Intel® 815 Chipset: Graphics Controller

Programmer's Reference Manual (PRM)

July 2000

Order Number: 298237-001

Contents

Figures

Tables

Revision History

Rev.	Description	Date
1.0	• Initial Release	July 2000

This page is intentionally left blank.

1. Introduction

The Intel® 815 chipset is a highly flexible chipset designed to extend from the basic graphics/multimedia PC platform up to the mainstream performance desktop platform. The chipset consists of an Intel® 82815 chipset Graphics and Memory Controller Hub (GMCH), an I/O Controller Hub (ICH) for the I/O subsystem, and a Firmware* Hub (FWH). For this chipset, the graphics capability resides in the Graphics and Memory Controller Hub (GMCH) chip.

The GMCH's Graphics Controller (GC) contains an extensive set of registers and instructions for configuration, 2D, 3D, and Video systems. This document describes the Intel® 815 chipset registers/instructions and provides detailed bit/field descriptions.

This Programmer's Reference Manual (PRM) is intended for hardware, software, and Firmware* designers who seek to implement or utilize the graphic functions of the Intel® 815 chipset. Familiarity with 2D and 3D graphics programming is assumed.

1.1. Terminology

Term	Description
AGP Mode	The GMCH is using its capability to interface with and AGP card. The internal graphics controller is disabled in this mode.
GPA Card	Graphics Performance Accelerator Card. This is a new implementation which allows local memory devices to be placed on a card that plugs into the AGP slot. When an AGP card is not present, an GPA card can be added to improve performance by acting as a display cache, for the Z-buffer only, of up to 4 MB. The GPA card was previously known as the AIMM (Add-In Memory Module).
CSI	Command Stream Interface (same as instruction stream interface)
GC	Graphics Controller
GFX Mode	The GMCH is using its internal graphics capability. This means that the ability to interface with an AGP card is disabled.
GMCH	The Graphics and Memory Controller Hub component that contains the functionality of an MCH plus an internal graphics controller.
Group 0 Protection (register)	As per the original IBM VGA specification, CRT Controller registers CR[0:7] can be write-protected via CR11[bit 7]. In BIOS code, this write protection is set following each mode change. Note that other group protection levels have no current use and are not supported by the GC. Only Group 0 Protection is supported.
Instruction	The GC has a set of graphics instructions. In some documents the term "command" is used for instruction.
IP	Instruction Parser
MBZ	Must Be Zero

Term	Description
MCH	The Memory Controller Hub component that contains the processor interface, DRAM controller, and AGP interface. The MCH communicates with the ICH over a proprietary interconnect called the hub interface (previously known as HubLink). The MCH was called the North Bridge (NB) in previous chip sets. MCH will be used to refer to non-graphics portion of the GMCH.
MM	Memory Mapped address space.
MMIO	Memory Mapped I/O space.
QW	Quad Word = 64 bits = 8 Bytes.
R/W (register)	Read/Write. A register with this attribute can be read and written.
R/WC (register)	Read/Write Clear. A register bit with this attribute can be read and written. However, a write of 1 clears (sets to 0) the corresponding bit and a write of 0 has no effect.
R/WO (register)	Read/Write Once. A register bit with this attribute can be written to only once after power up. After the first write, the bit becomes read only.
RO (register)	Read Only. In some cases, if a register is read only, writes to this register location have no effect. However, in other cases, two separate registers are located at the same location where a read accesses one of the registers and a write accesses the other register. See the I/O and memory map tables for details.
WO (register)	Write Only. In some cases, If a register is write only, reads to this register location have no effect. However, in other cases, two separate registers are located at the same location where a read accesses one of the registers and a write accesses the other register. See the I/O and memory map tables for details.

1.2. Reference Documents

The following documents should be available for reference when using this specification:

- Intel® 815 Chipset: Intel® 82815 chipset Graphics and Memory Controller Hub (GMCH) External Design Specification (EDS)
- Intel® 82801AA (ICH) and Intel® 82801AB (ICH0) I/O Controller Hub Datasheet
- Intel® 82801BA (ICH2) I/O Controller Hub External Design Specification (EDS)
- Intel® 82802AB/82802AC Firmware* Hub (FWH) Datasheet

2. Intel® 815 Chipset Overview

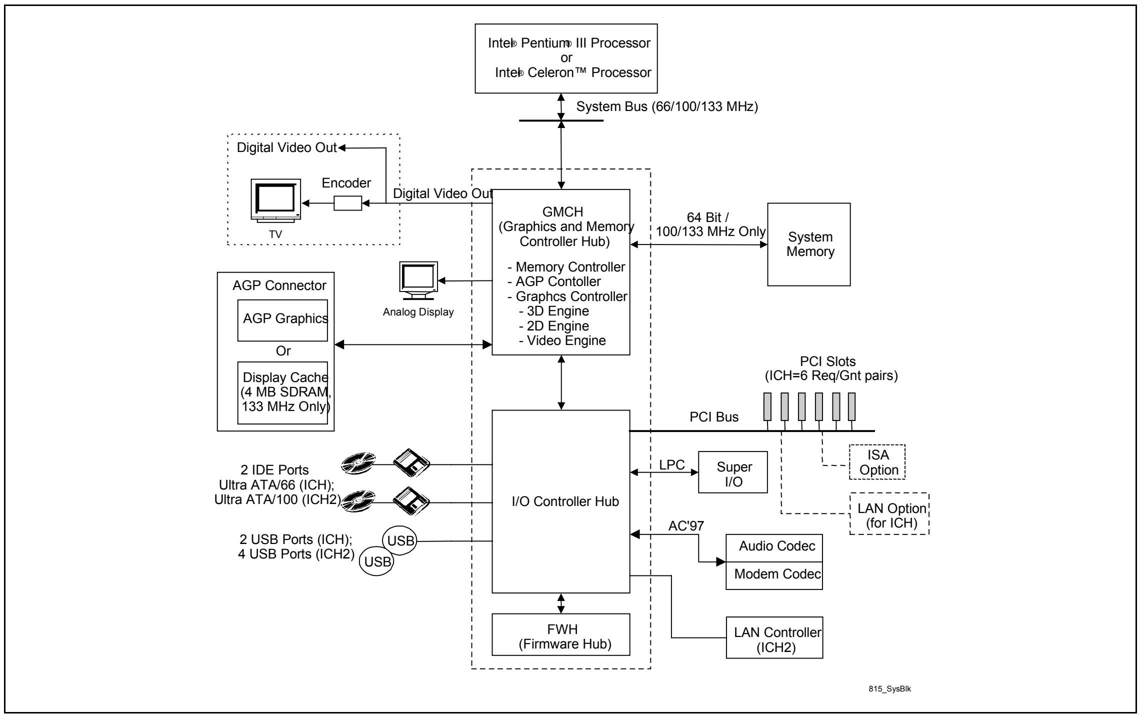
The chipset consists of an Intel® 82815 chipset Graphics and Memory Controller Hub (GMCH), an I/O Controller Hub (ICH) for the I/O subsystem, and a Firmware* Hub (FWH). The GMCH integrates a system memory SDRAM controller that supports a 64-bit 100/133 MHz SDRAM array. The Intel® 815 chipset family includes:

- **Intel® 815 chipset:** This chipset consists of the Intel® 82815 chipset GMCH, the 82801AA ICH, and the 82802AB/82802AC FWH.
- **Intel® 815E chipset:** This chipset consists of the Intel® 82815 chipset GMCH, the 82801BA ICH2, and the 82802AB/82802AC FWH.

The Intel® 82815 chipset GMCH integrates a Display Cache SDRAM controller that supports a 32-bit 133 MHz SDRAM array for enhanced integrated 2D and 3D graphics performance. Multiplexed with the display cache interface is an AGP controller interface to enable graphics configuration and upgrade flexibility with the Intel® 815 chipset. The AGP interface and the internal graphics device are mutually exclusive. When the AGP port is populated with an AGP graphics card, the integrated graphics is disabled; thus, the display cache interface is not needed.

The Intel® 815 chipset family uses a hub architecture with the Intel® 82815 chipset GMCH as the host bridge hub and the I/O Controller Hub (ICH) as the I/O hub. The Intel® 815 chipset supports the 82801AA ICH and the Intel® 815E chipset supports the 82801BA ICH2. The ICH is a highly integrated multifunctional I/O Controller Hub that provides the interface to the PCI Bus and integrates many of the functions needed in today’s PC platforms. The GMCH and ICH communicate over a dedicated hub interface.

Figure 1. Intel® 815 Chipset System Block Diagram



2.1. I/O Controller Hub

The 82801AA ICH/82801BA ICH2 functions and capabilites are listed below. Unless otherwise specified, the function/capability applies to both ICH and ICH2.

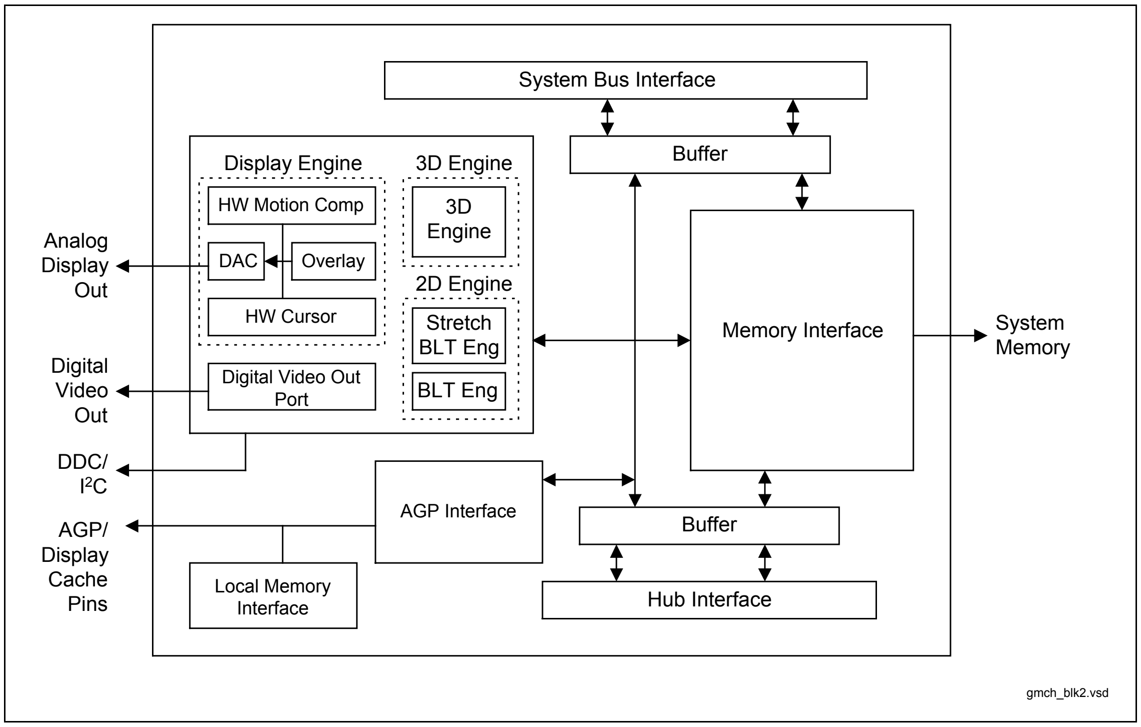
- PCI Rev 2.2 compliant with support for 33 MHz PCI operations
- ICH supports up to 6 Req/Gnt pairs
- Power Management Logic Support
- Enhanced DMA Controller, Interrupt Controller & Timer Functions
- Integrated IDE controller; ICH supports Ultra ATA/66 (ICH); Ultra ATA/100/66/33 (ICH2)
- Integrated LAN Controller (ICH2 only)
- USB host interface with support for two USB ports (ICH); Four ports (ICH2)
- System Management Bus (SMBus) compatible with most I^2C devices
- AC'97 2.1 Compliant Link for Audio and Telephony CODECs
- Low Pin Count (LPC) interface
- Firmware* Hub (FWH) interface support
- Alert On LAN*

2.2. Intel® 82815 Chipset GMCH Overview

Figure 2 is a block diagram of the GMCH illustrating the various interfaces and integrated functions. The functions and capabilities include:

- Support for a single processor configuration
- 64-bit AGTL+ based System Bus Interface at 66/100/133 MHz
- 32-bit Host Address Support
- 64-bit System Memory Interface with optimized support for SDRAM at 100/133 MHz
- Integrated 2D & 3D Graphics Engines
- Integrated H/W Motion Compensation Engine
- Integrated 230 MHz DAC
- Integrated Digital Video Out Port
- 133 MHz Display Cache
- AGP 1X/2X/4X Controller

Figure 2. Intel® 82815 Chipset GMCH Block Diagram



2.2.1. Host Interface

The host interface of the GMCH is optimized to support the Intel® Pentium® III processor and Intel® Celeron™ processor in the FC-PGA package. The GMCH implements the host address, control, and data bus interfaces within a single device. The GMCH supports a 4-deep in-order queue (i.e., supports pipelining of up to 4 outstanding transaction requests on the host bus). Host bus addresses are decoded by the GMCH for accesses to system memory, PCI memory and PCI I/O (via hub interface), PCI configuration space and Graphics memory. The GMCH takes advantage of the pipelined addressing capability of the processor to improve the overall system performance.

The Intel® 82815 chipset GMCH supports the 370-pin socket processor.

- **370-pin socket** (PGA370). The PGA370 is a zero insertion force (ZIF) socket that a processor in the FC-PGA package will use to interface with a system board.

2.2.2. System Memory Interface

The GMCH integrates a system memory controller that supports a 64-bit 100/133 MHz SDRAM array. The only DRAM type supported is industry standard Synchronous DRAM (SDRAM). The SDRAM controller interface is fully configurable through a set of control registers.

The GMCH supports industry standard 64-bit wide DIMMs with SDRAM devices. The thirteen multiplexed address lines, SMAA[12:0], along with the two bank select lines, SBS[1:0], allow the GMCH to support 2M, 4M, 8M, 16M, and 32M x64 DIMMs. Only asymmetric addressing is supported. The GMCH has six SCS# lines (two copies of each for electrical loading), enabling the support of up to six 64-bit rows of SDRAM. The GMCH targets SDRAM with CL2 and CL3 and supports both single and double-sided DIMMs. Additionally, the GMCH also provides a 1024 entry deep refresh queue. The GMCH can be configured to keep up to 4 pages open within the memory array. Pages can be kept open in any one bank of memory.

SCKE[5:0] are used in configurations requiring powerdown mode for the SDRAM.

2.2.3. Multiplexed AGP and Display Cache Interface

The Intel® 82815 chipset GMCH multiplexes an AGP interface with a display cache interface for internal 3D graphics performance improvement. The display cache is used only in the internal graphics. When an AGP card is installed in the system, the GMCH internal graphics will be disabled and the AGP controller will be enabled.

AGP Interface

A single AGP connector is supported by the GMCH AGP interface. The AGP buffers operate in one of two selectable modes in order to support the AGP Universal Connector:

- 3.3V drive, **not** 5 volt safe: This mode is compliant to the AGP 1.0 and 2.0 specifications.
- 1.5V drive, **not** 3.3 volt safe: This mode is compliant with the AGP 2.0 specification.

The following table shows the AGP Data Rate and the Signaling Levels supported by the GMCH.

Data Rate	Signaling Level	
	1.5V	3.3V
1x AGP	Yes	Yes
2x AGP	Yes	Yes
4x AGP	Yes	No

The AGP interface supports 4x AGP signaling. AGP semantic (PIPE# or SBA[7:0]) cycles to SDRAM are not snooped on the host bus. AGP FRAME# cycles to SDRAM are snooped on the host bus. The GMCH supports PIPE# or SBA[7:0] AGP address mechanisms, but not both simultaneously. Either the PIPE# or the SBA[7:0] mechanism must be selected during system initialization. High priority accesses are supported. Only memory writes from the hub interface to AGP are allowed. No transactions from AGP to the hub interface are allowed.

Display Cache Interface

The GMCH supports a Display Cache SDRAM controller with a 32-bit 133 MHz SDRAM array. The DRAM type supported is industry standard Synchronous DRAM (SDRAM) like that of the system memory. The local memory SDRAM controller interface is fully configurable through a set of control registers.

2.2.4. Hub Interface

The hub interface is a private interconnect between the GMCH and the ICH.

2.2.5. Intel® 82815 Chipset GMCH Integrated Graphics Support

The GMCH includes a highly integrated graphics accelerator. Its architecture consists of dedicated multi-media engines executing in parallel to deliver high performance 3D, 2D and motion compensation video capabilities. The 3D and 2D engines are managed by a 3D/2D pipeline preprocessor allowing a sustained flow of graphics data to be rendered and displayed. The deeply pipelined 3D accelerator engine provides 3D graphics quality and performance via per-pixel 3D rendering and parallel data paths which allow each pipeline stage to simultaneously operate on different primitives or portions of the same primitive. The GMCH graphics accelerator engine supports perspective-correct texture mapping, trilinear and anisotropic Mip-Map filtering, Gouraud shading, alpha-blending, fogging and Z-buffering. A rich set of 3D instructions permit these features to be independently enabled or disabled.

For the GMCH, a Display Cache (DC) can be used for the Z-buffer (textures and display buffer(s) are located only in system memory). If the display cache is not used, the Z-buffer is located in system memory.

The GMCH integrated graphics accelerator's 2D capabilities include BLT and arithmetic STRBLT engines, a hardware cursor and an extensive set of 2D registers and instructions. The high performance 64-bit BitBLT engine provides hardware acceleration for many common Windows* operations.

In addition to its 2D/3D capabilities, the GMCH integrated graphics accelerator also supports full MPEG-2 motion compensation for software-assisted DVD video playback, a VESA DDC2B compliant display interface and a digital video out port which may support (via an external video encoder) NTSC and PAL broadcast standards and (via an external TMDS transmitter) digital Flat Panel or Digital CRT displays.

Display, Digital Video Out, and LCD/Flat Panel/Digital CRT

The GMCH provides interfaces to a standard progressive scan monitor, TV-Out device, and TMDS transmitter. These interfaces are only active when running in internal graphics mode.

- The GMCH directly drives a standard progressive scan monitor up to a resolution of 1600x1200 pixels.
- The GMCH provides a Digital Video Out interface to connect an external device to drive a 1280x1024 resolution non-scalar DDP digital Flat Panel with appropriate EDID 1.2 data or digital CRTs. The interface has 1.8V signaling to allow it to operate at higher frequencies. This interface can also connect to a 1.8V TV-Out encoder.

2.2.6. System Clocking

The Intel® 82815 chipset GMCH has a new type of clocking architecture. It has integrated SDRAM buffers that run at either 100 or 133 MHz, independent of the system bus frequency. See table below for supported system bus and system memory bus frequencies. The system bus frequency is selectable between 66 MHz, 100 MHz or 133 MHz. The GMCH uses a copy of the USB clock as the DOT Clock input for the graphics pixel clock PLL.

Table 1. Supported System Bus and System Memory Bus Frequencies

Front Side Bus Frequency	System Memory Bus Frequency	Display Cache Interface Frequency
66 MHz	100 MHz	133 MHz or DVMT
100 MHz	100 MHz	133 MHz or DVMT
133 MHz	100 MHz	133 MHz or DVMT
133 MHz	133 MHz	133 MHz or DVMT

2.2.7. GMCH Power Delivery

The Intel® 82815 chipset GMCH core voltage is 1.85V. System Memory runs off of a 3.3V supply. Display cache memory runs off of the AGP 3.3V supply. AGP 1X/2X I/O can run off of either a 3.3V or a 1.5V supply. AGP 4X I/O require a 1.5V supply. The AGP interface voltage is determined by the VDDQ generation on the motherboard.

2.3. Three PCI Devices on GMCH

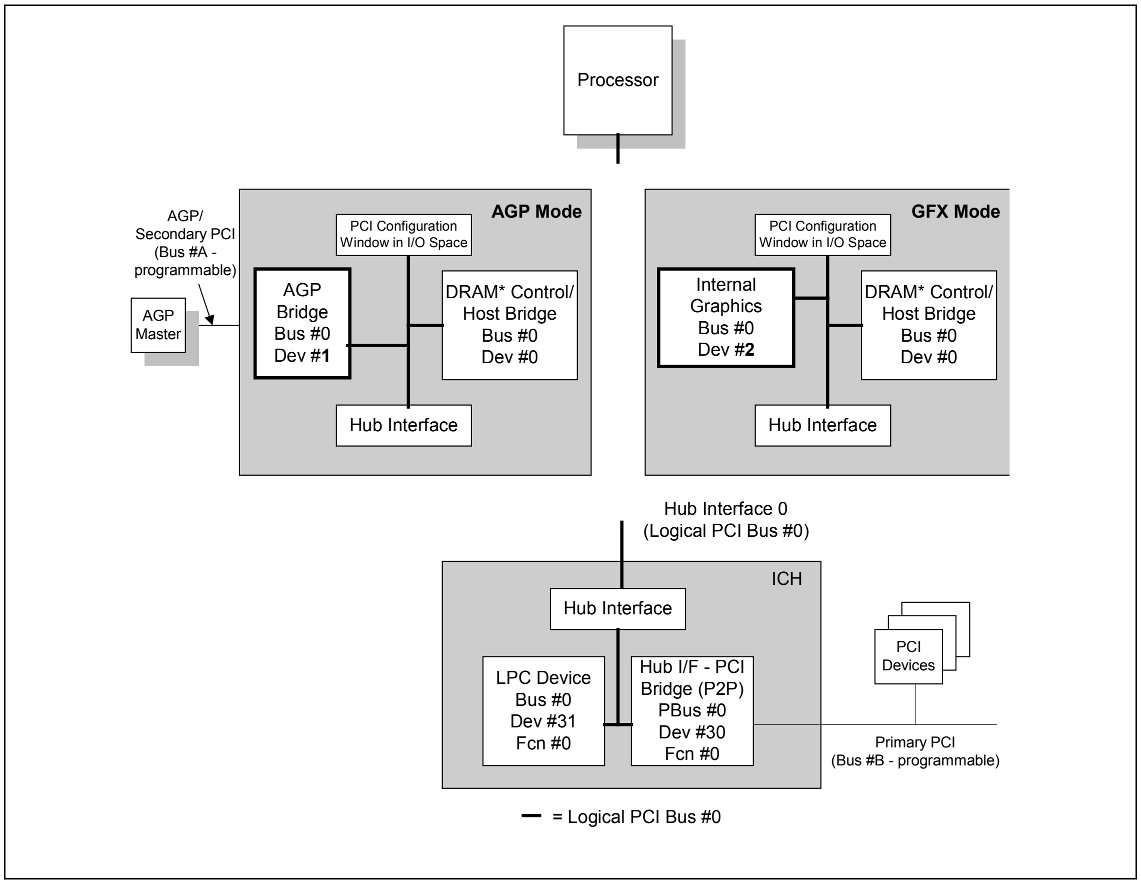
The Intel® 82815 chipset GMCH contains three PCI Devices. The management of active devices is controlled via bit 0 of the APCONT register (See the *Intel® 815 Chipset: 82815 Graphics and Memory Controller (GMCH) EDS* for details on PCI configuration registers).

