Platinum Memory Controller
ERS v1.0

Apple P/N 343S1184

Jay Rickard
x4-6118

Last Modified: 5/15/95

Apple Computer, Inc.

Table of Contents

1.0 Overview

The Platinum ASIC is a high-speed single chip memory subsystem controller for a PowerPC 601 based Macintosh® computer. It provides all of the functionality needed for a high performance DRAM based memory system as well as a high performance VRAM based video frame buffer. It is designed to work with 60–70 ns DRAMs and 60–70ns VRAMs, and a system bus clock of up to 50 MHz. Platinum also supports one bank of burst mode ROM with an access time of 120/60 or 100/50 ns.

A Platinum based memory subsystem will consist of some or all of the following components (as shown in Figure 1-1):

- Up to eight 128 MB banks of dynamic RAM (DRAM) — the memory which contains program code and data. The memory DIMM's must be the 168-pin, 64-bit wide type to meet the timing and capacitive loading requirements of Platinum.
- One bank of read-only memory (ROM) — the memory which contains the low level toolbox code and the start up routines. Burst mode ROMs are supported by Platinum. An EEPROM DIMM is supported for ROM code development work.
- Up to two 2 MB banks of video RAM (VRAM) — the memory which contains the graphics image to be displayed. VRAM is dual ported memory: one port is accessible via the system bus, the other is used to output pixel data for refreshing the video display. The VRAM DIMMs must be the 112-pin, 32-bit wide type with 1 CAS* and 4 WE* lines.
- A second level cache with 256, 512, or 1024 KB of synchronous static RAM (SRAM) — fast memory which holds recently used data from DRAM and ROM.
- A companion datapath chip, the Iridium ASIC.
- the DACula CLUT/DAC/Pixel Clock Generator — inputs pixel data from the VRAM's and video timing from Platinum; unpacks the pixel data; and produces the output RGB analog video signal.

The Platinum chip features a programmable load clock generator and a frame buffer controller which can be configured to support all Apple monitors in a variety of pixel depths, as shown in Figure 1-2.

The implementation of the Platinum ASIC is discussed in more detail in the following sections.

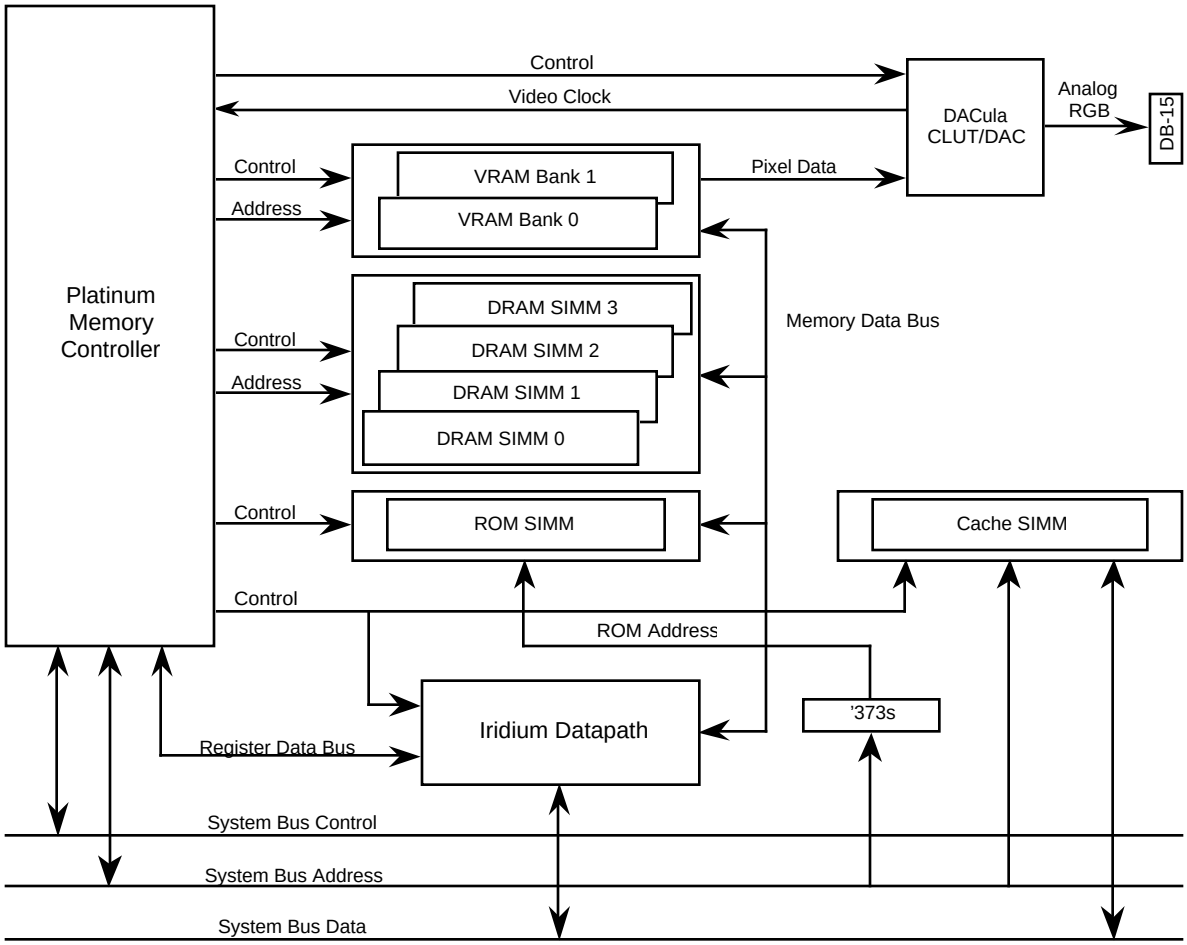


Figure 1-1. Platinum based Memory Subsystem Block Diagram

Display Resolution	*Single Buffered*			*Double Buffered*		
	1 MB VRAM	**2 MB VRAM**	**4 MB VRAM**	**1 MB VRAM**	**2 MB VRAM**	**4 MB VRAM**
512 x 384	8/16/32	8/16/32	8/16/32	N/A	N/A	8/16/32
640 x 400	8/16/32	8/16/32	8/16/32	N/A	N/A	8/16/32
640 x 480 (Mac, VGA, NTSC)	8/16	8/16/32	8/16/32	N/A	N/A	8/16/32
768 x 576 (PAL)	8/16	8/16/32	8/16/32	N/A	N/A	8/16/32
800 x 600	8/16	8/16/32	8/16/32	N/A	N/A	8/16/32
832 x 624	8/16	8/16/32	8/16/32	N/A	N/A	8/16/32
1024 x 768	8	8/16	8/16/32	N/A	N/A	8/16
1152 x 870	8	8/16	8/16 †	N/A	N/A	8/16
1280 x 960	N/A	8	8/16	N/A	N/A	8
1280 x 1024	N/A	8	8/16	N/A	N/A	8

† This case is limited by VRAM serial port bandwidth.

Figure 1-2. Display resolutions supported by Platinum/DACula

2.0 Implementation

Platinum is composed of ten major functional blocks (as shown in Figure 2-1), each of which will be discussed in the following sections:

1) System Bus Interface
2) System Bus Arbiter
3) Configuration/Status Registers
4) Cache Controller
5) DRAM Controller
6) ROM Controller
7) VRAM Controller
8) Video Refresh Timing Generator
9) Video Timing Generator (Swatch)
10) QuickDraw Accelerator

2.1 System Bus Interface

The System Bus Interface logic provides the interface to the processor's system bus for all of Platinum's other subsystems. It is responsible for generating all bus transaction acknowledgment signals and controlling the internal execution order of data bus cycles. It controls the posting of write data to the companion datapath chip, Iridium. It also handles bus timeouts for accesses to non-existent address spaces.

The system bus interface generates AACK to acknowledge address bus transactions for all Platinum controlled address spaces. These include DRAM ($0000 0000 – $7FFF FFFF), VRAM ($F1xx xxxx), ROM ($FFxx xxxx), and Platinum's configuration register space ($F000 0000 – $F2FF FFFF). Normally AACK is asserted one clock after TS, but it may be delayed if there is a data bus transaction to memory in progress. This enforces the strict ordering of data bus cycles and eliminates the need to latch more than one address inside Platinum. Along with AACK, the bus interface logic generates an internal request signal that tells the various subsystems when a valid access is occurring.

All writes to memory are posted to the datapath chip Iridium so they complete very quickly as far as any system bus master is concerned. The system bus interface generates the control signals needed to write the data into Iridium and the data bus acknowledge signal TA. When all words of the write transaction are loaded into Iridium, the bus interface generates an internal write request to the DRAM, VRAM, or ROM controller.

The system bus interface maintains an 8-bit counter for address bus timeouts. If an address bus cycle to a non-PCI address is not acknowledged within 256 clocks, Platinum will assert AACK, the appropriate data bus grant, and then TA and TEA to terminate the transaction with a bus error. PCI versus non-PCI addresses are determined by the contents of the PCI Address Mask register. PCI spaces are ignored by the watchdog timer; they must be acknowledged by the PCI subsystem logic for proper behavior of the system.

Platinum treats the address range $F8300 0000 – $F8FF FFFF differently than any other space. It will assert AACK to acknowledge accesses to this space, but it will not assert TA to terminate the data bus transaction. Any reference to this space will cause the system to fail, so this space must be treated as an invalid address range.

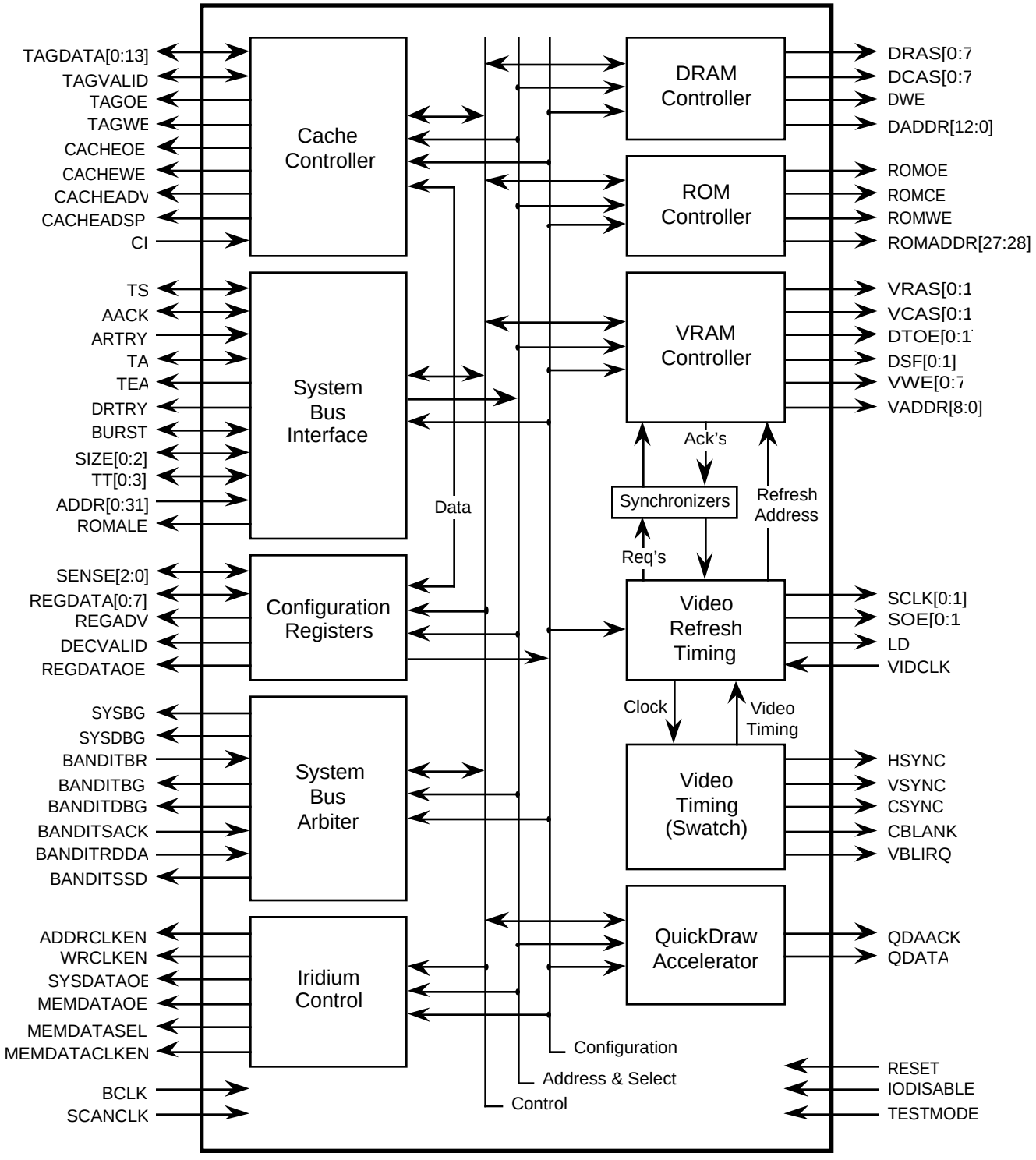


Figure 2-1. Platinum Block Diagram

2.2 System Bus Arbiter

The system bus arbiter logic controls all master accesses to the address and data buses. It is designed to support a default master (a PPC 601 or 603 processor), an I/O interface bus master (the Bandit ASIC), and an internal master that generates address-only cycles (the QuickDraw Accelerator, or QDA). The bus protocol is a modified form of the ArBus split transaction protocol. Address bus cycles may be pipelined and overlap data bus cycles, but data bus cycles are strictly ordered with respect to their associated address bus cycle. The arbiter consists of two state machines, one to control the address bus arbitration, and one to control the data bus arbitration. The two state machines are synchronized to maintain the correct ordering of the data bus transactions. All accelerator accesses and some processor accesses are address-only cycles. Both address and data arbiters recognize the address-only cycles. The data arbiter ignores them and the address arbiter responds immediately, completing an address-only bus cycle in 2 clock periods.

Figure 2-2 shows a sample sequence of address and data bus transactions. It illustrates how address transactions may be pipelined. In order to maintain the strict data bus ordering, the arbiter/bus interface logic does not acknowledge a pipelined address cycle until the completion of the previous data cycle. The only exception to this behavior occurs during address-only cycles, which have no corresponding data cycles. Address-only cycles 3 and 4 in the illustration are pipelined and completed during the data bus cycle for transaction number 2. The address bus cycle for transaction 5 is allowed to start before the data bus cycle for transaction 2 finishes. Since transaction 5 is not an address-only cycle, it is not acknowledged until data bus transaction 2 completes.

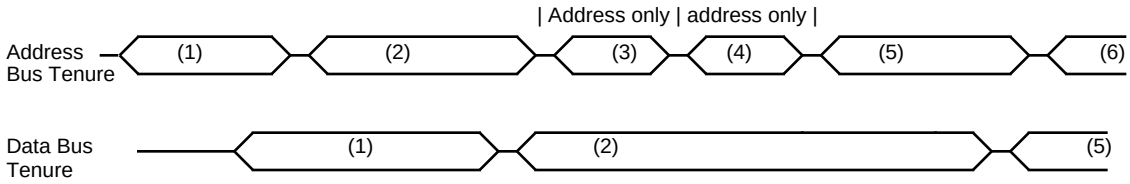


Figure 2-2 Sample of address/data bus transactions

2.2.1 Address Bus Arbiter

The address bus arbiter controls ownership of the pipelined system address bus. It grants the bus to the three possible masters, the processor, Bandit, and the QDA, in a more or less round-robin fashion. The processor is always granted the bus for at least one clock period during each round of arbitration, even if it doesn't need the bus. The bus request signal from the processor is not used by the arbiter , which assumes the processor always wants the bus. Also, since the address bus busy signal is not used by any device, the arbiter guarantees the various address bus grants do not overlap to prevent any bus contention. If no other master requests the address bus, it is granted to the processor by default.

Normally one of the three bus masters will always have control of the bus, but there are two special cases where the arbiter denies the bus to all masters. When the cache controller needs to update a cache tag, either to allocate an new line or to invalidate an old line, it must use the address bus to access the L2 cache tag store. The arbiter takes the address bus away from all masters temporarily so the cache controller can do its update. The arbiter then returns to its normal sequence of bus grants. The second special case occurs when Bandit is accessed as a slave device. Bandit was designed for a full split transaction bus and is capable of pipelining two accesses. The arbiter is designed to handle only one pipelined access. To prevent Bandit from acknowledging two accesses and thereby confusing the arbiter logic, the arbiter negates all address bus grants until Platinum tells Bandit to start the data transaction, which it indicates by asserting BANDITSSD. This forces Bandit to conform to the overlapped address

transaction protocol the arbiter supports. Once the data bus transaction starts the arbiter will grant the address bus to the next requesting master.

Figure 2-3 shows the state transition diagram for the address bus arbiter. The numbered transitions are described below.

(1) Negate SYSBG and go to state IDLE_1 when Bandit or the QDA request the address bus, or when the processor has started an address cycle and not received AACK, or when the cache controller needs to update an L2 tag entry, or when the processor starts a PCI space access.

(2) Assert SYSBG and go to state CPU when Bandit and the QDA negated their bus requests without starting a new address cycle, and there is no address transaction in progress, and tag updates are not needed, and BANDITSSD was asserted for a previous PCI space access.

(3) Assert *QDA_BG* (an internal signal) and go to state QDA when it requests the bus and Bandit does not, and the previous address transaction is done, and tag updates are not needed, and BANDITSSD was asserted for a previous PCI space access.

(4) Assert BANDITBG and go to state BANDIT when it requests the bus, and the previous address transaction is done, and tag updates are not needed, and BANDITSSD was asserted for a previous PCI space access.

(5) Negate BANDITBG bus grant and go to state IDLE_2 when Bandit negates its request, or when Bandit has started an address cycle and not received AACK, or when the cache controller need to update a tag entry.

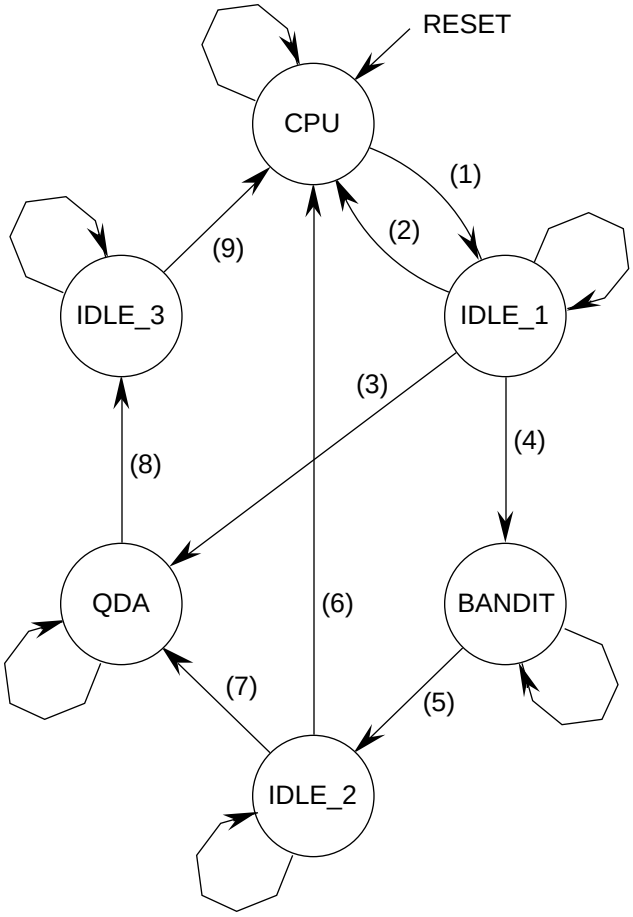


Figure 2-3 Address Bus Arbiter State Diagram

(6) Assert SYSBG and go to state CPU when the QDA is not requesting the bus, and the previous address transaction is done, and tag updates are not needed.

(7) Assert *QDA_BG* and go to state QDA when the QDA requests the bus, and the previous address transaction is done, and tag updates are not needed.

(8) Negate *QDA_BG* and go to state IDLE_3 when the QDA no longer needs the bus, or when the QDA has started an address cycle and not received AACK, or when the cache controller needs to update a tag entry.

(9) Assert SYSBG and go to state CPU when the previous address transaction is done, and tag updates are not needed.

2.2.2 Data Bus Arbiter

The data bus arbiter controls ownership of the system data bus. It is designed to handle two masters only, the processor and Bandit. The third address bus master, the QDA, performs address-only bus transactions so it does not need to access the data bus. The data bus arbiter enforces strict data bus tenure ordering

The arbiter is based on a four entry FIFO that tracks data bus requests by the two masters. Each entry is a single bit with 0 representing the processor and 1 representing Bandit. Two pointers are used to indicate the current master (PTR1) and the next, pipelined, master (PTR2). When the two pointers point to the same FIFO entry there are no pending data bus requests, and by default the processor is given the data bus. This allows for the fastest possible cache read hit timing for the processor. PTR1 is incremented when the last TA of a data transaction is seen. When a non-address-only cycle starts, the data arbiter sets the FIFO entry pointed to by PTR2 to 1 if Bandit is the master and to 0 if the processor is the master. PTR2 is then incremented to point to the next FIFO entry. PTR2 is decremented if the address transaction is terminated by ARTRY.

Figure 2-4 and the following paragraphs illustrate the behavior of the data bus arbitration logic for a sample sequence of address and data bus tenures.

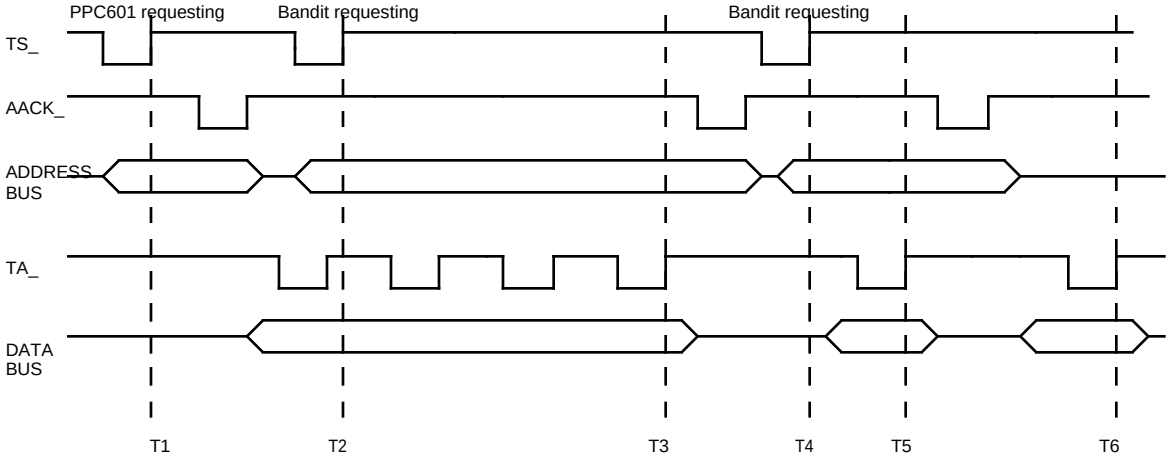
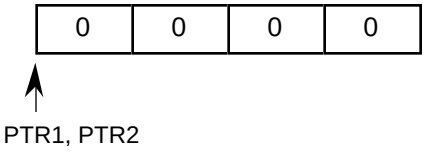
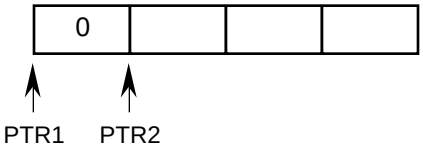


Figure 2-4 Address/Data bus arbitration sequence

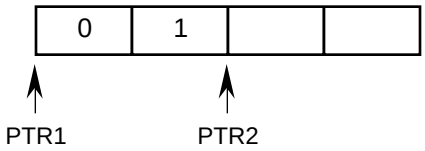
(T0) At the beginning of time, the pointers PTR1 and PTR2 and the FIFO are initialized to zero.



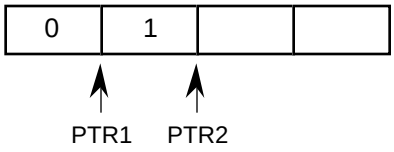
(T1) The start of a processor cycle is detected when TS is asserted. The FIFO element pointed to by PTR2 is set to "0" since the current address bus master is the processor, and then PTR2 is incremented to the next entry.



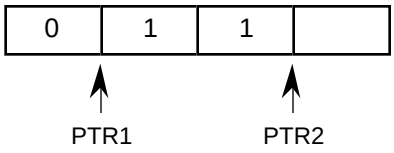
(T2) Bandit starts a pipelined address cycle. The FIFO element pointed to by PTR2 is set to “1” since the current address bus master is Bandit, and PTR2 is increment to the next entry.



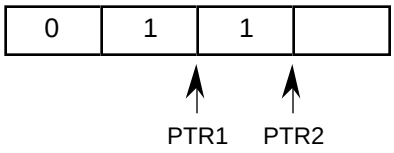
(T3) The end of the data bus transaction associated with the first address bus cycle is detected when the fourth TA of the burst is asserted. PTR1 is incremented to remove the first entry from the FIFO.



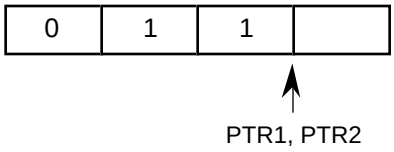
(T4) Bandit starts another pipelined address cycle. The FIFO element pointed to by PTR2 is set to "1" since the current address bus master is Bandit, and PTR2 is incremented.



(T5) The end of the data bus transaction associated with the first Bandit address bus cycle is detected when TA is asserted. PTR1 is incremented to remove the second entry from the FIFO.



(T6) The end of the data bus transaction associated with the second Bandit address bus cycle is detected when TA is asserted. PTR1 is incremented to remove the third entry from the FIFO. Since there are no more pending data bus cycles, the two pointers point to the same FIFO entry.



2.3 Configuration/Status Registers

Platinum contains a large number of configuration and status registers. These are used for setting up the DRAM addressing and timing, configuring the ROM access timing, enabling the L2 cache controller, programming the behavior of the video timing block, setting up the VRAM addressing and video refresh scheme, setting the VRAM operating mode, controlling and clearing interrupts, and resetting the Platinum hardware. A complete description of the function of all the Platinum registers is found in section 4.7.

The Registers module contains the physical registers used to hold the above configuration values. From these registers, configuration information is bused to other Platinum modules. This module also inputs interrupt status from the DRAM and VRAM controllers and the Swatch block in order to make this information available via a ‘register’ read operation.

