn the specific failure mechanism in the PROTO1 and to determin if we are dealing with a marginallity problem that is also in EVT1s, but not surfacing at SuperMAC.

At this point, SuperMAC does not have any issues with Weitek. Nor do we have any issues with SuperMAC. The issues resolve around Apple's contention that the SuperMAC board does not work in the PowerPC.

TESTS ----------

We first developed two tests and made some basic scope measurements of FRAME#, DEVSEL#, TRDY#, and data bits 4,5 and 6 using the PROTO1 + EVT1 systems. I have source to these programs. They run under MACSBUGS.

The BBTest2.c program is a scope test:
Write 1s to SYSCONFIG, Write 0s to pseudo device coordinate register

The BBTest3.c program counts failures in 10,000 correct writes. The writes (not Reads) are known to fail because of DFB errors. Bit positions 4 and 5 were identified early on. BBTest3 displays 3 counters:

c1= write failures in D4 or D5 bit positions. c2= write failures not in D4 or D5.
c3= number of successful writes (always 10,000).

Both tests have applications that are hardwired to 9000 0000 and D000 0000 for different slot addresses.

A third test was an application that used offscreen memory to blit a 64x64 pattern.

SYSTEMS ----------

PROTO1 system:
20 MHz PCI CLK, 2 bridge system
P9000 based main graphics board (1st bridge?)
P9100 SM (SuperMAC) and P9100 N4C WTK (2nd bridge?)

EVT1 system:
25 MHz PCI CLK, D7++ BIOS/OS
P9100 SM based main graphics board (1st bridge?)
P9100 SM (SuperMAC) or P9100 N4C WTK or P9001 WTK (2nd bridge?)

Our experience with the two systems revealed many differences. Too many differences to list or sort without Apple's help.

RESULTS ----------

Both Systems:
The Scope measurements (BBTest2) confirmed a MED speed DEVSEL and a 1 cycle FRAME#. All register writes seemed to completed in the minimum bus cycle time for a write. Set up times were large (CLK cycle times were 40 to 50 nS), hold times were in the 4 to 7 nS range.

PROTO1:
The BBTest3 program produced failed writes on both the SuperMAC board and the Weitek P9100 boards. Repeated runs (10 or so) were made:

SM Board with and without 2 K ohm pullups on PCI AD4 and AD5 bits: c1= 1 to 3
c2= 0 to 1 (1 in only one run)

WTK Board:
c1= 200 to 300
c2= 1 to 179

Tests were repeated with a technique to determin the failing bits in c2 count:

SM Board (no special pullups):
c2=4 (readback was FFFF FFE7 every time)

The blitting application produced bad pixels.

WEITEK

# POWER 9100 TECHNICAL OVERVIEW

## PRELIMINARY DATA

November 1993

The WEITEK Power 9100 is a high-performance display controller for use with graphical user interfaces such as Microsoft Windows. It combines an extremely high-speed frame buffer controller with a workstation-style accelerated display controller and a local-bus host interface for maximum performance.

## Contents

Power 9100 Technical Overview
November 1993



WEITEK Corporation
1060 East Arques Avenue
Sunnyvale, California 94086
Telephone (408) 738–8400



Written and illustrated by Claire-Marie Costanza and Robert Plamondon

Printed in the United States of America
93 94 6 5 4 3 2 1

4710–9214–00 Rev. B

PRELIMINARY DATA
November 1993

## Features

### *Single-Chip 2-D Graphics Accelerator*

- ◇ Ultra-high-speed local-bus display controller uses workstation display technology to give maximum acceleration with graphical user interfaces such as Microsoft Windows
- ◇ Sophisticated architecture, full-custom chip design, and advanced 0.8 micron process technology deliver maximum power at an affordable cost
- ◇ On-chip SVGA unit for DOS compatibility
- ◇ 208-pin plastic QFP package

### *High Bandwidth*

- ◇ Interleaved VRAM controller uses a PLL timing generator to extract 100% of VRAM bandwidth
- ◇ Sophisticated local-bus controller extracts full bandwidth from PCI and VL host buses
- ◇ Supports 32- and 64-bit RAMDACs at speeds up to 200 MHz for maximum display bandwidth
- ◇ Pipelined internal architecture is 32 bits wide throughout to ensure maximum throughput

### *Full Software Support*

- ◇ Drivers for Windows 3.1, Windows NT, OS/2 Presentation Manager, and AutoCAD R10–R12
- ◇ VESA-compatible BIOS
- ◇ DOS application drivers

### *Powerful Graphics features*

- ◇ Supports high- and true-color at large screen sizes:
  24-bit true-color to 1280x1024 pixels
  16-bit high-color to 1600x1200 pixels
- ◇ Fully accelerates 8-, 15-, 16-, 24- and 32-bits/pixel
- ◇ Draws lines and polygons (with optional pattern fill) at full VRAM bandwidth
- ◇ Performs bit block-transfer (BitBlt) from screen to screen at VRAM bandwidth, and from host to screen (with color expansion) at host bus bandwidth
- ◇ Provides automatic clipping against window and screen edges
- ◇ Supports patterning, plane masking, and boolean operations of pixels during drawing; supports polylines and mesh polygons; supports pick mode
- ◇ Directly mapped linear frame buffer allows efficient CPU access to frame-buffer memory without banking
- ◇ Multimedia support via frame-buffer coprocessor interface
- ◇ Supports VESA power management