- **Device 0** = Host Bridge = PCI bus #0 interface, Main Memory Controller, Graphics Aperture controller
- **Device 1** = AGP Bridge = AGP 2X/4X interface (AGP Mode)
- **Device 2** = Internal Graphics Device (GFX Mode)

Note: Devices 1 and 2 are mutually exclusive. Only one of these two devices can be active at any given time. Device selection is performed during the start-up sequence and can only be set once. The lock bit must be set after device selection.

The following diagram shows more detail at a platform level. The GMCH is shown in both AGP Mode (left side) and GFX Mode (right side). Only one mode can be active at any given time. The processor and ICH functions remain unchanged by the GMCH mode of operation.

Figure 3. Conceptual Platform PCI Configuration Diagram


Processor
AGP Mode
GFX Mode
AGP/ Secondary PCI (Bus #A - programmable)
AGP Master
PCI Configuration Window in I/O Space
AGP Bridge Bus #0 Dev #1
DRAM* Control/ Host Bridge Bus #0 Dev #0
Hub Interface
PCI Configuration Window in I/O Space
Internal Graphics Bus #0 Dev #2
DRAM* Control/ Host Bridge Bus #0 Dev #0
Hub Interface
Hub Interface 0 (Logical PCI Bus #0)
ICH
Hub Interface
LPC Device Bus #0 Dev #31 Fcn #0
Hub I/F - PCI Bridge (P2P) PBus #0 Dev #30 Fcn #0
PCI Devices
Primary PCI (Bus #B - programmable)
= Logical PCI Bus #0

2.3.1. Multi-Mode Capability Requirements

There are multiple **Device Modes** that are supported by the three PCI devices within this component:

1. **AGP Mode** – The AGP slot is populated with an AGP graphics card and the AGP interface device is active. The internal graphics controller is inactive.
2. **GFX Mode with Local Memory** – The internal graphics device is active. The AGP slot is populated with a GPA card and the AGP interface device is inactive.
3. **GFX Mode with Shared Memory** – The internal graphics device is active. The AGP slot is not populated with a GPA card or an AGP graphics card and the AGP interface device is inactive.
4. **PCI Mode** – The internal graphics controller and the AGP interface device are both inactive. The display data cycles are routed through the hub interface to the PCI display device.

2.3.1.1. Supported Single Monitor and Multi-monitor Configurations

For modern operating systems that have multiple monitor support, the primary graphics device must not be the Intel® 815 chipset internal graphics controller or an AGP graphics card. These graphics devices can serve only as secondary graphics devices in a multi-monitor configuration. It is important to understand that there is no support for simultaneous operation of the internal graphics device and an AGP graphics card.

To support a PCI graphics device, the Intel® 815 chipset simply passes all of that device's cycles to the hub interface as it would for any other PCI device.

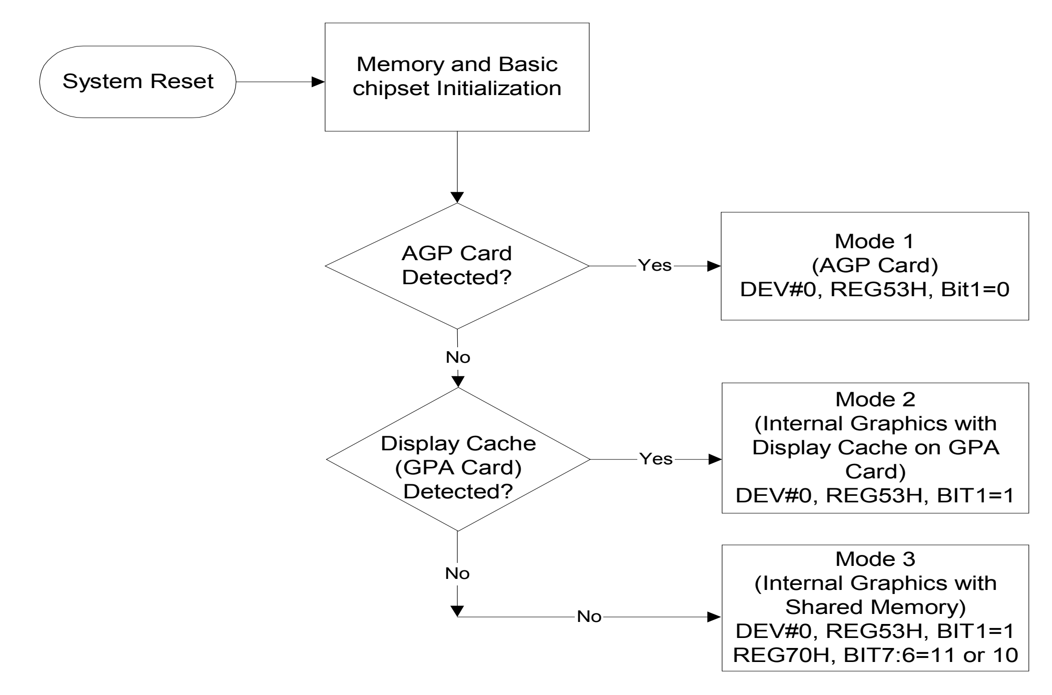
Configuration	**Single Monitor**			**Multi-monitor**		
Primary Graphics Device	Internal Graphics (with LM or UMA)	AGP Graphics Card	Non-AGP Graphics Card (PCI)	Non-AGP Graphics Card (PCI)	Non-AGP Graphics Card (PCI)	Non-AGP Graphics Card (PCI)
Secondary Graphics Device	None	None	None	Internal Graphics (with LM or UMA)	AGP Graphics Card	Non-AGP Graphics Card (PCI)

2.3.1.2. System Startup

The Intel® 815 chipset has multiple possible device modes. The selection of which mode will be auto-detected is represented in the following flow chart. The software requirements for implementing this high level flow are detailed in the next section.

Multi-monitor configurations are not addressed, as that is a function of the operating system when supported. System BIOS also provides the ability to select any of the display devices as the primary display device.

Figure 4. Device Mode Auto-Detect Flowchart


System Reset
Memory and Basic chipset Initialization
AGP Card Detected?
Yes
Mode 1 (AGP Card) DEV#0, REG53H, Bit1=0
No
Display Cache (GPA Card) Detected?
Yes
Mode 2 (Internal Graphics with Display Cache on GPA Card) DEV#0, REG53H, BIT1=1
No
No
Mode 3 (Internal Graphics with Shared Memory) DEV#0, REG53H, BIT1=1 REG70H, BIT7:6=11 or 10

2.3.1.3. Software Start-Up Sequence

The following sequence of events will occur during the initialization of an Intel® 815 chipset-based system:

1. The ICH asserts PCIRST# either in response to an initial assertion of PWROK or a write to an I/O Port.
2. System BIOS runs basic POST code to test the processor.
3. System BIOS initializes the minimum set of Intel® 815 chipset configuration registers required to initiate a PCI configuration cycle towards the AGP/PCI interface (e.g., bus #1). Note that following a system reset condition, the Intel® 815 chipset will always be in "AGP Mode" (Internal Graphics disabled).
4. System BIOS detects if an AGP graphics card is present by attempting a configuration read cycle to the Intel® 815 chipset AGP/PCI interface. If the configuration cycle completes successfully, then it is assumed that an external graphics device is present on the AGP interface, and the Intel® 815 chipset remains in "AGP Mode" (APCONT[0] = 0). If the AGP/PCI configuration cycle results in a Master Abort, then it is assumed no external AGP graphics device is present, and the System BIOS programs the Intel® 815 chipset to "Internal Graphics Mode" by setting APCONT[0] = 1. During the same configuration cycle as setting the APCONT[0] to either 0 or 1, the BIOS should set APCONT[2] (GFX AGP Select Lock) = 1 to lock the configuration mode.
5. System BIOS will then initialize the minimum set of the Intel® 815 chipset configuration registers required to detect and test system memory.
6. System BIOS interrogates each System Memory DIMM via the Serial Presence Detect (SPD) mechanism (ICH I2C cycles).
7. System BIOS determines if system configuration is capable of 133 MHz operation. Requirements for 133 MHz operation include: the Intel® 815 chipset in AGP Mode, and System Memory populated with up to two double-sided, or three single-sided, PC133-compliant DIMMs.
8. If system is 133 MHz capable (as determined by the previous step), System BIOS "upshifts" the System Memory operating frequency from 100 MHz to 133 MHz as follows (Note that the Intel® 815 chipset typically resets internally to 100 MHz SM operation):
   a) Program external clock generator chip (i.e., "CK815") for 133 MHz System Memory clocking via ICH I2C port (Byte 3, Bit 0 set to 1).
   b) Wait for 1 µs to allow the Clock Generator to stabilize the external System Memory clocks (i.e., SCLK). Note that the Clock Generator must guarantee a "clean" Host Clock (HCLK) during this entire transition!
   c) Program Intel® 815 chipset System Memory Frequency Select bit (GMCHCFG[2]) to 1 to enable internal System Memory clocking to operate at 133 MHz.
   d) Wait at least 0.5 µs for the Intel® 815 chipset to re-synchronize internal clocks before accessing anything other than Intel® 815 chipset Configuration registers or the Hub interface/ICH interface.
9. System BIOS then detects and configures all of system memory, and tests enough memory to support a stack and an interrupt area.
10. System BIOS should shadow itself and complete system initialization.
11. If and only if the Intel® 815 chipset is in Internal Graphics Mode (APCONT[0] = 1), pass control to the system BIOS internal graphics mode initialization routines described below.
12. System BIOS programs the Intel® 815 chipset base addresses using PCI configuration cycles as described in the PCI Local Bus Specification.
13. PCI enumeration takes place at this point.

14. During PCI enumeration, the system BIOS will identify and initialize the primary display device. The selection of the primary display device is typically OEM dependent. An OEM may use a BIOS setup question to allow the system user to select the primary display device. Intel recommends the system BIOS to give preference to the higher performance display device when performing an "auto-selection" of the primary display device. If the GMCH is in AGP mode, this algorithm gives preference to the AGP graphics device over a PCI VGA device. If the Intel® 815 chipset is in GFX mode, this algorithm gives preference to the internal graphics controller over a PCI VGA device.
15. Once the primary display device has been initialized, system BIOS will pass control to the Video Option ROM corresponding to that display device.
16. If and only if the GMCH is in GFX Mode (APCONT[0] = 1), the Video Option ROM will perform initialization routines described below.
17. System BIOS then completes the system test including testing the rest of main memory.

The following System BIOS initialization steps only apply if the GMCH is in "Internal Graphics Mode" (Not AGP):

1. System BIOS determines if a GPA card (i.e., Local Memory) is present. This is accomplished the same as on the Intel® 810 chipset: Enable Local Memory via the DRT, DRAMCL, and DRAMCH (offsets 3000-3002h) and attempt to write and readback a location in Local Memory. If the readback returns the same data that was written previously, then it can be assumed that Local Memory is present. If a GPA card is present, the following additional steps take place:
   - System BIOS configures the Local Memory Timing Options via the DRAMCL register at Device 2 Memory Mapped Register offset 3001h. There is no Serial Presence Detect mechanism available for the Local Memory / GPA Card interface, so the method used to determine these timing options is entirely up to the OEM. One conservative option is to leave the DRAMCL register with its default settings, which are the slowest available. Another option is to enforce certain minimum timings on any GPA cards used by that OEM, and program the DRAMCL with those minimums. While not recommended, it is also possible to determine optimal timings empirically.

The following Video BIOS initialization steps only apply if the GMCH is in "Internal Graphics Mode" (Not AGP):

1. Video BIOS assumes a VGA monitor type, and initializes the 2D-display controller in text mode.
2. 2D Video BIOS should take care to keep track of BIOS changes due to chip version changes.
3. The 3D section should be software-reset and initialized. At this point, no 3D operation should be enabled, since this is done at the application/driver level.

2.3.1.3.1.Graphics Driver Startup

At this point the GMCH internal graphics controller has been configured and initialized. The remainder of the graphics initialization occurs when the operating system calls the graphics driver. The graphics driver must initialize the graphics component address re-mapping hardware. Tasks include:

1. Allocate memory for the GTT re-mapping table. Set the host memory type of graphics memory for both physically local and system memory (i.e. enable Write Combining for local and non-local memory: OS calls graphics driver which calls OS services for setting host memory type).
2. Establish policy for limiting the amount of non-local video memory. The OS determines the maximum amount of system memory that can be allocated as non-local video memory. The graphics driver can make a guess on what is likely to be provided by the OS based on the same information that the OS uses to make its decision. The graphics driver can query the OS to check the type of memory surface allocation (i.e., system vs. local vs. non-local video memory.)

3. After the conclusion of the graphics driver startup code the internal graphics functions will be ready for run-time activity and commands can be written into the ring buffer.

2.3.1.4. Switching Device modes

Under normal conditions, the GMCH Device Mode will be switched at most once: from AGP to Internal Graphics mode (via APCONT[0]) when no external AGP graphics device is present. This is handled automatically by the system BIOS during system initialization. There is no case where the GMCH will be switched from Internal Graphics mode to AGP mode.

3. System Address Map

This chapter provides address maps of the graphics controller (GC) I/O and memory-mapped registers. Individual register bit field descriptions are provided in the following chapters. Note that PCI configuration register descriptions are not covered in this document. For details on PCI configuration registers, refer to the *Intel® 815 Chipset: 82815 Graphics and Memory Controller (GMCH) EDS*.

Figure 5 shows a high-level representation of the system memory address map. Figure 6 provides additional details on mapping specific memory regions as defined and supported by the GMCH chipset.

Figure 5. System Memory Address Map

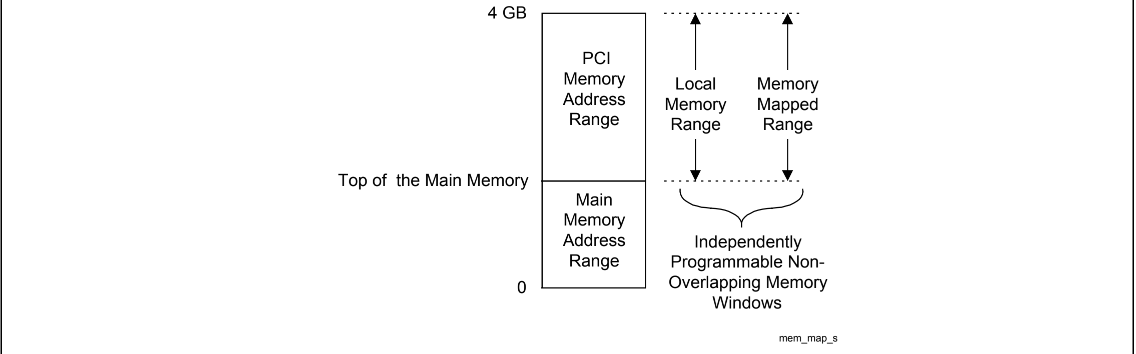
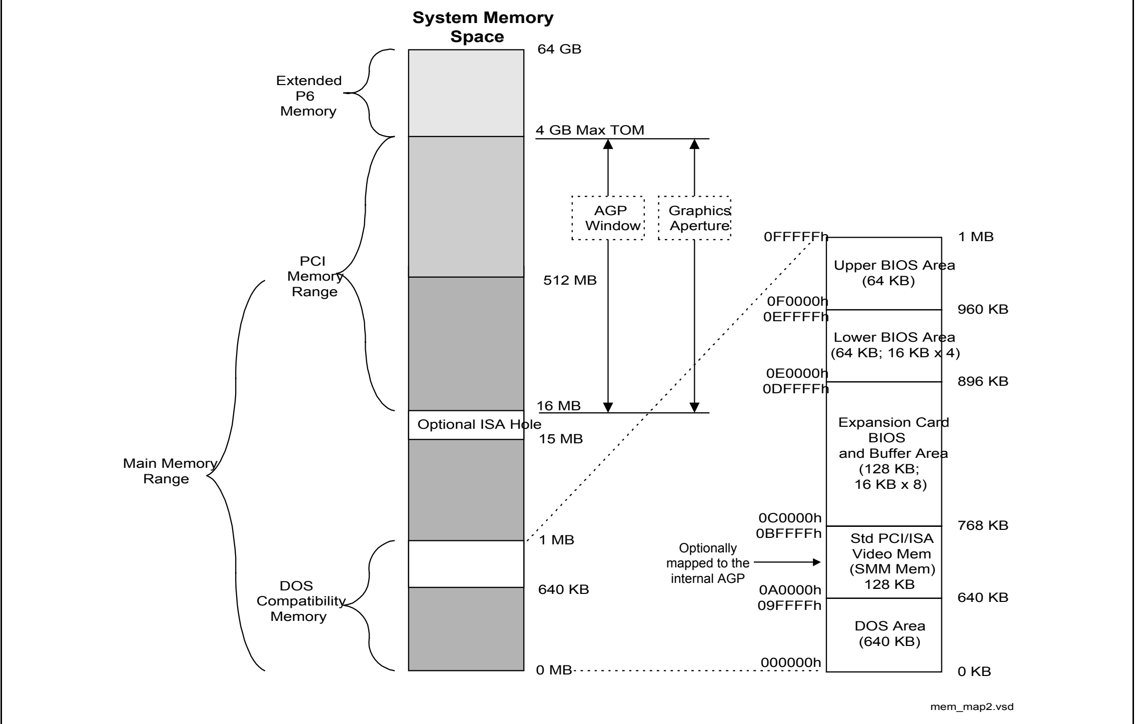


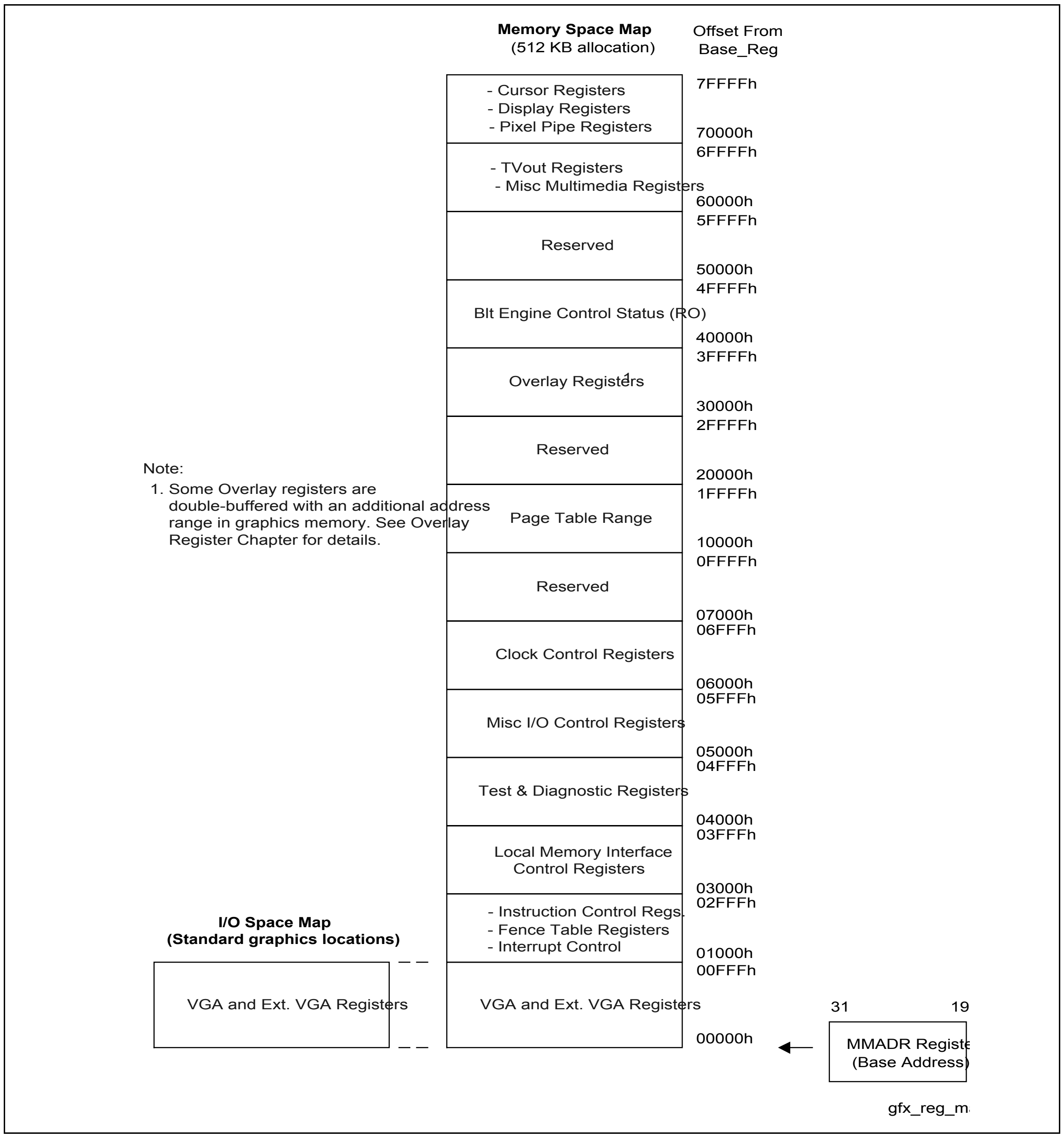
Figure 6. Detailed Memory System Address Map



3.1. Memory and I/O Space Registers

This section provides a high-level register map (register groupings per function). The memory and I/O maps for the GC registers are shown in the following figure. The VGA and Extended VGA registers can be accessed via standard VGA I/O locations as well as via memory-mapped locations. In addition, the memory map contains allocation ranges for various functions. The memory space address listed for each register is an offset from the base memory address programmed into the MMADR register (PCI configuration offset 14h).