Because of pin number constraints, Platinum does not directly connect to the system data bus. Instead it passes all register data through Iridium to the system bus. Both Platinum and Iridium contain logic that multiplexes 32-bit register data to and from the 8 bit REGDATA bus that goes between Platinum and Iridium. Logic in Platinum controls the interface in Iridium via the REGDATAOE and REGADV signals.

Individual register select lines are decoded from the latched system address. These register select lines are combined with control signals from the System Bus Interface module to individually enable read/write operations from/to the registers. All registers in Platinum are 32 bits wide (although not all bits are used in each register) and must be accessed as long word quantities with a byte offset of 0. This means all Platinum registers are read and written on the most significant 4 bytes of the processor's 8 byte

wide data bus. Individual registers are located at 16 byte offsets to be consistent with the MacRISC architectural conventions.

The QuickDraw accelerator registers are present in both Platinum and Iridium. To facilitate diagnostic testing of the two chips, these registers can be written at $F800 04xx and at $F800 05xx. The registers physically located in Platinum can be read at $F800 04xx, and the registers physically located in Iridium can be read at $F800 05xx.

2.4 Cache Controller

The cache controller logic handles all accesses to a secondary, or L2, cache made up of fast static RAM. The L2 cache is a direct-mapped, write-through cache with allocate on burst read miss. Platinum is designed to use synchronous static RAM for the cache data store and asynchronous static RAM for the cache tag store. The size of the cache can be 256, 512, or 1024 KB, and is automatically determined during system reset via dAddr[8:6]. By design, only DRAM ($0000 0000 – $7FFF FFFF) and ROM ($FFxx xxxx) spaces are cacheable in the L2 cache.

The cache controller is designed to support back to back 2-1-1-1 burst reads. It achieves this by assuming a burst read to cacheable DRAM and ROM addresses is a hit and immediately responding with TA and cache data. One clock later the cache tag comparison is complete and the controller can decide if the access was really a hit. If the access is a valid hit it proceeds normally. If the access is really a miss, the controller asserts DRTRY to invalidate the first TA cycle and then proceeds with a burst read miss cycle. If the cache is enabled and CI is not asserted, the cache controller will allocate a new line by writing the current address to the tag store along with a valid bit. As the memory subsystem returns the burst read data the cache controller will write it into the cache data store to complete the line allocation.

Since the L2 cache is write-through, all writes are posted to the memory subsystem. Writes to DRAM and ROM space that hit in the cache and meet certain requirements cause the L2 cache to be updated. Burst writes and 8-byte single writes that hit will generate a cache data write regardless of the state of the CI signal. If the cycle is a single beat write of 1–7 bytes, the cache line is invalidated rather than updated since the controller has only one write signal for the entire 8 byte word.

The cache controller will invalidate lines in the L2 cache under a number of circumstances. Whenever a QuickDraw accelerator address-only cycle hits in the L2 cache, the referenced line is invalidated. Writes of less than 8 bytes that hit also cause the line to be invalidated. Accesses to cacheable memory spaces (DRAM and ROM) that end with ARTRY asserted cause the referenced cache line to be invalidated regardless of the state of the CI signal or the internal hit/miss determination.

2.5 DRAM Controller

The DRAM Controller logic generates all memory cycles for the 8 DRAM banks based on access requests from the System Bus Interface module or the DRAM refresh logic. The module is composed of a master state machine and additional logic for row/column address multiplexing, page mode access detection, and DRAM refresh timing. The DRAM Controller operates synchronously with the System Bus Interface module (i.e., it is clocked by the system bus clock).

The core of the DRAM Controller module is the state machine which generates all cycles and control signals (DRAS, DCAS, and DWE). A diagram of the state machine is shown in Figure 2-5. The state machine normally resides in one of two idle states, one for page mode operation and one for "normal" operation. From the idle states, a number of different types of cycles may be started. A valid access request will cause a transition from the idle state:

- If the request is for a DRAM refresh cycle, the state machine asserts an acknowledge signal (rACK) and then performs a CAS-before-RAS (CBR) refresh cycle on all 8 banks of memory. The DRAS pulses are staggered to reduce peak current loads. After one refresh cycle is performed, the state machine returns to the IDLE or PIDLE state after one or more RAS precharge states. Extra wait states are added if the ‘CBR Delay 1’ and/or ‘CBR Delay 2’ configuration bits are set.

- If the request is for a QuickDraw accelerator access, the controller asserts QDAACK and starts a read or write cycle. Following the first read or write cycle a check is made to see if the access is a burst operation. If so, then an additional 3 read or write cycles are performed Once the transfer is completed, the state machine returns to the IDLE or PIDLE state after one or more RAS precharge states.

- If the request is for a system read or write access, the controller starts a read or write cycle. Following the first memory cycle a check is made to see if the access is a burst operation. If so, an additional 3 read or write cycles are performed. Once the transfer is completed, the state machine returns to the IDLE or PIDLE state after one or more RAS precharge states.

- If the state machine has started a system read or write access and the request signal is negated, the L2 cache has detected a hit and responded to the cycle (or ARTRY was asserted to abort the access). The controller will finish a RAS-only refresh cycle by entering the BAIL1 state and then return to the IDLE or PIDLE state after one or more RAS precharge states.

If the state machine is in the PIDLE state and a refresh request or an access request to a different page occurs, the state machine will perform one or more RAS precharge cycles. After the precharge cycles the IDLE state will be entered, and the request will then be serviced.

2.6 ROM Controller

The ROM controller logic handles all accesses to one bank of ROM memory. The access speed is programmable with 6–9 cycles for the first access and 3–6 cycles for the second to fourth accesses. There are two parameters to allow for burst mode ROMs. Writes to ROM space are acknowledged normally, but the ROM is not enabled so no bus contention occurs.

The ROM controller can also control one bank of Flash Memory in place of the normal ROM bank. A Flash Memory DIMM is used to speed up the ROM development cycle. If the ROM Write Enable bit is set, single beat writes to the ROM space cause the controller to generate a write enable signal to allow programming of the Flash Memory DIMM. Writes must be 8-byte single accesses since burst writes to ROM space are not supported and there is a single write enable for the entire bank of ROM.

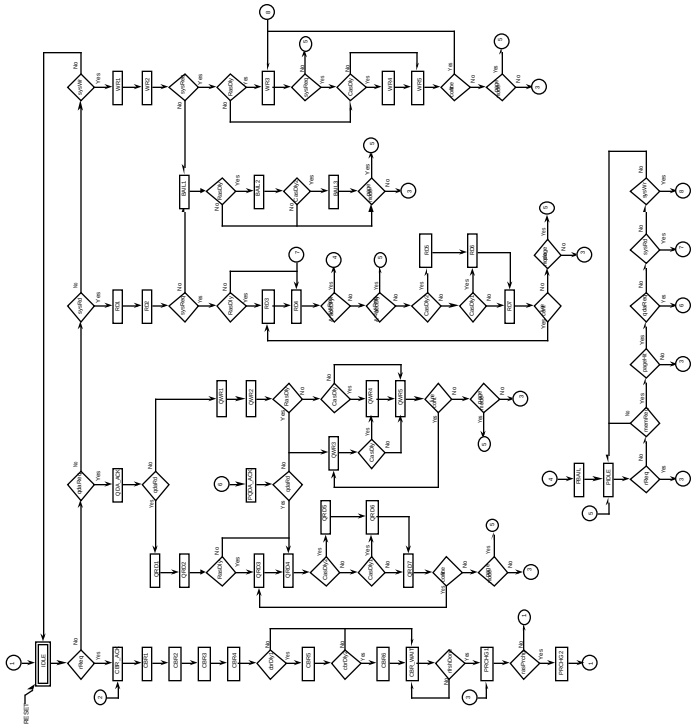


Figure 2-5 DRAM State Machine State Diagram

2.7 VRAM Controller

The VRAM Controller logic generates all memory cycles for the 2 VRAM banks based on access requests from the System Bus Interface module or the Video Refresh Timing Generator. The module is composed of a master state machine and additional logic for the generation of bank-specific control signals, row/column address MUXing, and page mode access detection. The VRAM Controller operates synchronously with the System Bus Interface module (i.e., it is clocked by the system bus clock).

The core of the VRAM Controller module is the state machine which generates all cycles and control signals (VRAS, VCAS, VWE, DSF, and TR/OE). A diagram of the state machine is shown in Figure 2-6. The state machine normally resides in one of two idle states, one for page mode operation and one for "normal" operation. From the idle states, a number of different types of cycles may be started. A valid access request will cause a transition from the current idle state:

- If the request is for a SAM read transfer cycle, the state machine asserts an acknowledge signal (VACK) and then performs either a full read transfer or a split read transfer cycle, based on the state of the splitXfer input. Extra wait states will be inserted if the ‘Xfer Delay 1’ and/or ‘Xfer Delay 2’ configuration bits are set. Once the transfer is completed, the state machine returns to the IDLE or PIDLE state after one or more RAS precharge states.

- If the request is for a VRAM refresh cycle, the state machine asserts an acknowledge signal (RACK) and then performs a CAS-before-RAS (CBR) refresh cycle. After one refresh cycle is performed, the state machine returns to the IDLE or PIDLE state after one or more RAS precharge states. Extra wait states are added if the ‘CBR Delay 1’ and/or ‘CBR Delay 2’ configuration bits are set.

- If the request is for a QuickDraw accelerator access, the controller asserts QDAACK and starts a read or write cycle. Following the first read or write cycle a check is made to see if the access is a burst operation. If so, then an additional 3 read or write cycles are performed Once the transfer is completed, the state machine returns to the IDLE or PIDLE state after one or more RAS precharge states.

- If the request is for a system read or write access, the controller starts a read or write cycle. Following the first memory cycle a check is made to see if the access is a burst operation. If so, an additional 3 read or write cycles are performed. Once the transfer is completed, the state machine returns to the IDLE or PIDLE state after one or more RAS precharge states.

- If the state machine has started a system read or write access and the request signal is negated, the L2 cache has detected a hit and responded to the cycle (or ARTRY was asserted to abort the access). The controller will finish a RAS-only refresh cycle by entering the BAIL1 state and then return to the IDLE or PIDLE state after one or more RAS precharge states.

If the state machine is idling in page mode and a SAM read transfer request, a refresh request, or a memory request to a different page occurs, then the state machine will perform one or more RAS precharge cycles. If the request is for a memory access or refresh cycle, then the idle state will be entered and the request will then be serviced. Otherwise it’s a SAM read transfer and the request is serviced directly after the RAS precharge cycle(s) since the VACK-assertion state satisfy the remainder of the RAS precharge requirement.

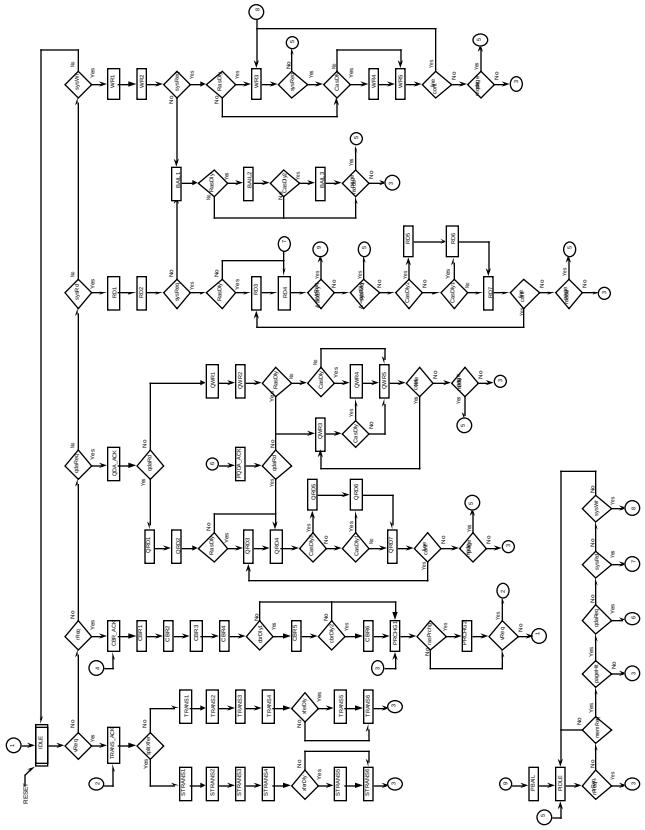


Figure 2-6 VRAM State Machine State Diagram

2.8 Video Refresh Timing Generator

The Video Refresh Timing Generator module generates the addresses and control signals required to refresh the video display (i.e., it controls the VRAMs to generate the pixel data stream used by DACula to produce an analog video output) and controls the incoming video clocks to generate the required video output clock.

The module is controlled by display configuration data obtained from the Registers module. Based on the display format configuration, it generates video refresh timing and addresses. The video refresh generation is synchronized to the basic video timing information input from Swatch. At the beginning of every blanking interval (using a special blanking signal generated by Swatch), the module computes the VRAM address at which pixel data for the next display line is located, and issues a request to the VRAM Controller module for a read transfer cycle to load the VRAM serial access memories (SAMs) with the pixel data (see section 4.1.3 for a description of SAM loading). The address for the SAM read transfer is also supplied to the VRAM Controller. Since this module and the VRAM Controller operate from asynchronous clocks, the request signal is synchronized to the VRAM controller clock before being passed to the VRAM controller. When the VRAM Controller acknowledges the SAM read transfer request(s), the acknowledge is in turn synchronized to the video clock. After the full read transfer a split read transfer is performed to finish loading the SAMs for the next display line.

When the video timing signals from Swatch indicate that the active video portion of a horizontal line is to begin, the Video Refresh Timing Generator begins producing the VRAM serial clock(s) required to clock data out of the VRAM SAM(s). A pixel load clock is also produced to clock the VRAM pixel data into the DACula CLUT/DAC. Depending on the display configuration, it may be required to reload the

VRAM SAM(s) during the middle of an active video line. If so, one or more requests for split SAM read transfer cycles are sent to the VRAM Controller in a manner similar to that used at the beginning of a horizontal blanking interval.

Requests to the VRAM Controller for full and split read transfer cycles are controlled by a state machine, shown in Figure 2-7. From the power-up idle state, the state machine first looks for the beginning of a blanking interval, and then issues a request for a full read transfer cycle, followed immediately by a request for a split read transfer cycle. The state machine then waits for the end of the blanking period. Once the active video line time has begun, the state machine continually checks to see if another blanking interval has started, in which case the above sequence is repeated, or if a mid-line split SAM read transfer is required. A change in the QSF input to the state machine is used to trigger the issuing of a split read transfer request. After a split read transfer request/acknowledge cycle has been completed, the state machine once again continually checks for a new blanking interval start, or for a change in QSF. This process repeats indefinitely.

The Video Refresh Timing Generator module also uses the configuration data from the Registers to divide the incoming pixel data clock by the amount specified by a configuration register.

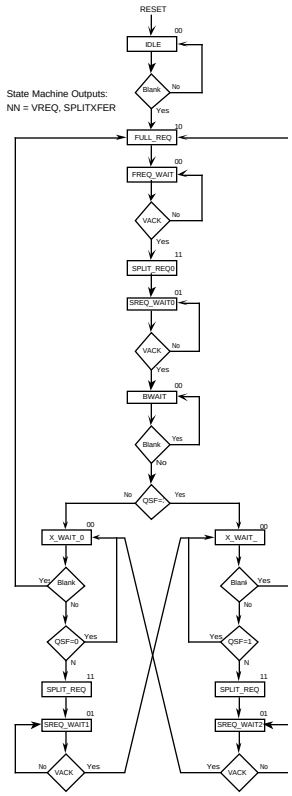


Figure 2-7 Vide Refresh State Machine State Diagram

2.9 Video Timing Generator (Swatch)

Swatch is the video timing generator for Platinum. It receives configuration information from the Registers module and a video clock from the Video Refresh Timing Generator, and then produces a number of video timing signals, including Sync, Blank, and video-related interrupts. It provides several other video timing signals to the Video Refresh Timing Generator for use in synchronizing the pixel data stream to the basic video timing.

Swatch is intended to be a generic, reusable video timing block for Apple video timing applications. Platinum uses the standard version of Swatch, but with a number of modifications for performance and gate reduction:

- Since Platinum does not support genlock to an external sync signal, the genlock circuitry in Swatch has been removed.
- Extra pipeline stages have been added to the horizontal timing section to allow it to operate at a higher clock frequency (up to 60 MHz). Several of the horizontal timing register values must be changed (decreased by 1) to allow for this pipelining.
- The vertical timing block has been modified to run at half the clock frequency of the horizontal timing block for performance reasons. The half speed clock is synchronized with the ‘half line’ signal from the horizontal timing block.
- The horizontal blanking signal has been deleted as a Swatch output, and composite blanking signals (with Platinum specific timing required by the Video Refresh Timing Generator) have been added. In addition, the standard Swatch pipeline delay has been deleted, and a more flexible, programmable delay has been added to the one of the special composite blanking signals used by the Video Refresh module.
- A "Swatch enable" input signal has been added.
- The sync and blank signals can be forced to an inactive (i.e., non-toggling) state to support video monitor power-down modes.

For more complete information about the internal operation of Swatch, consult the Swatch Specification, Version 2.01, February 15, 1990.

2.10 QuickDraw Accelerator

The QuickDraw accelerator is a logic block designed to speed up certain common operations performed by QuickDraw, the Macintosh’s graphics imaging software. Since all drawing to the screen goes through QuickDraw, any speedup here has a significant impact on overall system performance. A hardware implementation is able to perform many operations in parallel, so it can do memory intensive QuickDraw operations many times faster than a general purpose processor running software routines. The accelerator is designed specifically to speed up CopyBits type operations, pattern tiling, and bit map depth expansion (which is very useful for drawing bit mapped fonts to the screen).

In Platinum, the QuickDraw accelerator is made up of four state machines, three address generators, five pixel counters, and various other bits of logic. (The companion datapath chip, Iridium, contains all of the actual data manipulation logic.) The relationship between the various state machines and the other accelerator logic is show in Figure 2-8

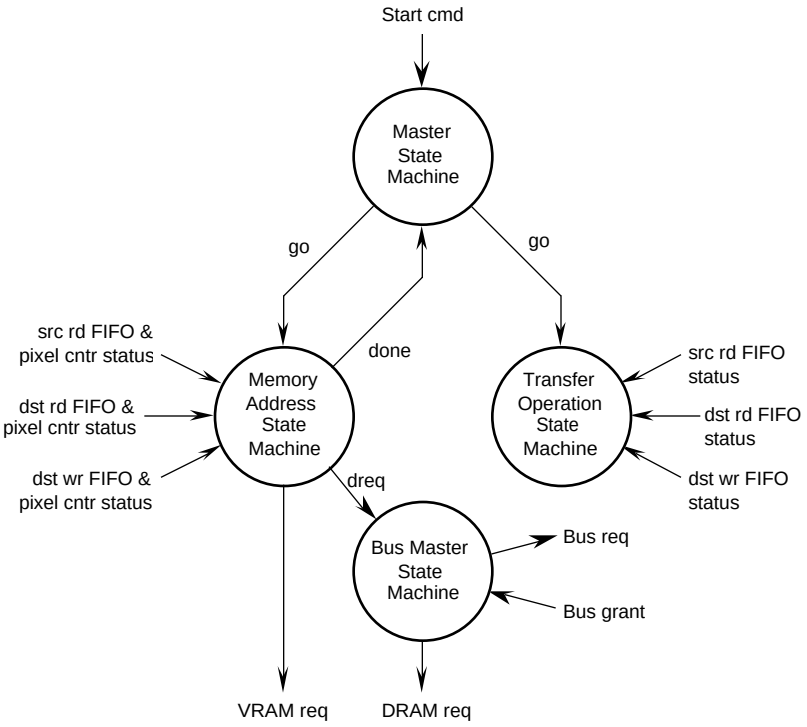


Figure 2-8 Relationships between the QuickDraw accelerator state machines

The QDA Master state machine controls the overall operation of the accelerator, monitoring line by line execution of the current command. Figure 2-9 shows the state diagram for the state machine. When it receives a “go” command from the Command register the state machine selects the base addresses from the register file (state SEL_BASE), and then enters a processing loop. For each line of the destination image, the QDA Master state machine loads the line’s starting addresses into the various address counters (state LD_ADDR) and then issues “go” commands to the Memory Address and Transfer Operation state machines (state DO_LINE). It then holds in state WAIT until the QDA Memory Address state machine returns a “line done” indication. If the current command is not done, the state machine loops back to state LD_ADDR to process the next line. Once the command is finished, the QDA Master state machine sets the “done” bit in the Status register (state DONE) and halts.

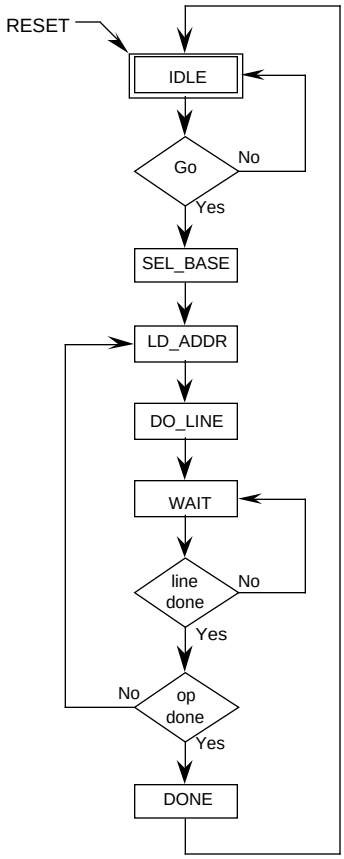


Figure 2-9 QDA Master State Machine

The QDA Memory Access state machine generates all memory requests for the accelerator. Status information from the three data FIFOs, the address counters, and the pixel counters is funneled through this state machine to generate DRAM and VRAM memory requests. Figure 2-10 shows the state diagram for this state machine.

The state machine waits in the IDLE state until it receives a “go” command from the QDA Master state machine. It then waits in the STALL state until one of the read FIFOs is ready to receive data or the write FIFO is ready to provide data. A request is generated for a DRAM or VRAM access based on the most significant bit of the read or write address, with a 1 indicating a VRAM space access. The REQ state is where the state machine waits until it receives an acknowledge from either the DRAM or VRAM controller. Depending on how much data is to be read or written and the address, the state machine will follow one of two paths.

The state SINGLE is used when one 8-byte word is to be read from or written to memory. The state machine loops on this state until the DRAM or VRAM controller asserts the transfer acknowledge signal QDATA. It then returns to the STALL state to wait for the next access, or goes back to the IDLE state if all reads and writes for the current line are finished.

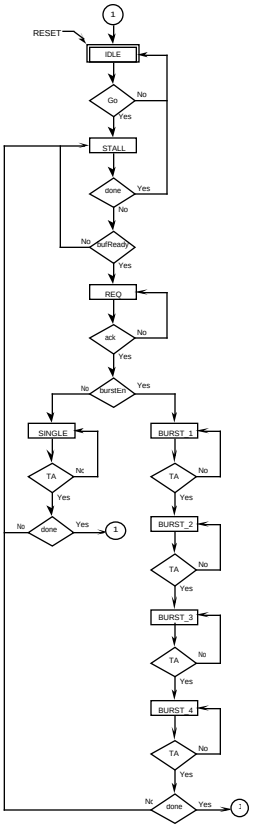


Figure 2-10 QDA Memory Access State Machine

The four BURST_x states are used when the state machine performs a 32-byte burst access. Each state is used to transfer one 8-byte word to or from memory. The state machine advances to the next BURST_x state when QDATA is asserted by the DRAM or VRAM controller. Once the fourth word is transferred, the state machine will return to the STALL state to wait for another transfer cycle, or to the IDLE state to wait for the start of the next line if all reads and writes for the current line are done.

The QDA Transfer Operation state machine controls the data manipulation logic in the accelerator. It accepts data from the source and destination read FIFOs, combines it according to the contents of the command registers, and writes the results to the destination write FIFO. Fill and copy operations are handled specially to improve the overall performance of the accelerator. The state machine waits in the IDLE state for a “go” command from the Master state machine. From IDLE it goes to the FILL, COPY, or XFER state branches based on what operation was programmed into the command registers. Figure 2-11 shows the state diagram for the state machine.

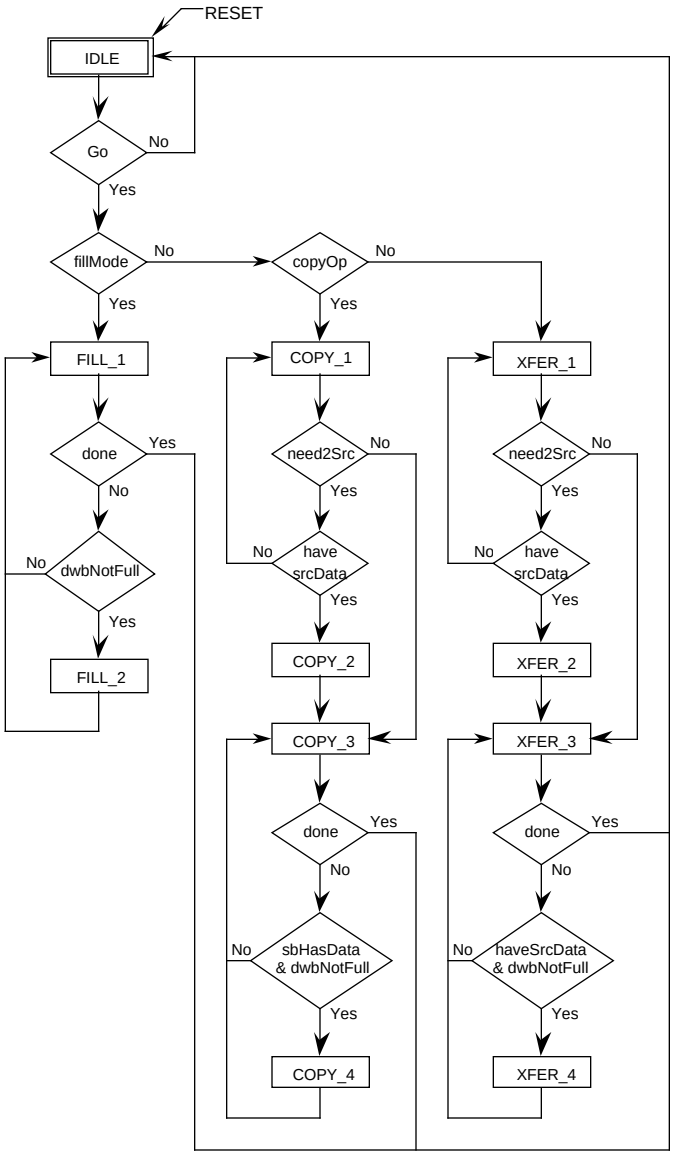


Figure 2-11 Transfer Operation State Machine Diagram

For a fill operation, the state machine repeatedly writes the contents of the foreground color register to the destination write FIFO. The FILL_1 state is used to wait until the destination FIFO is not full and the FILL_2 state writes data into that FIFO. The state machine loops on FILL_1 and FILL_2 until enough pixels have been written into the destination write FIFO to cover the current line.