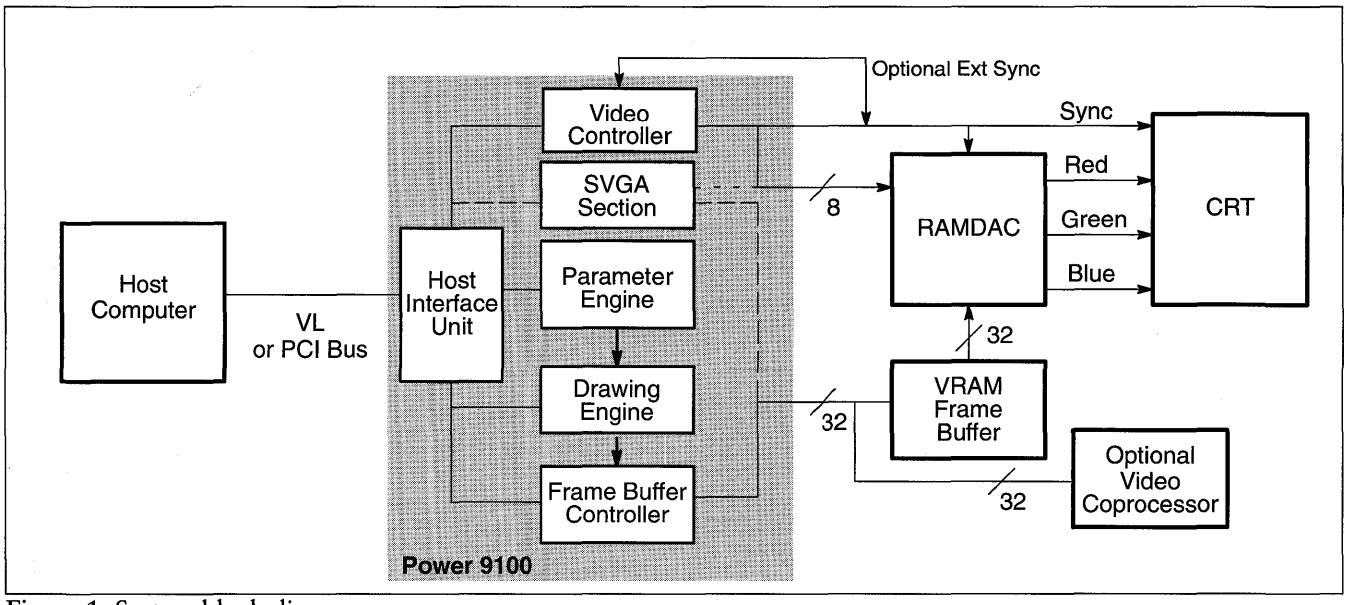



Figure 1. System block diagram

## Architecture

The WEITEK Power 9100 is a hard-wired display processor. Designed specifically for high-performance personal computers, its architecture takes full advantage of the speed of video RAM and high-speed local-bus interfaces.

Based on WEITEK's workstation display controllers, the Power 9100 is the latest in a series of controllers designed from an architectural strategy that can be summarized in two sentences:

1. Get the highest possible bandwidth.
2. Use all of it.

We have found that these simple statements capture the essence of high-performance controller design.

The following sections describe how these principles were applied in the main sections of the Power 9100:

1. Frame buffer controller
2. Parameter engine
3. Drawing engine
4. Host bus interface
5. Video controller

The Power 9100 contains an additional unit:

6. SVGA unit

for backward compatibility.

### FRAME BUFFER CONTROLLER

Capturing the highest possible bandwidth starts at the frame buffer. The Power 9100 uses three techniques to achieve (and use) the highest possible frame buffer bandwidth:

1. VRAM frame buffer
2. Interleaving
3. PLL-controlled memory timing

#### *VRAM FRAME BUFFER*

The Power 9100 uses a VRAM frame buffer for maximum performance. While VRAM is more expensive than DRAM, it is dual-ported; the stream of pixel data needed to refresh the display does not detract from the drawing rate. DRAM systems, in contrast, divide their bandwidth between these two functions, causing drawing rates to degrade as screen sizes increase.

The Power 9100 supports screen sizes of up to four megapixels, and full drawing bandwidth is retained even at these large screen resolutions.

#### *INTERLEAVING*

The Power 9100's two 32-bit banks of VRAM are interleaved for maximum performance.

Interleaving is a technique that takes advantage of the fact that RAM access times are much longer than their write-enable times. Two banks of RAM are placed in parallel, with common address and data lines, but separate write enable lines. One bank contains the even-numbered data words; the other contains the odd-numbered words. With interleaved RAM, writing two words of data only takes one cycle longer than writing one, giving the performance of 64-bit memory, while saving pins and giving a better architectural match to a 32-bit controller.

#### *PLL-CONTROLLED MEMORY TIMING*

To get maximum speed out of memory devices, you have to precisely match a large number of timing specifications. Typical memory controllers approximate this with timing patterns based on the system clock. Since memory speeds and system clock speeds are similar, this approach is too granular to match the memory parameters closely, and it results in significant performance loss.