Figure 7. Graphics Controller Register Memory and I/O Map



- **VGA and Extended VGA Control Registers (00000h–00FFFh).** These registers are located in both I/O space and memory space. The VGA and Extended VGA registers contain the following register sets: General Control/Status, Sequencer (SRxx), Graphics Controller (GRxx), Attribute Controller (ARxx), VGA Color Palette, and CRT Controller (CRxx) registers. Detailed bit descriptions are provided in the VGA and Extended VGA Register Chapter. The registers within a set are accessed using an indirect addressing mechanism as described at the beginning of each section. Note that some of the register description sections have additional operational information at the beginning of the section.
- **Instruction, Memory, Interrupt Control, and Error Registers (01000h–02FFFh).** The Instruction and Interrupt Control registers are located in main memory space and contain the following types of registers:
  - Instruction Control Registers. Ring Buffer registers and page table control registers are located in this address range. Various instruction status, error, and operating registers are located in this group of registers.
  - Graphics Memory Fence Registers. The Graphics Memory Fence registers are used for memory tiling capabilities.
  - Interrupt Control/Status Registers. This register set provides interrupt control/status for various GC functions.
  - Display Interface Control Register. This register controls the FIFO watermark and provides burst length control.
- **Local Memory Registers (03000h–03FFFh).** These registers are located in main memory space and provide local memory DRAM control.
- **Reserved (04000h–04FFFh).**
- **Miscellaneous I/O Control Registers (05000h–05FFFh).** This chapter provides miscellaneous I/O control register functions.
- **Clock Control Registers (06000h–06FFFh).** This memory address space is the location of the GC clock control and power management registers.
- **Reserved (07000h–0FFFFh).**
- **Page Table Range (10000h–1FFFFh).**
- **Reserved (20000h–2FFFFh).**
- **Overlay Registers (30000h–3FFFFh).** These registers provide control of the GC overlay engine. The overlay registers are double-buffered with one register buffer located in graphics memory and the other on the GC chip. On-chip registers are not directly writeable. To update the on-chip registers software writes to the register buffer area in graphics memory and instructs the GC to update the on-chip registers.
- **Blitter Status Registers (40000h–4FFFFh).** For debug purposes only, a set of read-only registers provide visibility into the BLT engine status.
- **Reserved (50000h–5FFFFh). (Reserved in the Intel® 815 chipset**).
- **LCD/TV-Out Registers (60000h–6FFFFh).** This memory address range is used for LCD/TV-Out control registers.
- **Cursor, Display, and Pixel Pipe Registers (70000h–7FFFFh).** This memory address range is used for cursor control, display, and pixel pipe control registers.

3.2. GC Register Memory Address Map

All GC registers are memory-mapped. In addition, the VGA and Extended VGA registers are I/O mapped.

Table 2. Memory-Mapped Registers

Address Offset	Symbol	Register Name	Access
00000h–00FFFh	—	**VGA and VGA Extended Registers** These registers are both memory and I/O mapped and are listed in the following table. Note that the I/O address and memory offset address are the same value for each register.	—
Instruction and Interrupt Control Registers (01000h–02FFFh)			
01000h–01FFFh	—	Reserved. Do not write	—
02000h–0201Fh	FENCE[0:7]	Graphics Memory Fence Table Register [0:7]	R/W
02020h–02023h	PGTBL_CTL	Page Table Control Register	R/W
02024h–02027h	PGTBL_ER	Page Table Error Register	RO
02028h–0202Bh	PGTBL_ERRMSK	Page Table Error Mask Register	R/W
02030h–0207Fh 02030h–0203Fh 02040h–0204Fh 02050h–0207Fh	RINGBUF	Ring Buffer Registers Low Priority Ring Buffer (4 DWs) Interrupt Ring Buffer (4 DWs) Reserved	R/W
02080h–02083h	HWS_PGA	Hardware Status Page Address Register	R/W
02084h–02087h	—	Reserved	—
02088h–0208Bh	IPEIR	Instruction Parser Error Identification Register	RO
0208Ch–0208Fh	IPEHR	Instruction Parser Error Header Register	RO
02090h–02091h	INSTDONE	Instruction Stream Interface Done Register	RO
02092h–02093h	—	Reserved	—
02094h–02097h	NOPID	NOP Identification Register	RO
02098h–002099h	HWSTAM	Hardware Status Mask Register	R/W
0209Ah–0209Fh	—	Reserved	—
020A0h–020A1h	IER	Interrupt Enable Register	R/W
020A2h–020A3h	—	Reserved	—
020A4h–020A5h	IIR	Interrupt Identity Register	R/WC
020A6h–020A7h	—	Reserved	—
020A8h–020A9h	IMR	Interrupt Mask Register	R/W
020AAh–020ABh	—	Reserved	—
020ACh–020ADh	ISR	Interrupt Status Register	RO
020AEh–020AFh	—	Reserved	—
020B0h–020B1h	EIR	Error Identity Register	R/WC
020B2h–020B3h	—	Reserved	—
020B4h–020B5h	EMR	Error Mask Register	R/W
020B6h–020B7h	—	Reserved	—
020B8h–020B9h	ESR	Error Status Register	RO
020BAh–020BFh	—	Reserved	—

Table 2. Memory-Mapped Registers

Address Offset	Symbol	Register Name	Access
020C0h	INSTPM	Instruction Parser Mode Register	R/W
020C1h–020C3h	—	Reserved	—
020C4h–020C7h	INSTPS	Instruction Parser State Register	RO
020C8h–020CBh	BBP_PTR	Batch Buffer Parser Pointer Register	RO
020CCh–020CFh	ABB_SRT	Active Batch Buffer Start Address Register	RO
020D0h–020D3h	ABB_END	Active Batch Buffer End Address Register	RO
020D4h–020D7h	DMA_FADD	DMA Engine Fetch Address Register	RO
020D8h–020DBh	FW_BLC	FIFO Watermark and Burst Length Control	R/W
020DCh–020DBh	—	Reserved	—
020DCh–020DFh	MEM_MODE	Memory Interface Mode Register	R/W
020E0h–02FFFh	—	Reserved	—
Memory Control Registers (03000h–03FFFh)			
03000h	DRT	DRAM Row Type	R/W
03001h	DRAMCL	DRAM Control Low	R/W
03002h	DRAMCH	DRAM Control High	R/W
03003h–03FFFh	—	Reserved	—
Reserved (04000h–04FFFh)			
04000h–04FFFh	—	Reserved	—
I/O Control Registers (05000h–05FFFh)			
05000h–05003h	HVSYNC	HSYNC/VSYNC Control	R/W
05010h–05013h	GPIOA	General Purpose I/O Control A	R/W
05014h–05017h	GPIOB	General Purpose I/O Control B	R/W
05018h–05FFFh	—	Reserved	—
Clock Control and Power Management Registers (06000h–06FFFh)			
06000h–06003h	DCLK_0D	Display Clock 0 Divisor	R/W
06004h–06007h	DCLK_1D	Display Clock 1 Divisor	R/W
06008h–0600Bh	DCLK_2D	Display Clock 2 Divisor	R/W
0600Ch–0600Fh	LCD_CLKD	LCD Clock Divisor	R/W
06010h–06013h	DCLK_0DS	Display and LCD Clock Divisor Select	R/W
06014h–06017h	PWR_CLKC	Power Management and Miscellaneous Clock Control	R/W
Reserved (07000h–0FFFFh)			
07000h–0FFFFh	—	Reserved	—
Graphics Translation Table Range Definition (10000h–1FFFFh)			
10000h–1FFFFh	GTT	Graphics Translation Table Range Definition	WO
Reserved (20000h–2FFFFh)			
20000h–2FFFFh	—	Reserved	—

Table 2. Memory-Mapped Registers

Address Offset	Symbol	Register Name	Access
Overlay Registers (30000h–03FFFFh) (For additional address offsets in the double-buffering scheme, see Overlay Chapter)			
30000h–30003h	OV0ADD	Overlay 0 Register Update Address Overlay 0	R/W
30004h–30007h	—	Reserved	—
30008h–3000Bh	DOV0STA	Display/Overlay 0 Status	RO
3000Ch–3000Fh	—	Reserved	—
30010h–30027h	GAMMA[5:0]	Gamma Correction [5:0] (6 registers)	R/W
30028h–300FFh	—	Reserved	—
30100h–30103	OBUF_0Y	Overlay Buffer 0 Y Pointer	RO
30104h–30107h	OBUF_1Y	Overlay Buffer 1 Y Pointer	RO
30108h–3010Bh	OBUF_0U	Overlay Buffer 0 U Pointer	RO
3010Ch–3010Fh	OBUF_0V	Overlay Buffer 0 V Pointer	RO
30110h–30113h	OBUF_1U	Overlay Buffer 1 U Pointer	RO
30114h–30117h	OBUF_1V	Overlay Buffer 1 V Pointer	RO
30118h–3011Bh	OV0STRIDE	Overlay 0 Stride	RO
3011Ch–3011Fh	YRGB_VPH	Y/RGB Vertical Phase	RO
30120h–30123h	UV_VPH	UV Vertical Phase	RO
30124h–30127h	HORZ_PH	Horzontal Phase	RO
30128h–3012Bh	INIT_PH	Initial Phase	RO
3012Ch–3012Fh	DWINPOS	Destination Window Position	RO
30130h–30133h	DWINSZ	Destination Window Size	RO
30134h–30137h	SWID	Source Width	RO
30138h–3013Bh	SWIDQW	Source Width In QWords	RO
3013Ch–3013Fh	SHEIGHT	Source Height	RO
30140h–30143h	YRGBSCALE	Y/RGB Scale Factor	RO
30144h–30147h	UVSCALE	U V Scale Factor	RO
30148h–3014Bh	OV0CLRC0	Overlay 0 Color Correction 0 `	RO
3014Ch–3014Fh	OV0CLRC1	Overlay 0 Color Correction 1	RO
30150h–30153h	DCLRKV	Destination Color Key Value	RO
30154h–30157h	DCLRKM	Destination Color Key Mask	RO
30158h–3015Bh	SCLRKVH	Source Color Key Value High	RO
3015Ch–3015Fh	SCLRKVL	Source Color Key Value Low	RO
30160h–30163h	SCLRKM	Source Color Key Mask	RO
30164h–30167h	OV0CONF	Overlay 0 Configuration	RO
30168h–3016Bh	OV0CMD	Overlay 0 Command	RO
30170h–30173h	AWINPOS	Alpha Blend Window Position	RO
30174h–30177h	AWINZ	Alpha Blend Window Size	RO
30178h–3FFFFh	—	Reserved	—

Table 2. Memory-Mapped Registers

Address Offset	Symbol	Register Name	Access
BLT Engine Status (40000h–4FFFFh) (Software Debug)			
40000h–40003h	BR00	BLT Opcode and Control	RO
40004h–40007h	BR01	Setup BLT Raster OP, Control, and Destination Offset	RO
40008h–4000Bh	BR02	Clip Rectangle Y1 Address	RO
4000Ch–4000Fh	BR03	Clip Rectangle Y1 Address	RO
40010h–40013h	BR04	Clip Rectangle X1 and X2 Address	RO
40014h–40017h	BR05	Setup Expansion Background Color	RO
40018h–4001Bh	BR06	Setup Expansion Foreground Color	RO
4001Ch–4001Fh	BR07	Setup Color Pattern Address	RO
40020h–40023h	BR08	Destination X1 and X2	RO
40024h–40027h	BR09	Destination Address and Destination Y1 Address	RO
40028h–4002Bh	BR10	Destination Y2 Address	RO
4002Ch–4002Fh	BR11	BLT Source Pitch (Offset) or Monochrome Source Quadwords	RO
40030h–40033h	BR12	Source Address	RO
40034h–40037h	BR13	BLT Raster OP Control, and Destination Pitch	RO
40038h–4003Bh	BR14	Destination Width and Height	RO
4003Ch–4003Fh	BR15	Color Pattern Address	RO
40040h–40043h	BR16	Pattern Expansion Background and Solid Pattern Color	RO
40044h–40047h	BR17	Pattern Expansion Foreground Color	RO
40048h–4004Bh	BR18	Source Expansion Background and Destination Colr	RO
4004Ch–4004Fh	BR19	Source Expansion Foreground Color	RO
40074h–40077h	SSLADD	Source Scan Line Address	RO
40078h–4007Bh	DSLH	Destination Scan Line Height	RO
4007Ch–4007Fh	DSLRADD	Destination Scan Line Read Address	RO
40080h–4FFFFh	—	Reserved	—
Reserved (50000h–5FFFFh)			
50000h–5FFFFh	—	Reserved	—

Table 2. Memory-Mapped Registers

Address Offset	Symbol	Register Name	Access
LCD/TV-Out Registers (60000h–6FFFFh)			
LCD/TV-Out			
60000h–60003h	HTOTAL	Horizontal Total	R/W
60004h–60007h	HBLANK	Horizontal Blank	R/W
60008h–6000Bh	HSYNC	Horizontal Sync	R/W
6000Ch–6000Fh	VTOTAL	Vertical Total	R/W
60010h–60013h	VBLANK	Vertical Blank	R/W
60014h–60017h	VSYNC	Vertical Sync	R/W
60018h–6001Bh	LCDTV_C	LCD / TV-Out Control	R/W
6001Ch–6001Fh	OVRACT	Overlay Active Register	R/W
60020h–60023h	BCLRPAT	Border Color Pattern	R/W
60024h–6FFFFh	—	Reserved	—
Display and Cursor Control Registers (70000h–7FFFFh)			
70000h–70003h	DISP_SL	Display Scan Line Count	R/W
70004h–70007h	DISP_SLC	Display Scan Line Count Range Compare	R/W
70008h–7000Bh	PIXCONF	Pixel Pipeline Configuration	R/W
7000Ch–7000Fh	BLTCNTL	BLT Control	R/W
70010h–70013h	DIAG	Diagnostic	R/W
70014h–7001Fh	SWF[1:3]	Software Flags [1:3] (3 registers)	R/W
70020h–70023h	DPLYBASE	Display Base Address	R/W
70024h–70027h	DPLYSTAS	Display Status Select	R/W
70080h–70083h	CURCNTR	Cursor Control and Vertical Extension	R/W
70084h–70087h	CURBASE	Cursor Base Address	R/W
70028h–7002Bh	CURPOS	Cursor Position	R/W
7002Ch–7FFFFh	—	Reserved	—

3.3. VGA and Extended VGA Register Map

For I/O locations, the value in the address column represents the register I/O address. For memory mapped locations, this address is an offset from the base address programmed in the MMADR register (PCI Configuration register).

3.3.1. VGA and Extended VGA I/O and Memory Register Map

Table 3. I/O and Memory Register Map

Address	Register Name (Read)	Register Name (Write)
2D Registers		
3B0h–3B3h	Reserved	Reserved
3B4h	VGA CRTC Index (CRX) (monochome)	VGA CRTC Index (CRX) (monochome)
3B5h	VGA CRTC Data (monochome)	VGA CRTC Data (monochome)
3B6h–3B9h	Reserved	Reserved
3BAh	VGA Status Register (ST01)	VGA Feature Control Register (FCR)
3BBh–3BFh	Reserved	Reserved
3C0h	VGA Attribute Controller Index (ARX)	VGA Attribute Controller Index (ARX)/ VGA Attribute Controller Data (alternating writes select ARX or write ARxx Data)
3C1h	VGA Attribute Controller Data (read ARxx data)	Reserved
3C2h	VGA Feature Read Register (ST00)	VGA Miscellaneous Output Register (MSR)
3C3h	Reserved	Reserved
3C4h	VGA Sequencer Index (SRX)	VGA Sequencer Index (SRX)
3C5h	VGA Sequencer Data (SRxx)	VGA Sequencer Data (SRxx)
3C6h	VGA Color Palette Mask (DACMASK)	VGA Color Palette Mask (DACMASK)
3C7h	VGA Color Palette State (DACSTATE)	VGA Color Palette Read Mode Index (DACRX)
3C8h	VGA Color Palette Write Mode Index (DACWX)	VGA Color Palette Write Mode Index (DACWX)
3C9h	VGA Color Palette Data (DACDATA)	VGA Color Palette Data (DACDATA)
3CAh	VGA Feature Control Register (FCR)	Reserved
3CBh	Reserved	Reserved
3CCh	VGA Miscellaneous Output Register (MSR)	Reserved
3CDh	Reserved	Reserved
3CEh	VGA Graphics Controller Index (GRX)	VGA Graphics Controller Index (GRX)
3CFh	VGA Graphics Controller Data (GRxx)	VGA Graphics Controller Data (GRxx)
3D0h–3D1h	Reserved	Reserved
2D Registers		
3D4h	VGA CRTC Index (CRX)	VGA CRTC Index (CRX)
3D5h	VGA CRTC Data (CRxx)	VGA CRTC Data (CRxx)
2D Registers		
3D8h–3D9h	Reserved	Reserved
3DAh	VGA Status Register (ST01)	VGA Feature Control Register (FCR)
3DBh–3DFh	Reserved	Reserved

3.4. Indirect VGA and Extended VGA Register Indices

Programming an index value into the appropriate SRX, GRX, ARX, or CRX register indirectly accesses the registers listed in this section. The index and data register address locations are listed in the previous section. Additional details concerning the indirect access mechanism are provided in the *VGA and Extended VGA Registers* Chapter (see SRxx, GRxx, ARxx or CRxx sections).

Table 4. 2D Sequence Registers (3C4h / 3C5h)

Index	Sym	Description
00h	SR00	Sequencer Reset
01h	SR01	Clocking Mode
02h	SR02	Plane / Map Mask
03h	SR03	Character Font
04h	SR04	Memory Mode
07h	SR07	Horizontal Character Counter Reset

Table 5. 2D Graphics Controller Registers (3CEh / 3CFh)

Index	Sym	Register Name
00h	GR00	Set / Reset
01h	GR01	Enable Set / Reset
02h	GR02	Color Compare
03h	GR03	Data Rotate
04h	GR04	Read Plane Select
05h	GR05	Graphics Mode
06h	GR06	Miscellaneous
07h	GR07	Color Don’t Care
08h	GR08	Bit Mask
10h	GR10	Address Mapping
11h	GR11	Page Selector
14:1Fh	GR[14:1F]	Software Flags

Table 6. 2D Attribute Controller Registers (3C0h / 3C1h)

Index	Sym	Register Name
00h	AR00	Palette Register 0
01h	AR01	Palette Register 1
02h	AR02	Palette Register 2
03h	AR03	Palette Register 3
04h	AR04	Palette Register 4
05h	AR05	Palette Register 5
06h	AR06	Palette Register 6
07h	AR07	Palette Register 7
08h	AR08	Palette Register 8
09h	AR09	Palette Register 9
0Ah	AR0A	Palette Register A
0Bh	AR0B	Palette Register B
0Ch	AR0C	Palette Register C
0Dh	AR0D	Palette Register D
0Eh	AR0E	Palette Register E
0Fh	AR0F	Palette Register F
10h	AR10	Mode Control
11h	AR11	Overscan Color
12h	AR12	Memory Plane Enable
13h	AR13	Horizontal Pixel Panning
14h	AR14	Color Select

Table 7. 2D CRT Controller Registers (3B4h / 3D4h / 3B5h / 3D5h)

Index	Sym	Register Name
00h	CR00	Horizontal Total
01h	CR01	Horizontal Display Enable End
02h	CR02	Horizontal Blanking Start
03h	CR03	Horizontal Blanking End
04h	CR04	Horizontal Sync Start
05h	CR05	Horizontal Sync End
06h	CR06	Vertical Total
07h	CR07	Overflow
08h	CR08	Preset Row Scan
09h	CR09	Maximum Scan Line
0Ah	CR0A	Text Cursor Start
0Bh	CR0B	Text Cursor End
0Ch	CR0C	Start Address High
0Dh	CR0D	Start Address Low
0Eh	CR0E	Text Cursor Location High

Index	Sym	Register Name
0Fh	CR0F	Text Cursor Location Low
10h	CR10	Vertical Sync Start
11h	CR11	Vertical Sync End
12h	CR12	Vertical Display Enable End
13h	CR13	Offset
14h	CR14	Underline Location
15h	CR15	Vertical Blanking Start
16h	CR16	Vertical Blanking End
17h	CR17	CRT Mode
18h	CR18	Line Compare
22h	CR22	Memory Read Latch Data
30h	CR30	Extended Vertical Total
31h	CR31	Extended Vertical Display End
32h	CR32	Extended Vertical Sync Start
33h	CR33	Extended Vertical Blanking Start
35h	CR35	Extended Horizontal Total Time
39h	CR39	Extended Horizontal Blank Time
40h	CR40	Extended Start Address
41h	CR41	Extended Offset
42h	CR42	Extended Start Address High
70h	CR70	Interlace Control
71h	CR71	NTSC/PAL Video Output Control
72h	CR72	Horizontal Serration 1 Start
73h	CR73	Horizontal Serration 2 Start
74h	CR74	NTSC/PAL Horizontal Pulse Width
75h	CR75	TV-Out Control
76h	CR76	TV-Out Horizontal Count Upper
77h	CD77	TV-Out Horizontal Count Lower
78h	CR78	TV-Out Vertical Count Upper
79h	CR79	TV-Out Vertical Count Lower
80h	CR80	I/O Control
81h	CR81	Reserved
82h	CR82	Blink Rate Control

3.4.1. Graphics Address Translation

The Intel® 815 chipset uses a logical memory-addressing concept for accessing graphics data. The GC supports a 64-MB logical address space, where each 4-KB logical page can be mapped to a physical memory page in System RAM, PCI Memory, or an optional Display Cache memory. This mapping is performed through the use of a Graphics Translation Table (GTT).