For a copy operation, the state machine reads data from the source read FIFO, aligns it with the destination rectangle, and writes it to the destination write FIFO. The COPY_1 state is used to wait until two 8-byte source words are available in the source FIFO for those situations where parts of two source words are needed to create the first destination word. COPY_2 writes that special first word to the destination FIFO. In the COPY_3 state, the state machine waits for the source FIFO to have data and the destination FIFO to be not full. The COPY_4 state writes data into the destination FIFO. The state machine loops on COPY_3 and COPY_4 until all source pixels on the current line have been copied to the destination FIFO.

For the general transfer operation, the state machine reads data from the source read and destination read FIFOs, combines the data according to the op mode field of the command register, and writes the results to the destination write FIFO. The XFER_1 state is used to wait until two 8-byte source words are available in the source FIFO for those situations where parts of two source words are needed to compute the first destination word. XFER_2 writes that special first word to the destination FIFO. In the XFER_3 state, the state machine waits for the source and destination read FIFOs to have data and the destination write FIFO to be not full. The XFER_4 state writes data into the destination write FIFO. The state machine loops on XFER_3 and XFER_4 until all the pixels of the source and original destination images on the current line have been combined and written to the destination write FIFO.

The QDA Bus Master state machine handles system bus address-only transactions for the accelerator. Since the accelerator's memory accesses are invisible to the processor and L2 cache, a special bus transaction is used to maintain cache coherency. Whenever the accelerator accesses DRAM, it first performs an address-only bus transaction to alert the processor's L1 cache and the Platinum L2 cache. This gives the processor a chance to flush modified cache lines to memory so the accelerator will not operate on stale data. The L2 cache will invalidate any line referenced by the accelerator's address-only cycle. Figure 2-12 shows the state diagram for the QDA Bus Master state machine.

When the accelerator requests a DRAM access, the Bus Master state machine blocks the memory request and generates a system bus request instead while in the IDLE state. When the system bus arbiter grants the bus, the state machine performs an address-only bus cycle with the TT(0:3) bits set to 0010 (flush sector) or 0110 (kill sector). Reads always use flush sector while writes use kill sector if the accelerator wants to write a full cache line. The DELAY state allows an extra clock cycle for the address and address modifiers to settle before TS is asserted in the START state. The state machine waits in the WAIT state for AACK to be asserted. It then transitions into the RETRY state where it checks for the assertion of ARTRY. If the processor asserts ARTRY, it needs to flush a modified cache line to memory. The REDO state allows time for the processor to get its bus request asserted so that it will get the bus before the QDA Bus Master state machine tries again. If ARTRY did not get asserted, the state machine allows the accelerator request to go to the DRAM controller. It waits in the PASS state until the DRAM controller acknowledges the QDA request.

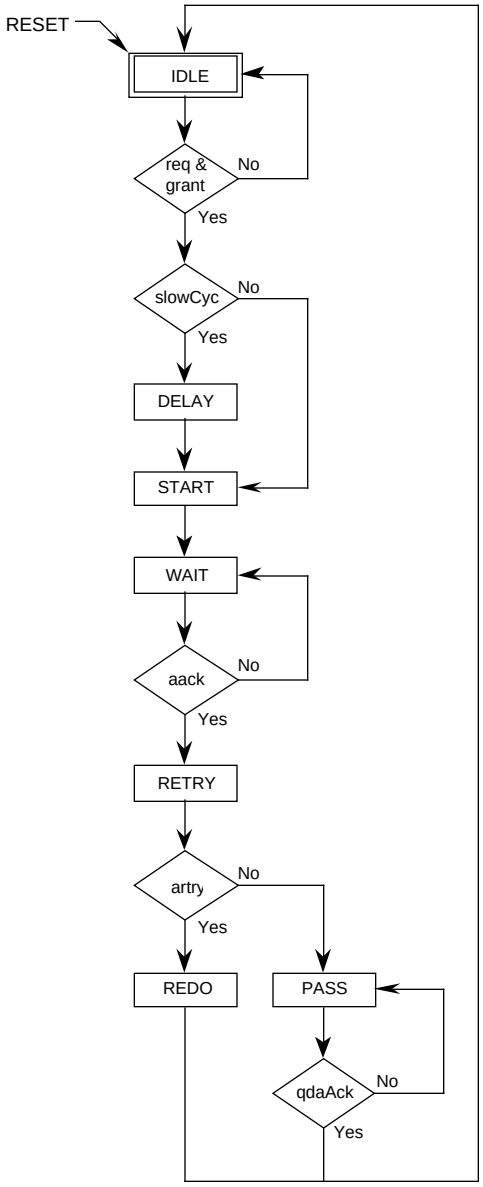


Figure 2-12 QDA Bus Master State Machine

Table 2-1 below shows how the various QuickDraw transfer modes are implemented in the accelerator. Although only the QuickDraw source transfer modes are shown, the table applies equally to the pattern transfer modes. Within the transfer operation block of the accelerator source images and patterns are treated identically. When in expand mode, the 1 bpp source image (or pattern) is expanded out to the destination bit depth before going through the transfer mode logic.

Table 2-1 Logical operations performed for QuickDraw transfer modes

```
invSrc = (src ^ invertMask), where for  8 bpp invertMask = 0xFF
                                  for 16 bpp invertMask = 0x7FFF
                                  for 32 bpp invertMask = 0x00FFFFFF

Direct destination (16 or 32 bpp):
       Non-colorizing:
              srcCopy:      dst = src
              srcOr:        dst = src & dst
              srcXor:       dst = invSrc ^ dst
              srcBic:       dst = invSrc | dst
              notSrcCopy:   dst = invSrc
              notSrcOr:     dst = invSrc & dst
              notSrcXor:    dst = src ^ dst
              notSrcBic:    dst = src | dst
       Colorizing:
              srcCopy:      dst = (invSrc & fgColor) | (src & bgColor)
              srcOr:        dst = (src & (dst ^ fgColor)) ^ fgColor
              srcXor:       dst =  invSrc ^ dst
              srcBic:       dst = (src & (dst ^ bgColor)) ^ bgColor
              notSrcCopy:   dst = (src & fgColor) | (invSrc & bgColor)
              notSrcOr:     dst = (invSrc & (dst ^ fgColor)) ^ fgColor
              notSrcXor:    dst =  src ^ dst
              notSrcBic:    dst = (invSrc & (dst ^ bgColor)) ^ bgColor

Indexed destination (8 bpp only):
       Non-colorizing:
              srcCopy:      dst =  src
              srcOr:        dst =  src | dst
              srcXor:       dst =  src ^ dst
              srcBic:       dst = ~src & dst
              notSrcCopy:   †
              notSrcOr:     dst = ~src | dst
              notSrcXor:    dst = ~src ^ dst
              notSrcBic:    dst =  src & dst
       Colorizing:
              srcCopy:      †
              srcOr:        dst = (~src & (dst ^ fgColor)) ^ fgColor
              srcXor:       dst =   src ^ dst
              srcBic:       dst = (~src & (dst ^ bgColor)) ^ bgColor
              notSrcCopy:   †
              notSrcOr:     dst = (src & (dst ^ fgColor)) ^ fgColor
              notSrcXor:    dst = ~src^ dst
              notSrcBic:    dst = (src & (dst ^ bgColor)) ^ bgColor

Expanded 1 bpp source (B&W):
       Colorized destination (8, 16, or 32 bpp):
              srcCopy:      dst = ( src & fgColor) | (~src & bgColor)
              srcOr:        dst = ( src & fgColor) | (~src & dst)
              srcXor:       dst =   src ^ dst
              srcBic:       dst = ( src & bgColor) | (~src & dst)
              notSrcCopy:   dst = (~src & fgColor) | ( src & bgColor)
              notSrcOr:     dst = (~src & fgColor) | ( src & dst)
              notSrcXor:    dst =  ~src ^ dst
              notSrcBic:    dst = (~src & bgColor) | ( src & dst)
```

† These operations are not supported since the accelerator cannot perform the required color space conversions. For each of these modes QuickDraw maps the 8-bit indexed color pixel to a 48-bit true color pixel, performs the logical operation, and maps the resulting 48-bit pixel to the closest 8-bit indexed color value.

3.0 Signal Description

Signal Name	Type	Description
ADDR[0:31]	I/O	System Address bus. ADDR[0] is the MSB, ADDR[31] is the LSB.
SIZE[0:2]	I/O	System bus signals which indicate the size of the current bus transfer. These signals are outputs only during QuickDraw accelerator initiated address only bus transactions, which are done to maintain cache coherency.
TT[0:3]	I/O	Transfer Type. System bus signals which indicate the type of the current bus transfer. These signals are outputs only during QuickDraw accelerator initiated address only bus transactions, which are done to maintain cache coherency.
BURST	I/O	Transfer Burst. System bus signal which indicates the current bus transaction is for 32 bytes (the size of a PowerPC cache line). It is asserted as an output by the QuickDraw accelerator during address only cycles for cache coherency.
CI	Input	Cache Inhibit. System bus signal which tells the level 2 cache not to update the cache with the current bus cycle's data.
TS	I/O	Transfer Start. System bus signal which indicates the start of an address bus transaction. It is asserted as an output by the QuickDraw accelerator during address only cycles for cache coherency.
AACK	I/O	Address Acknowledge. System bus signal which indicates the completion of an address bus transaction.
ARTRY	Input	Address Retry. System bus signal which indicates the just completed address bus transaction must be discarded and retried.
TA	I/O	Transfer Acknowledge. System bus signal which indicates the successful completion of a data bus transaction.
TEA	I/O	Transfer Error Acknowledge. System bus signal which indicates an error occurred during the current data bus transaction.
DRTRY	Output	Data Retry. System bus signal which indicates the data from the just completed data transfer is invalid and must be discarded. This signal is only used by the L2 cache controller to allow assertion of TA before a cache hit is detected during reads.
ADDRLATCHEN	Output	Address Latch Enable. This signal is asserted to latch the low 24 bits of the system address bus for use with the ROMs.
BANDITBR	Input	Bus Request (Bandit). Address bus request from the PCI bridge chip.

Signal Name	Type	Description
BANDITBG	Output	Bus Grant (Bandit). Address bus grant for the PCI bridge chip.
BANDITDBG	Output	Data Bus Grant (Bandit). Data bus grant for the PCI bridge chip.
BANDITSSD	Output	Source/Sink Data (Bandit). This signal acts as a slave data bus grant to the PCI bridge chip. When it is asserted, Bandit may drive read data onto the data bus or capture write data from the data bus.
BANDITSACK	Input	Slave Acknowledge (Bandit). This signal is asserted by the PCI bridge chip to tell the system bus arbiter that it will respond to the current bus transaction. In response, the bus arbiter will assert BANDITSSD when the data bus is available.
BANDITRDDA	Input	Read Data Available (Bandit). The PCI bridge chip will assert this signal when it has completed a PCI bus read transaction and is ready to return that data to the processor.
SYSBG	Output	Bus Grant (601). This is the address bus grant for the processor. Unless another master requests the bus, this signal is asserted to park the processor on the system address bus.
SYSDBG	Output	Data Bus Grant (601). This is the data bus grant for the processor. Unless another master requests the bus, this signal is asserted to park the processor on the system data bus.
TAGOE	Output	Tag Output Enable. This is the output enable for the L2 cache tag SRAMs.
TAGWE	Output	Tag Write Enable. This signal is asserted to allow new tag data to be written into the L2 cache tag SRAMs.
TAGVALID	I/O	Tag Valid. This is the tag valid bit. When this bit is asserted, the corresponding tag and cache data are valid.
TAGDATA[0:13]	I/O	Tag Data.. This is the L2 cache tag data that is compared against the current address to determine if the referenced data is in the L2 cache.
CACHEOE	Output	Cache Output Enable. This is the output enable for the L2 cache data SRAMs.
CACHEWE	Output	Cache Write Enables This is the write enable for the L2 cache data SRAMs.
CACHEADSP	Output	Cache Access Start Pulse. This signal is asserted to cause to L2 cache data SRAMs to latch the address and start the access.
CACHEADV	Output	Cache Advance. This signal is asserted to cause the L2 cache data SRAMs to increment their internal burst address counter in preparation for the next cache access.

Signal Name	Type	Description
DADDR[12:0]	I/O	DRAM Address. The row and column addresses for a DRAM access are multiplexed onto these lines.
DRAS[0:7]	Output	DRAM Row Address Strobe. Each DRAM bank has its own DRAS signal.
DCAS[0:7]	Output	DRAM Column Address Strobe. Each DRAM byte lane has its own DCAS signal.
DWE	Output	DRAM Write Enable. Write control signal for all banks of DRAM.
ROMOE	Output	ROM Output Enable. This signal is asserted during an access to the ROMs.
ROMWE	Output	ROM Output Enable. This signal is asserted during a write access to the ROM space when the Flash Write Enable bit is set in the ROM Configuration register. It is intended for use with Flash memory SIMM during development.
ROMCE	Output	ROM Chip Enable. This signal, which is normally left asserted, is negated during system sleep to reduce the ROMs' power consumption.
ROMADDR[27:28]	Output	ROM Address (bits 27–28). Least significant ROM address bits. They are incremented during burst accesses to ROM space to cycle through the four double long words which make up a PowerPC 60x cache line.
VADDR[8:0]	Output	VRAM Address. The row and column addresses for a VRAM access are multiplexed onto these lines.
VRAS[0:1]	Output	VRAM Row Address Strobe. Each VRAM bank has its own VRAS signal.
VCAS[0:1]	Output	VRAM Column Address Strobe. Each VRAM bank has its own VCAS signal.
VWE[0:7]	Output	VRAM Write Enable. These signals are the byte lane write enables for the VRAM array. For a VRAM cycle, these lines are high when VRAS falls, and low when VCAS falls for the byte lane(s) to be written.
DT/OE[0:1]	Output	VRAM Data Transfer/Outpute Enable. If this signal is asserted when VRAS falls, the VRAM cycle is a transfer operation to the VRAM serial access memory (SAM), otherwise a standard DRAM cycle is performed. After VRAS falls during a VRAM read cycle, this signal is used to enable data onto the data bus. Each VRAM bank has its own DT/OE signal.
DSF[0:1]	Output	VRAM Dual Special Function. This signal is used to enable split SAM read transfer operations, If high when VRAS falls and DT/OE indicates a read transfer operation, a split SAM trasnfer is performed. Each VRAM bank has its own DSF signal.

Signal Name	Type	Description
SOE[0:1]	Output	VRAM Serial Output Enable. This signal enables the VRAM data onto the serial data bus. Each VRAM bank has its own distinct output enable.
SCLK[0:1]	Output	VRAM Shift Clock. A rising edge on this signal clocks data out of the VRAM serial data output port. This signal is synchronous to LD, which clocks the VRAM serial data output into the CLUT/DAC. Each VRAM bank has its own serial data clock.
VIDCLK	Input	Video Clock. This is the input clock for the frame buffer controller logic
LD	Output	Pixel Load Clock. This signal is used to clock VRAM pixel data into the CLUT/DAC's pixel data input port. LD also serves as the master clock for the CLUT/DAC when the data rate into its pixel data port exceeds 30 MHz.
HSYNC	Output	Horizontal Sync. This horizontal synchronization signal is fed to the monitor connector.
VSYNC	Output	Vertical Sync. This vertical synchronization signal is fed to the monitor connector.
CSYNC	Output	Composite Sync. This composite synchronization sgnal is fed to the monitor connector.
CBLANK	Output	Composite Blank. This signal is used by the CLUT/DAC to blank the video signal during horizontal and vertical retrace.
VBLIRQ	Output	Vertical Blanking Interval Interrupt. If the vertical blanking interrupt is enabled, this line is asserted at the beginning of each vertical blanking interval.
ADDRCLKEN	Output	Address Clock Enable. This signal is asserted to tell the Iridium datapath chip to capture the information on its address inputs.
WRCLKEN	Output	Write Clock Enable. This signal is asserted to write data from the system data bus into a cache line sized write back buffer inside Iridium.
SYSDATAOE	Output	System Data Output Enable. This signal is asserted during reads from any part of the memory subsystem (DRAM, VRAM, or ROM).
MEMDATAOE	Output	Memory Data Output Enable. This signal is asserted during writes to any part of the memory subsyste (DRAM, VRAM, or ROM).
MEMDATASEL	Output	Memory Data Select. This is a multiplexer control signal for Iridium. It is used for accesses to VRAM when only half of one VRAM bank is present (this is the base 1 MB configuration).

Signal Name	Type	Description
MEMDATACLKEN	Output	Memory Data Clock Enable. This signal is asserted to capture read data when reading from the base VRAM configuration (half of one bank).
DECVALID	Output	Address Space Decode Qualifier. When this signal is asserted, the address space decodes are valid. (The decodes are sent to Iridium on the REGDATA bus during address bus cycles.)
QDAACK	Output	QuickDraw Accelerator Acknowledge. This signal tells the QuickDraw accelerator logic in Iridium that the next bus transaction is for the accelerator.
QDATA	Output	QuickDraw Accelerator Transfer Acknowledge. This cycle acknowledge signal is used by the QuickDraw accelerator logic in Iridium to control the flow of data through its buffers.
REGDATAOE	Output	Register Data Output Enable. This signal is asserted to allow Iridium to drive the register data bus during writes to Platinum or Iridium registers.
REGADV	Output	Register Shift Advance. This signal is asserted to tell Iridium to shift the next byte on to or off of the register data bus. This signal is asserted for four clocks to transfer all 32 bits of a register between Platinum and Iridium. During scan test mode this signal is the output pin for the scan chain.
REGDATA[0:7]	I/O	Register Data Bus. This 8-bit bus is used to send address information to Iridium during address bus transactions and to pass register data back and forth between Platinum and Iridium during register accesses.
SENSE[2:0]	I/O	Monitor ID Sense Lines. These lines are capable of being driven to accomodate Apple's extended monitor type sense line scheme (see Appendix C for the mapping of sense codes to monitors). SENSE[1] is the scan chain input pin during scan test mode. SENSE[2] is a secondary test clock during scan test mode.
BCLK	Input	System Bus Clock.
RESET	Input	Hardware Reset. This input initializes all Platinum circuits.
IODISABLE	Input	Test Mode Output Disable. When this signal is asserted, all output and bidirectional signals are tri-stated. This signal is intended to aid board level testing.
TESTMODE	Input	Test Mode Enable. When this signal is asserted, the chip is placed in a test mode (for IC test only).
SCANCLK	Input	Scan Test Clock.

4.0 Programmer's Guide

4.1 Memory Map

The memory map for a Platinum-based memory subsystem is shown is Figure 4-1. DRAM is allocated the first 2 GB of the physical address space ($0000 0000 – $7FFF FFFF), even though Platinum only has control signals to support 1 GB of DRAM. ROM is allocated the last 16 MB of the physical address space ($FF00 0000 – $FFFF FFFF), with the last physical location in ROM coinciding with address $FFFF FFFF. The system control and configuration registers (i.e., Platinum and Iridium registers) are located at $F80x xxxx. The L2 cache data and tag store test area is located from $F810 0000 – $F82F FFFF. Platinum controls the rest of the $F8xx xxxx space although nothing resides there. The VRAM requires 16 MB of address space to support double buffering and big and little endian access to the frame buffer. VRAM is located at $F1xx xxxx, with the big endian aperture starting at $F100 0000 and the little endian aperture starting at $F180 0000. All other address spaces are controlled by the Bandit PCI interface chip.

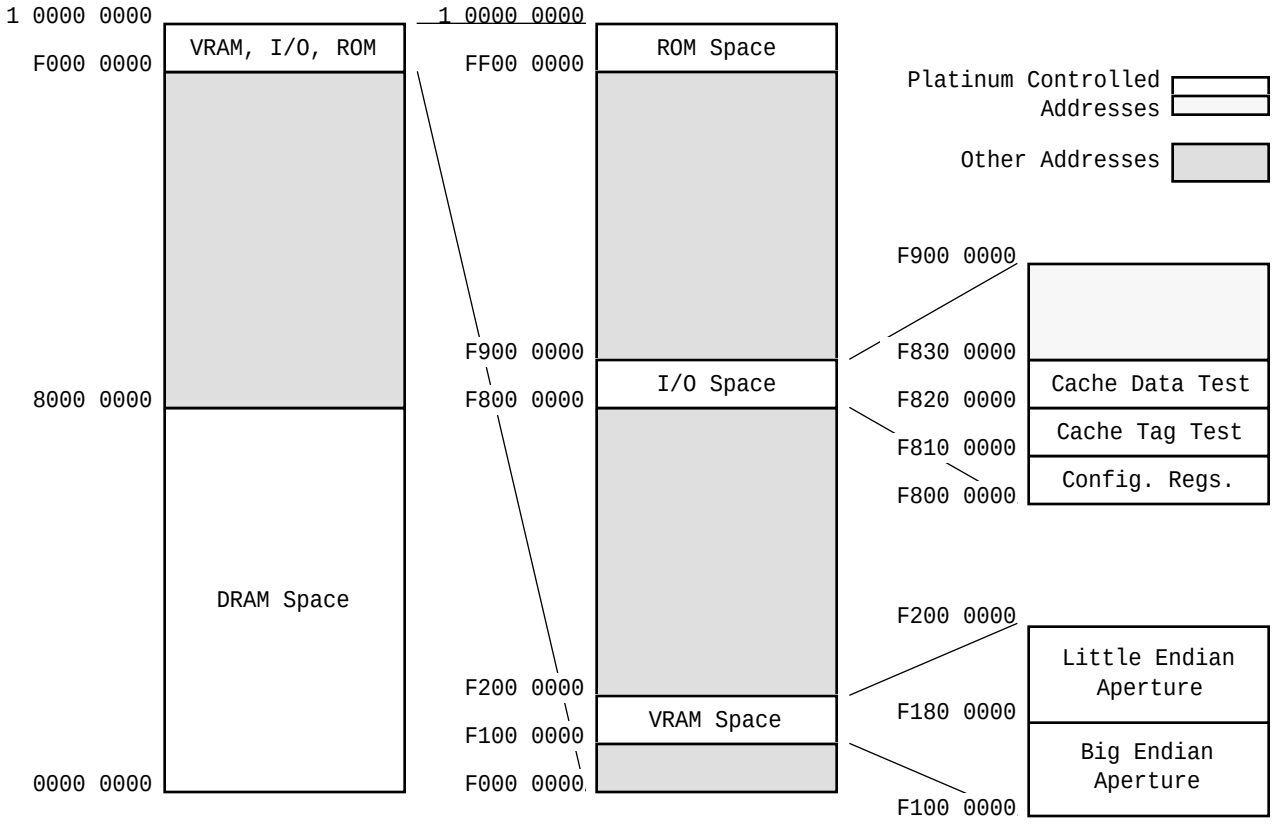


Figure 4-1 Platinum Memory Map

4.2 Cache

Platinum supports a direct-mapped, write-through L2 cache. The cache can be 256, 512, or 1024 KB in size and is organized in lines of 32 bytes each, with a single address tag and valid bit per line.

4.2.1 Cache Initialization and Test

Since the tag store is made up of standard asynchronous static RAM, it must be initialized before the cache is first used. Setting the Cache Enable bit to 0 and the Tag Test Enable bit to 1 in the Cache Configuration register enables direct access to the L2 cache tag store in the address space $F81x xxxx. Writing zeros to every tag will invalidate all entries and prepare the cache for subsequent use. Since each tag is associated with 32 bytes of data, they are read or written at 32-byte offsets, i.e. tag 0 is located at $F810 0000, tag 1 is at $F810 0020, tag 2 is at $F810 0040, and so on. The direct tag store access may also be used to test the integrity of the L2 tag store RAMs. All tag store accesses must be 4-byte single beat reads or writes or the system will hang.

The L2 cache data store can also be directly accessed by the processor for testing. Setting the Cache Enable bit to 0 and the Cache Test Enable bit to 1 in the Cache Configuration register enables the direct access mode. in the address range $F820 0000 – $F82F FFFF. Reads and writes must be performed as 8-byte single beat accesses or the system will hang.

4.2.2 Cache Sizing

The presence and size of an L2 cache DIMM is automatically detected by hardware at system startup. Software need only read the Cache Configuration register to determine what size, if any, cache is installed in the system.

4.2.3 Cache Low Power Mode

When a Platinum-based system is put into low power mode by setting the sleep bit in the Sleep Mode register, the cache SRAMs are tri-stated. Since the cache parts are inaccessible, no updates can be performed and the data stored in the cache may become stale (the data may no longer match what is in memory). Also, the tag store is not automatically invalidated when low power mode is activated. The cache should be disabled by software when the system enters low power mode and the tag store should be invalidated when the system is brought out of low power mode. This insures that any stale data in the cache is discarded.

4.2.4 Cache Access Times

The cache controller is designed to work with 15 ns SRAMs for bus clock speeds up to 45 MHz and 12 ns SRAMs at bus clock speeds of 50 MHz or higher. The access time in bus clock cycles does not change with increasing clock speeds, hence the need for faster parts at higher frequencies. Table 4-1 below shows the cache access times for various bus transactions.

Table 4-1 Cache Access Times

Operation Type	Single Read	Single Write	Burst Read	Burst Write
PPC 601 as master				
cache hit from idle bus	3	3	2-1-1-1	3-1-1-1
back-to-back hit with SYSDBG active	3	3	2-1-1-1	3-1-1-1
back-to-back hit with SYSDBG inactive	3	3	3-1-1-1	3-1-1-1
cache miss	—	3	—	3-1-1-1
Bandit PCI bridge as master				
cache hit from idle bus	5	4	5-1-1-1	4-1-1-1
back-to-back hit	5	4	5-1-1-1	4-1-1-1
cache miss	—	4	—	4-1-1-1

4.3 ROM

The ROM controller is designed to work with burst mode ROMs. EEPROMs, also known as “FlashROMs”, are supported to make firmware development easier. When the Flash Write Enable bit in the ROM Timing register is set, writes to ROM space are allowed. Table 4-6 show the appropriate setting for the Platinum ROM Timing register based on the system bus clock frequency and the ROM speed. Table 4-6 also indicates the number of bus clocks required to read from/write to ROM based on those register settings. The access times include the cycle in which TS is asserted. This table assumes that 120/60 ns burst mode ROMs are used at system bus clock speeds of 33–50 MHz, and 100/50 ns burst mode ROMs are used at 60/66 MHz.