The Power 9100 can match memory parameters exactly, using a PLL (phase-locked loop) timing generator and a programmable memory controller. This allows the frame buffer to be used at its full theoretical capacity.

### PARAMETER ENGINE

The graphics core of the Power 9100 is crucial to using all of the frame buffer's bandwidth. In addition to providing a low-latency path between the host and the frame buffer, the core must provide large amounts of acceleration if the full frame buffer bandwidth is to be used. The importance of acceleration stems from three factors:

1. Local-bus interfaces, while fast, are not as fast as the Power 9100's frame buffer.
2. General-purpose CPUs are inefficient at simple drawing operations, and cannot perform them at full host-bus bandwidth (let alone full frame-buffer bandwidth).
3. The user is better served if the CPU is running applications, not serving as an inefficient graphics controller.

## Architecture, continued

The Power 9100's graphics accelerator is divided into two loosely coupled graphics engines: the *parameter engine* and the *drawing engine*. The *parameter engine* prepares drawing operations for execution by the drawing engine (which is described in the next section). The parameter engine's basic function is to take input coordinates from the host and convert them to a form usable by the drawing engine. The input parameters include the *x,y* vertices of polygons and the corners of bit block-transfer *(BitBlt)* regions. The parameter engine tests the vertices against window and screen boundaries, tests for exceptions, and performs trivial rejection. Finally, it transfers commands that pass these tests to the drawing engine to be rendered into the frame buffer.

The parameter engine works independently of the drawing engine; the parameters for a new operation can be loaded while the drawing engine is busy.

The parameter engine prepares four kinds of "polygons" for drawing: quadrilaterals, triangles, lines, and points. It also handles screen-to-screen BitBlt and host-to-screen BitBlt.

The parameter engine handles all exception testing, trivial rejection, status reporting, and access to parameter engine registers. Once the parameter engine verifies that a drawing command should be performed (that is, it has valid parameters and has not been trivially rejected), it passes the operation to the drawing engine. The parameter engine's exception testing is very fast, completing in a few cycles.

### DRAWING ENGINE

The drawing engine performs three basic functions:

1. It draws quadrilaterals, triangles, and lines (the *quad* operation).
2. It performs screen-to-screen BitBlt (the *blit* operation).
3. It performs host-to-screen BitBlt (the *pixel1* and *pixel8* operations).

The quad operation draws quadrilaterals, triangles, lines, and points (the last three being treated by the hardware as special cases: quadrilaterals with one or more identical vertices). Triangles, lines, and points can always be rendered correctly, but the drawing engine cannot draw horizontally convex quadrilaterals. That is, it cannot cross from the inside to the outside of the same object more than once per scan line. This means that "bow ties" cannot be drawn (such quads are decomposed into two triangles by the driver software), though "hourglasses" can. See figure 2.

The blit operation copies a rectangular area of the display from one screen location to another.

The pixel1 operation takes monochrome, one-bit-per-pixel data from the host, expands the pixels internally to the current pixel depth (typically by assigning the foreground and background colors to one and zero bits), and writes them to the frame buffer. Up to 32 pixels can be transferred to the Power 9100 in a single word.

The pixel8 operation takes color pixels of 8, 16, or 32 bits, packed as four, two, or one pixel per word, respectively, and writes them to the frame buffer.

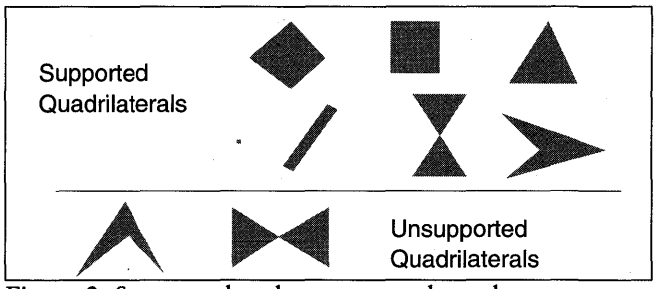



Figure 2. Supported and unsupported quads

### HOST BUS INTERFACE

The Power 9100 connects directly to PCI and VESA local buses. With small amounts of glue logic, it can be connected directly to processor buses.

The Power 9100 itself runs asynchronously to its bus clock. It supports both big-endian and little-endian address formats.

The Power 9100 is memory-mapped; each command has a unique memory address. In many cases, this means that both command and data are specified in a single write operation: The address specifies what operation is to be performed on the data being transferred.

## Architecture, continued

### VIDEO CONTROLLER

A typical board design feeds the VRAM shift registers into a RAMDAC, which converts digital pixel data into an analog RGB video signal. The host initializes the control registers and look-up tables in the RAMDAC through the Power 9100's RAMDAC access instructions.

The Power 9100 also generates horizontal and vertical synchronization and blanking signals, and controls the clocking of the video data. The timing of these signals is programmed through the video controller registers. The Power 9100's divided dot clock is completely asynchronous to the Power 9100's main system clock.

The Power 9100 supports 32- and 64-bit RAMDACs, such as the IBM RGB525, the Brooktree Bt 484 and Bt 485, and the Texas Instruments TVP3010 and TVP3020. By supporting RAMDACs with wide input buses at speeds of up to 200 MHz, the Power 9100 has no difficulty supporting large screens at high color depths and ergonomic refresh rates.