GC engines can address the full 64-MB logical address space. The processor is provided access to either the full 64-MB space, or just the lower 32 MB, via a PCI memory range associated with the graphics device.

The GTT is allocated in system RAM and maintained by the graphics driver. The 4 KB-aligned physical address of the 64 KB GTT is programmed via the GC's PGTBL register.

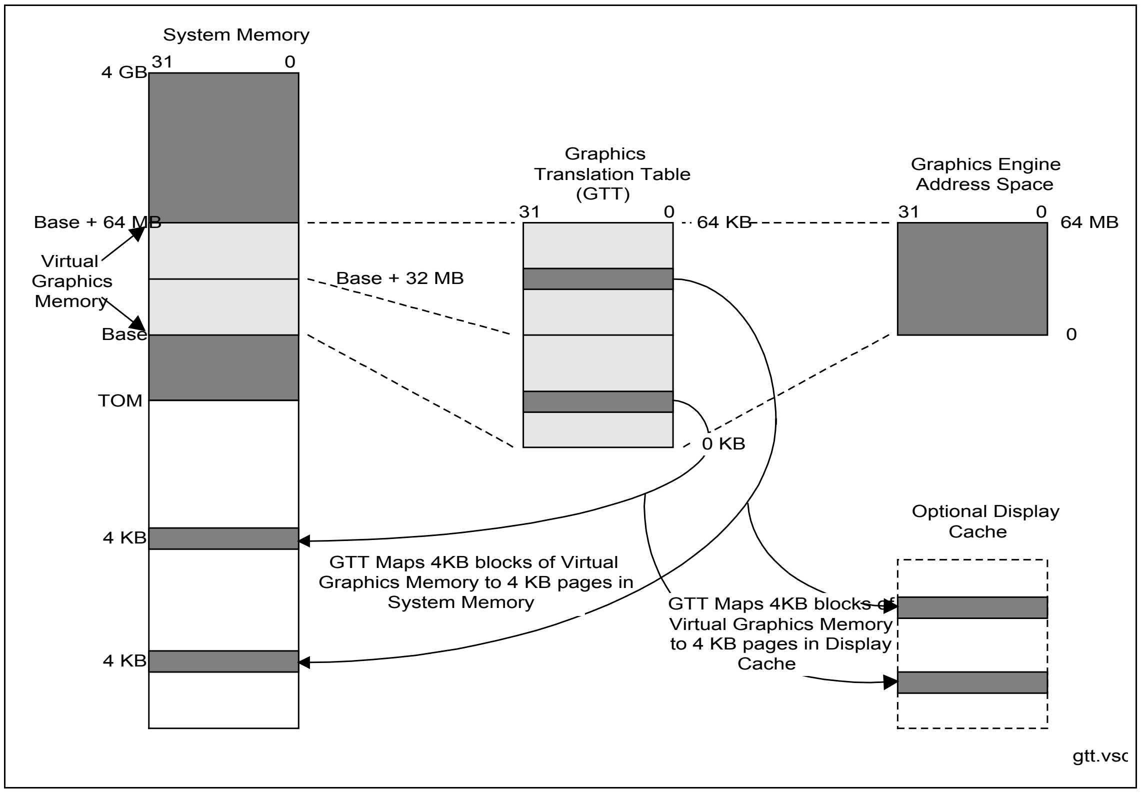
Each of the 16K DWord GTT entries can map a 4-KB logical page to a physical memory page. Fields in the GTT entry control the mapping of that logical page in the following manner:

- (V) whether or not that logical 4 KB page is mapped to a physical memory page. Accesses to invalid pages will result in an error interrupt.
- (T1T0) the physical memory address space of the mapped page:
  - System RAM page (no processor cache snoop)
  - PCI Memory page (processor cache snooped if below TOM)
  - Display Cache page
- the page number of the mapped page (within the particular physical memory address space)

Although the GTT format permits any logical page to be mapped to any page in the supported physical memory address spaces, the GC imposes restrictions on how specific graphics operands (buffers, etc.) can be mapped to physical memory.

The GTT entries must be written via a GTT alias in the graphics device's memory-mapped register space (10000h–1FFFFh). This allows the GC to snoop GTT entry writes and invalidate graphics TLBs as required. The GTT entries must *not* be written directly in system memory.

Figure 8. GTT Mapping



3.4.2. Memory Buffers for GC's Instruction Interface

The GC provides two Ring Buffer (RB) mechanisms through which instructions can be passed to the GMCH's Instruction Parser. In addition, the GC provides for the execution of instruction sequences *external* to the ring buffers. These sequences are called "batch buffers", and are initiated through the use of GFXCMDPARSER_BATCH_BUFFER instructions that specify the starting address and length of the batch buffers. For detailed information on these buffers, refer to the *Programming Interface* Chapter.

4. Graphics Translation Table (GTT) Range Definition

Address Offset:	10000h–FFFFh
Default Value:	Page table range 64 KB
Access:	aligned DWord-QWord Write Only

This range defined within the graphics memory mapped register space is for the memory manager to access the graphics translation table. A page table write will invalidate that entry in internal translation table caches (TLBs). The translation table resided in system memory and can be accessed by the memory manager directly. However, to ensure coherency between hardware maintained translation caches and the translation table in main memory, the memory manager must use this range to update the translation table.

The page table is required to be QW aligned with each entry being DWord aligned such that each QW stores the translation for 2, 4 KB pages. Page Table base address for graphics memory is programmed in the PGTB_CNTL register. For graphics memory of 64 MB with a TLB block size of 4 KB, 16K entries are needed. Each entry is 4 bytes; hence, the page table size is 64 KB.

Page Table Entry: 1 DWord per 4-KB page.

31 30	29 12	11 3	2 1	0
XX=00	Physical Address 29:12	Reserved	T1T0	V

V:	1 = Valid page table entry (PTE). 0 = Invalid page table entry (PTE). An access to an invalid PTE will result in an interrupt.
T1T0:	01 = Physical address targets Local Memory 00 = Physical address targets main memory (not snooped) 11 = Physical address targets cacheable main memory (results in snoop on processor bus) 10 = Reserved.

Note: T1T0 = 11 is used only if the surface is a Blit soure or destination operand used within the context of a source copy command.

Note that the 4-KB pages of physical main memory (that have been mapped to the graphics aperture through the GTT) must be accessed strictly through the aperture. The GMCH does not guarantee data coherency if any of these pages are accessed directly using their physical addresses. For example, a 4-KB page of main memory has been mapped via the GTT to a 4-KB aperture page. Although, the GMCH still allows this 4-KB page to be accessed directly through its physical memory address, the chipset does not guarantee data coherency with respect to accesses through the normal graphics aperture address range. This is because a read to the aperture memory can result in prefetching and caching of data, while a write to the aperture can result in temporary write data buffering in the graphics controller of the GMCH. Accesses to these same memory locations through their physical address take a different logical path through the chipset controller side of the GMCH. There is **no** hardware support for ensuring coherency between these two access paths.

This page is intentionally left blank.

5. Basic Initialization Procedures

5.1. Initialization Sequence

The initialization of graphics driver resources can be broken down into three categories: hardware detection, frame buffer initialization, and hardware register initialization. Each category is discussed in more detail in the following sections.

In all discussions that follow, there is a basic assumption that the graphics adapter has completed the power-on video BIOS initialization or video BIOS reset. Therefore, the adapter is in a known state and will respond in compliance with the VGA and VESA specifications.

5.2. Hardware Detection (Probe)

Most operating systems will probe for installed devices. The Intel 8281x family of devices advertise their presence in PCI space by using unique values in the PCI VendorId and DeviceId locations. The following table lists the device IDs used to identify the members of the 8281x family of graphics adapters.

Vendor Id PCI Offset: 0	Device Id PCI Offset: 2	PCI Device Number	Characteristics
8086	7120	0	Intel® 82810 chipset bridge
8086	7121	1	Intel® 82810 chipset
8086	7122	0	Intel® 82810 chipset DC100 bridge
8086	7123	1	Intel® 82810 chipset DC100
8086	7124	0	Intel® 82810E chipset bridge
8086	7125	1	Intel® 82810E chipset
8086	1100	0	Intel® 82815 chipset GMCH host-hub interface bridge / DRAM controller FSB limited to 100 MHz
8086	1101	1	Intel® 82815 chipset FSB limited to 100 MHz; AGP bridge
8086	1102	2	Intel® 82815 chipset FSB limited to 100 MHz; Internal graphics device
8086	1110	0	Intel® 82815 chipset no AGP, internal graphics only; GMCH host-hub interface bridge / DRAM controller
8086	1112	2	Intel® 82815 chipset no AGP, internal graphics only; Internal graphics device
8086	1120	0	Intel® 82815 chipset no internal graphics, AGP only; GMCH host-hub interface bridge / DRAM controller
8086	1121	1	No internal graphics, AGP only; AGP bridge
8086	1130	0	Intel® 82815 chipset fully featured; GMCH host-hub interface bridge / DRAM controller
8086	1131	1	Intel® 82815 chipset fully featured; AGP bridge
8086	1132	2	Intel® 82815 chipset fully featured; Internal graphics device

Once the operating system has identified the device, it can load the appropriate driver.

One of the first tasks of the driver is to make sure that the device matches the driver. Checking that the driver and device match is done in much the same way that the operating system identifies the graphics adapter. That is, the PCI VendorId and ProductId values are examined. Some operating systems will make available to the driver the values it found during its scan. If not, the driver must scan the PCI space until it finds a match on the VendorId and ProductId values. The driver normally caches this information so that it is accessible by other driver modules, when needed.

The next task of the device driver is to ensure that required resources are present. These resources include the minimum memory requirements, IO address space requirements, and operating system support requirements (such as GART support). If the driver detects that the operating system or the physical hardware does not meet the driver's minimum requirements, the driver should not load. The operating system should then be able to make use of the graphics adapter in its VGA- and VESA-compliant mode.

If the operating system and hardware support are present, the driver should acquire the blocks of memory and IO address space that will be required. These blocks should include at least the following:

- Memory-mapped IO address space: 512 KB beginning at 0x80000
- Linear frame buffer space: 32 or 64 MB beginning at 0xFE000000
- Legacy IO addresses to support monochrome or color monitors
- VGA IO addresses

5.3. Frame Buffer Initialization

The frame buffer initialization is responsible for setting up the memory that will contain the display data. Other objects also can be stored in display memory.

The following steps should be performed:

- Map a 0x80000-byte region in memory to the MMIO base address. The base address of the memory-mapped region should be programmed into the MMADDR register, offset 14 in the PCI address space.
- Allocate enough memory for the frame buffer from a memory pool created during initialization. The amount of memory is determined by system characteristics, but should default to at least 8 MB.
- If a hardware cursor is being used, allocate memory for the hardware cursor from the same memory pool. The hardware does not use the GART to access the memory for the cursor, so local-to-physical memory address translation must be performed. The hardware cursor memory address should be programmed into the CURBASE register, memory-mapped address 70084h.
- The low-priority ring buffer memory should be initialized to 0. The low-priority ring buffer pointers should be programmed into the ring buffer pointer registers, RINGBUF, which begin at offset 2030h in the memory-mapped IO space.

5.4. Hardware Register Initialization

5.4.1. Color vs. Monochrome Monitors

The mapping and initialization of some hardware registers depends in part on whether the graphics adapter is attached to a monochrome or color monitor. The following steps illustrate how to determine the type of output device attached to the graphics adapter:

- Read the Miscellaneous Output Register (0x3CC).
- Test the low-order bit of the Miscellaneous Output Register, and interpret it as follows:
  - 0: The adapter is in monochrome monitor mode. In this mode, the control register is 3B4 and 3B5, and status is at 3BA.
  - 1: The adapter is in color monitor mode. In this mode, the control register is 3D4 and 3D5, and status is at 3DA.

See the section on VGA compatibility for a description of the register space that must be acquired.

5.4.2. Protect Registers: Locking and Unlocking

To make use of some protected VGA registers, a locking and unlocking mechanism needs to be implemented. The following steps illustrate how to unlock (or unprotect) the VGA registers:

- Send a VERT_SYNC_END value to the register at vgaBase + 4.
- Read the value in the register at vgaBase + 5.
- Clear the high-order bit of the value just read.
- Write the resulting value back into the register at vgaBase + 5.

5.4.3. Checking Memory Frequency

The driver behavior occasionally must be modified, depending on the frequency at which the memory is running. The following steps illustrate how to determine the local memory frequency:

- Read the contents of the Intel® 82815 chipset Configuration Register (PCI address space 0x50).
- Examine the value of bit 4.
- The value is interpreted as follows:
  - 0: Frequency is 100 MHz.
  - 1: Frequency is 133 MHz.

5.5. Hardware State

Under certain conditions, it may be necessary to save and restore the hardware state of the graphics adapter. These conditions include mode switching, output device switching, processing changes in power state, and others. The next two sections provide a brief description of the state saving and restoration requirements.

5.6. Saving the Hardware State

Note that the VGA register unlocking protocol must be performed in order to access some of the registers described below.

During a state change, the driver should preserve the following registers to provide complete state restoration in the future:

• IO Control	CR80
• Address Mapping	GR10
• Bit Blit Control	MM 0x7000C
• Video Clock 2 / M	MM 0x6008
• Video Clock 2 / N	MM 0x600C
• Video Clock 2 / Divisor Select	MM 0x6012
• Vertical Total	CRX 30
• Vertical Display End	CRX 31
• Vertical Sync Start	CRX 32
• Vertical Blank Start	CRX 33
• Horizontal Total	CRX 35
• Horizontal Blnk	CRX 39
• Ext Offset	CRX 41
• Interlace Control	CRX 70
• Hardware Status Mask Register	MM 0x2098
• Interrupt Enable Register	MM 0x20A0
• Interrupt Identity Register	MM 0x20A4
• Interrupt Mask Register	MM 0x20A8
• Error Mask Register	MM 0x20B4
• Display Control Register	MM 0x70008
• Pixel Pipeline Configuration 0 Register	MM 0x70009
• Pixel Pipeline Configuration 1 Register	MM 0x7000A
• Pixel Pipeline Configuration 2 Register	MM 0x7000B
• Watermark and Burstlength Control	MM 0x20D8
• Low-priority ring information	MM 0x2030 – 0x203F

5.7. Restoring the Hardware State

The graphics adapter state should be restored by performing the following steps. Note some of the synchronization operations, especially those that ensure that the local memory is idle during the state restore. Also, much of the work involves reprogramming the registers with the values captured during the save-state operation.

- Blank the screen.
- Turn off DRAM refresh.
  - Read the value of the DRAM_CONTROL_HI Register (MM 0x3002).
  - Set the DRAM Refresh Rate bits (DDR Bits 4:3) to Disable_Refresh (value 0).
  - Write the modified value back to the DRAM_CONTROL_HI Register.
- Write the M, N, and P (i.e., the Divisor Select value) values from the saved state information.
- Restore the 8-bit DAC mode to what it was when the state was saved, but preserve the current value of the rest of the register containing this flag:
  - Read the Pixel Pipeline Configuration 0 Register.
  - Clear the current value of the 8- or 6-bit DAC mode.
  - OR–in (only) the value of the DAC_8_BIT from saved register information of the Pixel Pipeline Configuration 0 Register.
  - Write the result back to the Pixel Pipeline Configuration 0 Register.
- Restore the generic VGA registers to the values captured at save-state time.
- Restore the following registers to their saved state values:

— Vertical Total	CRX 30
— Vertical Display End	CRX 31
— Vertical Sync Start	CRX 32
— Vertical Blank Start	CRX 33
— Horizontal Total	CRX 35
— Horizontal Blnk	CRX 39
— Ext Offset	CRX 41

- The following registers should restore only certain bits from the saved state values:

  Interlace Control CRX 70
  - Read the current value.
  - Clear the interlace enable bit.
  - OR–in the saved value of the Interlace Control Register.
  - Write the result back into the Interlace Control Register.

  Address Mapping: GR10
  - Read the current value of the Address Mapping Register.
  - Save only the reserved bits values (bits 7:5).
  - OR–in the saved value of the Address Mapping Register.
  - Write the result back into the Address Mapping Register.
- Now the DRAM refresh can be turned on:
  - Read the value of the DRAM_CONTROL_HI Register.
  - Turn off the DRAM_REFRESH_RATE bits.
  - OR–in a 60-Hz refresh rate value.
  - Write the result back into the DRAM_CONTROL_HI Register.

- Other registers that should restore only certain bits from the saved-state values:

  Bit Blit Control MM 0x7000c
  - Read the current value of the Bit Blit Control Register.
  - Clear the bits pertaining to the Color Expansion Mode (bits 5:4).
  - OR–in the saved value of the Bit Blit Control Register.
  - Write the result back into the Bit Blit Control Register.

  Display Control Field MM 0x70008
  - Read the current value of the Display Control Register.
  - OR–in the saved value of the Display Control Register.
  - Write the result back into the Display Control Register.

  Pixel Pipeline Configuration 0 Field MM 0x70009
  - Read the current value of the Pixel Pipeline Configuration 0 Register.
  - Save reserved bits 6:5 and 2. Clear all other bits.
  - OR–in the saved value of the Pixel Pipeline Configuration 0 Register.
  - Write the result back into the Pixel Pipeline Configuration 0 Register.

  Pixel Pipeline Configuration 2 Field MM 0x7000b
  - Read the current value of the Pixel Pipeline Configuration 2 Register.
  - Save reserved bits 7:4 and 1:0. Clear all other bits.
  - OR–in the saved value of the Pixel Pipeline Configuration 2 Register.
  - Write the result back into the Pixel Pipeline Configuration 2 Register.

  Pixel Pipeline Configuration 1 Field MM 0x7000a
  - Read the current value of the Pixel Pipeline Configuration 1 Register.
  - Clear the Display Color Mode bit (bits 3:0).
  - OR–in the saved value of the Pixel Pipeline Configuration 1 Register.
  - Write the result back into the Pixel Pipeline Configuration 1 Register.

  Hardware Status Mask Register MM 0x2098
  - Read the current value of the Hardware Status Mask Register.
  - Clear everything but the reserved bits (14:13).
  - OR–in the saved value of the Hardware Status Mask Register.
  - Write the result back into the Hardware Status Mask Register.

  Interrupt Enable Register MM 0x20A0
  - Read the current value of the Interrupt Enable Register.
  - Clear everything but the reserved bits (14:13).
  - OR–in the saved value of the Interrupt Enable Register.
  - Write the result back into the Interrupt Enable Register.

  Interrupt Mask Register MM 0x20A8
  - Read the current value of the Interrupt Mask Register.
  - Clear everything but the reserved bits (14:13).
  - OR–in the saved value of the Interrupt Mask Register.
  - Write the result back into the Interrupt Mask Register.

  Error Mask Register MM 0x20B4
  - Read the current value of the Error Mask Register.
  - Clear everything but the reserved bits (15:6).
  - OR–in the saved value of the Error Mask Register.
  - Write the result back into the Error Mask Register.

Watermark and Burstlength Control MM 0x20D8
- Read the current value of the Watermark and Burstlength Control Register.
- Clear the burst length and watermark bits (bits 22:20, 17:12, 10:8 and 5:0).
- OR–in the saved value of the Watermark and Burstlength Control Register.
- Write the result back into the Watermark and Burstlength Control Register.

- Disable the low-priority ring buffer, in preparation for setting new values, by clearing the RING_VALID bit in the Low-Priority Ring Buffer Length field at MM 0x203C.
  - Read the current value of the Low-Priority Ring Buffer Length field (MM 0x203C).
  - Clear the valid bit (bit 0).
  - Write the result back into the Low-Priority Ring Buffer Length field.
- Set up the low-priority ring buffer.
  - Write a 0 to the low-priority ring buffer tail at MM 0x2030.
  - Write a 0 to the low-priority ring buffer head at MM 0x2034.
  - Restore the low-priority ring buffer start at MM 0x2038, but preserve the reserved bits.
  - Restore the Low-Priority Ring Buffer Length field, but preserve the Automatic Report Header Pointer bits and set the Ring Buffer Valid flag.
- Turn on the screen.
- Relock the protected register space in order to complete the state restoration process.

At this point the graphics adapter should function completely, in the mode identified by the saved-state information.

This page is intentionally left blank.