Table 4-2 ROM Timing Register Values and Access Times

System Configuration	flashWE	firstDly1	firstDly0	burstDly1	burstDly0	Single Read	Burst Read	Single Write
33 MHz	—	1	1	1	0	6	6-4-4-4	6
37.5 MHz	—	1	0	1	0	7	7-4-4-4	7
40 MHz	—	1	0	1	0	7	7-4-4-4	7
45 MHz	—	0	1	0	1	8	8-5-5-5	8
50 MHz	—	0	0	0	1	9	9-5-5-5	9
60/66 MHz	—	0	0	0	1	9	9-5-5-5	9

4.4 DRAM

Platinum supports up to 8 banks of DRAM. Each bank may have 2–128 MBytes of DRAM, organized as a 64-bit wide array. Each 64-bit DIMM can have 1 or 2 banks of DRAM, so Platinum supports a total of 4 DIMM sockets.

4.4.1 DRAM Addressing Modes

Platinum supports a wide variety of DRAMs which use several different addressing schemes. Each bank of DRAM has a control bit, the “Addressing Mode” bit in the bank configuration register, that is used to configure the DRAM address bus. The processor address to DRAM address mapping changes depending on the state of the mode bit, as shown in Table 4-3. Platinum supports all 1, 4, or 16 Mbit DRAMs, although using x1 parts is not recommended due to capacitive loading problems on the DRAM address bus. Platinum also supports most varieties of 64 MBit DRAMs, but these parts are 3.3V only so there are system level issues with using them. Table 4-4 lists the DRAM types that Platinum can handle and the correct addressing mode bit setting for each one. Platinum supports only a linear memory organization; interleaved memory is not supported.

Table 4-3 PPC 601 Address to DRAM Address Mapping

dAddr bit	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Address mode = 1													
Row Address	6	7	8	10	11	12	13	14	15	16	17	18	19

Col Address	4	5	7	9	20	21	22	23	24	25	26	27	28
Address mode = 0													
Row Address	5	9	10	20	11	12	13	14	15	16	17	18	19
Col Address	4	5	7	6	8	21	22	23	24	25	26	27	28

Table 4-4 Address Mode Bit Settings for Various DRAMs

DRAM Type	Row Bits	Col Bits	Mode Bit Setting
1 MBit DRAMs			
1M x 1	10	10	1
256K x 4	9	9	1
4 MBit DRAMs			
4M x 1	11	11	1
1M x 4	10	10	1
512K x 8	10	9	1
256K x 16	9	9	1
256K x 16	10	8	0
16 MBit DRAMs			
16M x 1	12	12	0
4M x 4	12	10	1
4M x 4	11	11	1
2M x 8	11	10	1
2M x 8	12	9	0
1M x 16	12	8	0
1M x 16	10	10	1
64 MBit DRAMs			
64M x 1	13	13	—
16M x 4	13	11	0
16M x 4	12	12	0
8M x 8	13	10	1
8M x 8	12	11	0
4M x 16	13	9	—
4M x 16	12	10	1
4M x 16	11	11	1
2M x 32	13	8	—
2M x 32	12	9	0
2M x 32	11	10	1

4.4.2 DRAM Bank Sizing

Platinum contains a number of base registers used to “stitch” all the banks of DRAM together into one contiguous address range. In order to set these base registers correctly, the size of each bank must be determined. Since Platinum supports a variety of DRAM sizes, with a number of different internal addressing schemes, sizing memory is a bit more complicated than it was with previous memory controllers. The following steps should be used to determine the size of each bank. (This procedure assumes the sizing routine checks the last location in each 1 MByte of address space to determine the bank size. If a different method is used, a new procedure will be needed.)

1) Set the Bank *n* Configuration register (described in section 4.7.1) to 0x(0nnn n0000)00 0000, where n is the bank number (in binary), and size the bank. If the bank size is not a power of 2 (i.e. a size of 5 MB, or 13 MB, or …), go to step 2. If the bank size is a power of 2 (i.e. a size of 1 MB, or 4 MB, or …), go to step 3.

2) Set the Bank *n* Configuration Register to 0x(1nnn n0000)00 0000, where n is the bank number (in binary),and size the bank.

3) Program the base address field for the bank with bits 1:10 of the first address in the bank. (The first address of a bank depends on the sum of the sizes of all lower numbered banks of DRAM.)

4.4.3 DRAM Refresh

Since DRAM is dynamic memory, it must be refreshed periodically to retain its contents. A single refresh cycle refreshes the contents of a single row, and the entire DRAM must be refreshed at a rate of one row every 15.6 µs. Some DRAMs require a refresh rate of one row every 7.8 µs, a rate that usually is not supported by the firmware in ROM. Platinum refreshes all DRAM banks in parallel, so only a single refresh cycle is needed to refresh all five banks. Also, CAS-before-RAS refreshes are performed, thus utilizing the DRAMs' internal refresh address counters. Platinum performs DRAM refreshes at periodic intervals, based on the value programmed into the DRAM Refresh Interval register (see the Control Registers section).

4.4.4 Fast Page Mode

The DRAM features a fast page mode for quickly accessing multiple column locations in the same row by performing multiple CAS cycles during a single active RAS cycle. In page mode, the initial access to a row occurs as a 'standard' DRAM access. However, at the end of the cycle, RAS remains active. As long as consecutive DRAM accesses are within the same row (row size depends on DRAM type and will comprise 1024–8192 double words) the access time is significantly reduced since only the column address need be supplied to the DRAM.

Platinum provides support for fast page mode DRAM operation. Page mode is enabled or disabled simply by writing to a Platinum register (see the DRAM Configuration Register description). After page mode has been enabled, all subsequent DRAM read/write operations will occur in page mode. Disabling page mode will cause the following DRAM operations to occur as 'standard' DRAM accesses. It should be noted that just as there is a performance increase for in-page 'hits', there is a performance penalty associated with each page 'miss' since RAS must be precharged before the next row access.

4.4.5 DRAM Access Times

The DRAM controller is designed to work with 60 or 70ns DRAMs. Table 4-4 show the appropriate setting for the Platinum Configuration register based on the system bus clock frequency and the DRAM speed, Table 4-5 indicates the number of bus clocks required to read/write from/to DRAM, using the register values shown in Table 4-4. The access times include the cycle in which TS is asserted. Read accesses are one bus clock slower if a second level cache is present in the system. This table assumes that 70ns DRAMs are used at system bus clock speeds of 33–50 MHz, and 60 ns DRAMs are used at 60/66 MHz.

Table 4-5 DRAM Configuration Register Values

System Configuration	AOCycDly	qdaRasDly	qdaCasDly1	qdaCasDly2	pageMode	rdRasDly	rdCasDly1	rdCasDly2	wrRasDly	wrCasDly	rasPrchg	cbrDly1	cbrDly2
33 MHz	0	0	0	0	—	0	0	0	0	0	0	0	0
37.5 MHz	0	0	0	0	—	0	1	0	0	0	0	0	0
40 MHz	0	0	0	0	—	0	1	0	0	0	1	0	0

45 MHz	1	0	0	0	—	0	1	0	0	1	1	1	0
50 MHz	1	0	1	0	—	1	1	0	0	1	1	1	0
60/66 MHz	1	1	1	0	—	1	1	1	1	1	1	1	1

Table 4-6 DRAM Access Time

Operation Type	Single Read	Single Write	Burst Read	Burst Write
33 MHz				
isolated access	6	3	6-3-3-3	3-2-2-2
page mode (with in-page hit)	4	2	4-3-3-3	2-2-2-2
37.5 MHz				
isolated access	7	3	6-3-3-3	3-2-2-2
page mode (with in-page hit)	5	2	4-3-3-3	2-2-2-2
40 MHz				
isolated access	7	3	7-4-4-4	3-2-2-2
page mode (with in-page hit)	5	2	5-4-4-4	2-2-2-2
45 MHz				
isolated access	7	4	7-4-4-4	4-3-3-3
page mode (with in-page hit)	5	3	5-4-4-4	3-3-3-3
50 MHz				
isolated access	8	4	8-4-4-4	4-3-3-3
page mode (with in-page hit)	5	3	5-4-4-4	3-3-3-3
60/66 MHz				
isolated access	9	5	9-5-5-5	5-3-3-3
page mode (with in-page hit)	6	3	6-5-5-5	3-3-3-3

4.5 Frame Buffer

Platinum supports frame buffers with 1/2, 1 or 2 banks of VRAM. Each full bank contains 2 MB of VRAM which is organized as 256Kx64. A 1/2 bank configuration has 1 MB of VRAM organized as 256Kx32. A number of display resolutions are supported for each of the VRAM amounts, as shown earlier in Figure 1-2. Platinum also provides some limited support for double buffering.

4.5.1 Addressing Modes

Platinum only supports a linear addressing mode for accessing VRAM. All VRAM in the frame buffer appears as a single address space starting at $F100 0000. Platinum does support banks of different sizes, though, so the video driver software must determine how much VRAM is installed and how it is arranged. All configurations of Catalyst will use 2 (or 4) Mbit VRAMs, so the “2 Mbit VRAMs” bit in FBConfig1 should always be set. Some of the 4 Mbit parts have only a 256-bit SAM, so the “512 Bit SAM” bit in FBConfig1 should always be cleared. (All of the approved 2 and 4 Mbit VRAMs will work with the 256-bit SAM timing.) Platinum supports 4 VRAM configurations:

1) 1 MB VRAM arranged as a 32-bit wide half bank (this is the base configuration on Catalyst)

2) 2 MB VRAM arranged as a 64-bit wide full bank

3) 2 MB VRAM arranged as two 32-bit half banks (this configuration may not be supported by the video driver)

4) 4 MB VRAM arranged as two 64-bit wide full banks

To determine which configuration is present, the following steps should be performed:

1) Set the “Full VRAM Banks” and “2 Mbit VRAMs” bits in FBConfig1, and clear the “512 bit SAM” bit in FBConfig1.

2) Write a 64-bit data value to $F120 0000 and then write a different 64-bit data value to $F100 0000.

3) Read the data back from $F120 0000 and then from $F100 0000, and compare the data values with what was written.

4) If both 64-bit values are valid, there is 4 MB of VRAM installed and sizing is done (this is configuration 4 from above).

5) If only the lower bits (32-63) of the data from $F120 0000 are valid and only the lower bits of the data from $F100 0000 are valid, configuration 3 is present and the “Full VRAM Banks” bit should be cleared.

6) If the data from $F120 0000 is bad and the data from $F100 0000 is valid, configuration 2 is present and sizing is done.

7) If the data from $F120 0000 is bad and only the lower bits (32-63) of the data from $F100 0000 are valid, configuration 1 is present and the “Full VRAM Banks” bit in FBConfig1 should be cleared.

Note: Platinum also supports 1 Mbit VRAMs, but no currently planned configuration will use them. The table below shows how the “2 Mbit VRAMs” configuration bit affects VRAM addressing.

Table 4-7 PPC 60x Address to VRAM Address Mapping

vAddr bit	8	7	6	5	4	3	2	1	0
“2 Mbit VRAMs” = 0									
Row Address	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
Col Address	**—**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**
“2 Mbit VRAMs” = 1									
Row Address	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**
Col Address	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**

4.5.2 SAM Loading

Platinum-based frame buffer VRAM is dual ported - each bank includes a 256 (or 512) x 64-bit serial access memory (or SAM - it functions much like a parallel-loaded shift register) used to clock pixel data out of the VRAM to be sent to the CLUT/DAC. The SAM is loaded via a VRAM read transfer cycle which loads an entire VRAM row into the SAM. The frame buffer VRAM also features the capability to perform a split read transfer, which only loads one half of the SAM (i.e., 128 (or 256) 64-bit words).

The split transfer capability eases timing requirements when reloads occur during the active display time.

Platinum initiates a SAM read transfer at the beginning of every horizontal blanking interval (except during vertical blanking). All VRAM banks perform a (split) read transfer cycle in parallel. Immediately after the full read transfer, a split read transfer cycle is executed to finish loading the SAM for the next display line. One or more split read transfers may be performed during the active line time if required to keep the SAMs filled (i.e., whenever QSF toggles).

The address of the VRAM row loaded into the SAMs is determined by Platinum based on the video base address, rowwords, double buffer state, and interlace registers, and internal video timing information such as vertical and horizontal blanking and row and field numbers. A new row address is calculated for each new horizontal line. The address is updated continuously in a 19-bit counter during the active video line time since a split SAM read transfer may be required.

4.5.3 VRAM Refresh

Since VRAM is dynamic memory, it must be refreshed periodically to retain its contents. A single refresh cycle refreshes the contents of a single row, and the entire VRAM must be refreshed at a rate of one row every 15.6 µs. Platinum refreshes all VRAM banks in parallel, so only a single refresh cycle is needed to refresh both banks. Also, CAS-before-RAS refreshes are performed, thus utilizing the VRAMs' internal refresh address counters. Platinum performs VRAM refreshes at periodic intervals, based on the value programmed into the VRAM Refresh Interval register (see the Control Registers section).

Note: The SAMs are not dynamic memory, and therefore do not need to be refreshed.

4.5.4 Fast Page Mode

The frame buffer VRAM features a fast page mode for quickly accessing multiple column locations in the same VRAM row by performing multiple CAS cycles during a single active RAS cycle. In page mode, the initial VRAM access to a row occurs as a 'standard' VRAM access. However, at the end of the cycle, RAS remains active. As long as consecutive VRAM accesses are within the same row (rows consist of either 256 or 512 double words) the VRAM access time is significantly reduced since only the column address need be supplied to the VRAM.

Platinum provides support for fast page mode VRAM operation. Page mode is enabled or disabled simply by writing to a Platinum register (see the Frame Buffer Control Registers section). After page mode has been enabled, all subsequent VRAM read/write operations will occur in page mode. Disabling page mode will cause the following VRAM operations to occur as 'standard' VRAM accesses. It should be noted that just as there is a performance increase for in-page 'hits', there is a performance penalty associated with each page 'miss' since RAS must be precharged before the next row access. However, since the Platinum register operation to change page mode is relatively fast (only 8 clock cycles), page mode may be enabled/disabled on an operation by operation basis, thus allowing the VRAM to operated in the mode most efficient for the current graphics operation. For instance, drawing a vertical or steep line would result in consecutive accesses to different pages, so page mode should be left off to achieve best performance for this type of operation. However, an area fill would cause a number of consecutive accesses to the same row, so turning on page mode for this type of operation should result in a performance increase.

4.5.5 VRAM Access Times

The VRAM controller is designed to work with 60–70 ns VRAMs. Table 4-8 shows the appropriate setting for the Frame Buffer Configuration 2 register based on the system bus clock frequency and the VRAM speed. Table 4-9 indicates the number of clock cycles required to read/write data from/to VRAM, using the register values shown in Table 4-8. (The bold entry for each clock speed is the one that would normally be used in a real system.) The access times include the cycle in which TS is

asserted. Access times are shown for individual transactions and for page mode accesses with in-page hits. Page mode accesses which have an in-page miss incur a 2 cycle penalty for RAS precharge (or 3 cycles if the 'RAS Precharge' bit is set). After the precharge time, the access proceeds with the isolated access timing.

Table 4-8 Frame Buffer Configuration 2 Register Values

System Configuration	qdaRasDly	qdaCasDly1	qdaCasDly2	rdRasDly	rdCasDly1	rdCasDly2	wrRasDly	wrCasDly	rasPrchg	cbrDly1	cbrDly2	xferDly1	xferDly2
33 MHz	0	0	0	0	0	0	0	0	0	0	0	0	0
37.5 MHz	0	0	0	0	1	0	0	0	0	0	0	0	0
40 MHz	0	0	0	0	1	0	0	0	1	0	0	0	0
45 MHz	0	0	0	0	1	0	0	1	1	1	0	0	0
50 MHz	1	0	0	1	1	0	0	1	1	1	0	1	0
60/66 MHz	1	1	0	1	1	1	1	1	1	1	1	1	1

Table 4-9 VRAM Access Time

Operation Type	Single Read	Single Write	Burst Read	Burst Write
33 MHz				
isolated access	6	3	6-3-3-3	3-2-2-2
page mode (with in-page hit)	4	2	4-3-3-3	2-2-2-2
37.5 MHz				
isolated access	7	3	7-4-4-4	3-2-2-2
page mode (with in-page hit)	5	2	5-4-4-4	2-2-2-2
40 MHz				
isolated access	7	3	7-4-4-4	3-2-2-2
page mode (with in-page hit)	5	2	5-4-4-4	2-2-2-2
45 MHz				
isolated access	7	4	7-4-4-4	4-3-3-3
page mode (with in-page hit)	5	3	5-4-4-4	3-3-3-3
50 MHz				
isolated access	8	4	8-4-4-4	4-3-3-3
page mode (with in-page hit)	5	3	5-4-4-4	3-3-3-3
60/66 MHz				
isolated access	9	4	9-5-5-5	5-3-3-3
page mode (with in-page hit)	6	3	6-5-5-5	3-3-3-3

4.5.6 Double Buffering

Platinum provides support for double buffering. Two banks of VRAM are needed for double buffer support. Although the physical addresses of the two buffers are dependent on the VRAM bank size (1 or 2 MB), software need not be concerned with the actual buffer addresses as the frame buffer memory is always located at the same logical address regardless of which buffer (Buffer0 or Buffer1) is currently active. A single configuration register (see section 4.4) controls which buffer is used to update the display and which buffer is selected for read/write operations. Double buffering can also be enabled/disabled via the double buffer configuration register. The base address should always be set as if there were no double buffering. Platinum double buffer hardware will generate the correct address for display refresh and read/write operations based on the buffer select bits in the double buffer register. The actual bank assignments for Buffer0 and Buffer1 are shown in Figure 4-1 for different VRAM configurations.

Changes to the buffer select bit for the display buffer in the double buffer register take effect during the vertical blanking period. If the display double buffer select bit is changed during the active video portion of a video frame, no change in the selected buffer will take effect until the vertical blanking period. Register changes made during vertical blanking will take effect immediately.

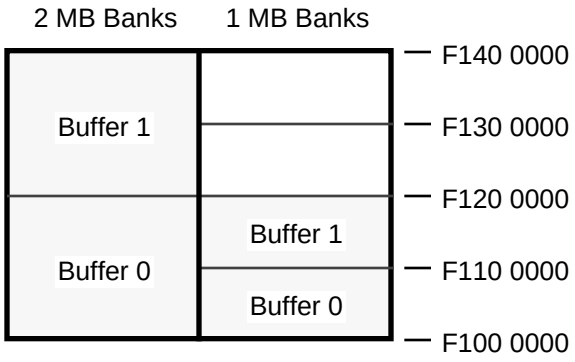


Figure 4-1 Double Buffer Physical Locations

To provide compatibility with processes that are not ‘double buffer aware’, a virtual buffer space is provided at $F140 0000 – $F17F FFFF. Reads from the virtual buffer space will always come from the front buffer, as determined by the read/write buffer select bit. Writes to virtual buffer space will occur to both buffers simultaneously.

4.5.7 Little Endian Aperture

The 4 MB of VRAM supported by the Platinum VRAM controller is mirrored at $F180 0000 – $F1FF FFFF to provide a little endian port into the display buffer. This will make it easier to port a little endian oriented OS to run on a Platinum-based computer. This also makes it easier to use PCI cards that generate little endian ordered pixel data. (The Iridium datapath chip contains logic to rearrange little endian pixel data to the big endian ordering required by the video subsystem, primarily the CLUT/DAC and the QuickDraw accelerator).

4.6 QuickDraw Accelerator

All pixels must be aligned in memory on their natural boundary: 8-bit pixels on any byte address, 16-bit pixels on any even byte address, and 32-bit pixels on any address evenly divisible by 4. 1 bpp source images can be aligned on any bit within a byte.

Not every accelerator operation will use every configuration register. In general, only the registers and fields used by a particular command need to be programmed. Unneeded registers are considered “don’t cares” by the accelerator logic. The accelerator modifies the contents of the Command/Status register only; all other register contents are undisturbed by the execution of any command. If a series of commands uses the same value for one or more registers, those registers do not need to be reloaded after the first command.

In general, a software driver for the accelerator will check its status by reading the Command/Status register. If the “Command in Progress” bit is not set, the accelerator is idle and can be programmed for the next command. Software should program the required registers — some or all of the first 8 accelerator registers — and then write a 1 into the “Go” bit of the Command/Status register. Software can then poll the Command/Status register to see when the accelerator has finished the command, when the busy bit changes from 1 to 0. Alternatively, software can set the interrupt enable bit in the Control 2 register and simply wait for the command completion interrupt from the accelerator.

4.6.1 Foreground and Background Colors

The Foreground and Background Color registers will normally be set to QuickDraw's current foreground and background color data before each command. The accelerator will use these colors during execution according to the table 2-1 shown earlier.

The Or and Bic modes treat the foreground and background color registers differently than the other transfer modes. In order to simplify the hardware implementation, these modes use both registers even though QuickDraw would only use one of the two colors. When performing an Or mode operation (srcOr, notSrcOr, patOr, or notPatOr), both foreground and background registers should be programmed with the current foreground color. When performing a Bic mode operation (srcBic, notSrcBic, patBic, or notPatBic), both foreground and background registers should be programmed with the current background color.

4.6.2 Source and Destination Base Addresses

The accelerator is capable of accessing all of DRAM and VRAM and can move data to/from both areas. It does no bounds checking on the addresses it is given, so software must be careful to program realistic values into the base registers. The accelerator assumes that if the most significant bit of a base register is set the address points into the VRAM memory space, and if it is cleared the address points into the DRAM memory space. DRAM accesses incur a slight penalty since a system address bus cycle is performed for every reference to DRAM. This is done to guarantee coherency in the processor's L1 cache and the system's L2 cache. Referenced data in the L1 cache will be written back to memory before the accelerator performs its DRAM access.

4.6.3 Row Bytes

The Row Bytes register holds the QuickDraw row bytes value from the pixmap data structure for the source and destination images. Each field is 16 bits wide even though the QuickDraw row bytes parameter can be only 13 bits. Separate fields for the source and destination images allow for doing operations like copying a packed source image (one where row bytes is equal to the horizontal size) from DRAM to the non-packed frame buffer in VRAM. Unlike QuickDraw, the accelerator has no requirement that the row bytes value be even; restricting the value to be even would not have made the hardware any easier to implement and providing a superset of QuickDraw's functionality may prove useful in the future. The destination row bytes value is required for all operations while the source row bytes value is required only for operations that read a source (or pattern) image.

4.6.4 Destination Size

The horizontal and vertical dimensions (in pixels) of the destination image are required for all accelerator operations. The minimum size is 1 pixel by 1 pixel and the maximum size is $2^{16} - 1$ pixels by $2^{16} - 1$ pixels. Since there is only one register for the horizontal and vertical dimensions of the images, both the source and destination must be the same size in pixels. The accelerator does not perform any shrinking or stretching of the source image to match it to the destination's size.

4.6.5 Pattern Sizes

Patterns are limited to 8, 16, 32, 64, 128, or 256 pixels in each dimension. Patterns do not have to be square; for example 8 x 64 is a valid pattern size. Patterns must be aligned on their natural boundary in memory, i.e. an 8 x 8 8 bpp pattern must be aligned on a 64 byte boundary in memory and a 256 x 256 32 bpp pattern must be aligned on a 256K byte boundary. The source starting address can be any pixel within the pattern. This allows software to correctly align the pattern with the origin of the display without having to manually rotate the pattern before use. Patterns whose line size (horizontal pixels times bytes per pixel) is 64 bytes or less are more efficient in terms of memory bandwidth since the accelerator will load the pattern's current line only once per line of the destination rectangle. The pattern size fields need to be programmed for pattern transfer operations only.

4.6.6 Direction Control

The Processing Direction bit in the Command 1 register controls how the accelerator traverses the source and destination rectangles. If it is a 1 the accelerator increments the addresses, covering the images left to right, top to bottom. If it is a 0 the accelerator decrements the addresses, covering the images from right to left, bottom to top. The appropriate choice depends on the relative position of the destination rectangle to the source rectangle. If the two rectangles are identical or if there is no overlap between them, either direction is valid. If the two rectangles overlap in memory, the setting of the direction bit is critical to insure the source is not overwritten before it is used. Figure 4-2 shows the possible overlaps between the source and destination rectangles, and the proper setting of the direction control bit. (The center square shows the case where source and destination rectangles are the same and either value for the direction bit is useable.)

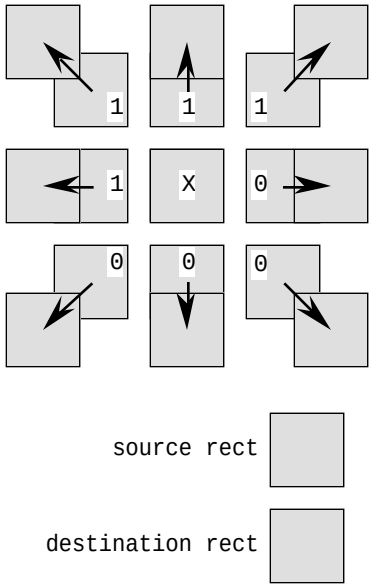


Figure 4-2 Source/Destination Rectangle Overlaps

4.6.7 Pattern Mode and Raster Op

The four bits of the combined pattern mode and raster op fields map directly to the 16 transfer mode constants defined by QuickDraw. These bits control the basic functions of the transfer operation logic block in Iridium. Their behavior is modified by the Fill, Colorize, and Expand control bits.

4.6.8 Fill Mode

Fill mode writes a single color to every pixel of the destination image. Fill mode uses the Destination Starting Address, Destination Row Bytes, Destination Size, and parts of the Control 1 and Control 2 registers to define the area of memory to be filled with the contents of the Foreground Color register. The pattern size and bit offset fields of the Control 1 register are ignored. The expand, colorize, and pattern mode bits and the raster operation field of the Control 2 register are also ignored.

4.6.9 Colorize Mode

The colorize, or “Ted Turner Memorial”, mode bit controls the accelerator’s use of the Foreground and Background color registers. If this bit is cleared, the accelerator ignores the color registers for all commands but the fill command. If this bit is set, the accelerator uses the color registers for all commands. Setting the foreground color to black (all 1s for 8 bpp or all 0s for 16 and 32 bpp) and the background color to white (all 0s for 8 bpp or all 1s for 16 and 32 bpp) while in colorize mode is the equivalent of turning off colorize mode. The majority of QuickDraw operations will have the colorize bit set.