### SVGA UNIT

For compatibility with older applications, the Power 9100 contains an on-chip SVGA unit which supports screen sizes to 1280x1024, and color depths of up to 24 bits. This SVGA unit is independent of the main graphics engine of the Power 9100, although the two share bus, frame buffer, and video interfaces. Control can be switched between the two graphics units under software control.

## Major Differences Between the Power 9100 and the Power 9000

The Power 9100 builds upon the strengths of its predecessor, the Power 9000. This section summarizes the most important differences between the two.

### ADDITIONAL FEATURES

1. On-chip VESA Local (VL) Bus interface
2. On-chip PCI Bus interface
3. On-chip SVGA unit
4. Power 9100 and SVGA modes
5. Full acceleration of 16- through 32-bit graphics
6. Higher clock speeds
7. Four-color pattern RAM (but reduced from 16x16 to 8x8) can be used for four-color dither at full speed
8. Aperture memory mapping (frame buffer can be mapped into a 64 KB aperture in addition to its usual 4 MB linear mapping)
9. Video coprocessor support
10. ROM BIOS control logic
11. Configuration EEPROM control logic
12. Clock generator control logic
13. Serial clock and serial enable generation for VRAMs
14. Text transparency
15. 256 raster-ops

### ADDRESSING

Power 9100 addressing and general address formats are different from those on the Power 9000.

# Details of Graphics Operation

## THE GRAPHICS PIPELINE

Graphics operations flow through the *graphics pipeline,* first through the parameter engine, and then through the stages of the drawing engine.

### *THE PARAMETER ENGINE*

*Functions.* The parameter engine determines what will happen as the result of each drawing operation. It calculates status information, including whether the operation is clipped by the viewing window or the screen edge, and whether the operation can be drawn at all before passing the operation to the drawing engine. If the drawing engine is busy, or if the request contains illegal parameters, the parameter engine does not pass on the request. (The parameter engine performs these tests only on coordinate register loads and drawing commands. Other operations, such as setting color registers or video timing registers, bypass the parameter engine.)

The parameter engine performs clipping calculations on each *x,y* vertex. It compares these points against all four edges of the screen and viewing window, and against each other. It also tests for trivial rejection and trivial acceptance.

The parameter engine detects, but cannot correct, an illegal request, such as a request for a horizontally convex quad. Such quadrilaterals must be rendered in software. The status register flags such problems. In addition, the interrupt signal can be used to interrupt the host when such exceptions occur.

*Access.* All accesses to parameter engine functions and registers complete in a few cycles; the parameter engine is always accessible.

### *THE DRAWING ENGINE*

*Functions.* The drawing engine accepts one drawing operation at a time from the parameter engine. When processing a drawing operation, it first determines which pixels are "touched" by the drawing process *(scan conversion)*, then determines the color value of each touched pixel *(raster ops)*.

*Scan conversion.* Two line-drawing engines follow the left and right edges of the quadrilateral on each scan line, and a pixel-processing engine fills the region between these edges. (Pixel1, pixel8, and blit operations are limited to rectangular areas, while the quad operation can have edges at any angle).

*Raster ops.* The color of each touched pixel is determined by the raster-op function, which is further conditioned by the contents of other registers (see the "Color Selection" section).

*Access.* The drawing engine can remain busy for long periods when drawing large quads or performing a blit operation on a large portion of the screen. The quad and blit operations are started by a read operation. The read requests that the operation take place, and returns the contents of the status register, which indicates whether or not the request has been granted. The Power 9100 does not accept a request if the drawing engine is busy, or if an exception has occurred, nor does it queue requests. Therefore, the software must check the status register, which contains both exception and drawing engine status bits, and resubmit any quad or blit request that is not accepted. While no new drawing operations can be started until the drawing engine is idle, the host can load the parameter engine's coordinate registers with the vertices of a new operation while the drawing engine is busy, thereby overlapping the processing of two objects.

## DRAWING QUADRILATERALS

The drawing engine assigns the two edges at either the topmost or bottom-most vertex to its two Bresenham *line-drawing engines.* These engines do not actually draw anything in the frame buffer, but they traverse the boundaries of the quadrilateral on each scan line. The *pixel-processing engine* then fills in the pixels between the boundaries found by the line-drawing engines. (This filling operation is also clipped against the clipping window and screen boundaries.) See figure 3. When a line-drawing engine reaches another vertex, it starts down the new edge.

In *oversized* mode, the Power 9100 draws perimeter pixels according to the Bresenham algorithm. Oversized mode must be selected to draw points and lines, as X11's drawing rules do not "touch" pixels in zero-width objects (meaning nothing is drawn).