6. Blt Engine Programming

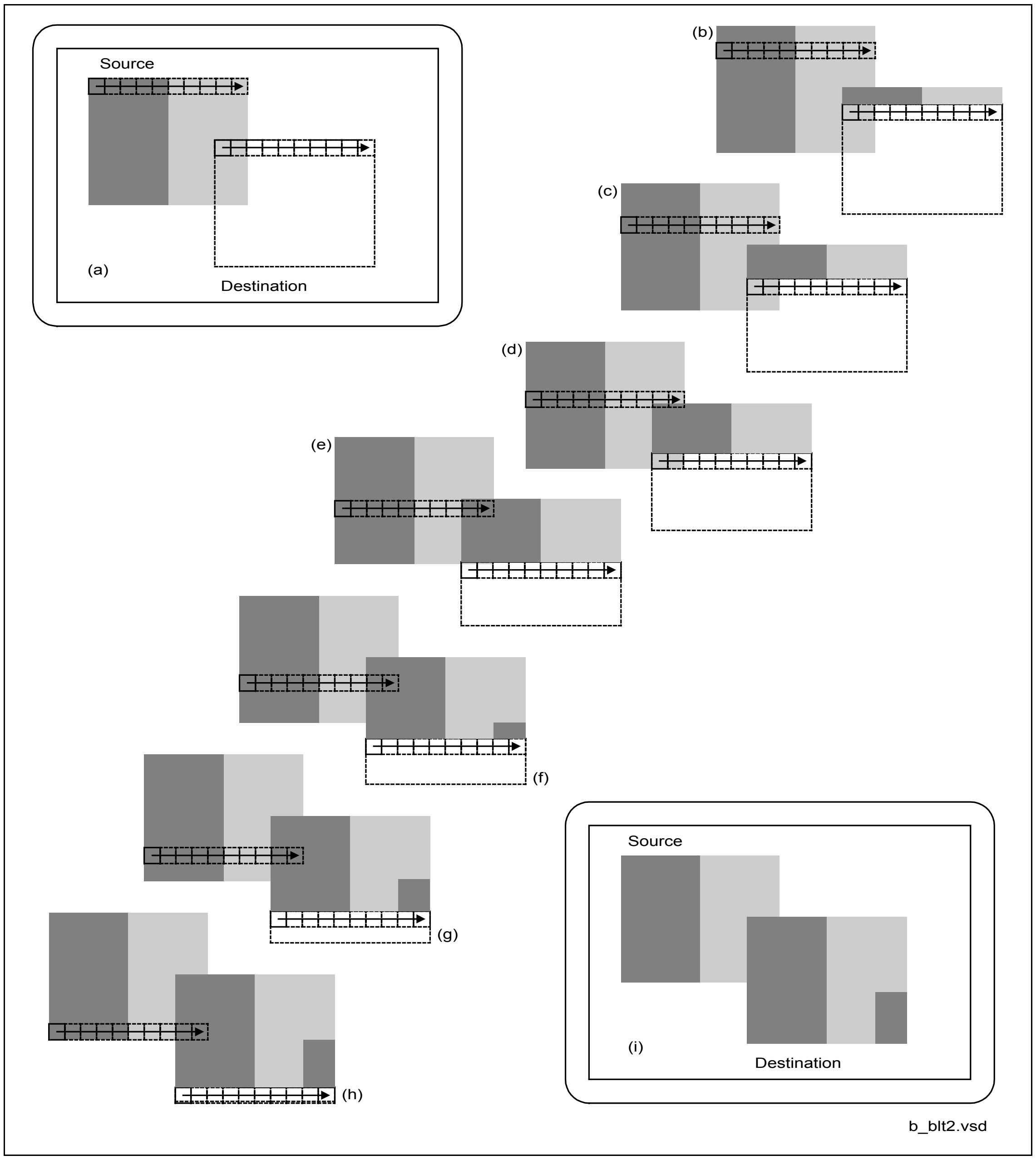
6.1. BLT Engine Programming Considerations

6.1.1. When the Source and Destination Locations Overlap

It is possible to have BLT operations in which the locations of the source and destination data overlap. This frequently occurs in BLT operations where a user is shifting the position of a graphical item on the display by only a few pixels. In these situations, the BLT engine must be programmed so that destination data is not written into destination locations that overlap with source locations before the source data at those locations has been read. Otherwise, the source data will become corrupted.

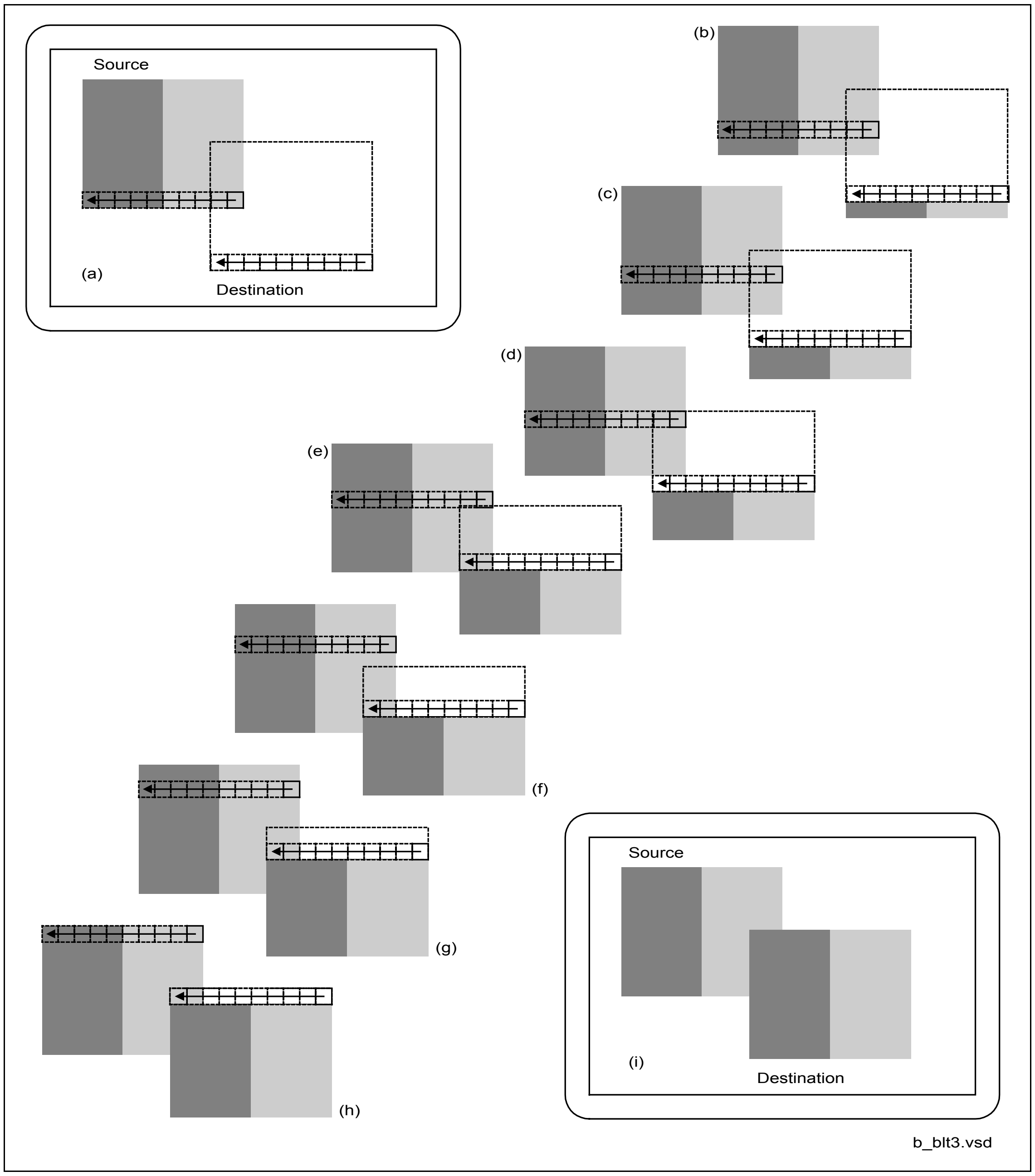
The following figure shows how the source data can be corrupted when a rectangular block is copied from a source location to an overlapping destination location. The BLT engine reads from the source location and writes to the destination location starting with the left-most pixel in the top-most line of both, as shown in step (a). As shown in step (b), corruption of the source data has already started with the copying of the top-most line in step (a) — part of the source that originally contained lighter-colored pixels has now been overwritten with darker-colored pixels. More source data corruption occurs as steps (b) through (d) are performed. At step (e), another line of the source data is read, but the two right-most pixels of this line are in the region where the source and destination locations overlap, and where the source has already been overwritten as a result of the copying of the top-most line in step (a). Starting in step (f), darker-colored pixels can be seen in the destination where lighter-colored pixels should be. This errant effect occurs repeatedly throughout the remaining steps in this BLT operation. As more lines are copied from the source location to the destination location, it becomes clear that the end result is not what was originally intended.

Figure 9. Source Corruption in BLT with Overlapping Source and Destination Locations



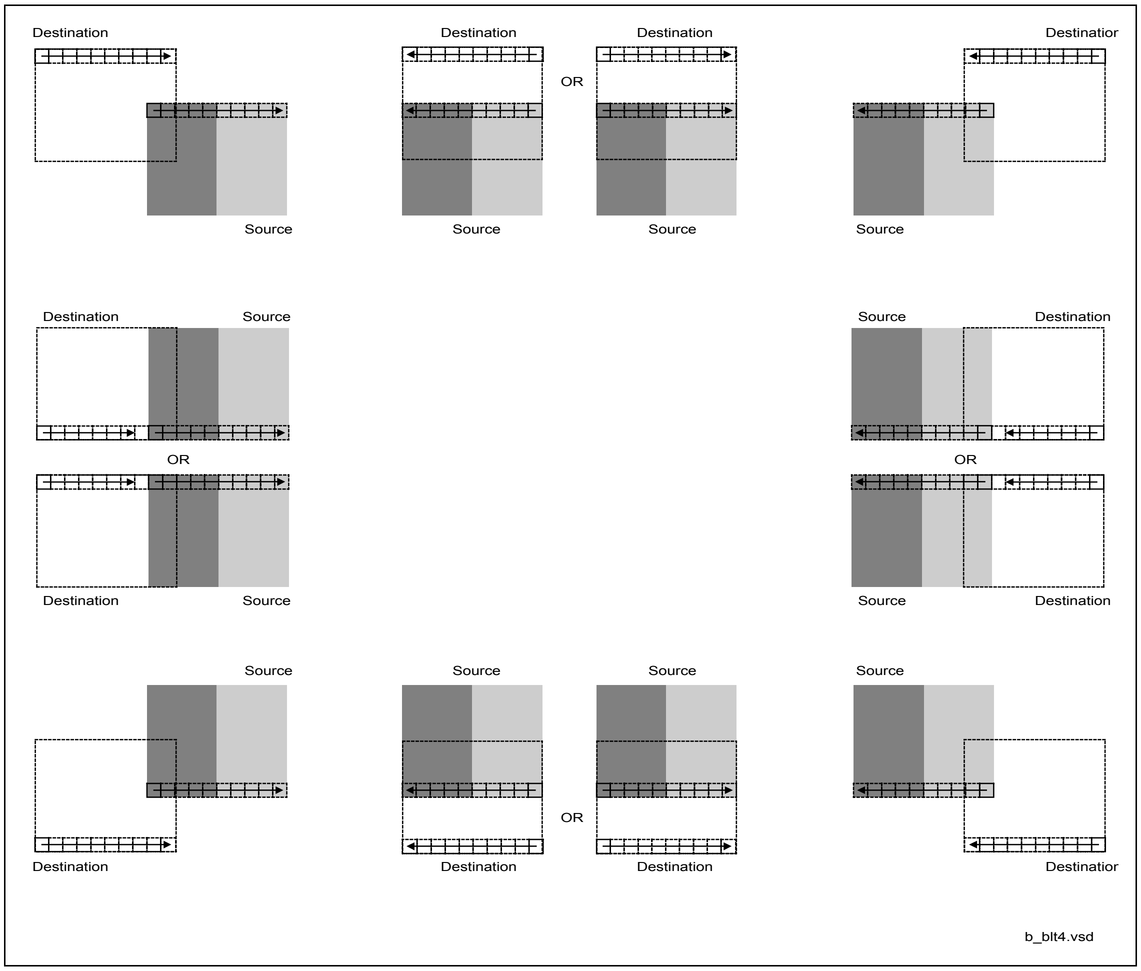
The BLT engine can alter the order in which source data is read and destination data is written when necessary to avoid source data corruption problems when the source and destination locations overlap. The command packets provide the ability to change the point at which the BLT engine begins reading and writing data from the upper left-hand corner (the usual starting point) to one of the other three corners. The BLT engine may be set to read data from the source and write it to the destination starting at any of the four corners of the panel.

Figure 10. Correctly Performed BLT with Overlapping Source and Destination Locations



The figure below illustrates how this feature of the BLT engine can be used to perform the same BLT operation as was illustrated in the figure above, while avoiding the corruption of source data. As shown in the figure below, the BLT engine reads the source data and writes the data to the destination starting with the right-most pixel of the bottom-most line. By doing this, no pixel existing where the source and destination locations overlap will ever be written to before it is read from by the BLT engine. By the time the BLT operation has reached step (e) where two pixels existing where the source and destination locations overlap are about to be over written, the source data for those two pixels has already been read.

Figure 11. Suggested Starting Points for Possible Source & Destination Overlap Situations



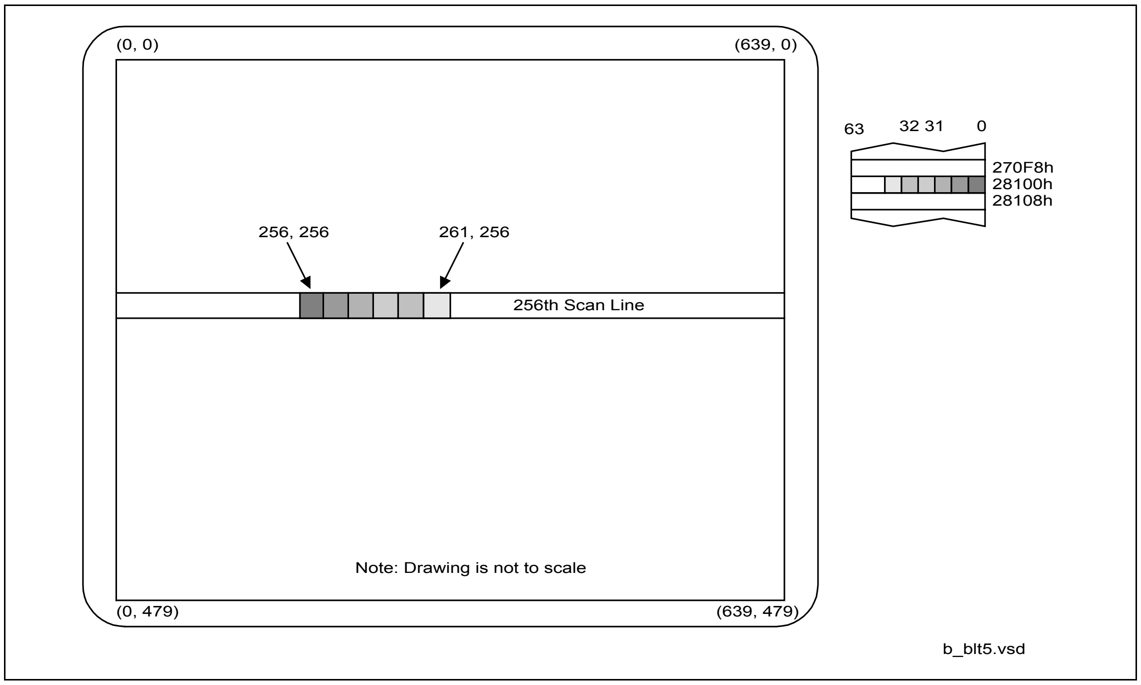
The figure above shows the recommended lines and pixels to be used as starting points in each of 8 possible ways in which the source and destination locations may overlap. In general, the starting point should be within the area in which the source and destination overlap.

6.2. Basic Graphics Data Considerations

6.2.1. Contiguous vs. Discontinuous Graphics Data

Graphics data stored in memory, particularly in the frame buffer of a graphics system, has organizational characteristics that often distinguish it from other varieties of data. The main distinctive feature is the tendency for graphics data to be organized in a discontinuous block of graphics data made up of multiple sub-blocks of bytes, instead of a single contiguous block of bytes.

Figure 12. Representation of On-Screen Single 6-Pixel Line in the Frame Buffer

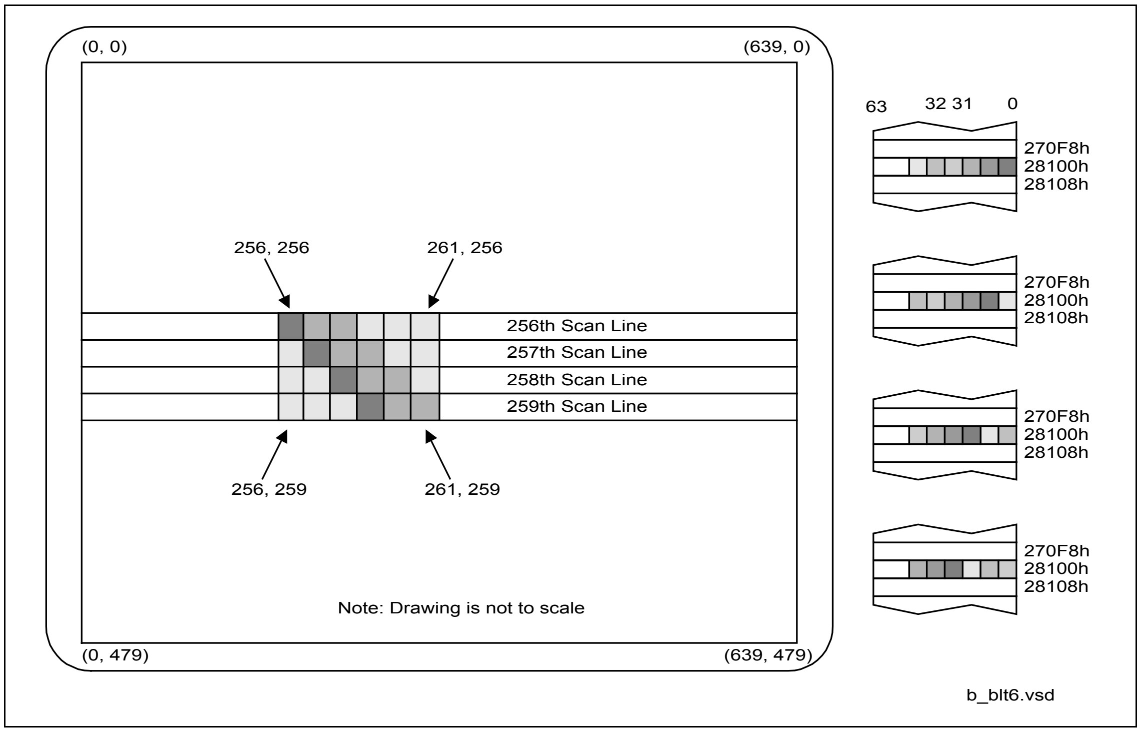


The figure above shows an example of contiguous graphics data — a horizontal line made up of six adjacent pixels within a single scan line on a display with a resolution of 640x480. Presuming that the graphics system driving this display has been set to 8 bits per pixel, and that the frame buffer's starting address of 0h corresponds to the upper left-most pixel of this display, then the six pixels that make this horizontal line starting at coordinates (256, 256) would occupy six bytes starting at frame buffer address 28100h, and ending at address 28105h.

In this case, there is only one scan line's worth of graphics data in this single horizontal line, so the block of graphics data for all six of these pixels exists as a single, contiguous block comprised of only these six bytes. The starting address and the number of bytes are the only pieces of information that a BLT engine would require to read this block of data.

The simplicity of the above example of a single horizontal line contrasts sharply to the example of discontinuous graphics data depicted in the figure below. The simple six-pixel line of the figure above is now accompanied by three more six-pixel lines placed on subsequent scan lines, resulting in the 6x4 block of pixels shown.

Figure 13. Representation of On-Screen 6x4 Array of Pixels in the Frame Buffer



Since there are other pixels on each of the scan lines on which this 6x4 block exists that are not part of this 6x4 block, what appears to be a single 6x4 block of pixels on the display must be represented by a discontinuous block of graphics data made up of 4 separate sub-blocks of six bytes apiece in the frame buffer at addresses 28100h, 28380h, 28600h, and 28880h. This situation makes the task of reading what appears to be a simple 6x4 block of pixels more complex. However, there are two characteristics of this 6x4 block of pixels that help simplify the task of specifying the locations of all 24 bytes of this discontinuous block of graphics data: all four of the sub-blocks are of the same length, and the four sub-blocks are separated from each other at equal intervals.

The BLT engine is designed to make use of these characteristics of graphics data to simplify the programming required to handle discontinuous blocks of graphics data. For such a situation, the BLT engine requires only four pieces of information: the starting address of the first sub-block, the length of a sub-block, the offset (in bytes), pitch, of the starting address of each subsequent sub-block, and the quantity of sub-blocks.

6.2.2. Source Data

The source data may exist in the frame buffer or main memory graphics memory where the BLT engine may read it directly, or it may be provided to the BLT engine by the host processor through the command packets. The block of source graphics data may be either contiguous or discontinuous, and may be either in color (with a color depth that matches that to which the BLT engine has been set) or monochrome.

The source select bit in the command packets specifies whether the source data exists in the frame buffer or is provided through the command packets. Monochrome source data is always specified as being supplied through an immediate command packet.

If the color source data resides within the frame buffer or main memory graphics memory, then the Source Address Register, specified in the command packets is used to specify the address of the source. However, if the host processor provides the source data, then this register takes on a different function and the three least-significant bits of the Source Address Register can be used to specify a number of bytes that must be skipped in the first quadword received from the command packet to reach the first byte of valid source data.

In cases where the host processor provides the source data, it does so by writing the source data to ring buffer directly after the BLT command that requires the data or uses an IMMEDIATE_INDIRECT_BLT command packet which has a size and pointer to the operand in Main memory graphics memory.

There is also an address space used for debug where the processor can write the source data. It is a 64-KB memory space on the host bus. There is no actual memory allocated to this memory space, so any data that is written to this location cannot be read back. This memory space is simply a range of memory addresses that the BLT engine's address decoder watches for the occurrence of any memory writes.

The BLT engine loads all data written to any memory address within this memory space or through the command packet in the order in which it is written, regardless of the specific memory address to which it is written and uses that data as the source data in the current BLT operation. The block of bytes sent by the host processor to either this data port or through the command packets must be quadword-aligned, although the source data contained within the block of bytes does not need to be aligned. As mentioned earlier, the least significant three bits of the Source Address Register are used to specify the number of bytes that must be skipped in the first quadword of color data to reach the first byte of valid source data.