4.6.10 Expand Mode

The expand mode bit in the Command 2 register causes the accelerator to treat the source image as a 1 bpp bit map. When the expand mode bit is set, the bit offset field in the Command 1 register is used to locate the first bit within the first byte of each line of the source image. The bit ordering is big endian with bit 0 being the leftmost, or most significant, bit of the byte, and bit 7 being the rightmost, or least significant, bit of the byte. In the 1 bpp source 1s are considered the black foreground pixels, and 0s are the white background pixels. The source is expanded out to the bit depth indicated by the Pixel Depth field in the Control 1 register before it is used in the transfer operation logic. The expand mode may be used in conjunction with the colorize mode and any of the source or pattern transfer modes.

4.7 Configuration and Status Registers

All registers are accessed as 32-bit quantities only; other size accesses will produce undefined results. Any unused bits are reserved; they must be written with 0's and are undefined on reads. Table 4-10 is a quick reference to the Platinum registers and where they reside in the address map. Each register is described in more detail in the following sections.

Table 4-10 Platinum Register Addresses

Address	Register Name	Read/Write	Description	Reference
0xF800 0000	CPUID	R	Catalyst CPU ID Register	Section 4.7.1
0xF800 0010	PLRev	R/W	Platinum Revision/Test Register	
0xF800 0020	ROMTiming	R/W	ROM Timing Configuration Register	
0xF800 0030	CacheCfg	R/W	Cache Configuration Register	
0xF800 0040	DRAMTiming	R/W	DRAM Timing Configuration Register	
0xF800 0050	RfrshTiming	R/W	DRAM Refresh Timing Register	
0xF800 0060	Base0	R/W	DRAM Bank 0 Base Address Register	
0xF800 0070	Base1	R/W	DRAM Bank 1 Base Address Register	
0xF800 0080	Base2	R/W	DRAM Bank 2 Base Address Register	
0xF800 0090	Base3	R/W	DRAM Bank 3 Base Address Register	
0xF800 00A0	Base4	R/W	DRAM Bank 4 Base Address Register	
0xF800 00B0	Base5	R/W	DRAM Bank 5 Base Address Register	
0xF800 00C0	Base6	R/W	DRAM Bank 6 Base Address Register	
0xF800 00D0	Base7	R/W	DRAM Bank 7 Base Address Register	
0xF800 00E0	SWReg	R/W	General Purpose Software Register	
0xF800 00F0	PCIMask	R/W	PCI Address Mask Register	
0xF800 0100	BaseAddr	R?W	Frame Buffer/Display Base Addresses	Section 4.7.2
0xF800 0110	—	—	—	
0xF800 0120	RowWords	R/W	Row Words	
0xF800 0130	ClkConfig	R/W	Video Clock Configuration	
0xF800 0140	FBConfig1	R/W	Frame Buffer Configuration 1	
0xF800 0150	FBConfig2	R/W	Frame Buffer Configuration 2	
0xF800 0160	PageMode	R/W	Page Mode Enable	
0xF800 0170	MonID	R/W	Monitor ID Sense Lines	
0xF800 0180	FBReset	R/W	Frame Buffer Reset	
0xF800 0190	DblBufCntl	R/W	Double Buffer Control	
0xF800 01A0	VTestReg	R/W	Frame Buffer Test	
0xF800 01B0	vRfrshTiming	R/W	VRAM Refresh Timing	
0xF800 01C0	—	—	—	
0xF800 01D0	—	—	—	
0xF800 01E0	—	—	—	
0xF800 01F0	—	—	—	

Table 4-10 Platinum Register Addresses (con't)

Address	Register Name	Read/Write	Description	Reference
0xF800 0200	SwatchMode	R/W	Swatch Mode	Section 4.7.3
0xF800 0210	IntMask	R/W	Interrupt Mask	
0xF800 0220	IntStatus	R/W	Interrupt Status	
0xF800 0230	ClrCInt	R/W	Clear Cursor Interrupt	
0xF800 0240	ClrAInt	R/W	Clear Animation Line Interrupt	
0xF800 0250	ClrVBLInt	R/W	Clear VBL Interrupt	
0xF800 0260	CursLine	R/W	Cursor Line	
0xF800 0270	AnimLine	R/W	Animation Line	
0xF800 0280	CntrTst	R/W	Counter Test	
0xF800 0290	hSERR	R/W	Horizontal Serration Pulse	
0xF800 02A0	hHALFLN	R/W	Horizontal Half Line	
0xF800 02B0	hEQ	R/W	Horizontal Equalization Pulse	
0xF800 02C0	hSP	R/W	Horizontal Sync Pulse	
0xF800 02D0	hBWAY	R/W	Horizontal Beezeway	
0xF800 02E0	hBRST	R/W	Horizontal Burst Gate	
0xF800 02F0	hBP	R/W	Horizontal Back Porch	
0xF800 0300	hAL	R/W	Horizontal Active Line	
0xF800 0310	hFP	R/W	Horizontal Front Porch	
0xF800 0320	hPIX	R/W	Horizontal Pixels	
0xF800 0330	vHLINE	R/W	Vertical Half Lines	
0xF800 0340	vSYNC	R/W	Vertical Sync Pulse	
0xF800 0350	vBPEQ	R/W	Vertical Back Porch Equalization	
0xF800 0360	vBP	R/W	Vertical Back Porch	
0xF800 0370	vAL	R/W	Vertical Active Line	
0xF800 0380	vFP	R/W	Vertical Front Porch	
0xF800 0390	vFPEQ	R/W	Vertical Front Porch Equalization	
0xF800 03A0	TimeAdj	R/W	Timing Adjust	
0xF800 03B0	CurLine	R	Current Scan Line	
0xF800 03C0	—	—	—	
0xF800 03D0	—	—	—	
0xF800 03E0	—	—	—	
0xF800 03F0	—	—	—	

Table 4-10 Platinum Register Addresses (con't)

Address	Register Name	Read/Write	Description	Reference
0xF800 0400	FGColor	R/W	Foreground Color	Section 4.7.4
0xF800 0410	BGColor	R/W	Background Color	
0xF800 0420	SrcBase	R/W	Source Image Base Address	
0xF800 0430	DstBase	R/W	Destination Image Base Address	
0xF800 0440	RowBytes	R/W	Source/Destination RowBytes	
0xF800 0450	DstSize	R/W	Destination Size (H, V pixels)	
0xF800 0460	QDACntl	R/W	Control 1	
0xF800 0470	QDACmd	R/W	Control 2	
0xF800 0480	QDAStatus	R/W	Command/Status	
0xF800 0490	QDAPtr	R/w	Command List Pointer	
0xF800 04A0	SysConfig	R/W	System Configuration Register	
0xF800 04B0	SleepMode	R/W	Sleep (power down) Mode Enable	
0xF800 04C0	—	—	—	
0xF800 04D0	—	—	—	
0xF800 04E0	—	—	—	
0xF800 04F0	—	—	—	

4.7.1 Memory Subsystem Registers

The Platinum memory subsystem configuration/status registers are shown below with a description of each register.

CPU ID Register This register helps to uniquely identify the Catalyst chip set and board. There are a number of fields in this register:

Design Center: These bits indicate the machine is from the High-End RISC Design Center (0x3).

Upper ID: These bits are non-zero for machines which have 8-bit registers and zero for machines which have wider registers. For Platinum, these bits are 0 and the machine ID is found in the Lower ID bits.

Lower ID: These bits indicate this is a Platinum-based machine (0x01).

Clock Source: These bits indicate what frequency crystal is present on the board. These bits are latched from the dAddr[10] and dAddr[11] pins at the rising edge of /RESET. The valid values for these bits are:

clkSrc[0]	clkSrc[1]	Crystal Frequency
0	0	33 MHz
0	1	Reserved
1	0	20 MHz
1	1	31.3344 MHz

Motherboard ID: Thses bits are latched from the dAddr[6:4] pins at the rising edge of /RESET to identify the type of motherboard and ASIC configuration. Legal values include :

mbID[0]	mbID[1]	mbID[2]	Motherboard Type
0	0	0	Catalyst motherboard with 1 MB of VRAM
0	0	1	Reserved
0	1	–	Reserved
1	–	–	Reserved

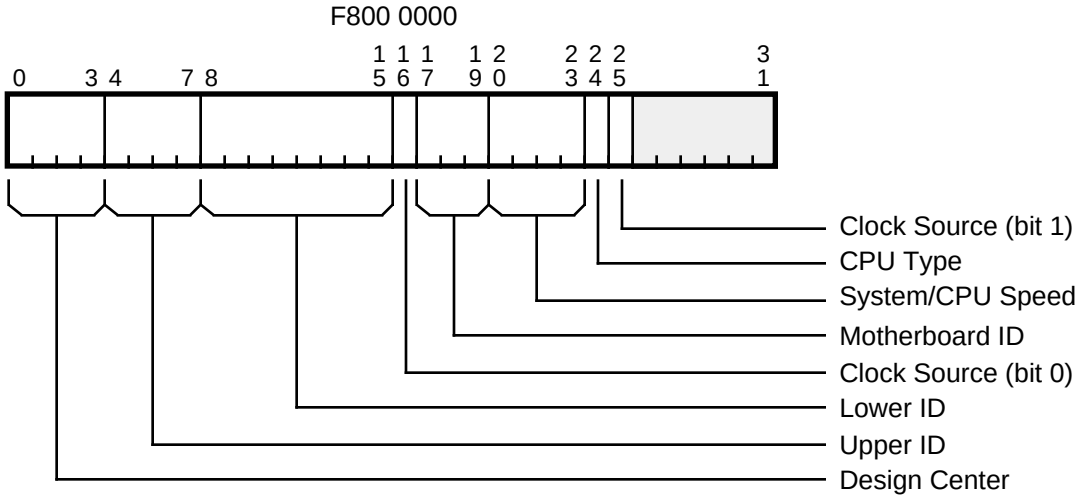
System/CPU Speed: These bits are latched from the dAddr[3:0] pins at the rising edge of /RESET. These bits, along with the Clock Source bits, identify the speed of the system bus clock and processor clock. The table below shows all valid combinations of these bits.

clkSrc[0]	clkSrc[1]	sSpeed[0]	sSpeed[1]	sSpeed[2]	sSpeed[3]	Processor/System speed
0	0	0	0	0	0	66 MHz / 33 MHz
0	0	0	0	0	1	80 MHz / 40 MHz
0	0	0	0	1	0	80 MHz / 27 MHz
0	0	0	0	1	1	100 MHZ / 50 MHz
0	0	0	1	0	0	100 MHz / 33 MHz
0	0	0	1	0	1	120 MHz / 60 MHz
0	0	0	1	1	0	120 MHz / 40 MHz
0	0	0	1	1	1	120 MHz/ 30 MHz
0	0	1	0	0	0	133 MHz / 66 MHz
0	0	1	0	0	1	133 MHz / 44 MHz
0	0	1	0	1	0	133 MHz / 33 MHz
0	0	1	0	1	1	150 MHz / 75 MHz
0	0	1	1	0	0	150 MHz / 50 MHz
0	0	1	1	0	1	150 MHz / 37 MHz

0	0	1	1	1	—	Reserved
0	1	—	—	—	—	Reserved
1	0	0	0	0	0	40 MHz / 20 MHz
1	0	0	0	0	1	48 MHz / 24 MHz
1	0	0	0	1	0	48 MHz / 16 MHz
1	0	0	0	1	1	60 MHz / 30 MHz
1	0	0	1	0	0	60 MHz / 20 MHz
1	0	0	1	0	1	72 MHz / 36 MHz
1	0	0	1	1	0	72 MHz / 24 MHz
1	0	0	1	1	1	72 MHz / 18 MHz
1	0	1	0	0	0	80 MHz / 40 MHz
1	0	1	0	0	1	80 MHz / 27 MHz
1	0	1	0	1	0	80 MHz / 20 MHz
1	0	1	0	1	1	90 MHZ / 45 MHz
1	0	1	1	0	0	90 MHz / 30 MHz
1	0	1	1	0	1	90 MHz / 22 MHz
1	0	1	1	1	—	Reserved
1	1	0	0	0	0	62 MHZ / 31 MHz
1	1	0	0	0	1	75 MHz / 38 MHz
1	1	0	0	1	0	75 MHz / 25 MHz
1	1	0	0	1	1	94 MHz / 47 MHz
1	1	0	1	0	0	94 MHz / 31 MHz
1	1	0	1	0	1	113MHz / 56 MHz
1	1	0	1	1	0	113 MHz / 38 MHz
1	1	0	1	1	1	113 MHz / 28 MHz
1	1	1	0	0	0	125 MHz / 63 MHz
1	1	1	0	0	1	125 MHz / 42 MHz
1	1	1	0	1	0	125 MHz / 31 MHz
1	1	1	0	1	1	141 MHz / 71 MHz
1	1	1	1	0	0	141 MHz / 47 MHz
1	1	1	1	0	1	141 MHz / 35 MHz
1	1	1	1	1	—	Reserved

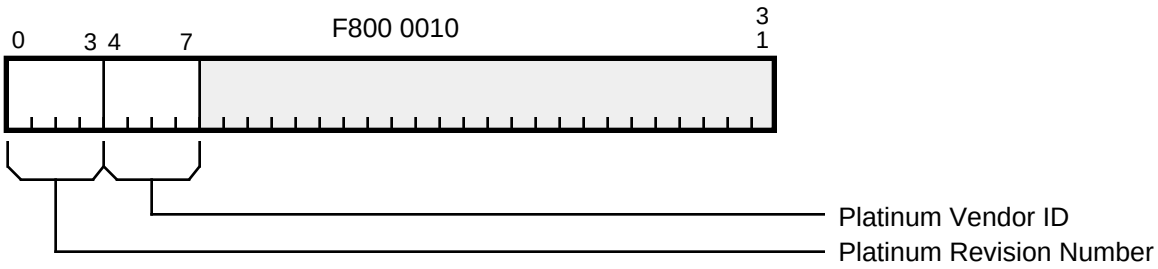
CPU Type: This bit is latched from the dAddr[12] pin at the rising edge of /RESET. When this bit is a 0 the board has a PPC 601 processor, and when this bit is a 1 the board has a PPC 603 (or 604) processor.

The entire register is read-only and has a default value of 0x3001???0.



Platinum Revision This register contains the revision level and vendor ID for Platinum. The current verison of Platinum has a revision level of 3. The vendor ID field is 0 for parts from VTI,

and 1 for parts from TI. The register is read-only and has a default value of 0x30000000 or 0x31000000.



ROM Timing This register allows software to configure ROM timing based on the system bus clock frequency. There are a number of fields in this register:

Burst Cycle Delay: This field specifies the number of system bus clocks for the second through fourth accesses of a burst transfer:

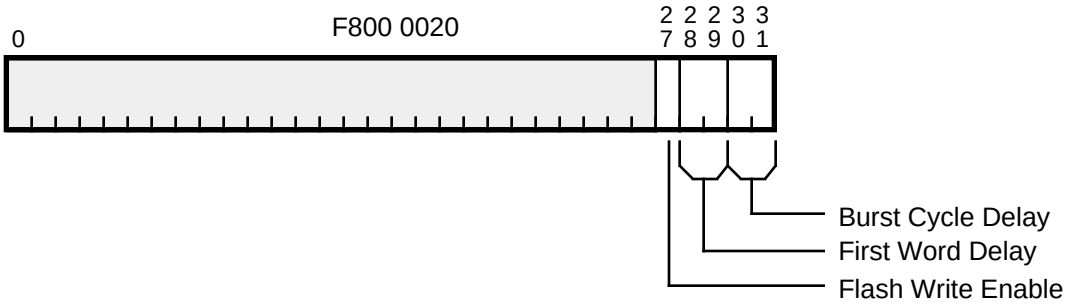
burstDly[1]	burstDly[0]	Access Time
0	0	6 clock cycles
0	1	5 clock cycles
1	0	4 clock cycles
1	1	3 clock cycles

First Word Delay: This field specifies the number of system bus clocks for the first ROM access:

firstDly[1]	firstDly[0]	Access Time
0	0	9 clock cycles
0	1	8 clock cycles
1	0	7 clock cycles
1	1	6 clock cycles

Flash Write Enable: This bit enables writing to the ROM space. When asserted, writes to ROM space are allowed (for programming Flash memories). When negated, writes to ROM space complete normally but without the write enable signal asserted.

The default value is 0x00000000.



Cache Configuration Register This register allows software to enable the second level cache and perform a number of diagnostic tests on the cache. There are a number of fields in this register:

Cache Enable: When this bit is a 1, the L2 cache is enabled. When this bit is a 0, the L2 cache is disabled and may be invalidated or tested.

Cache Tag Test Enable: When this bit is a 1, direct memory-mapped access to the L2 cache tag memory is enabled. This mode should be used to invalidate the entire cache on system powerup.

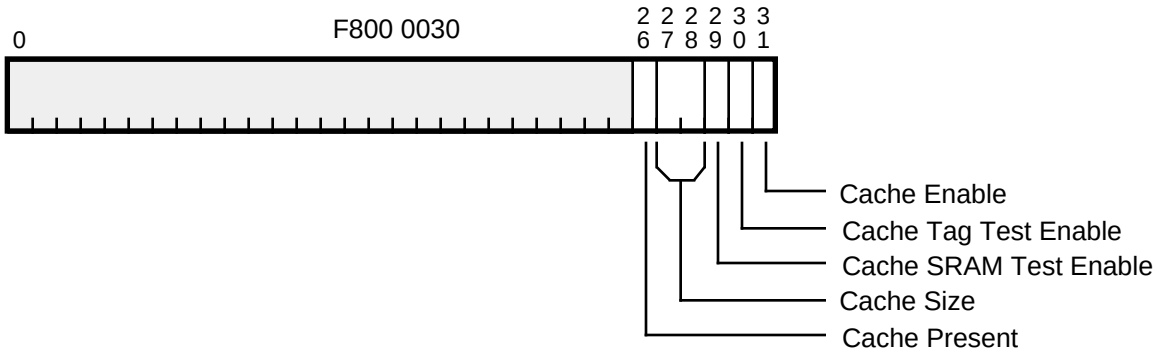
Cache SRAM Test Enable: When this bit is a 1, direct memory-mapped access to the L2 cache data memory is enabled.

Cache Size: This read-only field allows software to determine the size of the L2 cache. Size information from the cache SIMM (or the onboard L2 cache) is latched at the rising edge of /RESET from the dAddr[8:7] pins. The currently defined sizes are:

cSize[1]	cSize[0]	L2 Cache Size
0	0	512 KB
0	1	256 KB
1	0	1 MB
1	1	Reserved (treated as 1 MB)

Cache Present: This read-only bit will be a 1 if an L2 cache is present in the system. It will be a 0 if no L2 cache is present. This bit is latched at the rising edge of /RESET from the dAddr[9] pin.

The default value is 0x0000 0000.



DRAM Timing This register allows software to configure DRAM timing based on the system bus clock frequency. There are a number of fields in this register:

Slow Address Only Cycle: This bit, when a 1, delays the assertion of TS_ during QuickDraw accelerator address-only bus cycles by one system bus clock. This allows an extra clock cycle in which to drive the address onto the bus (useful for higher bus clock speeds).

QDA Read Ras Delay: This bit, when a 1, delays the assertion of RAS during QuickDraw accelerator reads by one system bus clock. This bit also delays the assertion of CAS by one clock cycle.

QDA Read Cas Delay 1: This bit, when a 1, extends the CAS low time during QuickDraw accelerator reads by one system bus clock.

QDA Read Cas Delay 2: This bit, when a 1, extends the CAS low time during QuickDraw accelerator reads by one system bus clock.

Page Mode Enable: This bit enables fast page mode operation for DRAM if a 1, and disables fast page mode operation if a 0.

Read Ras Delay: This bit, when a 1, delays the assertion of RAS during reads by one system bus clock. This bit also delays the assertion of CAS by one clock cycle.

Read Cas Delay 1: This bit, when a 1, extends the CAS low time during reads by one system bus clock.

Read Cas Delay 2: This bit, when a 1, extends the CAS low time during reads by one system bus clock. This bit affects read timing only if the “Read Cas Delay 1” bit is set.

Write Ras Delay: This bit, when a 1, delays the assertion of RAS during writes by one system bus clock. This bit also delays the assertion of CAS by one clock cycle.

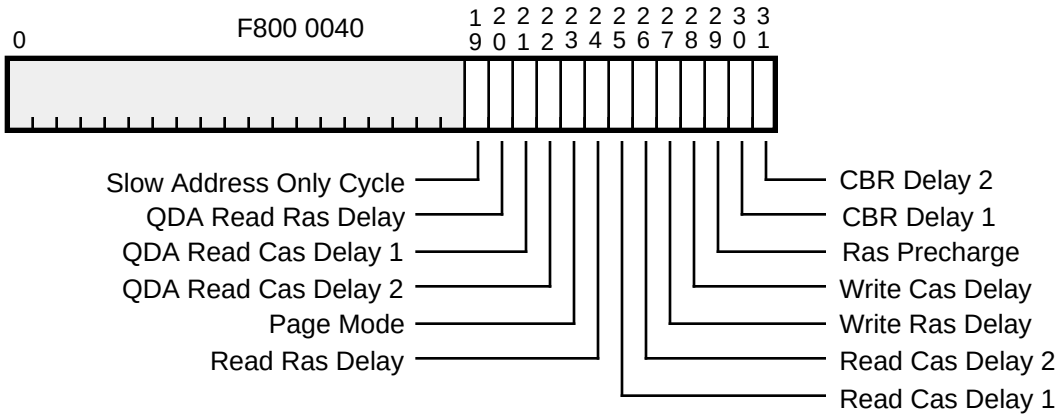
Write Cas Delay: This bit, when a 1, extends the CAS low time during writes by one system bus clock.

Ras Precharge: This bit, when a 1, extends the RAS high time by one system bus clock.

CBR Delay 1: Thsi bit, when a 1, extends the CAS-Before-RAS pulse widths by one system bus clock.

CBR Delay 2: This bit, when a 1, extends the CAS-Before-RAS pulse widths by one system bus clock. This bit affects refresh timing only if the “CBR Delay 1” bit is set.

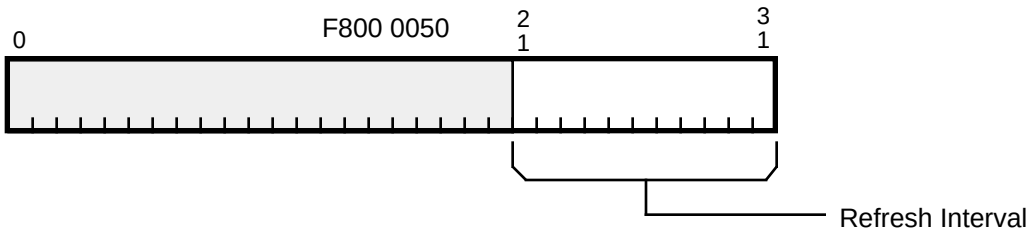
The default value is 0x0000 0EFF



DRAM Refresh Timing This register allows software to configure DRAM refresh timing based on the system bus clock frequency. The refresh interval is programmed with the number of clock cycles between DRAM refresh cycles. The value is a funtion of the system bus clock speed and the DRAM’s refresh period, and is equal to:

Refresh Interval = (Bus clock speed (in MHz) x DRAM refresh period (in µs)) -– 20

The 20 cycle “fudge factor” accounts for the maximum time it takes for a refresh request to be recognized by the DRAM controller, plus a few cycles margin. The default value is 0x0000 01F4.

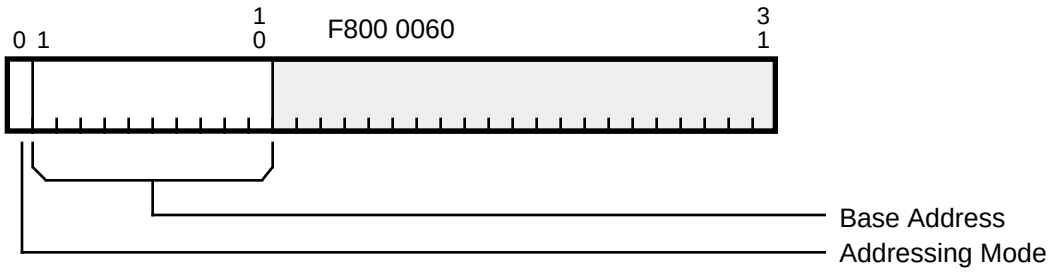


Bank 0 Base Register This register is used to set the DRAM configuration information for bank 0. There are a number of fields in this register:

Base Address: Bits 1–10 of the first address in bank 0. This field is read only (since bank 0 always starts at address 0x0000 0000) and will always read as 0x00.

Address Mode: This bit controls the DRAM address multiplexer when bank 0 is accessed.

The default value is 0x0000 0000.

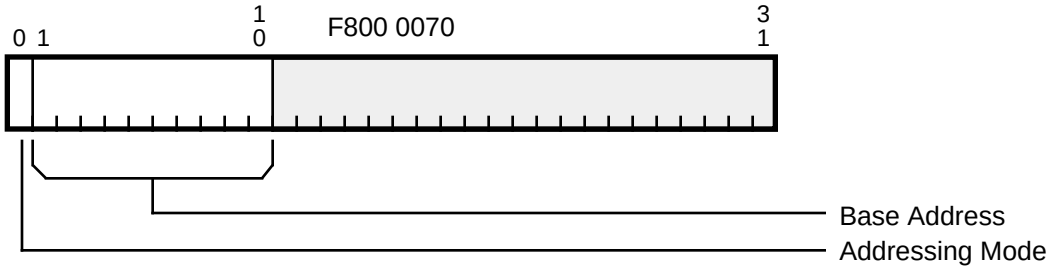


Bank 1 Base Register This register is used to set the DRAM configuration information for bank 1. There are a number of fields in this register:

Base Address: Bits 1–10 of the first address in bank 1.

Address Mode: This bit controls the DRAM address multiplexer when bank 1 is accessed.

The default value is 0x0800 0000.

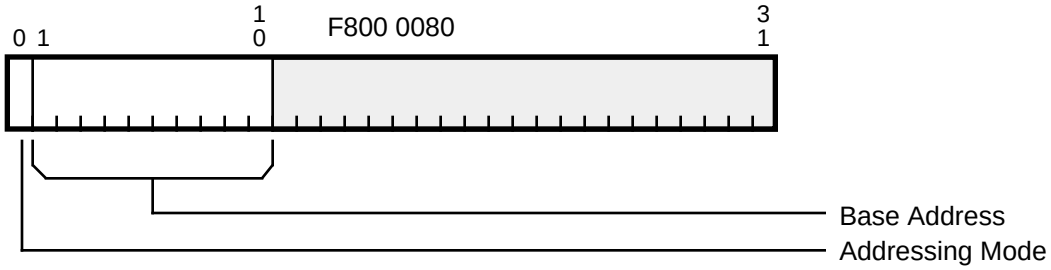


Bank 2 Base Register This register is used to set the DRAM configuration information for bank 2. There are a number of fields in this register:

Base Address: Bits 1–10 of the first address in bank 2.