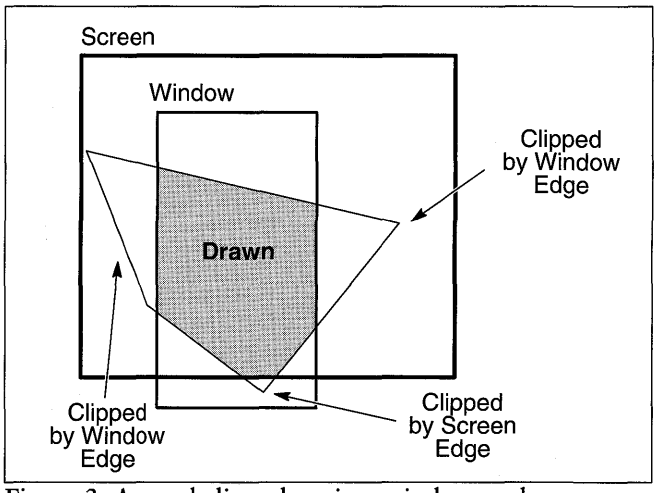



Figure 3. A quad clipped against window and screen

## Details of Graphics Operation, continued

In *X11 mode,* the Power 9100 draws only those pixels whose centers lie within the perimeter of the quad. Pixels whose centers lie precisely on the perimeter are drawn in accordance with X11's tie-breaker rules. (See figure 4.) Coordinates can either be specified relative to the clipping window or relative to the previous vertex.

Polylines, meshed triangles, and meshed quadrilaterals are all supported; they are drawn by transferring the new vertices and issuing another draw command.

### DRAWING BIT MAPS

To transfer a bit map to the screen, the host first sets up the *x,y* coordinates of the destination, and then transfers data to the Power 9100 with the pixel1 and pixel8 operations. The Power 9100 draws the pixels on the screen, auto-incrementing the current *x,y* location after each pixel. Bit-map drawing proceeds from left to right; when it reaches the right-hand edge of the target bit map, the Power 9100 automatically wraps to the next line.

The pixel8 operation draws colored pixels; up to four eight-bit pixels (or two 16-bit pixels, or one 32-bit pixel) are transferred and drawn through a single bus transfer from the host. This mode is used to transfer color images.

The pixel1 operation, which draws monochrome bit maps, is ideal for text transfer. Its basic operation is similar to pixel8, but instead of transferring, say, four eight-bit pixels per operation, it transfers up to 32 one-bit pixels. The Power 9100 expands the one and zero bits of the data word into pixels at the current color depth. (The Power 9100 expands these values according to the raster-op function described in the "Color Selection" section.)

Pixel data must be padded out to multiples of 32 bits per scan line for pixel8 transfers; leftover pixels in the current word do not wrap to the next line. Pixel1 allows the left-over pixels to wrap, however, making it especially suitable for sending narrow blocks of monochrome data, such as character bit maps.

### BIT BLOCK TRANSFER

The blit operation moves a rectangular block of pixels from one part of the screen to another. It handles overlapping source and destination blocks properly; the source block arrives unchanged at the destination, as if it were moved off-screen, then copied back at its new destination.

Like quad, blit is a "fire-and-forget" operation; once initiated, it runs to completion without additional attention from the host. As a large blit can involve over a million pixels, the host driver code should test the busy bit in the Power 9100's status register before attempting another drawing operation when a blit could be in progress.

### COLOR SELECTION

Once a pixel has been touched, there still remains the question of what color it will be. Screen color is selected at five levels:

1. the initial (source) color
2. the pattern color
3. the raster-op color
4. the plane mask
5. the RAMDAC look-up table color (8-bit modes only; higher pixel depths use direct color)

The Power 9100 applies each of these in turn. The *pattern RAM* allows imposition of an 8x8-bit repeating pattern on the data. See figure 5. The *raster-op* function is a three-input boolean function controlled by an eight-bit minterm array. The three inputs are:

1. The source color
2. The destination color (the color value currently at the *x,y* location being written)
3. The color registers color[3..0]

The Power 9100 applies the same function to each of the bits in the pixel.

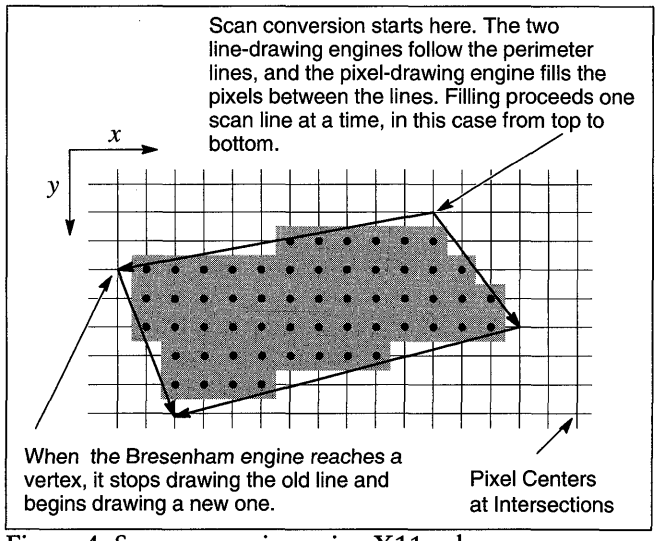



Figure 4. Scan conversion using X11 rules

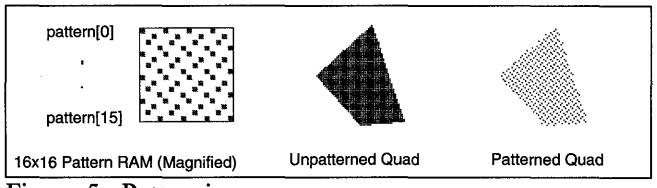



Figure 5. Patterning

## Host Bus Interface

The Power 9100 supports the VL and PCI buses directly, with no glue logic. See figures 6 through 9. Other buses can be accommodated with a small amount of external glue logic.