To accommodate discontinuous source data, the source and destination pitch registers can be used to specify the offset in bytes from the beginning of one scan line's worth source data to the next. Otherwise, if the source data is contiguous, then an offset equal to the length of a scan line's worth of source data should be specified.

6.2.3. Monochrome Source Data

The opcode of the command packet specifies whether the source data is color or monochrome. Since monochrome graphics data only uses one bit per pixel, each byte of monochrome source data typically carries data for 8 pixels, which hinders the use of byte-oriented parameters when specifying the location and size of valid source data. Monochrome source data is always supplied through the command stream, which avoids the read latency during BLT Engine operation. Some additional parameters must be specified to ensure the proper reading and use of monochrome source data by the BLT engine. The BLT engine also provides additional options for the manipulation of monochrome source data versus color source data.

The various bit-wise logical operations and per-pixel write-masking operations were designed to work with color data. In order to use monochrome data, the BLT engine converts it into color through a process called color expansion, which takes place as a BLT operation is performed. In color expansion, the single bits of monochrome source data are converted into one, two, three, or four bytes (depending on the color depth to which the BLT engine has been set) of color data that are set to carry value corresponding to either the foreground or background color that have been specified for use in this conversion process. If a given bit of monochrome source data carries a value of 1, then the byte(s) of color data resulting from the conversion process will be set to carry the value of the foreground color. If a given bit of monochrome source data carries a value of 0, then the resulting byte(s) will be set to the value of the background color. The foreground and background colors used in the color expansion of monochrome source data can be set in the source expansion foreground color register and the source expansion background color register.

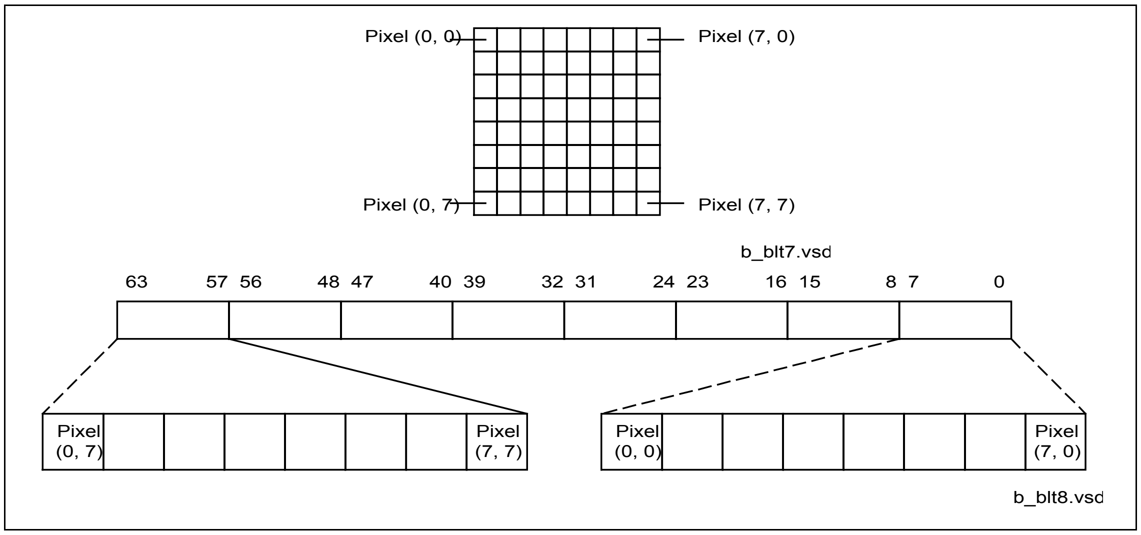
The BLT Engine requires that the bit alignment of each scan line's worth of monochrome source data be specified. Each scan line's worth of monochrome source data is word aligned, but can actually start on any bit boundary of the first byte. Monochrome text is special cased and it is bit packed, where there are no invalid pixels (bits) between scan lines. There is a 3 bit field which indicates the starting pixel position within the first byte for each scan line, Mono Source Start.

The BLT engine also provides various clipping options for use with specific BLT commands (BLT_TEXT) with a monochrome source. Clipping is supported through: Clip rectangle Y addresses and X coordinates along with scan line starting and ending addresses along with X starting and ending coordinates.

6.2.4. Pattern Data

The color pattern data must exist within the frame buffer or Main memory graphics memory where the BLT engine may read it directly. Monochrome pattern data is supplied by the command packet when it is to be used. As shown in figure below, the block of pattern graphics data always represents a block of 8x8 pixels. The bits or bytes of a block of pattern data may be organized in the frame buffer memory in only one of four ways, depending upon its color depth which may be 8, 16, 24, or 32 bits per pixel (whichever matches the color depth to which the BLT engine has been set), or monochrome.

Figure 14. Pattern Data -- Always an 8x8 Array of Pixels



The Pattern Address Register is used to specify the address of the color pattern data at which the block of pattern data begins. The three least significant bits of the address written to this register are ignored, because the address must be in terms of quadwords. This is because the pattern must always be located on an address boundary equal to its size. Monochrome patterns take up 8 bytes, or a single quadword of space, and are loaded through the command packet that uses it. Similarly, color patterns with color depths of 8, 16, and 32 bits per pixel must start on 64-byte, 128-byte and 256-byte boundaries, respectively. Color patterns with color depths of 24 bits per pixel must start on 256-byte boundaries, despite the fact that the actual color data fills only 3 bytes per pixel. The next 4 figures show how monochrome, 8bpp, 16bpp, 24bpp, and 32bpp pattern data, respectively, is organized in memory.

Figure 15. 8bpp Pattern Data -- Occupies 64 Bytes (8 quadwords)

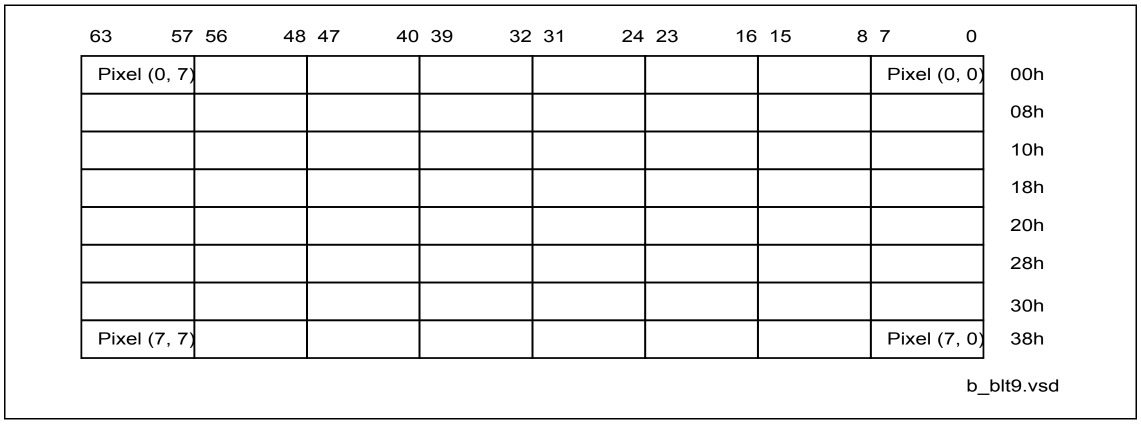


Figure 16. 16bpp Pattern Data -- Occupies 128 Bytes (16 quadwords)

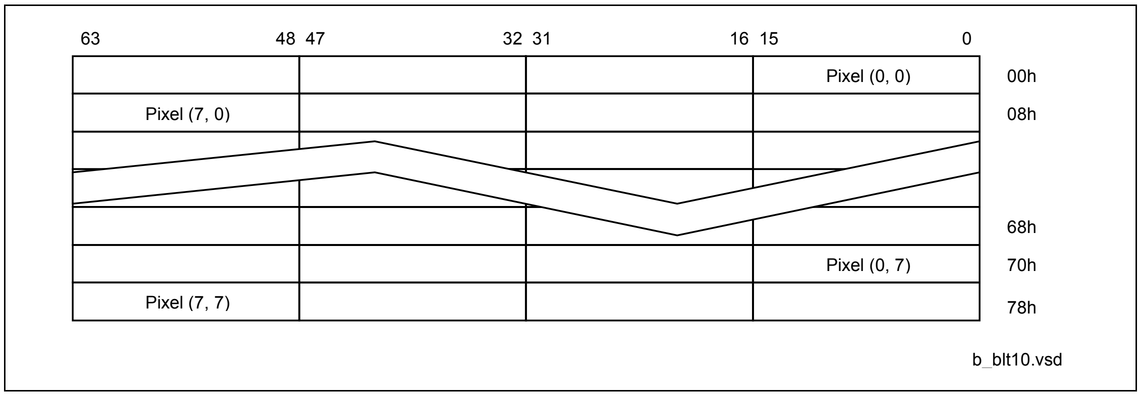


Figure 17. 24bpp Pattern Data -- Occupies 256 Bytes (32 quadwords)

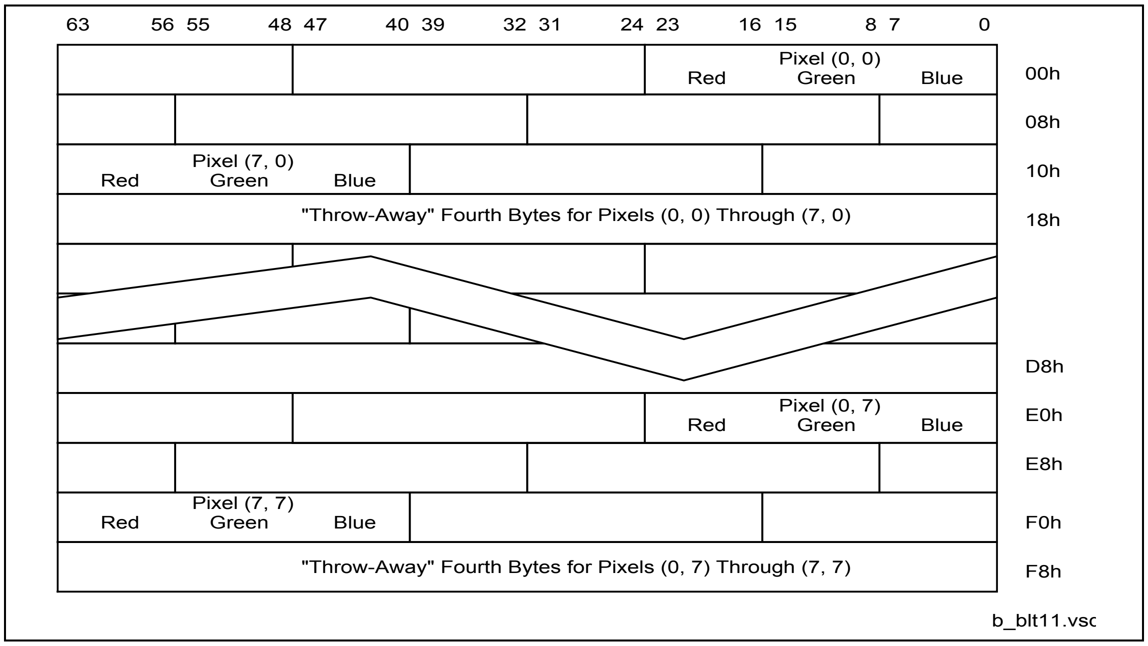
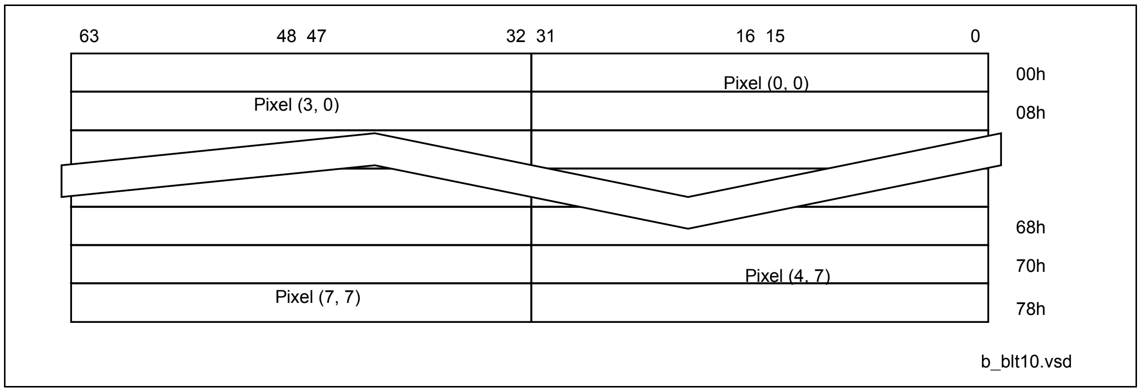

63 56 55 48 47 40 39 32 31 24 23 16 15 8 7 0
Pixel (0, 0) Red Green Blue
Pixel (7, 0) Red Green Blue
"Throw-Away" Fourth Bytes for Pixels (0, 0) Through (7, 0)
Pixel (0, 7) Red Green Blue
Pixel (7, 7) Red Green Blue
"Throw-Away" Fourth Bytes for Pixels (0, 7) Through (7, 7)
00h
08h
10h
18h
D8h
E0h
E8h
F0h
F8h
b_blt11.vsc

Figure 18. 2bpp Pattern Data -- Occupies 256 Bytes (32 quadwords)



As is shown in 24bpp pattern data figure, there are four bytes allocated for each pixel on each scan line's worth of pattern data, which allows each scan line's worth of 24bpp pattern data to begin on a 32-byte boundary. The extra ("fourth") unused bytes of each pixel on a scan line's worth of pattern data are collected together in the last 8 bytes (the last quadword) of each scan line's worth of pattern data.

The opcode of the command packet specifies whether the pattern data is color or monochrome. The various bit-wise logical operations and per-pixel write-masking operations were designed to work with color data. In order to use monochrome pattern data, the BLT engine is designed to convert it into color through a process called "color expansion" which takes place as a BLT operation is performed. In color expansion, the single bits of monochrome pattern data are converted into one, two, three, or four bytes (depending on the color depth to which the BLT engine has been set) of color data that are set to carry values corresponding to either the foreground or background color that have been specified for use in this process. The foreground color is used for pixels corresponding to a bit of monochrome pattern data that carry the value of 1, while the background color is used where the corresponding bit of monochrome pattern data carries the value of 0. The foreground and background colors used in the color expansion of monochrome pattern data can be set in the Pattern Expansion Foreground Color Register and Pattern Expansion Background Color Register.

6.2.5. Destination Data

There are actually two different types of "destination data": the graphics data already residing at the location that is designated as the destination, and the data that is to be written into that very same location as a result of a BLT operation.

The location designated as the destination must be within the frame buffer or Main memory graphics memory where the BLT engine can read from it and write to it directly. The blocks of destination data to be read from and written to the destination may be either contiguous or discontinuous. All data written to the destination will have the color depth to which the BLT engine has been set. It is presumed that any data already existing at the destination which will be read by the BLT engine will also be of this same color depth — the BLT engine neither reads nor writes monochrome destination data.

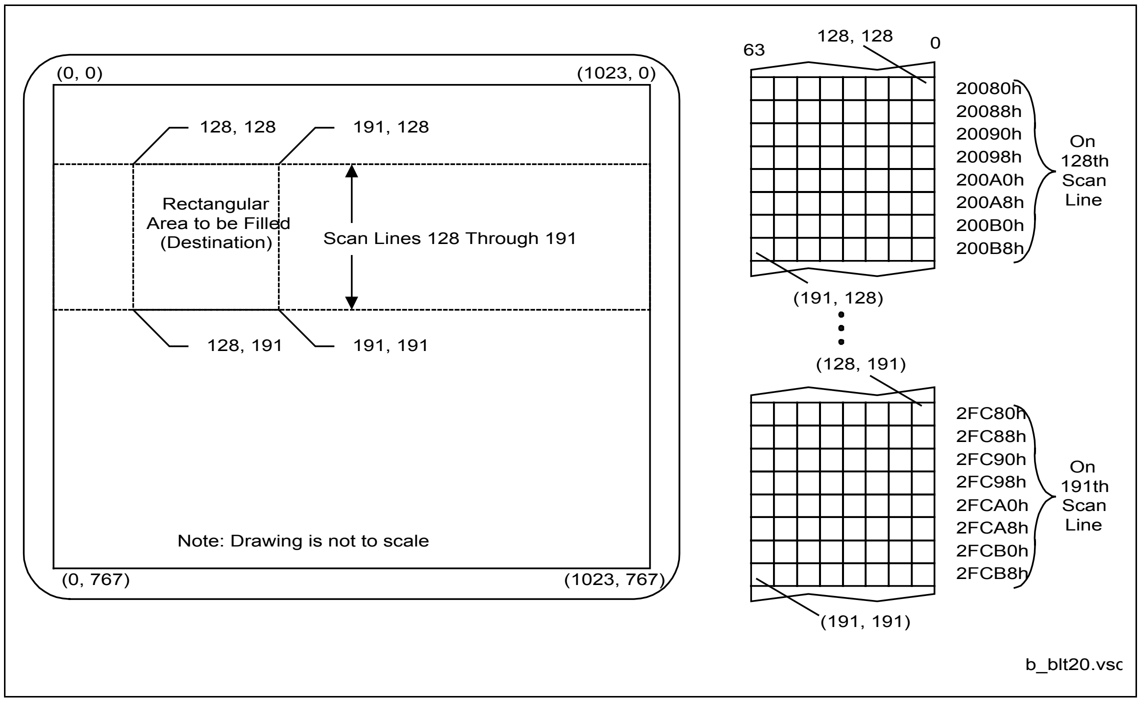
The Destination Address Register is used to specify the address of the destination. To accommodate discontinuous destination data, the Source and Destination Pitch Registers can be used to specify the offset in bytes from the beginning of one scan line's worth of destination data to the next. Otherwise, if the destination data is contiguous, then an offset equal to the length of a scan line's worth of destination data should be specified.

6.3. BLT Programming Examples

6.3.1. Pattern Fill -- A Very Simple BLT

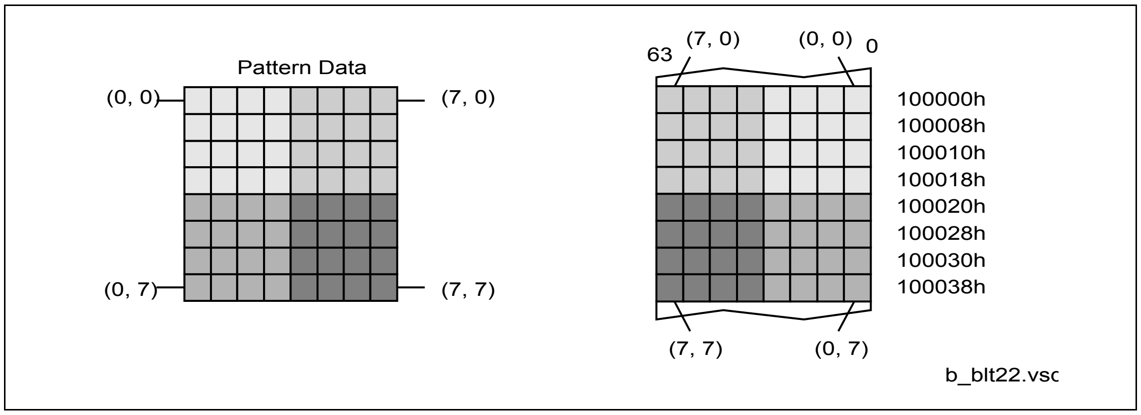
In this example, a rectangular area on the screen is to be filled with a color pattern stored as pattern data in off-screen memory. The screen has a resolution of 1024x768 and the graphics system has been set to a color depth of 8 bits per pixel.

Figure 19. On-Screen Destination for Example Pattern Fill BLT



As shown in the figure above, the rectangular area to be filled has its upper left-hand corner at coordinates (128, 128) and its lower right-hand corner at coordinates (191, 191). These coordinates define a rectangle covering 64 scan lines, each scan line's worth of which is 64 pixels in length — in other words, an array of 64x64 pixels. Presuming that the pixel at coordinates (0, 0) corresponds to the byte at address 00h in the frame buffer memory, the pixel at (128, 128) corresponds to the byte at address 20080h.

Figure 20. Pattern Data for Example Pattern Fill BLT



As shown in figure above, the pattern data occupies 64 bytes starting at address 100000h. As always, the pattern data represents an 8x8 array of pixels.

The BLT command packet is used to select the features to be used in this BLT operation, and must be programmed carefully. The vertical alignment field should be set to 0 to select the top-most horizontal row of the pattern as the starting row used in drawing the pattern starting with the top-most scan line covered by the destination. The pattern data is in color with a color depth of 8 bits per pixel, so the dynamic color enable should be asserted with the dynamic color depth field should be set to 0. Since this BLT operation does not use per-pixel write-masking (destination transparency mode), this field should be set to 0. Finally, the raster operation field should be programmed with the 8-bit value of F0h to select the bit-wise logical operation in which a simple copy of the pattern data to the destination takes place. Selecting this bit-wise operation in which no source data is used as an input causes the BLT engine to automatically forego either reading source data from the frame buffer or waiting for the host processor to provide it.

The Destination Pitch Register must be programmed with number of bytes in the interval from the start of one scan line’s worth of destination data to the next. Since the color depth is 8 bits per pixel and the horizontal resolution of the display is 1024, the value to be programmed into these bits is 400h, which is equal to the decimal value of 1024.