Address Mode: This bit controls the DRAM address multiplexer when bank 2 is accessed.

The default value is 0x1000 0000.

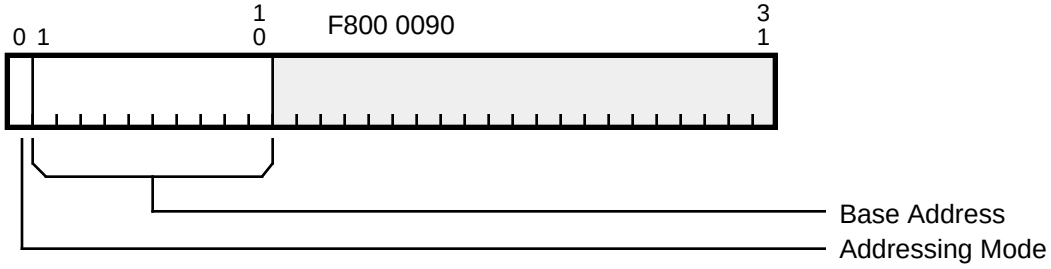


Bank 3 Base Register This register is used to set the DRAM configuration information for bank 3. There are a number of fields in this register:

Base Address: Bits 1–10 of the first address in bank 3.

Address Mode: This bit controls the DRAM address multiplexer when bank 3 is accessed.

The default value is 0x1800 0000.

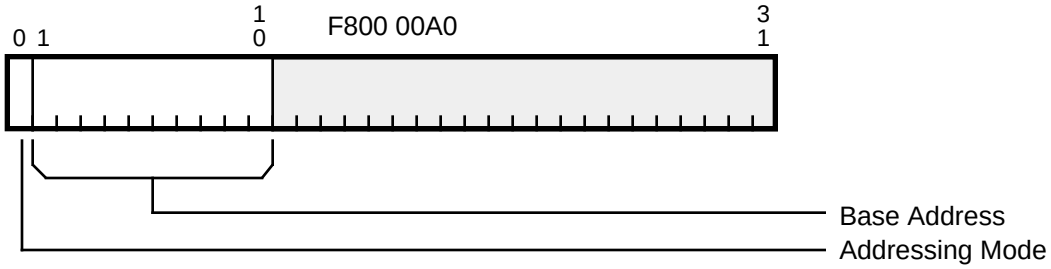


Bank 4 Base Register This register is used to set the DRAM configuration information for bank 4. There are a number of fields in this register:

Base Address: Bits 1–10 of the first address in bank 4.

Address Mode: This bit controls the DRAM address multiplexer when bank 4 is accessed.

The default value is 0x2000 0000.

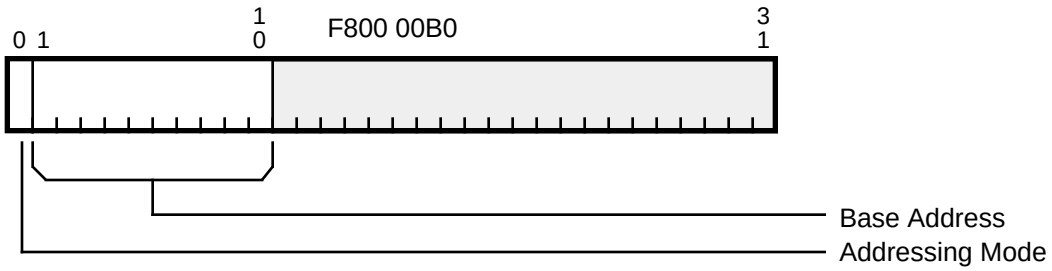


Bank 5 Base Register This register is used to set the DRAM configuration information for bank 5. There are a number of fields in this register:

Base Address: Bits 1–10 of the first address in bank 5.

Address Mode: This bit controls the DRAM address multiplexer when bank 5 is accessed.

The default value is 0x2800 0000.

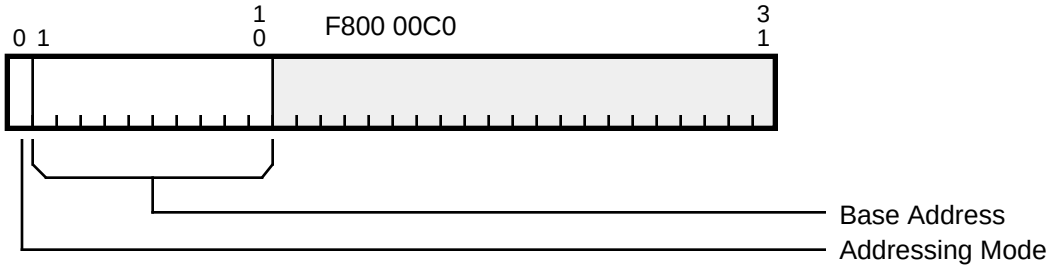


Bank 6 Base Register This register is used to set the DRAM configuration information for bank 6. There are a number of fields in this register:

Base Address: Bits 1–10 of the first address in bank 6.

Address Mode: This bit controls the DRAM address multiplexer when bank 6 is accessed.

The default value is 0x3000 0000.

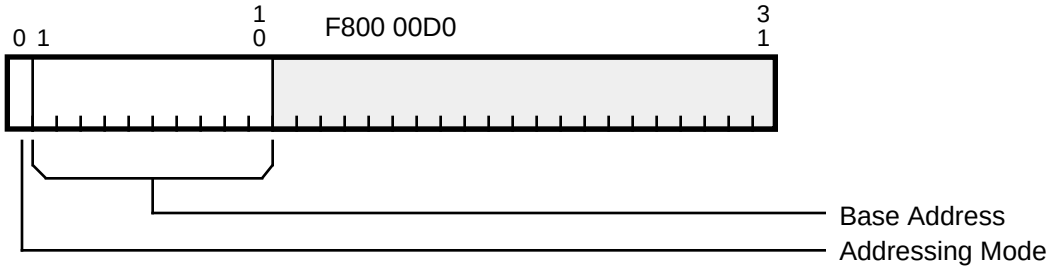


Bank 7 Base Register This register is used to set the DRAM configuration information for bank 7. There are a number of fields in this register:

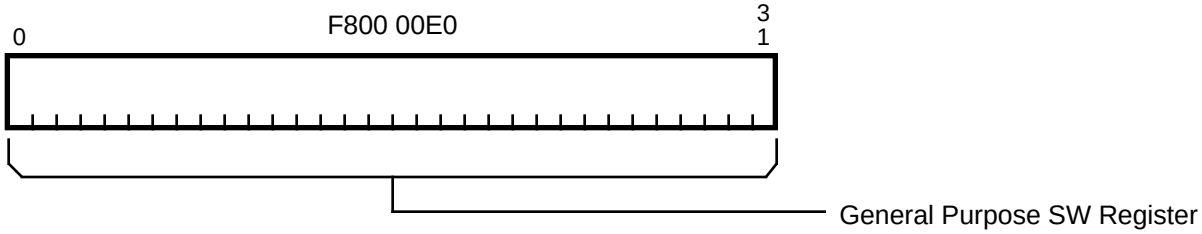
Base Address: Bits 1–10 of the first address in bank 7.

Address Mode: This bit controls the DRAM address multiplexer when bank 7 is accessed.

The default value is 0x3800 0000.



General Purpose Software Register This register is simply a 32-bit read/write location that can be used by software as a temporary storage place. Its intended use is to hold information needed by software during system initialization before the memory system has been configured.

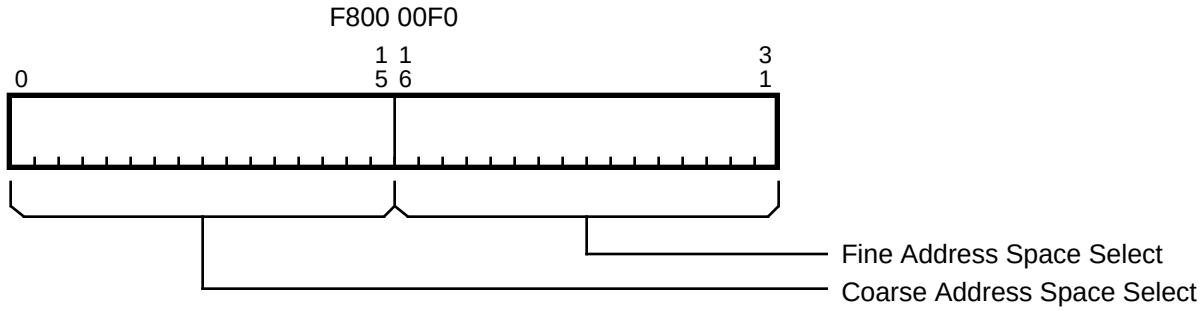


PCI Address Mask Register This register is a copy of the Address Mask Register in Bandit. These bits are used to control the address space watchdog timer in Platinum. Any space defined as belonging to Bandit will be ignored by the watchdog timer. Setting a bit prevents the watchdog timer from generating a bus timeout error for the corresponding portion of the address space. See the “TNT Bandit ASIC ERS” for on how this register functions. There are two fields in this register.

Fine Address Space Select: Bits 16–31 define which fine (16 MB) regions of the upper 256 MB of the system bus address space belong to Bandit and PCI.

Coarse Address Space Select: Bits 0–15 define which coarse (256 MB) regions of the system bus address space belong to Bandit and PCI.

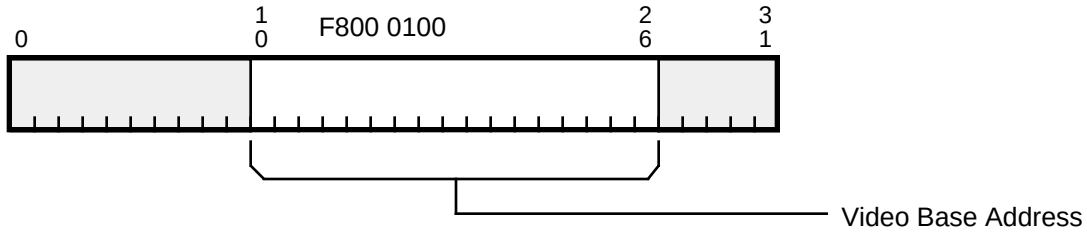
The default value is 0x0000 0000.



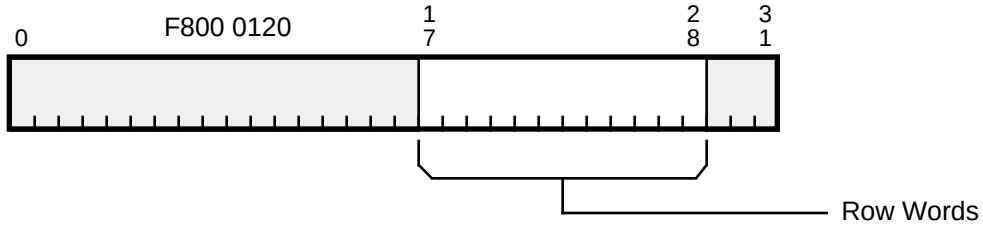
4.7.2 Frame Buffer Registers

The Platinum frame buffer configuration/status registers are shown below with a description of each register.

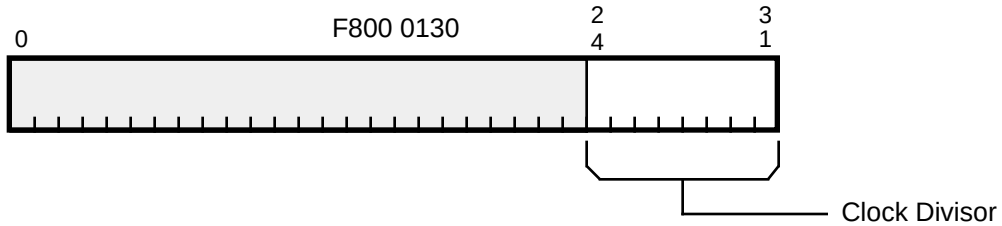
Display Base Address This register holds the upper 17 bits of the 22 bits that form the base address of the video display image in VRAM. The lower 5 bits are always 0. The default value is 0xF100 0000 (the upper bits match the physical base address of VRAM).



Row Words Bits 14–3 of the address offset from the start of one scan line to the start of the next (QuickDraw's RowBytes/8). The default value is 0x0000 0000



Clock Configuration This register holds the divisor that specifies the factor by which VidClk is divided down to generate the LD signal for the CLUT/DAC. 255 = divide by 1, 0 = divide by 2, 1 = divide by 3, etc. The default value is 0x00000000



Frame Buffer Configuration 1 This register holds a number of fields used to configure the frame buffer:

Full VRAM Banks: If this bit is cleared, the controller addresses VRAM as a 32-bit wide array (half the PowerPC's natural data bus width). If this bit is set, the controller addresses VRAM as a 64-bit wide array.

ROM Speed: This field is unused in Platinum (it is here for consistancy with the DaMFB ASIC). These bits are read only and will always return 0xF.

VRAM Refresh Count: This field is unused in Platinum (it is here for consistancy with the DaMFB ASIC). These bits are read only and will always return 0x0.

Video Enable: If this bit is set, video display refresh is enabled. If video refresh is disabled, sync and blank will still be generated but no pixel data will be sent to the CLUT/DAC.

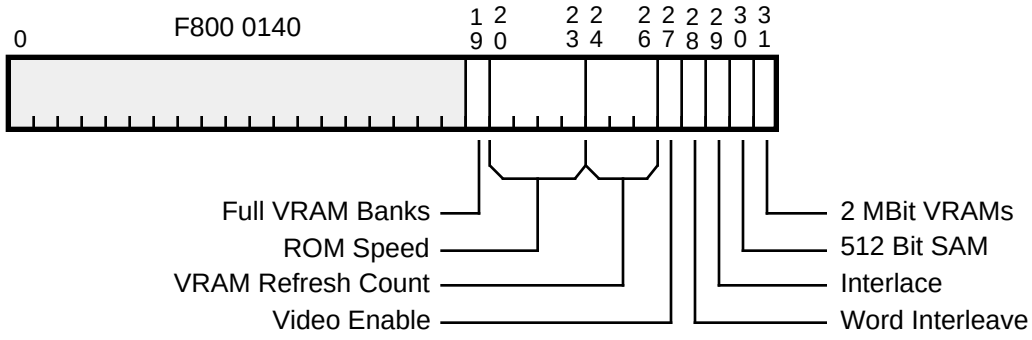
Word Interleave: This field is unused in Platinum (it is here for consistancy with the DaMFB ASIC). This bit is read-only and will always return 0x0.

Interlace: If this bit is set, interlaced video is enabled. If interlace is active, RowWords must be set to twice the row words value that QuickDraw uses and the active number of vertical half-lines must be odd.

512 Bit SAM: If this bit is set, the video refresh logic works with the 512 bit SAMs found in 256Kx4 and 256Kx8 VRAMs. If this bit is cleared, the video refresh logic is configured to work with 256 bit SAMs.

2 MB VRAMs: If this bit is set, the VRAM addressing scheme works with 256KxN parts. This results in a 2 MB VRAM bank size. If this bit is cleared, the VRAM addressing scheme works with 128Kx8 parts, resulting in a 1 MB VRAM bank size.

The default value is 0x00001F00.



Frame Buffer Configuration 2 This register allows software to configure VRAM timing based on the system bus clock frequency. There are a number of fields in this register:

QDA Read Ras Delay: This bit, when a 1, delays the assertion of RAS during QuickDraw accelerator reads by one system bus clock. This bit also delays the assertion of CAS by one clock cycle.

QDA Read Cas Delay 1: This bit, when a 1, extends the CAS low time during QuickDraw accelerator reads by one system bus clock.

QDA Read Cas Delay 2: This bit, when a 1, extends the CAS low time during QuickDraw accelerator reads by one system bus clock.

Read Ras Delay: This bit, when a 1, delays the assertion of RAS during reads by one system bus clock. This bit also delays the assertion of CAS by one clock cycle.

Read Cas Delay 1: This bit, when a 1, extends the CAS low time during reads by one system bus clock.

Read Cas Delay 2: This bit, when a 1, extends the CAS low time during reads by one system bus clock. This bit affects read timing only if the “Read Cas Delay 1” bit is set.

Write Ras Delay: This bit, when a 1, delays the assertion of RAS during writes by one system bus clock. This bit also delays the assertion of CAS by one clock cycle.

Write Cas Delay: This bit, when a 1, extends the CAS low time during writes by one system bus clock.

Ras Precharge: This bit, when a 1, extends the RAS high time by one system bus clock.

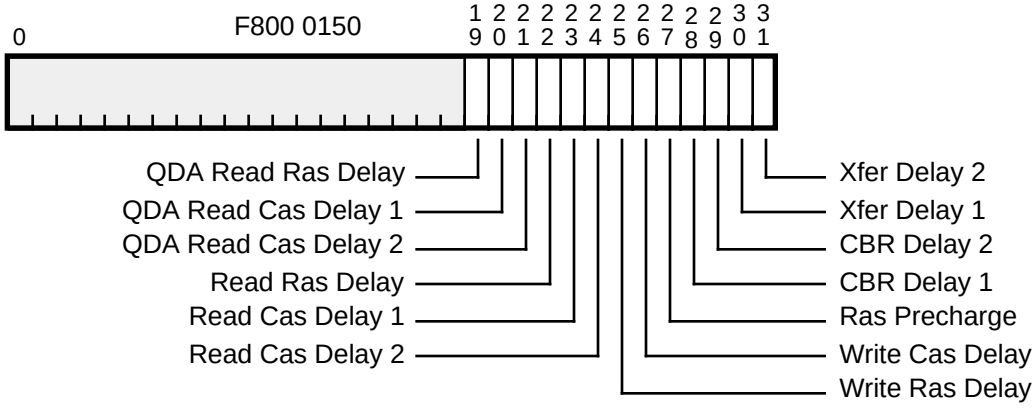
CBR Delay 1: Thsi bit, when a 1, extends the CAS-Before-RAS pulse widths by one system bus clock.

CBR Delay 2: This bit, when a 1, extends the CAS-Before-RAS pulse widths by one system bus clock. This bit affects refresh timing only if the “CBR Delay 1” bit is set.

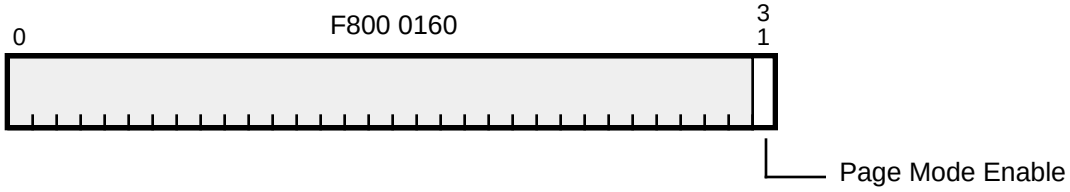
Xfer Delay 1: This bit, when a 1, extends the CAS low time by one system bus clock during full and split read transfer accesses.

Xfer Delay 2: This bit, when a 1, extends the CAS low time by two system bus clocks during full and split read transfer accesses.

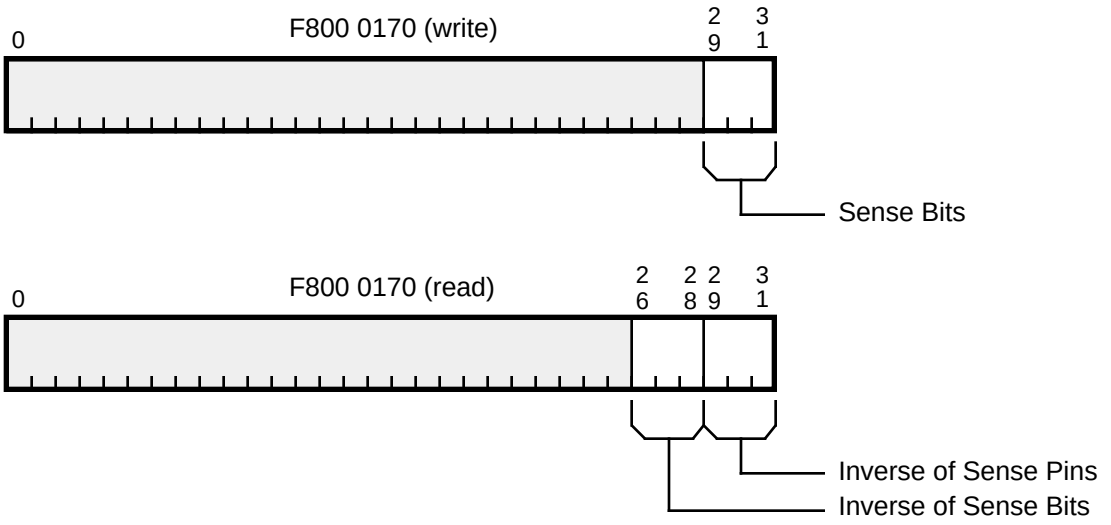
The default value is 0x00001FFF



Page Mode Enable This single bit register enables fast page mode operations for VRAM if a 1, and disables page mode if a 0. The default value is 0x00000000



Sense Line Enable This register may be written to in order to drive or tristate the 3 monitor sense lines, and may be read in order to determine the state of the lines. Writing a 1 will tristate the corresponding sense line output, and writing a 0 will enable that output. The value driven onto the sense lines depends on the contents of the test register (normally a 0). The value read from this register is inverted from the value written and from the value currently on the sense line pins, i.e. if the register is written with 0x0 and the test register is written with 0x001, the value read back will be 0x38. The default value (when read) is 0x00000000.

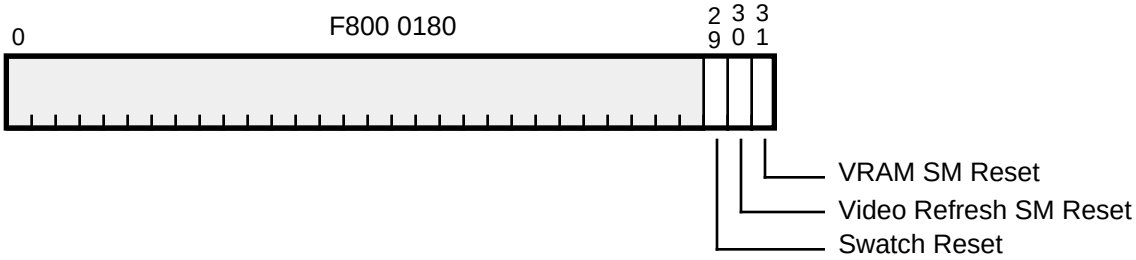


Reset This register holds a number of fields used to reset the components of the frame buffer controller:

VRAM SM Reset: This bit resets the VRAM controller state machine to its IDLE state. 1 = reset, 0 = enabled.

Video Refresh SM Reset: This bit resets the video refresh state machine to its IDLE state. 1 = reset, 0 = enabled.

Swatch Reset: This bit resets the Swatch video timing generator. 1 = reset, 0 = enabled.



The default value is 0x00000007, i.e., all three reset lines are asserted. Each of the three sections of the frame buffer controller must be explicitly taken out of reset by writing to this register in order for normal operation to begin. The recommended startup procedure is to first program all other configuration registers (except the Swatch interrupt mask register, which should only be written to after the frame buffer is fully initialized), including the CLU/DAC and pixel clock generator registers, and then to release the three sections from reset as follows:

- First, release Swatch from reset (i.e., write 0x3 to the register)
- Second, reassert the Swatch reset (0x7)
- Third, again release Swatch from reset (0x3)
- Fourth, release the VRAM state machine from reset (0x2)
- Finally, release the video refresh state machine (0x0)

At least 4 of the video clock periods should be allowed between successive writes to the reset register to assure that the change is reset state has time to propogate through internal synchronization logic.

*** WARNING ***: If an access is made to VRAM while the VRAM controller is held in reset, Platinum will ‘hang’ without returning a TA.

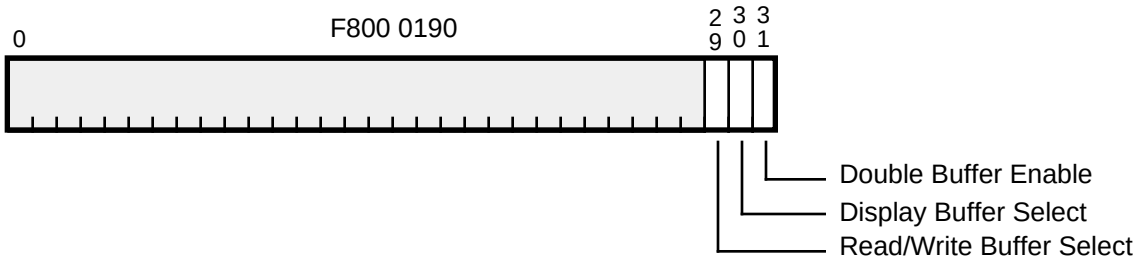
Double Buffer Control This register holds a number of fields used to control the double buffering feature in Platinum:

Double Buffer Enable: This bit enables double buffering. When enabled, display refreshes and read/write accesses will occur to the buffer selected by the two buffer select bits described below.
1 = enabled, 0 = disabled.

Display BufferSelect: This bit selects which of the two buffers is used to refresh the display.
1 = high order buffer, 0 = low order buffer.

Read/Write Buffer Select: This bit selects which of the two buffers is accessed by read/write operations.
1 = high order buffer, 0 = low order buffer.

The default value is 0x00000000



Frame Buffer Test Register This register is used for testing purposes only and contains a number of fields:

Sense Line Output Data: These bits hold the values driven onto the sense lines when the corresponding bits in the Sense Line Enable register are written to 0. These bits must be 0’s (low) for the monitor extended sense algorithm to work.

Syncs’ Output Enable: This bit controls the output enables for the HSYNC_, VSYNC_, and CSYNC_ pins. When this bit is written to 1, the sync outputs are tri-stated.

Frame Buffer Version Number: This is the version number of the frame buffer controller logic. (The value is non-zero since this is the 6th variation of the DAFB frame buffer controller.) These bits are read only.