Some pins on the Power 9100 change function depending on which bus is selected. Rather than introduce a confusing third signal nomenclature to that of PCI and VL, we have provided two sets of pin configurations: one for each bus, each using the signal naming conventions for that bus.

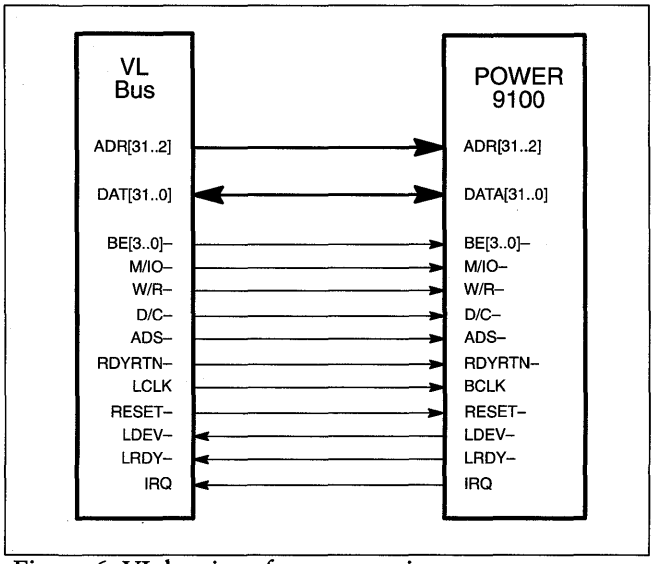



Figure 6. VL bus interface connection

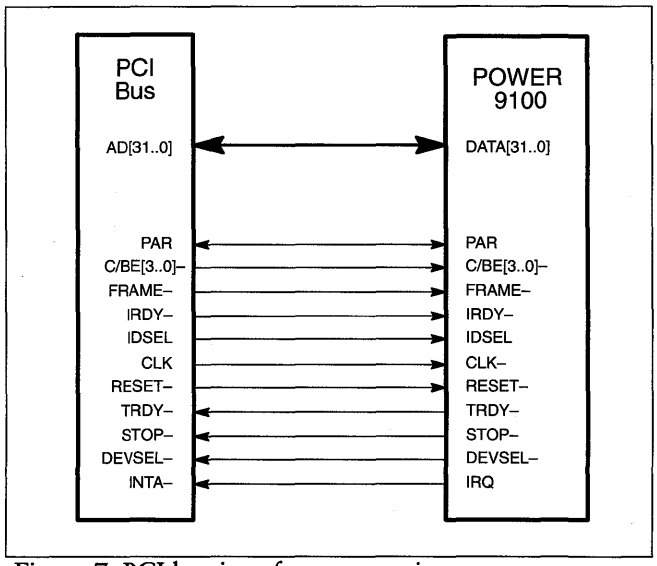



Figure 7. PCI bus interface connection

| VL Bus | | |
|---|---|---|
| **Signal** | **Type** | **Description** |
| DATA[31..0] | I/O | Data bus |
| ADR[31..2] | Input | Address bus |
| BE[3..0]– | Input | Byte enable |
| W/R– | Input | Write or read status |
| M/IO– | Input | Memory or I/O status |
| ADS– | Input | Address data strobe |
| LDEV– | Output | Local device |
| LRDY– | Output | Local ready |
| RDYRTN– | Input | Ready return |
| IRQ | Output | Interrupt request |
| BCLK | Input | VL clock |
| RESET– | Input | Reset |
| D/C– | Input | Data or code status |

Figure 8. VL bus interface signals

| PCI Bus | | |
|---|---|---|
| **Signal** | **Type** | **Description** |
| DATA[31..0] | I/O | Address and data bus |
| C/BE[3..0]– | Input | Bus command/byte enable |
| PAR | I/O | Parity |
| IDSEL | Input | Initialization device select |
| STOP– | Output | Stop |
| FRAME– | Input | Cycle frame |
| DEVSEL– | Output | Device select |
| TRDY– | Output | Target ready |
| IRDY– | Input | Initiator ready |
| IRQ | Output | Interrupt request |
| CLK– | Input | PCI clock |
| RESET– | Input | Reset |

Figure 9. PCI bus interface signals

## Frame Buffer and Video Interfaces

Figure 10 illustrates Power 9100 RAMDAC, frame buffer, and video pin connections.