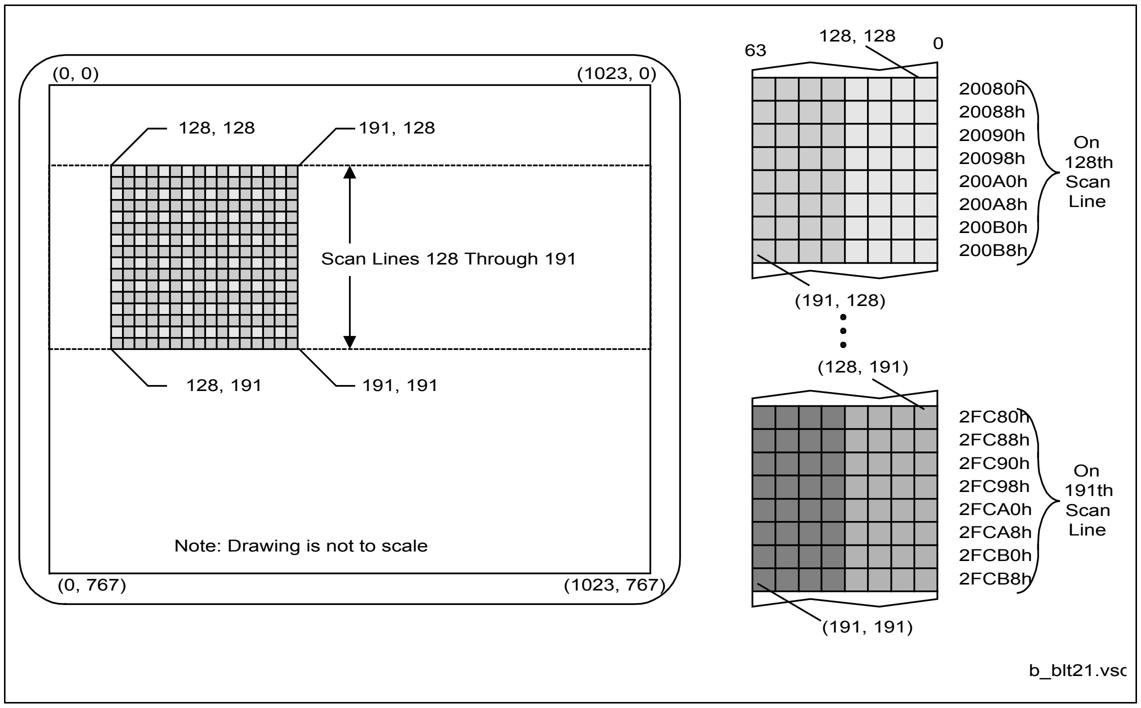
Bits [31:3] of the Pattern Address Register must be programmed with the address of the pattern data.

Similarly, bits [31:0] of the Destination Address Register must be programmed with the byte address at the destination that will be written to first. In this case, the address is 20080h, which corresponds to the byte representing the pixel at coordinates (128, 128).

This BLT operation does not use the values in the Source Address Register or the Source Expansion Background or Foreground Color Registers.

The Destination Width and Height Registers must be programmed with values that describe to the BLT engine the 64x64 pixel size of the destination location. The height should be set to carry the value of 40h, indicating that the destination location covers 64 scan lines. The width should be set to carry the value of 40h, indicating that each scan line’s worth of destination data occupies 64 bytes. All of this information is written to the ring buffer using the PAT_BLT command packet.

Figure 21. Results of Example Pattern Fill BLT

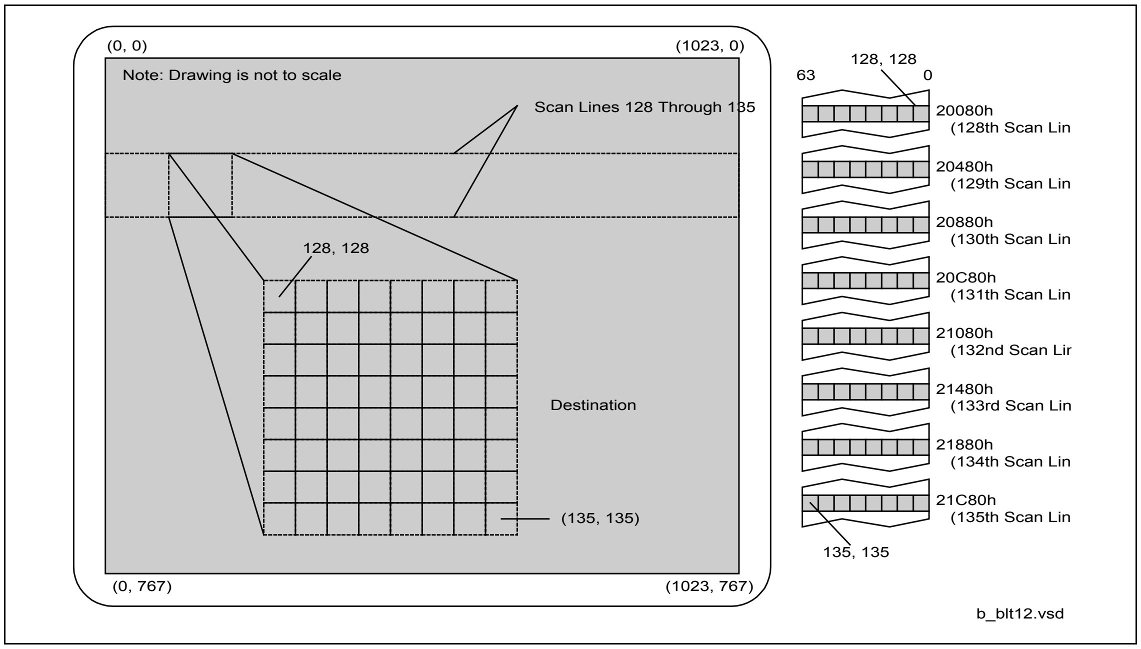


The figure above shows the end result of performing this BLT operation. The 8x8 pattern has been repeatedly copied ("tiled") into the entire 64x64 area at the destination.

6.3.2. Drawing Characters Using a Font Stored in System Memory

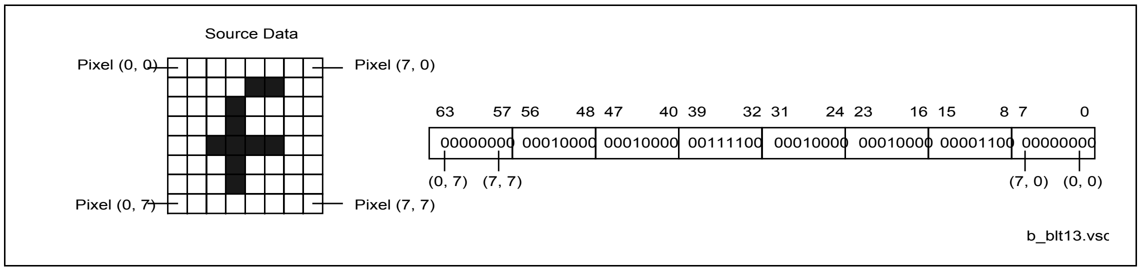
In this example BLT operation, a lowercase letter “f” is to be drawn in black on a display with a gray background. The resolution of the display is 1024x768, and the graphics system has been set to a color depth of 8 bits per pixel.

Figure 22. On-Screen Destination for Example Character Drawing BLT



The figure above shows the display on which this letter “f” is to be drawn. As shown in this figure, the entire display has been filled with a gray color. The letter “f” is to be drawn into an 8x8 region on the display with the upper left-hand corner at the coordinates (128, 128).

Figure 23. Source Data in System Memory for Example Character Drawing BLT



The figure above shows both the 8x8 pattern making up the letter “f” and how it is represented somewhere in the host’s system memory — the actual address in system memory is not important. The letter “f” is represented in system memory by a block of monochrome graphics data that occupies 8 bytes. Each byte carries the 8 bits needed to represent the 8 pixels in each scan line’s worth of this graphics data. This type of pattern is often used to store character fonts in system memory.

During this BLT operation, the host processor will read this representation of the letter “f” from system memory, and write it to the BLT engine by performing memory writes to the ring buffer as an immediate monochrome BLT operand following the BLT_TEXT command. The BLT engine will receive this data through the command stream and use it as the source data for this BLT operation. The BLT engine will be set to the same color depth as the graphics system — 8 bits per pixel, in this case. Since the source data in this BLT operation is monochrome, color expansion must be used to convert it to an 8 bpp color depth. To ensure that the gray background behind this letter “f” is preserved, per-pixel write masking will be performed, using the monochrome source data as the pixel mask.

The BLT Setup and Text command packets are used to select the features to be used in this BLT operation. Only the fields required by these two command packets must be programmed carefully. The BLT engine ignores all other registers and fields. The source select field must be set to 1, to indicate that the source data is provided by the host processor through the IMMEDIATE_BLT command packet. Finally, the raster operation field should be programmed with the 8-bit value CCh to select the bit-wise logical operation that simply copies the source data to the destination. Selecting this bit-wise operation in which no pattern data is used as an input, causes the BLT engine to automatically forego reading pattern data from the frame buffer.

The Setup Pattern/Source Expansion Foreground Color Register to specify the color with which the letter “f” will be drawn. There is no Source address. All scan lines of the glyph are bit packed and the clipping is controlled by the ClipRect registers from the SETUP_BLT command and the Destination Y1, Y2, X1, and X2 registers in the TEXT_BLT command. Only the pixels that are within (inclusive comparisons) the clip rectangle are written to the destination surface.

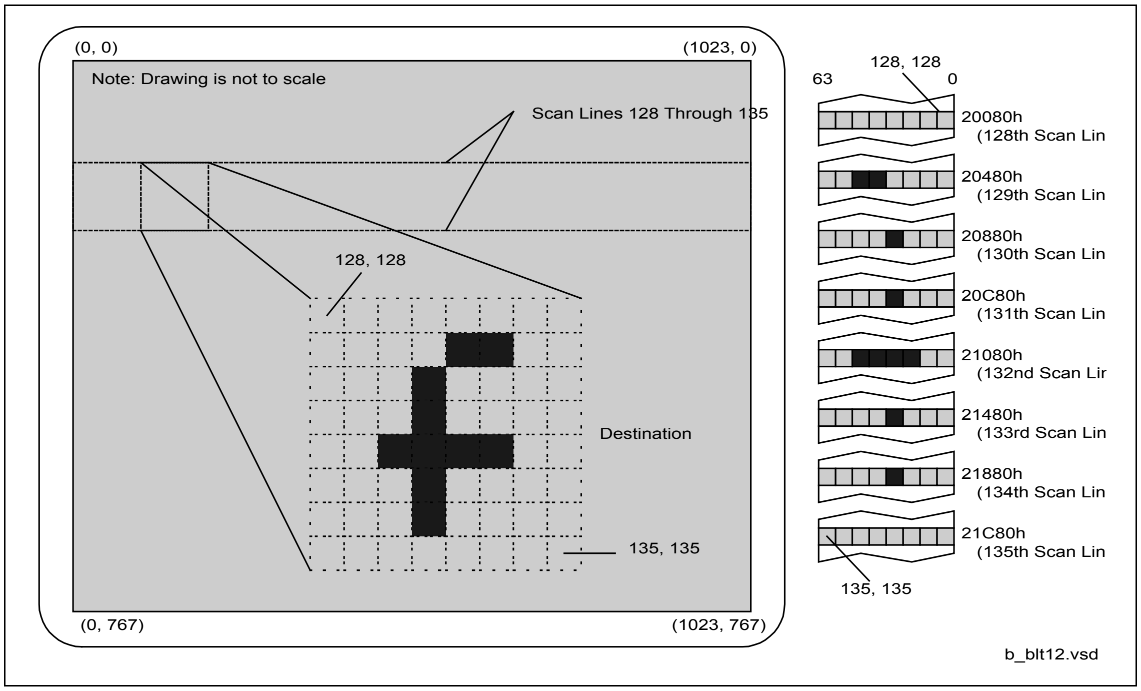
The Destination Pitch Register must be programmed with a value equal to the number of bytes in the interval between the first bytes of each adjacent scan line’s worth of destination data. Since the color depth is 8 bits per pixel and the horizontal resolution of the display is 1024 pixels, the value to be programmed into these bits is 400h, which is equal to the decimal value of 1024. Since the source data used in this BLT operation is monochrome, the BLT engine will not use a byte-oriented pitch value for the source data.

Since the source data is monochrome, color expansion is required to convert it to color with a color depth of 8 bits per pixel. Since the Setup Pattern/Source Expansion Foreground Color Register is selected to specify the foreground color of black to be used in drawing the letter “f”, this register must be programmed with the value for that color. With the graphics system set for a color depth of 8 bits per pixel, the actual colors are specified in the RAMDAC palette, and the 8 bits stored in the frame buffer for each pixel actually specify the index used to select a color from that palette. This example assumes that the color specified at index 00h in the palette is black, and therefore bits [7:0] of this register should be set to 00h to select black as the foreground color. The BLT engine ignores bits [23:8] of this register because the selected color depth is 8 bits per pixel. Even though the color expansion being performed on the source data normally requires that both the foreground and background colors be specified, the value used to specify the background color is not important in this example. Per-pixel write-masking is being performed with the monochrome source data as the pixel mask, which means that none of the pixels in the source data that will be converted to the background color will ever be written to the destination. Since these pixels will never be seen, the value programmed into the Pattern/Source Expansion Background Color Register to specify a background color is not important.

The Destination Width and Height Registers are not used. The Y1, Y2, X1, and X2 are used be program with values that describe to the BLT engine the 8x8 pixel size of the destination location. The Destination Y1 and Y2 address registers must be programmed with the starting and ending scan line address of the destination data. This address is specified as an offset from the start of the frame buffer of the scan line at the destination that will be written to first. The destination X1 and X2 registers must be programmed with the starting and ending pixel offsets from the beginning of the scan line.

This BLT operation does not use the values in the Pattern Address Register, the Source Expansion Background Color Register, or the Source Expansion Foreground Color Register.

Figure 24. Results of Example Character Drawing BLT



The above figure shows the end result of performing this BLT operation. Only the pixels that form part of the actual letter “f” have been drawn into the 8x8 destination location on the display, leaving the other pixels within the destination with their original gray color.

7. Initialization Registers

To function, all registers described in this section must be programmed for the Intel® 815 chipset family of products. The default states of these registers, with the exception of registers that deal with extended modes or performance enhancements, will prevent the Intel® 815 chipset family products from booting.

Note: The registers in this document are normally programmed by the video BIOS.
These registers also may be documented in other sections of this document.

7.1. Standard VGA Registers

All VGA registers are in standard locations and initialized by means of standard procedures. This section will document all nonstandard registers that are needed for initialization of the Intel® 815 chipset.

7.2. SMRAM Registers

The SMRAM register is in the chipset's PCI configuration space (Device 0). Since this register is not documented anywhere else, the entire register description is provided here.

7.2.1. SMRAM—System Management RAM Control Register (Device 0)

Address Offset:	70h
Default Value:	00h
Access:	Read/Write, Read Only
Size:	8 bits

The SMRAM register controls how accesses to Compatible and Extended SMRAM spaces are treated, and how much (if any) memory is "Stolen" from the System to support both SMRAM and Graphics Local Memory needs.

7 6	5 4	3 2	1	0
Graphics Mode Select	Upper SMM Select	Lower SMM Select	SMM Space Locked	E_SMRAM_ERR

Bit	Description
7:6	**Graphics Mode Select (GMS).** This field is used to enable/disable the Internal Graphics device and select the amount of Main Memory that is "Stolen" to support the Internal Graphics device in VGA (non-linear) mode only. These 2 bits only have meaning if we are not in AGP mode. 00 = Internal Graphics Device Disabled, No memory "Stolen" 01 = Internal Graphics Device Enabled, No memory "Stolen" 10 = Internal Graphics Device Enabled, 512K of memory "Stolen" for frame buffer. 11 = Internal Graphics Device Enabled, 1M of memory "Stolen" for frame buffer. **Note:** When the Internal Graphics Device is Disabled (00) the Graphics Device and all of its memory and I/O functions are disabled and the clocks to this logic are turned off, memory accesses to the VGA range (A0000-BFFFF) will be forwarded on to the hub interface, and the Graphics Local Memory space is NOT "stolen" from main memory. Any change to the SMRAM register will not affect AGP mode or cause the controller to go into AGP mode. When this field is non-0 the Internal Graphics Device and all of its memory and I/O functions are enabled, all non-SMM memory accesses to the VGA range will be handled internally and the selected amount of Graphics Local Memory space (0, 512K or 1M) is "stolen" from the main memory. Graphics Memory is "stolen" AFTER TSEG Memory is "stolen". Once D_LCK is set, these bits becomes read only. GMCH does not support VGA on local memory. Software must not use the 01 mode for VGA
5:4	**Upper SMM Select (USMM).** This field is used to enable/disable the various SMM memory ranges above 1 MB. TSEG is a block of memory ("Stolen" from Main Memory at [TOM-Size] : [TOM]) that is only accessable by the processor and only while operating in SMM mode. HSEG is a Remap of the AB segment at FEEA0000 : FEEBFFFF. Both of these areas, when enabled, are usable as SMM RAM. 00 = TSEG and HSEG are both Disabled 01 = TSEG is Disabled, HSEG is Conditionally Enabled 10 = TSEG is Enabled as 512 KB and HSEG is Conditionally Enabled 11 = TSEG is Enabled as 1 MB and HSEG is Conditionally Enabled **Note:** Non-SMM Operations (SMM processor accesses and all other access) that use these address ranges are forwarded to the hub interface. Once D_LCK is set, these bits becomes read only. HSEG is ONLY enabled if LSMM = 00.
3:2	**Lower SMM Select (LSMM).** This field controls the definition of the A&B segment SMM space 00 = AB segment Disabled (no one can write to it). 01 = AB segment Enabled as General System RAM (anyone can write to it). 10 = AB segment Enabled as SMM Code RAM Shadow. Only SMM Code Reads can access DRAM in the AB segment (processor code reads only). SMM Data operations and all Non-SMM Operations go to either the internal graphics device or are broadcast on the hub interface. 11 = AB segment Enabled as SMM RAM. All SMM operations to the AB segment are serviced by DRAM, all Non-SMM Operations go to either the internal Graphics Device or are broadcast on the hub interface (processor SMM R/W can access SMM space). When D_LCK is set bit 3 becomes Read Only, and bit 2 is Writable ONLY if bit 3 is a "1". When bit 3 is set only the processor can access it.

Bit	Description
1	**SMM Space Locked (D_LCK).** When D_LCK is set to 1 then D_LCK, GMS, USMM, and the most significant bit of LSMM become read only. D_LCK can be set to 1 via a normal configuration space write but can only be cleared by a reset. The combination of D_LCK and LSMM provide convenience with security. The BIOS can use LSMM=01 to initialize SMM space and then use D_LCK to "lock down" SMM space in the future so that no application software (or BIOS itself) can violate the integrity of SMM space, even if the program has knowledge of the LSMM function. This bit also Locks the DRP and DRP2 registers.
0	**E_SMRAM_ERR (E_SMERR).** This bit is set when processor accesses the defined memory ranges in Extended SMRAM (HSEG or TSEG) while not in SMM mode. It is software's responsibility to clear this bit. The software must write a 1 to this bit to clear it This bit is Not set for the case of an Explicit Write Back operation.

Initialization and Usage of "Stolen" Memory

SMRAM Register Bits 7:4 control the theft of memory from Main Memory space for use as Graphics Local Memory and SMM TSEG memory. The blocks of memory selected by these fields are NOT accessible as general system RAM. When Bit 5 of the SMRAM register is a "1" the TSEG segment of memory can ONLY be accessed by the processor in SMM mode (No other agent can access this memory). Therefore, BIOS should initialize this block of memory BEFORE setting either Bit 5 or Bit 7 of the SMRAM register. The memory for TSEG is "Stolen" first and then the Graphics Local Memory is "Stolen". An example of this theft mechanism is:

- TOM equal 64 MB,
- TSEG selected as 512 KB in size,
- Graphics Local Memory selected as 1 MB in size
- General System RAM available in system = 62.5 MB

— General System RAM Range	00000000h to 03E7FFFFh
— TSEG Address Range	03F80000h to 03FFFFFFh
— TSEG "Stolen" from	03F80000h to 03FFFFFFh
— Graphics Local Memory "Stolen" from	03E80000h to 03F7FFFFh

7.3. Display, I/O, GPIO, Clock, LCD, and Pixel Pipeline Registers

These registers are described elsewhere in this document. Refer to the appropriate sections of this PRM for detailed bit/field descriptions.

Address Offset	Symbol	Register Name	Access
Instruction and Interrupt Control Registers			
020D8h–020DBh	FW_BLC	FIFO Watermark and Burst Length Control	R/W
I/O Control Registers			
05000h–05003h	HVSYNC	HSYNC/VSYNC Control	R/W
05010h–05013h	GPIOA	General Purpose I/O Control A	R/W
05014h–05017h	GPIOB	General Purpose I/O Control B	R/W
Clock Control and Power Management Registers			
06000h–06003h	DCLK_0D	Display Clock 0 Divisor	R/W
06004h–06007h	DCLK_1D	Display Clock 1 Divisor	R/W
06008h–0600Bh	DCLK_2D	Display Clock 2 Divisor	R/W
0600Ch–0600Fh	LCD_CLKD	LCD Clock Divisor	R/W
06010h–06013h	DCLK_0DS	Display and LCD Clock Divisor Select	R/W
06014h–06017h	PWR_CLKC	Power Management and Miscellaneous Clock Control	R/W
LCD/TV-Out			
60000h–60003h	HTOTAL	Horizontal Total	R/W
60004h–60007h	HBLANK	Horizontal Blank	R/W
60008h–6000Bh	HSYNC	Horizontal Sync	R/W
6000Ch–6000Fh	VTOTAL	Vertical Total	R/W
60010h–60013h	VBLANK	Vertical Blank	R/W
60014h–60017h	VSYNC	Vertical Sync	R/W
60018h–6001Bh	LCDTV_C	LCD / TV-Out Control	R/W
6001Ch–6001Fh	OVRACT	Overlay Active Register	R/W
60020h–60023h	BCLRPAT	Border Color Pattern	R/W
Display and Cursor Control Registers			
70008h–7000Bh	PIXCONF	Pixel Pipeline Configuration	R/W

7.4. 2D Graphics Controller Registers (3CEh / 3CFh)

Refer to Chapter 9, "VGA and Extended VGA Registers" for detailed bit/field descriptions.