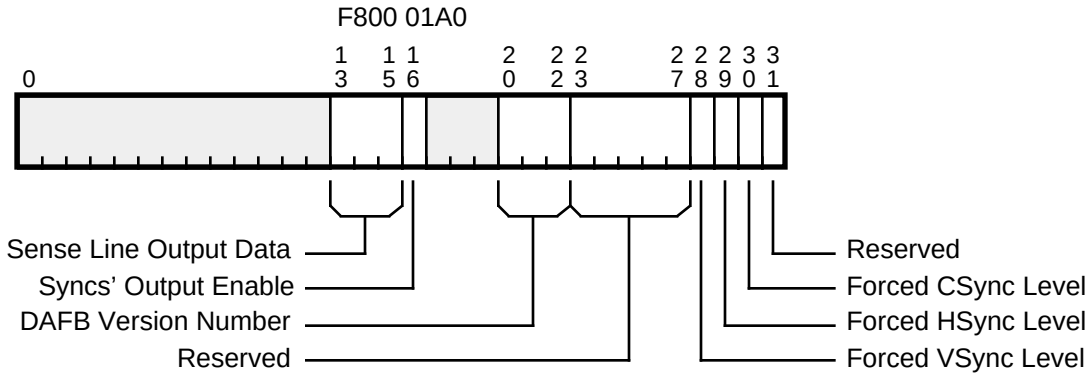
Reserved: These bits are unused in Platinum (they are used in the DaMFB ASIC). These bits are read only and will always return 0’s.

Forced VSync Level: This bit sets the level (high/low) that appears on the VSync (vertical sync) pin when VSync is forced to a static state via the Swatch Mode Register.

Forced HSync Level: This bit sets the level (high/low) that appears on the HSync (horizontal sync) pin when HSync is forced to a static state via the Swatch Mode Register.

Forced CSync Level: This bit sets the level (high/low) that appears on the CSync (composite sync) pin when CSync is forced to a static state via the Swatch Mode Register.

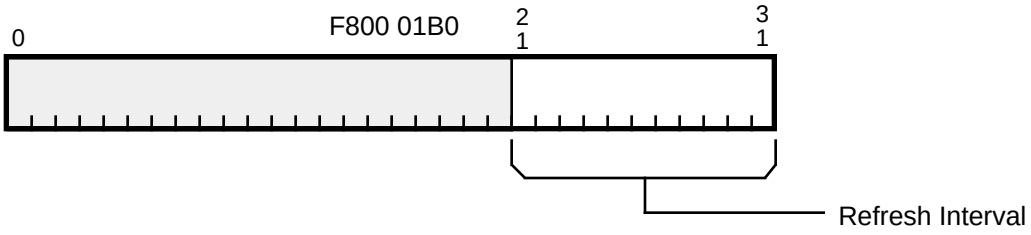
The default value is 0x00000C00.



VRAM Refresh Timing This register allows software to configure VRAM refresh timing based on the system bus clock frequency. The refresh interval is programmed with the number of clock cycles between VRAM refresh cycles. The value is a funtion of the system bus clock speed and the VRAM's refresh period, and is equal to:

Refresh Interval = (Bus clock speed (in MHz) x VRAM refresh period (in µs)) -– 20

The 20 cycle "fudge factor" accounts for the maximum time it takes for a refresh request to be recognized by the VRAM controller, plus a few cycles margin. The default value is 0x0000 01F4.



4.7.3 Video Timing (Swatch) Registers

Swatch registers are accessed in the same manner as any other Platinum register. For more complete information on how Swatch register values affect video timing, consult the Swatch Specification, version 2.01, February 15, 1990.

4.7.3.1 General Swatch Registers

Swatch Mode This is the Swatch configuration register. It holds a number of fields:

Reserved: These bits are unused in Platinum (they are used in the DaMFB ASIC). These bits are read only and will always return 1's.

CBlank Enable: If this bit is set, CBLANK behaves normally (i.e., it toggles). If this bit is cleared, CBLANK is static. The static value of CBLANK is always low.

VSync Enable: If this bit is set, VSYNC behaves normally (i.e., it toggles). If this bit is cleared, VSYNC is static. The static value of VSYNC is taken from the Test Register.

HSync Enable: If this bit is set, HSYNC behaves normally (i.e., it toggles). If this bit is cleared, HSYNC is static. The static value of HSYNC is taken from the Test Register.

CSync Enable: If this bit is set, CSYNC behaves normally (i.e., it toggles). If this bit is cleared, CSYNC is static. The static value of CSYNC is taken from the Test Register.

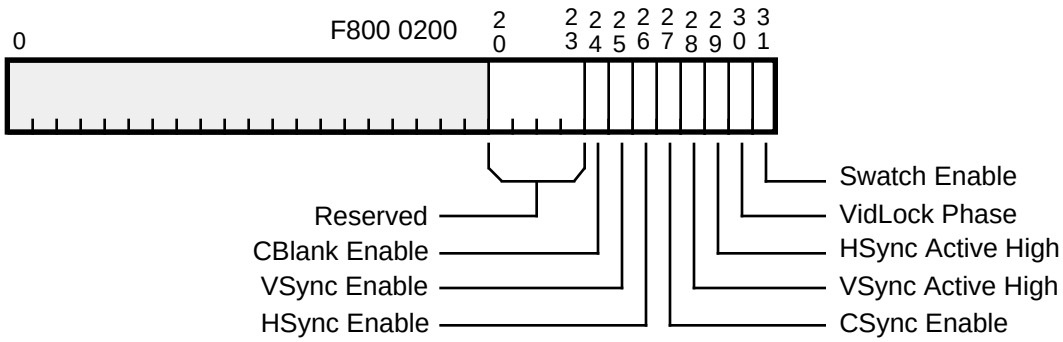
Vertical Active High: 1 = VSYNC active high, 0 = VSYNC active low. Does not affect CSYNC output.

Horizontal Active High: 1 = HSYNC active high, 0 = HSYNC active low. Does not affect CSYNC output.

VidLock Phase: sets the phase of the VIDCLK signal to which Swatch locks its timing generation. Since this bit is useful only if PIXCLK is available, it is read-only and will always be read as a '0'.

Swatch Enable: enables Swatch operation. 0 = enabled, 1 = disabled. Swatch is completely halted if disabled.

The default value is 0x00000FFD.



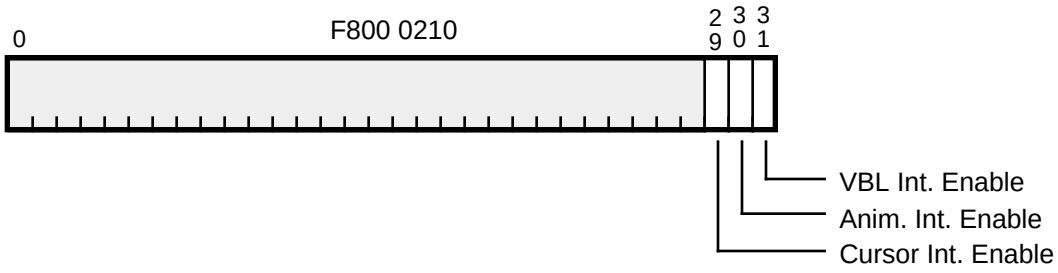
Interrupt Mask Interrupt mask register for Swatch interrupts.

VBL Interrupt Enable: 1 = vertical blanking interrupt enabled, 0 = VBL interrupt disabled.

Animation Interrupt Enable: 1 = animation line interrupt enabled, 0 = animation line interrupt disabled.

Cursor Interrupt Enable: 1 = cursor line interrupt enabled, 0 = cursor line interrupt disabled.

The interrupt mask register defaults to 0x00000000 (all interrupts disabled). Interrupts should not be enabled until the frame buffer has been programmed and all sections have been released from reset (see the description of the reset register).

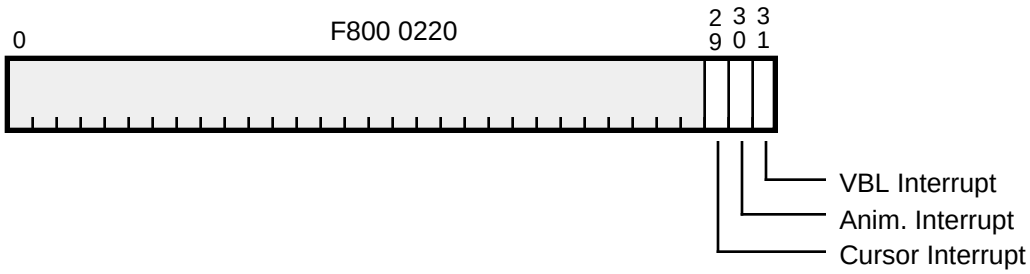


Interrupt Status Interrupt status register for Swatch interrupts (read only).

VBL Interrupt Status: 1 = vertical blanking interrupt asserted, 0 = VBL interrupt not asserted.

Animation Interrupt Status: 1 = animation line interrupt asserted, 0 = animation line interrupt not asserted.

Cursor Interrupt Status: 1 = cursor line interrupt asserted, 0 = cursor line interrupt not asserted.

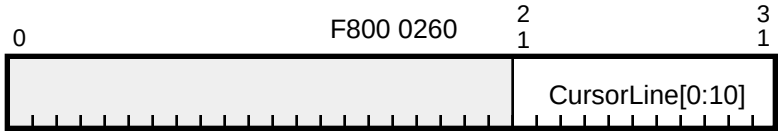


Clear Cursor Interrupt Access to this address (0xF8000230) clears the cursor line interrupt source.

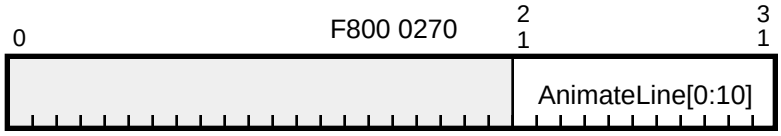
Clear Anim. Interrupt Access to this address (0xF8000240) clears the animation line interrupt source.

Clear VBL Interrupt Access to this address (0xF8000250) clears the vertical blanking interrupt source.

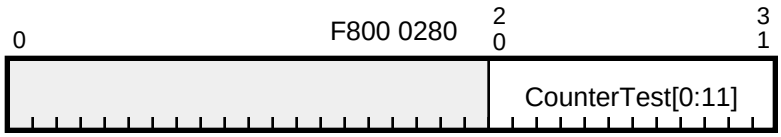
Cursor_Line This is the video line which will generate the cursor interrupt. It is specified in whole lines as an offset from the vertical zero reference point, which is normally at the beginning of vertical sync. The cursor line register defaults to 0x00000000.



Animate Line This is the line of output video which will generate the animation interrupt. The first line of vertical blanking is counted as line number 0. The animate line register defaults to 0x00000000.



Counter Test This value gets loaded into an internal Swatch counter when the appropriate load signal is asserted. This register is unused in Platinum (the Swatch counters are now tested with the scan test logic.) The counter test register defaults to 0x00000000.



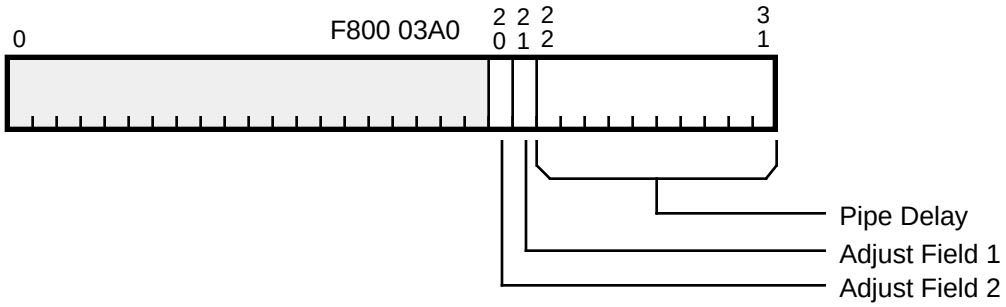
Timing Adjust Miscellaneous video timing adjustments. This register contains a number of fields:

Pipeline Delay: This parameter specifies the point in the horizontal timing at which pixel data from the VRAMs is started through the pixel data pipeline (VRAM to RaDACal). It is equal to the number of pixel clocks from the horizontal zero reference point at which the pixel data pipeline is to be started.

Adjust Field 1 and 2: Half-line field adjustments. A special adjustment is available to suppress the half-lines that appear in interlaced modes. Interlaced displays typically end one field in the middle of a horizontal line and begin the next field in the middle of a horizontal line. For some applications it may be desirable for the active video to contain no half-lines. By using the ADJF1 and ADJF2 bits, the vertical active areas for field I and field II can include only full lines. ADJF1 (adjust field I) and ADJF2 (adjust field II) are used to increase the size of the front and back porches by one half-line from field I to field II, as shown in the following table:

ADJF2	ADJF1	Result
0	0	No adjustments necessary
0	1	Field I parameters are increased by 1
1	0	Field II parameters are increased by 1
1	1	Not useful

The timing adjust register defaults to 0x00000000.



4.7.3.2 Swatch Horizontal Timing Registers

Swatch is composed of both horizontal and vertical timing generators. The horizontal timing generator is driven from the VIDCLK video clock input, Horizontal timing is specified by a number of parameters which control when transitions between different states in a horizontal line occur. These parameters and states are shown in Figure 4-2 below:

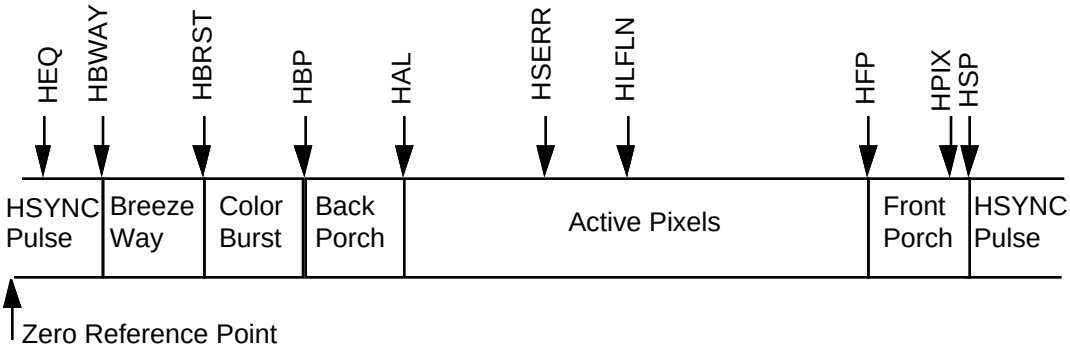
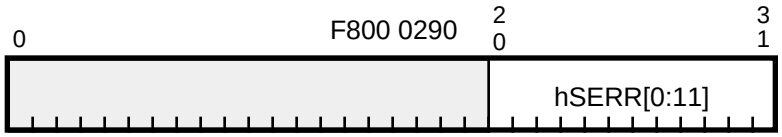


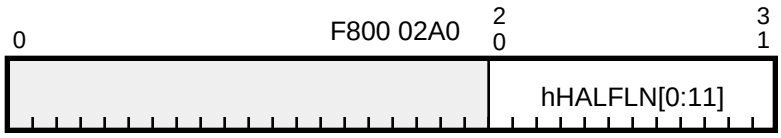
Figure 4-2. Swatch Horizontal Timing States

All the Swatch horizontal parameters are specified as the number of clocks from a horizontal zero reference point (shown in the above figure). The zero reference point must be located within the horizontal sync period.

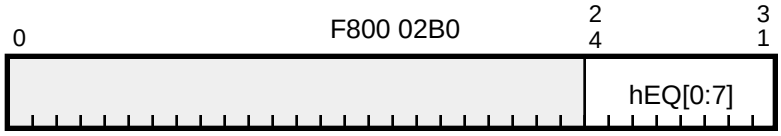
HSERR Where the serration pulse rises. Equal to the number of pixel clocks from the beginning of the horizontal sync period for the pulse in the first half of a line. Equal to the number of pixel clocks from the half line point for the pulse in the second half of a line. The HSERR register defaults to 0x00000000.



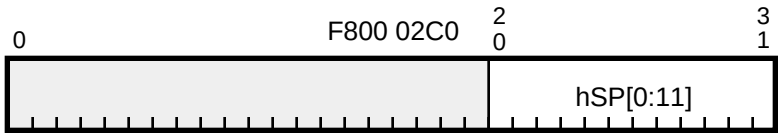
HLFLN Half-line point where equalizing pulses or serrations fall. Equal to the number of pixel clocks from the horizontal zero reference point. The HLFLN register defaults to 0x00000FFF.



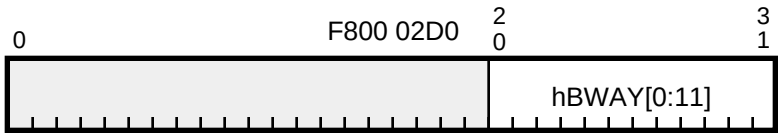
HEQ Horizontal equalizing pulse. Where an equalizing pulse would rise. Equal to the number of pixel clocks -1 from the beginning of the horizontal sync period for the pulse in the first half of a line. Equal to the number of pixel clocks - 1 from the half line point for the pulse in the second half of a line. The HEQ register defaults to 0x00000000.



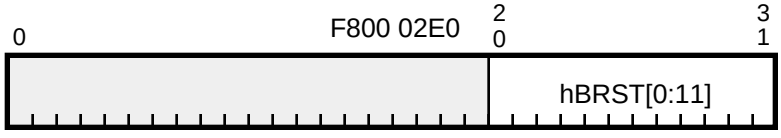
HSP Horizontal sync pulse. Where the horizontal front porch ends and the horizontal sync pulse begins. The only requirement for HSP programming is that the zero reference point be within the horizontal sync period, so HSP should be programmed to start before the zero reference point (i.e., at the end of the horizontal line). It is suggested that HSP be programmed to be one less than the number of clocks in a horizontal line. The HSP register defaults to 0x00000000.



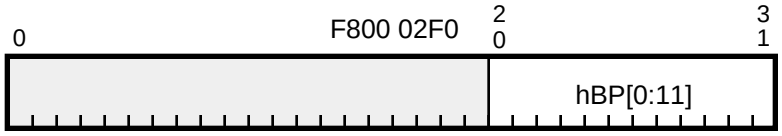
HBWAY Horizontal BreezeWay. Where the horizontal sync pulse ends. Equal to the number of pixel clocks -1 from the horizontal zero reference point. The HBWAY register defaults to 0x00000000.



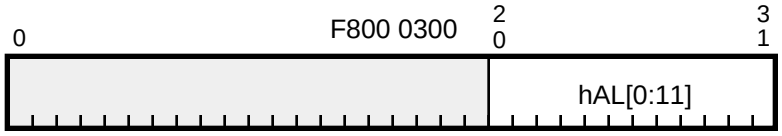
HBRST Horizontal Burst. Where the horizontal burst gate begins. Equal to the number of pixel clocks - 1 from the horizontal zero reference point. The HBRST register defaults to 0x00000000.



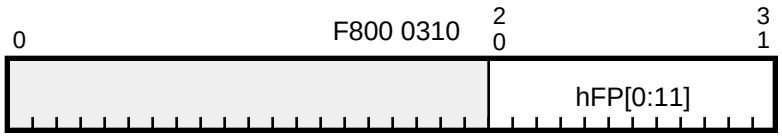
HBP Horizontal Back Porch. Where the horizontal burst gate pulse ends. Equal to the number of pixel clocks - 1 from the horizontal zero reference point. The HBP register defaults to 0x00000000.



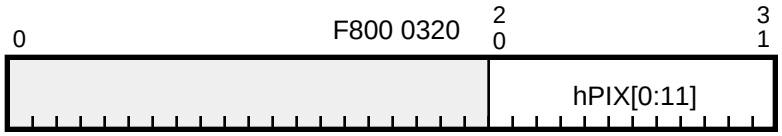
HAL Horizontal Active Line. Where the pixel data for a horizontal line begins. Equal to the number of pixel clocks -1 from the horizontal zero reference point. The HAL register defaults to 0x00000000.



HFP Horizontal Front Porch. Where the pixel data for a horizontal line ends. Equal to the number of pixel clocks -1 from the horizontal zero reference point. The HFP register defaults to 0x00000000.



HPIX Horizontal Pixels. HPIX is equal to the (total number of pixel clocks - 2) in a horizontal line. The HPIX register defaults to 0x00000000.



4.7.3.3 Swatch Vertical Timing Registers

The vertical timing generator is driven from a clock which is only half the frequency of the horizontal timing generator clock. However, vertical state transitions can only occur at intervals of half a horizontal line (i.e., at the beginning and middle of a line). Vertical timing is specified by a number of parameters which control when transitions between different states in a video field occur. These parameters and states are shown in Figure 4-3 below:

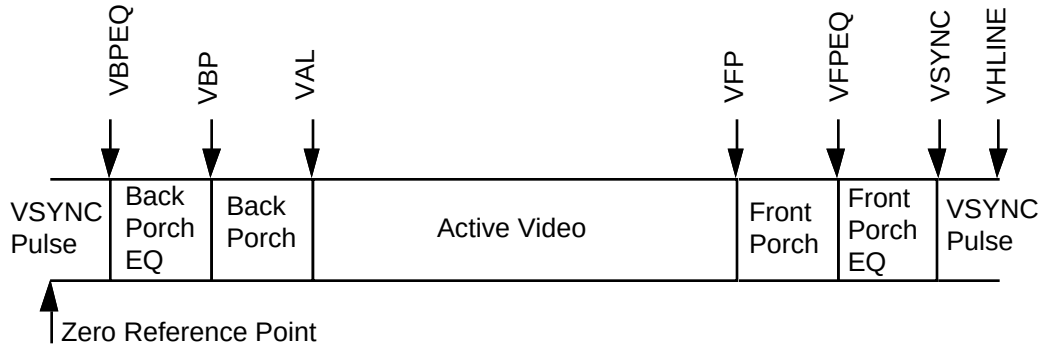
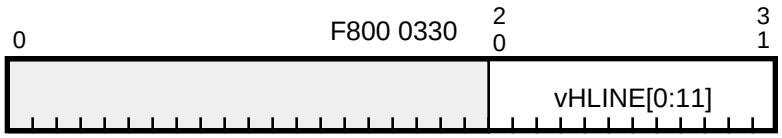


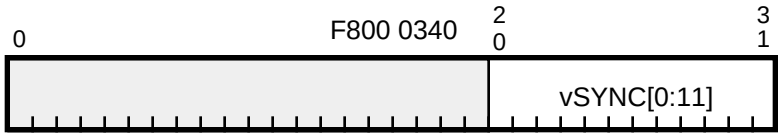
Figure 4-3 Swatch Vertical Timing States

All the Swatch vertical parameters are specified as the number of half-lines from a vertical zero reference point (shown in the above figure). The zero reference point is located within the vertical sync period.

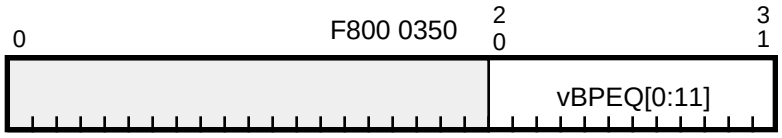
VHLINE Vertical Half Lines. The total number of half-lines in one field. VHLINE is odd for interlaced video and even for non-interlaced video. The VHLINE register defaults to 0x00000000.



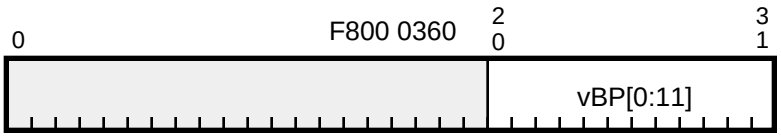
VSYNC Vertical Sync. Where the vertical sync pulse begins. For interlaced video it is VHLINE - 1. For non-interlaced video it is VHLINE - 2. The VSYNC register defaults to 0x00000000.



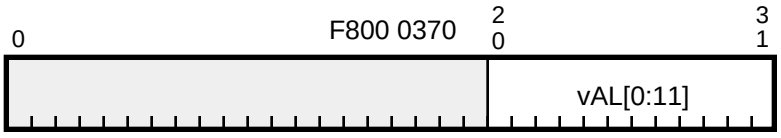
VBPEQ Vertical Back Porch Equalization. Where the vertical back porch with equalization pulses begins. Referenced from the vertical zero reference point. The VBPEQ register defaults to 0x00000000.



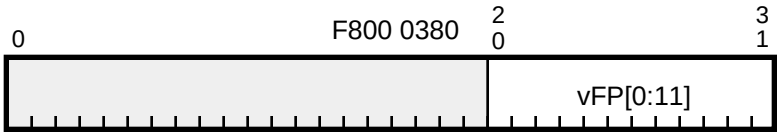
VBP Vertical Back Porch. Where the vertical back porch without equalization pulses begins. Referenced from the vertical zero reference point. The VBP register defaults to 0x00000000.



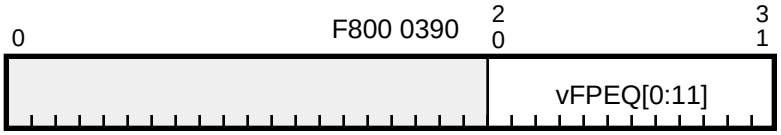
VAL Vertical Active Lines. Where the vertical active video area begins. Referenced from the vertical zero reference point. The VAL register defaults to 0x00000000.



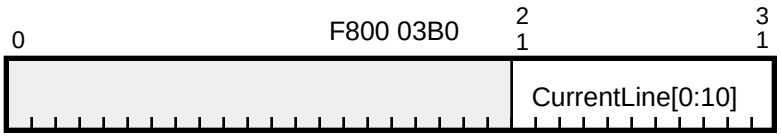
VFP Vertical Front Porch. Where the vertical front porch (without equalization pulses) begins. Referenced from the vertical zero reference point. The VFP register defaults to 0x00000000.



VFPEQ Vertical Front Porch Equalization. Where the vertical front porch with equalization pulses begins. Referenced from the vertical zero reference point. The VFPEQ register defaults to 0x00000000.