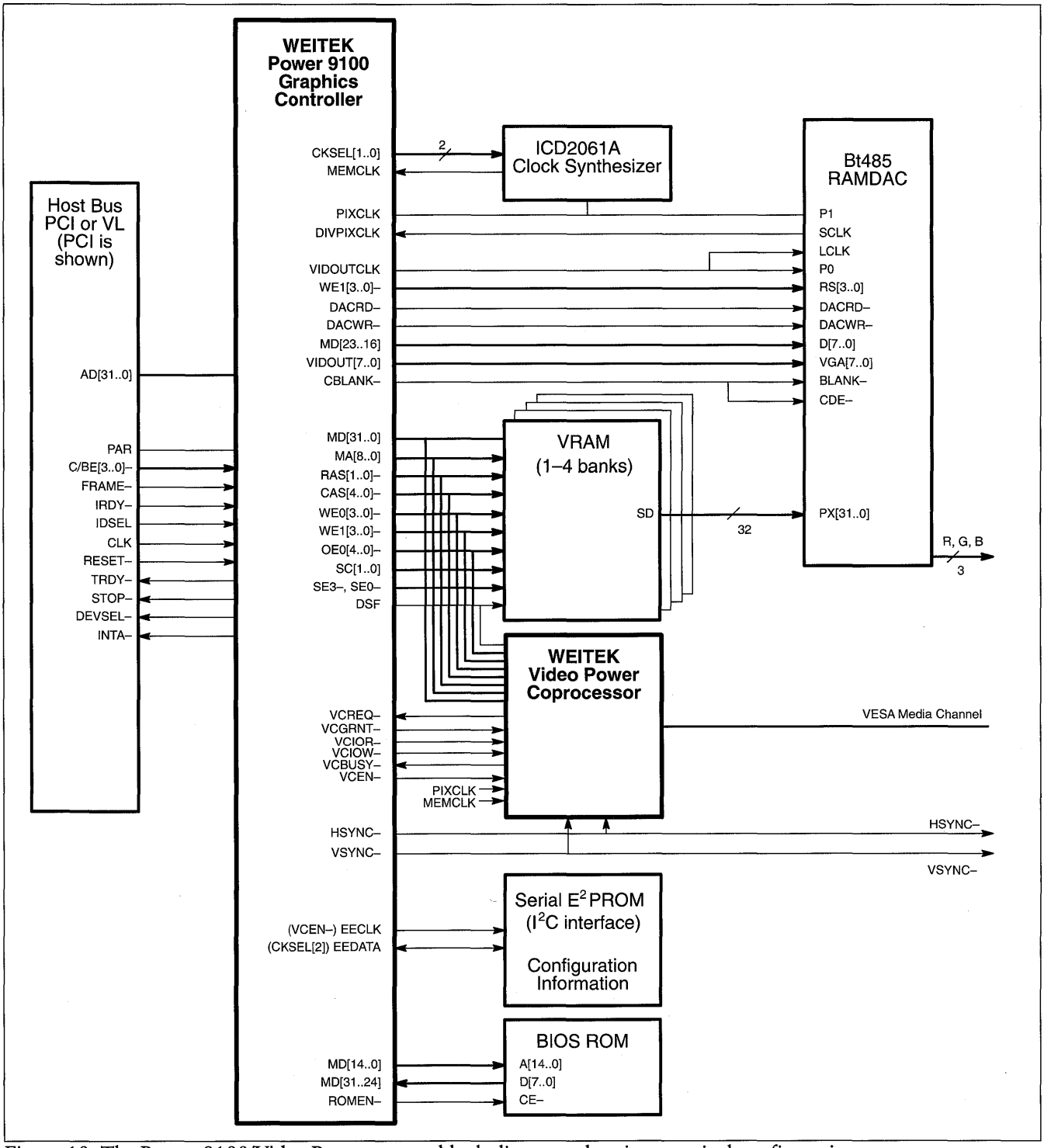



Figure 10. The Power 9100/Video Power system block diagram, showing a typical configuration

## Frame Buffer and Video Interfaces, continued

### SIGNAL DESCRIPTION

| Signal | Type | Description |
|---|---|---|
| MD[31..0] | I/O | Memory data bus |
| MA[8..0] | Output | Memory address bus |
| RAS[1..0]– | Output | Row address strobes |
| CAS[4..0]– | Output | Column address strobes. CAS[0]– controls the least-significant 16 bits of bank 0; CAS[4]– controls the most-significant bits of bank 0 (necessary for VGA transfers). Note that in two-bank systems, bank 1 is controlled by CAS[3]–, not CAS[1]– |
| WE0[3..0]– | Output | Write-enable for the individual bytes in bank 0 (2-bank systems), or banks 0–1 (4-bank systems) |
| WE1[3..0]– | Output | Write-enable for the individual bytes in bank 1 (2-bank systems), or banks 2–3 (4-bank systems) |
| OE[4..0]– | Output | Output Enables. OE[0]– controls the least-significant 16 bits of bank 0; OE[4]– controls the most-significant bits of bank 0 (necessary for VGA transfers) |
| DSF | Output | VRAM special function pin |
| DACRD– | Output | RAMDAC control line |
| DACWR– | Output | RAMDAC write control signal |
| MEMCLK | Input | Main chip clock |
| ROMEN– | Output | BIOS ROM enable |
| EECK | Output | EEPROM clock control signal |
| EEDA | I/O | EEPROM data control signal |
| VDDPLL | Input | Supply voltage for on-chip clock generator |
| VSSPLL | Output | Ground for on-chip clock generator |

Figure 11. Memory control signals

| Signal | Type | Description |
|---|---|---|
| SE[3]–, SE[0]– | Output | VRAM serial shift enable |
| SC[1..0] | Output | VRAM serial shift clock |
| VIDOUT[7..0] | Output | Video data out (SVGA modes) |
| VIDOUTCLK | Output | Video data clock out |
| HSYNC– | I/O | Horizontal synchronization. Normally an output; can also be used as an input for external sync |
| VSYNC– | I/O | Vertical synchronization. Normally an output; can also be used as an input for external sync |
| BLANK– | Output | Blanking interval |
| PIXCLK | Input | Pixel clock |
| DIVPIXCLK | Input | Divided pixel clock |
| CKSEL[2..0] | Output | Frequency synthesizer control |