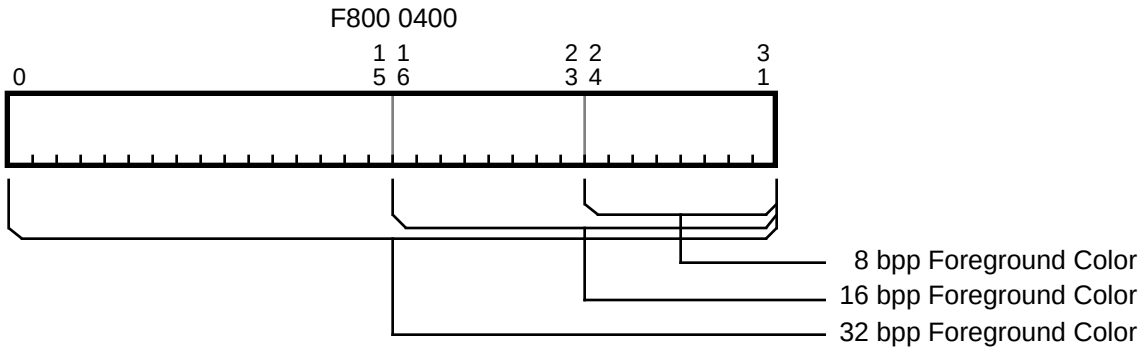
Current Line Current active video line. This is a read only register which returns the number of the currently active scan line. Note that this is the number of whole lines, not half-lines. Since the video timing logic is asynchronous to the register logic, this register should be read successively until two consecutive read operations return the same line number.



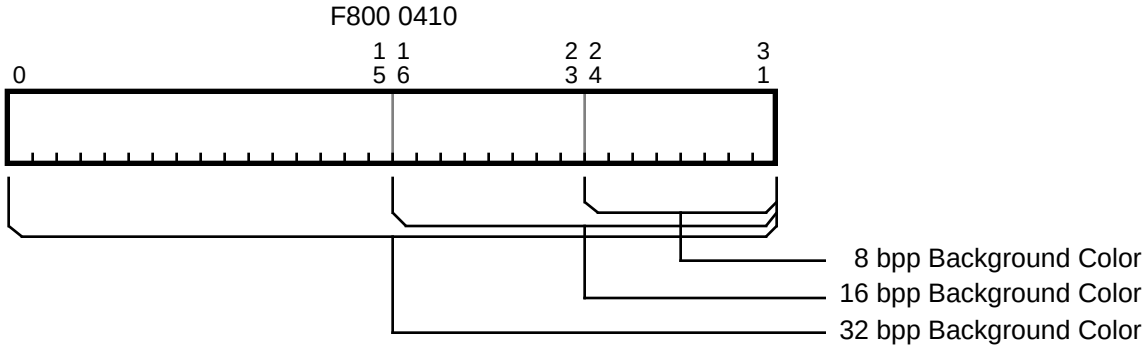
4.7.4 QuickDraw Accelerator Registers

The Platinum QuickDraw Accelerator configuration/status registers are shown below with a description of each register and its address.

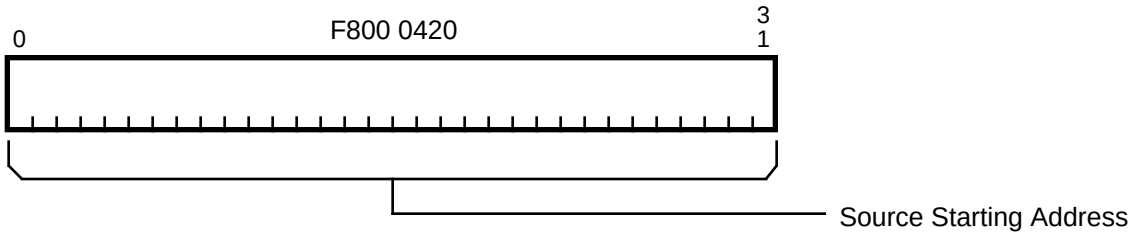
Foreground Color Register This 32-bit register holds the current foreground color. Valid data is right justified as shown below, and depends on the current pixel depth. This register physically exists in Iridium only, but can be read and written in the Platinum register space for ease of use. The default value is 0x00000000.



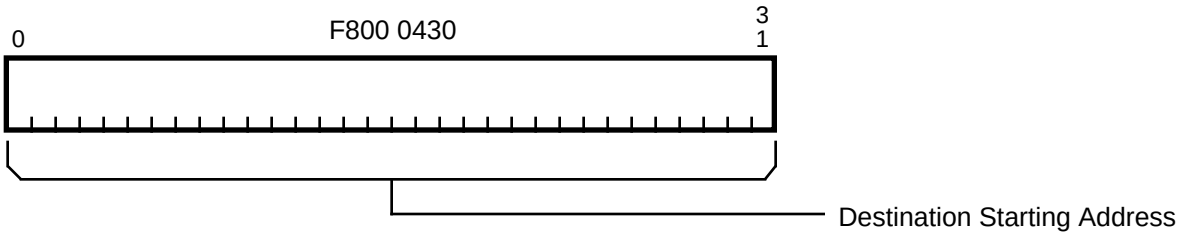
Background Color Register This 32-bit register holds the current background color. Valid data is right justified as shown below, and depends on the current pixel depth. This register physically exists in Iridium only, but can be read and written in the Platinum register space for ease of use. The default value is 0x00000000.



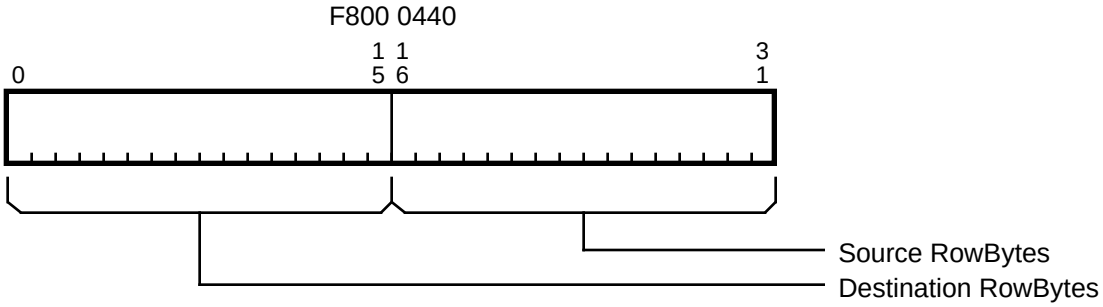
Source Starting Address This 32-bit register holds the address of the first pixel to be read from the source image. This is usually the pixel in the upper left or lower right corner of the source image, but may be any pixel in the source when a pattern transfer mode is selected. The default value is 0x00000000.



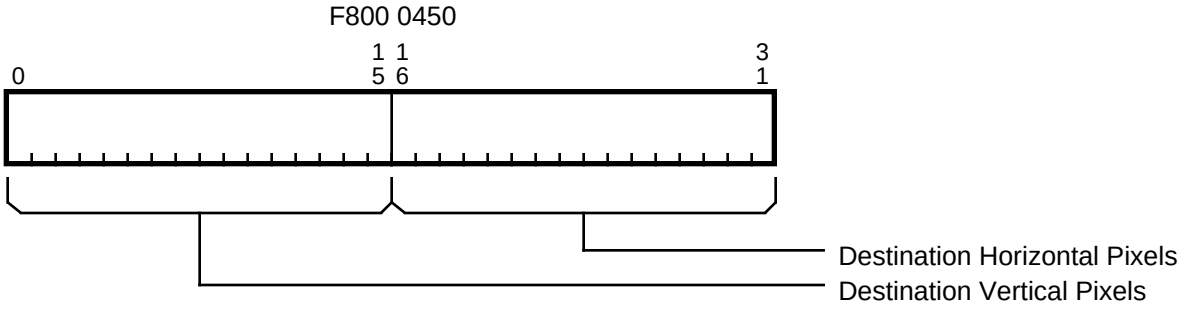
Destination Starting Address This 32-bit register holds the address of the first pixel to be written to in the destination image. This must be the pixel in the upper left or lower right corner of the destination image. (Which corner depends on how the source and destination images overlap. See the Control Register description for more details.) The default value is 0x00000000.



Destination/Source Row Bytes This 32-bit register holds the source and destination images' row bytes values. Row bytes is the number of bytes from the start of one line of the image to the start of the next line of the image. These values must be a multiple of the current pixel depth (in bytes). The default value is 0x00000000.



Destination Vertical/Horizontal Size This 32-bit register holds the vertical and horizontal dimensions, in pixels, of the destination image (and the source image if a pattern mode is not selected).
The default value is 0x00000000.



Control Register 1 This 32-bit register has a number of fields that help define the operation the accelerator is to perform:

Pattern Horizontal Pixels: These bits specify the horizontal size, in pixels, of the source pattern. The allowed pattern sizes are shown in the following table.

Pattern Horizontal Size	Register Contents
8	00_0001b
16	00_0010b
32	00_0100b
64	00_1000b
128	01_0000b
256	10_0000b

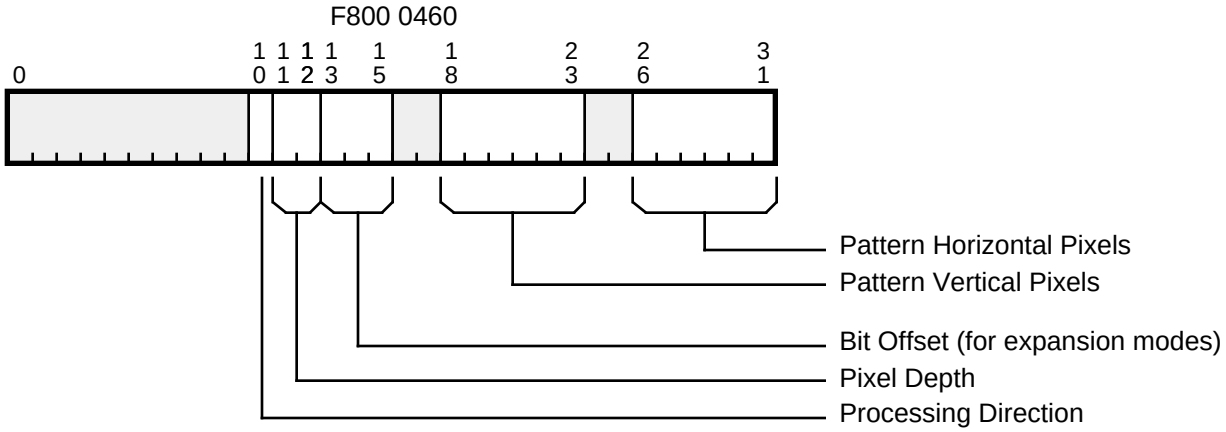
Pattern Vertical Pixels: These bits specify the vertical size, in pixels, of the source pattern. The allowed vertical sizes are the same as the allowed horizontal sizes shown above.

Bit Offset: These bits give the inital offset into the first byte of a 1 bpp source image, where 0 is the most significant bit and 7 is the least significant.

Pixel Depth: These bits encode the destination's bit depth. 0 = 8 bpp, 1 = 16 bpp, and 2= 32 bpp.

Processing Direction: This bit tells the accelerator how it should traverse the source and destination images. If this bit is a 1 addresses will be incremented so the image will be processed left to right, top to bottom. If this bit is a 0 address will be decremented, so the image will be processed right to left, bottom to top.

The default value is 0x00000000.



Control Register 2 This 32-bit register has a number of fields that help define the operation the accelerator is to perform. The lower 16 bits define the raster operation tthat will be used and the upper 16 bits define the command to be executed.

Raster Operation: This field determines which of the 8 standard QuickDraw transfer modes will be used during the next accelerator operation.

Pattern Operation: When this bit is set, the source image is a pattern that will be tiled onto the destination rectangle.

Fill: Setting this bit instructs the accelerator to fill the destination rectangle with the value in the Foreground Color register.

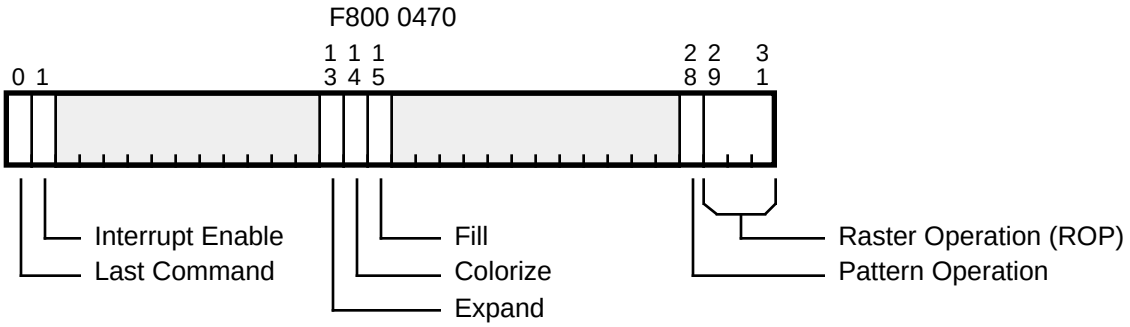
Colorize: Setting this bit instructs the accelerator to use the Foreground Color and Background Color registers when writing to the destination.

Expand: Setting this bit instructs the accelerator to treat the source image or pattern as a 1 bpp bit map. The source will be expanded out to the pixel depth specified in the Control Register before it is combined with the destination image.

Interrupt Enable: This bit, when set to 1, enables the command completion interrupt from the accelerator.

Last Command: Setting this bit tells the accelerator that the current command is the last one in the list of commands pointed to by the "Command List Pointer" register. The accelerator will complete the current command and halt. (Not implemented in this version of Platinum.)

The default value is 0x00000000.



Command/Status Register This 32-bit register has a number of fields:

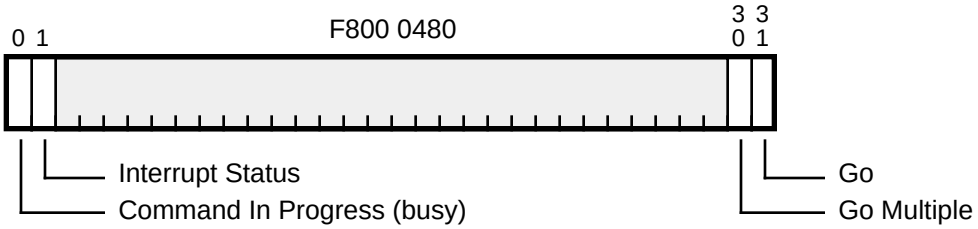
Go: Writing this bit to a 1 starts the accelerator. The accelerator will clear this bit once it starts.

Go Multiple: Writing this bit with a 1 instructs the accelerator to execute the list of commands pointed to by the "Command List Pointer" register. (Not implemented in this version of Platinum)

Interrupt Status: This bit gets set upon completion of the current command. If the interrupt enable bit in the Configuration Register is set, this bit will also generate an interrupt.

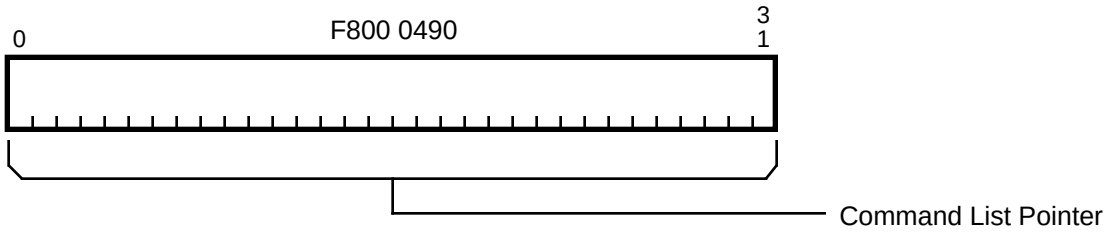
Command In Progress: This bit reflects the current state of the accelerator logic. If it is set, the accelerator is busy executing the current command. If it is cleared, the accelerator is idle and can be programmed for the next operation.

The default value is 0x00000000.



Command List Pointer Register This register holds a memory address that points to a list of accelerator commands to be executed. (Not implemented in this version of Platinum).

The default value is 0x000000000.



System Configuration Register This register is used to control the behavior of the Iridium datapath chip. There are a number of fields in this register:

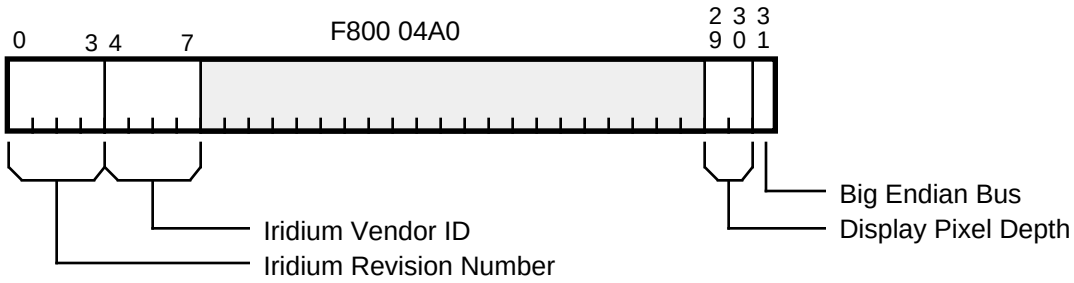
Iridium Revision Number: This field contains the revision level for Iridium. The first verison of Iridium will have a revision level of 0. These bits are read-only.

Iridium Vendor ID: This field contains the vendor ID for Iridium. The VTI verison of Iridium will have a vendor ID of 0 and the TI version will have a vendor ID of 1. These bits are read-only.

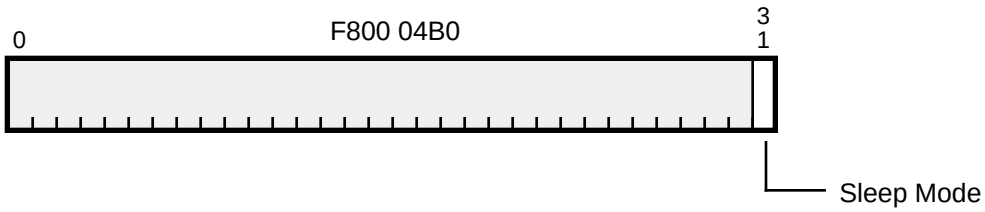
Big Endian Bus: This bit indicates the data orientation on the system data bus. If set (the default state), data and addresses on the system bus are assumed to be in big endian order. If cleared, data and addresses are assumed to be little endian. This bit is provided for compatibility with little endian operationg systems like Windows NT.

Display Pixel Depth: These bits encode the display's bit depth. 00b = 8 bpp, 01b = 16 bpp, and 10b = 32 bpp.

This register physically exists in Iridium only, but can be read and written in the Platinum register space for ease of use. The default value is 0x00000001 or 0x01000001.



Sleep Mode Register This single bit register is used to control the low power system sleep mode. When this bit is set, the system clocks run at a much lower rate than normal, significantly reducing the motherboard power requirements. The default value is 0x00000000.



4.7.5 Cache Diagnostic Access Areas

The address space from $F820 0000 to $F82F FFFF is allocated to diagnostic reads and writes of the cache data SRAMs. This is enough space for a 1 MB cache. Access to this space is allowed only if the Cache Enable bit is cleared and the Cache Test Enable bit is set in the Cache Configuration register. All accesses to this space must be single beat reads or writes; burst accesses will cause the machine to hang. Each entry in the cache is an 8-byte quantity and should be read or written as such. Cache data 0 is at location $F820 0000, data 1 is at $F820 0008, data 2 is at $F820 0010, and so on. If the cache is less than 1 MB in size, its image will be replicated enough times to fill the entire 1 MB space.

The address space from $F810 0000 to $F81F FFFF is allocated to diagnostics reads and writes of the cache tag SRAMs. This is enough space to accommodate the tag store for a 1 MB cache. Access to this space is allowed only if the Cache Enable bit is cleared and the Tag Test Enable bit is set in the Cache Configuration register. All accesses to this space must be 4-byte single beat reads or writes; burst accesses will cause the machine to hang. Since each tag corresponds to a 32-byte cache line, tag 0 is at location $F810 000, tag 1 is at $F810 0020, tag 2 is at $F810 0040, and so on. The tag bits are mapped to data bits 0–13 and the tag valid bit is mapped to data bit 14.

Appendix A: Revision History

Version 0.1 (5/25/94):
- — Initial release.

Version 0.2 (8/9/94):
- — Updated register map to reflect the current state of the ASIC.
- — Added tables for programming the DRAM and VRAM controllers, based on system bus speed and memory speed.

Version 0.3 (8/15/94):
- — Updated I/O pin list to reflect the current state of the ASIC.
- — Updated register map and descriptions to reflect the current state of the ASIC.
- — Corrected tables for programming the DRAM and VRAM controllers.

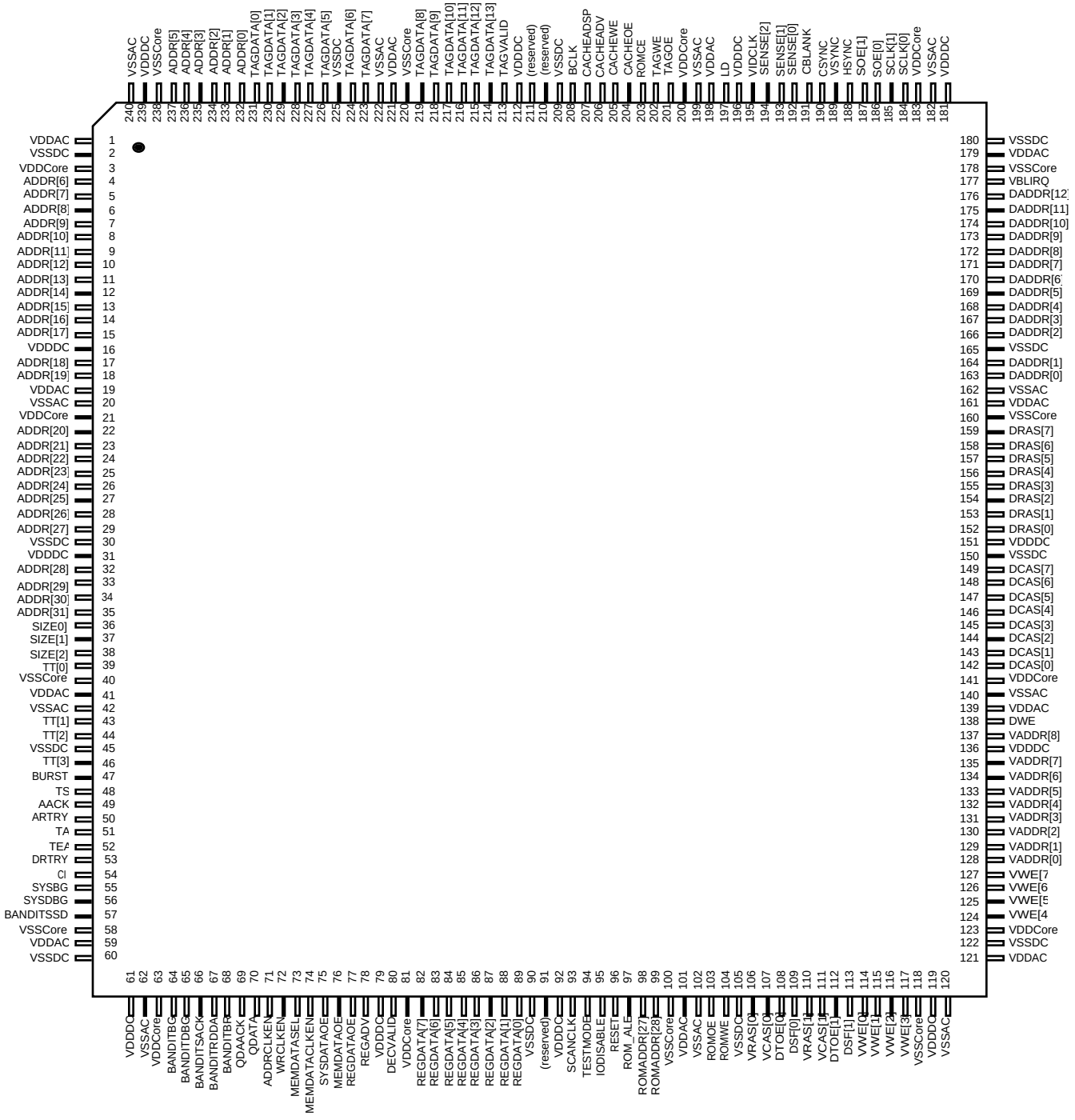
Version 0.4 (12/12/94):
- — Updated I/O pin list to reflect the current state of the ASIC.
- — Updated register map and descriptions to reflect the current state of the ASIC.
- — Fixed numerous errors in the Programmer's Guide (section 4.0).

Version 0.5 (3/15/94):
- — Updated I/O pin list to reflect the current state of the ASIC.
- — Updated register map and descriptions to reflect the current state of the ASIC.
- — Fixed numerous errors in the Programmer's Guide (section 4.0).

Version 0.9 (4/21/95):
- — Updated I/O pin list to reflect the current state of the ASIC.
- — Updated register map and descriptions to reflect the current state of the ASIC.
- — Fixed numerous errors in the Programmer's Guide (section 4.0).
- — Added much text in the Implementation section.

Version 1.0 (5/15/95):
- — Finished Implementation and Programmer's Model sections.
- — Added Monitor Sense Code table in Appendix C.

Appendix B: Pinout



Appendix C: Monitor Sense Line Code Assignments

The display controller in Platinum is capable of supporting all monitors made by Apple. Using the SENSE[2:0] lines and the following procedure, software can determine what type of monitor is attached to the computer:

1) Make sure the Sense Line Output Data bits in the Frame Buffer Test register ($F800 01A0) are zero.

2) Write 0x7 to the Sense Line Enable register ($F800 0170) and read back the same register. If the value returned is 0–5, the monitor type can be determined directly from Table C-1. If the value returned is 6 or 7, continue with the following steps.

3) For each sense line, write a zero bit to the Sense Line Enable register for that line and then read the state of the other two lines. For example, write 0x6 to the Sense Line Enable register (setting sense line 0 low), read back the same register, and save bits 26–27 of the result. These bits will be compared with the column in Table C-1 labeled (2,1).

4) Compare the resulting 6 bits of data with the entries in Table C-1 to determine which monitor is attached. If all reads return 1s, there is no monitor attached. Any code not listed in the table is currently reserved.

Table C-1 Extended Sense Codes for Apple Monitors

Monitor Type	Sense Lines			Extended Sense Codes		
	2	1	0	(1,0)	(2,0)	(2,1)
21” RGB	0	0	0	—	—	—
15” grayscale (portrait)	0	0	1	—	—	—
12” RGB	0	1	0	—	—	—
21” grayscale	0	1	1	—	—	—
NTSC	1	0	0	—	—	—
15” RGB (portrait)	1	0	1	—	—	—
Standard Mac II 12/13" grayscale/RGB	1	1	0	—	—	—
15” multi-sync	1	1	0	11	00	00
17” multi-sync	1	1	0	11	01	00
20” multi-sync	1	1	0	11	00	01
VESA	1	1	1	11	10	10
16” RGB	1	1	1	10	11	01
19” RGB	1	1	1	01	01	11
PAL	1	1	1	00	00	11
No monitor attached	1	1	1	11	11	11