Figure 12. Video control signals

## Frame Buffer and Video Interfaces, continued

| Signal | Type | Description |
|---|---|---|
| VCBUSY– | Input | Video coprocessor busy. Video coprocessor is busy; I/O reads and writes will not succeed. This signal should be pulled up to VCC with a 10 KΩ resistor |
| VCEN– | Output | Video coprocessor enable. When asserted, video coprocessor is active. Resets video coprocessor when de-asserted |
| VCGRNT– | Output | Video coprocessor bus grant. Signals that the Power 9100 has released the frame buffer |
| VCIOR– | Output | Video coprocessor I/O read. Requests a read from a video coprocessor register |
| VCIOW– | Output | Video coprocessor I/O write. Requests a write to a video coprocessor register |
| VCREQ– | Input | Video coprocessor bus request. Requests that the video coprocessor be given control of the bus. This signal should be pulled up to VCC with a 10 KΩ resistor |

Figure 13. Video coprocessor signal description

## Video Coprocessor Interface

The video coprocessor interface allows a separate coprocessor to share the Power 9100's frame buffer and host interface. Such a coprocessor could provide features to accelerate still pictures, video animation, or 3-D rendering.

The video coprocessor interface allows the host to read and write data from the video coprocessor, while the video coprocessor grant and release functions allow the video coprocessor to take and relinquish control of the frame buffer. The video coprocessor pre-empt function lets the Power 9100 regain control to perform high-priority tasks such as memory refresh.

## Related Documents

For availability of Power 9100 documentation, see your WEITEK sales representative. Upcoming titles include:

*Power 9100 Data Book.* Information on memory map, registers, VGA registers, commands, and host, frame buffer, video, RAMDAC, and video coprocessor interfaces.

*Power 9100 Programmer's Reference Manual.* Full information on registers and commands for both Power 9100 and SVGA modes.

*Power 9100 Application Notes.* Practical techniques for Power 9100 design, including examples of complete board designs for both VL and PCI buses.

*Power 9100 Manufacturing Kits.* Artwork, programmable logic equations, and manufacturing drawings for tested, cost-effective Power 9100 designs.

## Ordering Information

Contact your WEITEK sales representative for information on ordering, pricing, and availability of the Power 9100.

| Package Type | Speed Grade | Temperature Range (Case) | Order Number |
|---|---|---|---|
| 208-pin plastic QFP | 50 MHz | 0-85°C | P9100–050–PFP |

Figure 14. Ordering information

WEITEK

For additional information on WEITEK products, please fill out the form below and mail.

Name _____ Title _____

Company _____ Phone _____

Address _____

Comments _____

I am currently involved in a design with the following WEITEK products _____ and wish to be added to your design data base to insure that I receive status updates.

**APPLICATION:**

- ☐ PERSONAL COMPUTERS
- ☐ ENGINEERING WORKSTATIONS
- ☐ SCIENTIFIC COMPUTERS
- ☐ OTHER
- ☐ LASER PRINTERS
- ☐ TYPESETTERS
- ☐ OTHER PAGE PRINTERS

Check the products on which you would like to receive additional information:

**PERSONAL COMPUTER**

- ☐ Power 9100
- ☐ Video Power
- ☐ Power 9000
- ☐ Abacus 4167

**WORKSTATION**

- ☐ Power μP
- ☐ W8720
- ☐ W8701

| WEITEK use: | Red'd | Out | TPT | Source:DS |
|---|---|---|---|---|
| Status | | | | |

## Power 9100 Technical Overview

Please comment on the quality of this document, and on any corrections or clarifications you would like to see in a future edition.

Fold, Staple and Mail to Weitek Corp.

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

FIRST CLASS PERMIT NO. 1374 SUNNYVALE, CA

POSTAGE WILL BE PAID BY ADDRESSEE

**WEITEK Corporation**
**1060 E. Arques Ave.**
**Sunnyvale, CA 94088–9861**

**ATTN: Christina Parise**

WEITEK

**Headquarters**
Weitek Corporation
1060 E. Arques Avenue
Sunnyvale, CA 94086
TEL (408) 738-8400
FAX (408) 738-1185

**Domestic Sales Office**
Weitek Corporation
6213 Back Bay Lane
Circle "C" Ranch
Austin, TX 78748
TEL (512) 288-2102
FAX (512) 288-0210

**Domestic Sales Office**
Weitek Corporation
1500 West Park Drive, Building 5
Westborough Office Park
Westborough, MA 01581
TEL (508) 366-9030
FAX (508) 366-8121

**European Sales Headquarters**
Hertfordshire Business Centre
Unit 15B
Alexander Road
London Colney
Herts AL2 1JG
United Kingdom
TEL 44-72-782-6973
FAX 44-72-782-6975

**Japanese Representative**
4-8-1 Tsuchihashi
Miyamae-Ku
Kawasaki, Kanagawa-Pre
213 Japan
TEL (011) 81-44-852-1135
FAX (011) 81-44-888-3158