Index	Sym	Register Name
10h	GR10	Address Mapping
11h	GR11	Page Selector

7.5. 2D CRT Controller Registers (3B4h/3D4h/3B5h/3D5h)

Refer to Chapter 9, "VGA and Extended VGA Registers" for detailed bit/field descriptions.

Index	Sym	Register Name
30h	CR30	Extended Vertical Total
31h	CR31	Extended Vertical Display End
32h	CR32	Extended Vertical Sync Start
33h	CR33	Extended Vertical Blanking Start
35h	CR35	Extended Horizontal Total Time
39h	CR39	Extended Horizontal Blank Time
40h	CR40	Extended Start Address
41h	CR41	Extended Offset
42h	CR42	Extended Start Address High
70h	CR70	Interlace Control
80h	CR80	I/O Control
82h	CR82	Blink Rate Control

7.6. Initialization Values for VGA Registers

Mode ->	0	0*	0+	1	1*	1+	2	2*	2+	3	3*	3+	7	7+	132 col	132 col
Register															Opt 1	Opt 2
MSR	63h	A3h	67h	63h	A3h	67h	63h	A3h	67h	63h	A3h	67h	A6h	66h	6Bh	6Bh
CR00	2Dh	2Dh	2Dh	2Dh	2Dh	2Dh	5Fh	5Fh	5Fh	5Fh	5Fh	5Fh	5Fh	5Fh	A0h	9Eh
CR01	27h	27h	27h	27h	27h	27h	4Fh	4Fh	4Fh	4Fh	4Fh	4Fh	4Fh	4Fh	83h	83h
CR02	28h	28h	28h	28h	28h	28h	50h	50h	50h	50h	50h	50h	50h	50h	85h	84h
CR03	90h	90h	90h	90h	90h	90h	82h	82h	82h	82h	82h	82h	82h	82h	82h	81h
CR04	2Bh	2Bh	2Bh	2Bh	2Bh	2Bh	55h	55h	55h	55h	55h	55h	55h	55h	8Ah	8Ah
CR05	A0h	A0h	A0h	A0h	A0h	A0h	81h	81h	81h	81h	81h	81h	81h	81h	81h	9Eh
CR06	BFh	BFh	BFh	BFh	BFh	BFh	BFh	BFh	BFh	BFh	BFh	BFh	BFh	BFh	BFh	BFh
CR07	1Fh	1Fh	1Fh	1Fh	1Fh	1Fh	1Fh	1Fh	1Fh	1Fh	1Fh	1Fh	1Fh	1Fh	1Fh	1Fh
CR08	00h	00h	00h	00h	00h	00h	00h	00h	00h	00h	00h	00h	00h	00h	00h	00h
CR09	C7h	4Dh	4Fh	C7h	4Dh	4Fh	C7h	4Dh	4Fh	C7h	4Dh	4Fh	4Dh	4Fh	4Fh	4Fh
CR0A	06h	0Bh	0Dh	06h	0Bh	0Dh	06h	0Bh	0Dh	06h	0Bh	0Dh	0Bh	0Dh	0Dh	0Eh
CR0B	07h	0Ch	0Eh	07h	0Ch	0Eh	07h	0Ch	0Eh	07h	0Ch	0Eh	0Ch	0Eh	0Eh	0Fh
CR0C	00h	00h	00h	00h	00h	00h	00h	00h	00h	00h	00h	00h	00h	00h	00h	00h
CR0D	00h	00h	00h	00h	00h	00h	00h	00h	00h	00h	00h	00h	00h	00h	00h	00h
CR0E	00h	00h	00h	00h	00h	00h	00h	00h	00h	00h	00h	00h	00h	00h	00h	00h
CR0F	00h	00h	00h	00h	00h	00h	00h	00h	00h	00h	00h	00h	00h	00h	00h	00h
CR10	9Ch	83h	9Ch	9Ch	83h	9Ch	9Ch	83h	9Ch	9Ch	83h	9Ch	83h	9Ch	9Ch	9Ch
CR11	8Eh	85h	8Eh	8Eh	85h	8Eh	8Eh	85h	8Eh	8Eh	85h	8Eh	85h	8Eh	8Eh	8Eh
CR12	8Fh	5Dh	8Fh	8Fh	5Dh	8Fh	8Fh	5Dh	8Fh	8Fh	5Dh	8Fh	5Dh	8Fh	8Fh	8Fh
CR13	14h	14h	14h	14h	14h	14h	28h	28h	28h	28h	28h	28h	28h	28h	42h	42h
CR14	1Fh	1Fh	1Fh	1Fh	1Fh	1Fh	1Fh	1Fh	1Fh	1Fh	1Fh	1Fh	0Dh	0Fh	1Fh	1Fh
CR15	96h	63h	96h	96h	63h	96h	96h	63h	96h	96h	63h	96h	63h	96h	96h	96h
CR16	B9h	BAh	B9h	B9h	BAh	B9h	B9h	BAh	B9h	B9h	BAh	B9h	BAh	B9h	B9h	B9h
CR17	A3h	A3h	A3h	A3h	A3h	A3h	A3h	A3h	A3h	A3h	A3h	A3h	A3h	A3h	A3h	A3h
CR18	FFh	FFh	FFh	FFh	FFh	FFh	FFh	FFh	FFh	FFh	FFh	FFh	FFh	FFh	FFh	FFh
SR00	00h	00h	00h	00h	00h	00h	00h	00h	00h	00h	00h	00h	00h	00h	00h	00h
SR01	09h	09h	08h	09h	09h	08h	01h	01h	00h	01h	01h	00h	00h	00h	01h	01h
SR02	03h	03h	03h	03h	03h	03h	03h	03h	03h	03h	03h	03h	03h	03h	03h	03h
SR03	00h	00h	00h	00h	00h	00h	00h	00h	00h	00h	00h	00h	00h	00h	00h	00h
SR04	02h	02h	02h	02h	02h	02h	02h	02h	02h	02h	02h	02h	03h	02h	02h	02h
GR00	00h	00h	00h	00h	00h	00h	00h	00h	00h	00h	00h	00h	00h	00h	00h	00h

Mode ->	0	0*	0+	1	1*	1+	2	2*	2+	3	3*	3+	7	7+	132 col	132 col
Register															Opt 1	Opt 2
GR01	00h	00h	00h	00h	00h	00h	00h	00h	00h	00h	00h	00h	00h	00h	00h	00h
GR02	00h	00h	00h	00h	00h	00h	00h	00h	00h	00h	00h	00h	00h	00h	00h	00h
GR03	00h	00h	00h	00h	00h	00h	00h	00h	00h	00h	00h	00h	00h	00h	00h	00h
GR04	00h	00h	00h	00h	00h	00h	00h	00h	00h	00h	00h	00h	00h	00h	00h	00h
GR05	10h	10h	10h	10h	10h	10h	10h	10h	10h	10h	10h	10h	10h	10h	10h	10h
GR06	0Eh	0Eh	0Eh	0Eh	0Eh	0Eh	0Eh	0Eh	0Eh	0Eh	0Eh	0Eh	0Ah	0Ah	0Eh	0Eh
GR07	00h	00h	00h	00h	00h	00h	00h	00h	00h	00h	00h	00h	00h	00h	00h	00h
GR08	FFh	FFh	FFh	FFh	FFh	FFh	FFh	FFh	FFh	FFh	FFh	FFh	FFh	FFh	FFh	FFh
AR00	00h	00h	00h	00h	00h	00h	00h	00h	00h	00h	00h	00h	00h	00h	00h	
AR01	01h	01h	01h	01h	01h	01h	01h	01h	01h	01h	01h	01h	08h	08h	01h	
AR02	02h	02h	02h	02h	02h	02h	02h	02h	02h	02h	02h	02h	08h	08h	02h	
AR03	03h	03h	03h	03h	03h	03h	03h	03h	03h	03h	03h	03h	08h	08h	03h	
AR04	04h	04h	04h	04h	04h	04h	04h	04h	04h	04h	04h	04h	08h	08h	04h	
AR05	05h	05h	05h	05h	05h	05h	05h	05h	05h	05h	05h	05h	08h	08h	05h	
AR06	06h	14h	14h	06h	14h	14h	06h	14h	14h	06h	14h	14h	08h	08h	14h	
AR07	07h	07h	07h	07h	07h	07h	07h	07h	07h	07h	07h	07h	08h	08h	07h	
AR08	10h	38h	38h	10h	38h	38h	10h	38h	38h	10h	38h	38h	10h	10h	38h	
AR09	11h	39h	39h	11h	39h	39h	11h	39h	39h	11h	39h	39h	18h	18h	39h	
AR0A	12h	3Ah	3Ah	12h	3Ah	3Ah	12h	3Ah	3Ah	12h	3Ah	3Ah	18h	18h	3Ah	
AR0B	13h	3Bh	3Bh	13h	3Bh	3Bh	13h	3Bh	3Bh	13h	3Bh	3Bh	18h	18h	3Bh	
AR0C	14h	3Ch	3Ch	14h	3Ch	3Ch	14h	3Ch	3Ch	14h	3Ch	3Ch	18h	18h	3Ch	
AR0D	15h	3Dh	3Dh	15h	3Dh	3Dh	15h	3Dh	3Dh	15h	3Dh	3Dh	18h	18h	3Dh	
AR0E	16h	3Eh	3Eh	16h	3Eh	3Eh	16h	3Eh	3Eh	16h	3Eh	3Eh	18h	18h	3Eh	
AR0F	17h	3Fh	3Fh	17h	3Fh	3Fh	17h	3Fh	3Fh	17h	3Fh	3Fh	18h	18h	3Fh	
AR10	08h	08h	0Ch	08h	08h	0Ch	08h	08h	0Ch	08h	08h	0Ch	0Eh	0Eh	0Ch	0Ch
AR11	00h	00h	00h	00h	00h	00h	00h	00h	00h	00h	00h	00h	00h	00h	00h	00h
AR12	0Fh	0Fh	0Fh	0Fh	0Fh	0Fh	0Fh	0Fh	0Fh	0Fh	0Fh	0Fh	0Fh	0Fh	0Fh	0Fh
AR13	00h	00h	00h	00h	00h	00h	00h	00h	00h	00h	00h	00h	08h	00h	00h	00h
AR14	00h	00h	08h	00h	00h	08h	00h	00h	08h	00h	00h	08h	00h	08h		00h

This page is intentionally left blank.

8. Frame Buffer Access

The VGA frame buffer is located at A000h-BFFFh. This is the standard VGA frame buffer address.

The physical location of the frame buffer is at the top of main memory. The size can either be 512 KB or 1 MB. This is selected in the SMRAM register, which is documented in the *Initialization Registers* section of this document.

The frame buffer is not stored in local memory, but it is taken from the top of main memory, as described in the SMRAM register description.

This page is intentionally left blank.

9. VGA and Extended VGA Registers

This chapter describes the registers and the functional operation notations for the observable registers in the 2D section. Each register is documented and the various bit settings defined. It is important to note that not all combinations of bit settings result in functional operating modes. Note that these registers can be accessed via either I/O space or memory space. The memory space addresses listed are offsets from the base memory address programmed into the MMAPA register (PCI configuration offset 14h). For each register, the memory mapped address offset is the same address value as the I/O address.

9.1. General Control & Status Registers

The setup, enable and general registers are all directly accessible by the processor. A sub indexing scheme is not used to read from and write to these registers.

Name	Function	Read		Write	
		I/O	Memory Offset	I/O	Memory Offset
ST00	VGA Input Status Register 0	3C2h	3C2h	—	—
ST01	VGA Input Status Register 1	3BAh/3DAh[1]	3BAh/3DAh[1]	—	—
FCR	VGA Feature Control Register	3CAh	3CAh	3BAh/3DAh[1]	3BAh/3DAh[1]
MSR	VGA Miscellaneous Output Register	3CCh	3CCh	3C2h	3C2h

NOTES:

1. The address selection for ST01 reads and FCR writes is dependent on CGA or MDA emulation mode as selected via the MSR register.

Various bits in these registers provide control over and the real-time status of the horizontal sync signal, the horizontal retrace interval, the vertical sync signal, and the vertical retrace interval.

The horizontal retrace interval is the period during the drawing of each scan line containing active video data, when the active video data is not being displayed. This period includes the horizontal front and back porches, and the horizontal sync pulse. The horizontal retrace interval is always longer than the horizontal sync pulse.

The vertical retrace interval is the period during which the scan lines not containing active video data are drawn. It is the period that includes the vertical front and back porches, and the vertical sync pulse. The vertical retrace interval is always longer than the vertical sync pulse.

Display Enable is a status bit (bit 0) in VGA Input Status Register 1 that indicates when either a horizontal retrace interval or a vertical retrace interval is taking place. In the IBM* EGA graphics system (and the ones that preceded it, including MDA and CGA), it was important to check the status of this bit to ensure that one or the other retrace intervals was taking place before reading from or writing to the frame buffer. In these earlier systems, reading from or writing to frame buffer at times outside the retrace intervals meant that the CRT controller would be denied access to the frame buffer in while accessing pixel data needed to draw pixels on the display. This resulted in either “snow” or a flickering display. The term “Display Enable” is an inaccurate description for this status bit, since the name suggests a connection to the enabling or disabling the graphics system.

9.1.1. ST00—Input Status 0

I/O (and Memory Offset) Address:	3C2h
Default:	00h
Attributes:	Read Only

7	6 5	4	3 0
CRT Int	Reserved (00)	RGB Cmp / Sen	Reserved (0000)

Bit	Descriptions
7	**CRT Interrupt Pending.** Note that the generation of interrupts can be enabled, through bits [4,5] of the Vertical Retrace End Register (CR11). This ability to generate interrupts at the start of the vertical retrace interval is a feature that is typically unused by current software. This bit is here for EGA compatibility. 0 = CRT (vertical retrace interval) interrupt is not pending. 1 = CRT (vertical retrace interval) interrupt is pending
6:5	**Reserved.** Read as 0s.
4	**RGB Comparator / Sense.** This bit returns the state of the output of the RGB output comparator(s). BIOS uses this bit to determine whether the display is a color or monochrome CRT. 0 = Monochrome 1 = Color BIOS blanks the screen or clears the frame buffer to display only black. Next, BIOS configures the D-to-A converters and the comparators to test for the presence of a color display. Finally, if the BIOS does not detect any colors, it tests for the presence of a display. The result of each such test is read via this bit.
3:0	**Reserved.** Read as 0s.

9.1.2. ST01—Input Status 1

I/O (and Memory Offset) Address:	3BAh/3DAh
Default:	00h
Attributes:	Read Only

The address selection is dependent on CGA or MDA emulation mode as selected via the MSR register.

7	6	5 4	3	2 1	0
Reserved (0)	Reserved (0)	Video Feedback	Vertical Retrace	Reserved (00)	Display Enable

Bit	Descriptions
7	**Reserved (as per VGA specification).** Read as 0s.
6	**Reserved.** Read as 0.
5:4	**Video Feedback 1, 0.** These are diagnostic video bits that are selected by the Color Plane Enable Register. These bits that are programmably connected to 2 of the 8 color bits sent to the palette. Bits 4 and 5 of the Color Plane Enable Register (AR12) selects which two of the 8 possible color bits become connected to these 2 bits of this register. The current software normally does not use these 2 bits. They exist for EGA compatibility.
3	**Vertical Retrace/Video.** 0 = VSYNC inactive (Indicates that a vertical retrace interval is not taking place). 1 = VSYNC active (Indicates that a vertical retrace interval is taking place). **Note:** Bits 4 and 5 of the Vertical Retrace End Register (CR11) can program this bit to generate an interrupt at the start of the vertical retrace interval. This ability to generate interrupts at the start of the vertical retrace interval is a feature that is largely unused by current software.
2:1	**Reserved.** Read as 0s.
0	**Display Enable Output.** 0 = DE inactive. Active display area data is being drawn on the display. Neither a horizontal retrace interval or a vertical retrace interval is currently taking place. 1 = DE active. Either a horizontal retrace interval or a vertical retrace interval is currently taking place.

9.1.3. FCR—Feature Control

I/O (and Memory Offset) Address:	3BAh/3DAh— Write; 3CAh— Read
Default:	00h
Attributes:	See Address above

The address selection for reads is dependent on CGA or MDA emulation mode as selected via the MSR register.

7 4	3	2 0
Reserved (0000)	VSYNC Control	Reserved (000)

Bit	Descriptions
7:4	**Reserved.** Read as 0.
3	**VSYNC Control.** 0 = Vsync output on the VSYNC pin (default). 1 = Logical 'OR' of VSync and Display Enable output on the VSYNC pin. This capability is provided for IBM compatibility.
2:0	**Reserved.** Read as 0.

9.1.4. MSR—Miscellaneous Output

I/O (and Memory Offset) Address:	3C2h — Write;	3CCh— Read
Default:	00h	
Attributes:	See Address above	

7	6	5	4	3	2	1	0
VSYNC Polarity	HSYNC Polarity	Page Select	Reserved (0)	Clock Select		A0000–BFFFFh Acc En	I/O Address

Bit	Descriptions
7	**CRT VSync Polarity.** 0 = Positive Polarity (default). 1 = Negative Polarity.
6	**CRT HSync Polarity.** 0 = Positive Polarity (default). 1 = Negative Polarity
5	**Page Select.** In Odd/Even Memory Map Mode 1 (GR6), this bit selects the upper or lower 64 KB page in display memory for processor access: 0 = Upper page (default) 1 = Lower page. Selects between two 64 KB pages of frame buffer memory during standard VGA odd/even modes (modes 0h through 5h). Bit 1 of register GR06 can also program this bit in other modes. Note that this bit is always set to 1 by the driver software.
4	**Reserved.** Read as 0.
3:2	**Clock Select.** These bits usually select the dot clock source for the CRT interface. The bits select the dot clock in standard VGA modes. 00 = CLK0, 25 MHz (for standard VGA modes with 640 pixel horizontal resolution) (default) 01 = CLK1, 28 MHz. (for standard VGA modes with 720 pixel horizontal resolution) 1x = CLK2 (left “reserved” in standard VGA, used for all extended modes 6 MHz–135 MHz)
1	**A0000–BFFFFh Access Enable.** VGA Compatibility bit enables access to local video memory (frame buffer) at A0000–BFFFFh. When disabled, accesses to system memory are blocked in this region (by not asserting DEVSEL#). This bit does not block processor access to the video linear frame buffer at other addresses. 0 = Prevent processor access to frame buffer (default). 1 = Allow processor access to frame buffer.
0	**I/O Address Select.** This bit selects 3Bxh or 3Dxh as the I/O address for the CRT Controller registers, the Feature Control Register (FCR), and Input Status Register 1 (ST01). Presently ignored (whole range is claimed), but will “ignore” 3Bx for color configuration or 3Dx for monochrome. 0 = Select 3Bxh I/O address (MDA emulation) (default). 1 = Select 3Dxh I/O address (CGA emulation).

NOTES:

1. In standard VGA modes, bits 7 and 6 indicate which of the three standard VGA vertical resolutions the standard VGA display should use. All extended modes, including those with a vertical resolution of 480 scan lines, use a setting of 0 for both of these bits. This setting was “reserved” in the VGA standard.

Table 8. CRT Display Sync Polarities

V	H	Display	Horizontal Frequency	Vertical Frequency
P	P	>480 Line	Variable	Variable
P	P	200 Line	15.7 KHz	60 Hz
N	P	350 Line	21.8 KHz	60 Hz
P	N	400 Line	31.5 KHz	70 Hz
N	N	480 Line	31.5 KHz	60 Hz

9.2. Sequencer Registers

The sequencer registers are accessed via either I/O space or Memory space. To access the registers the VGA Sequencer Index register (SRX) at I/O address 3C4h (or memory address 3C4h) is written with the index of the desired register. Then the desired register is accessed through the data port for the sequencer registers at I/O address 3C5 (or memory address 3C5).

9.2.1. SRX—Sequencer Index

I/O (and Memory Offset) Address:	3C4h
Default:	00h
Attributes:	Read/Write

7 3	2 0
Reserved (00000)	Sequencer Index

Bit	Descriptions
7:3	**Reserved.** Read as 0s.
2:0	**Sequencer Index.** This field contains a 3-bit Sequencer Index value used to access sequencer data registers at indices 0 through 7. **Notes:** 1. SR02 is referred to in the VGA standard as the Map Mask Register. However, the word “map” is used with multiple meanings in the VGA standard and was, therefore, deemed too confusing; hence, the reason for calling it the Plane Mask Register. 2. SR07 is a standard VGA register that was not documented by IBM. It is not an graphics controller extension.

9.2.2. SR00—Sequencer Reset

I/O (and Memory Offset) Address:	3C5h(Index=00h)
Default:	00h
Attributes:	Read/Write

7 … 2	1	0
Reserved (000000)	Reserved (scratch bit)	Reserved (scratch bit)

Bit	Descriptions
7:2	**Reserved**. Read as 000000. Write has no effect.
1	**Read/Write scratch bit required for VGA compatibility**. Read previously written value. Write stores written value. This bit is a fully readable/writeable MMIO location in the hardware. It has no other functionality in the hardware.
0	**Read/Write scratch bit required for VGA compatibility**. Read previously written value. Write stores written value. This bit is a fully readable/writeable MMIO location in the hardware. It has no other functionality in the hardware.

Programming Hints :

- It has been noted that legacy Video BIOS code has the bits [1:0] programmed to “11” values for reason not fully understood. Experiment was done on 810 Video BIOS with leaving these 2 bits at the default values. It was found that full screen DOS box mode does not behave properly in that case. However, with Video BIOS programming up the bits to “11” values, the problem was corrected.

9.2.3. SR01—Clocking Mode

I/O (and Memory Offset) Address:	3C5h (Index=01h)
Default:	00h
Attributes:	Read/Write

7 6	5	4	3	2	1	0
Reserved (00)	Screen Off	Shift 4	Dot Clock Divide	Shift Load	Reserved (0)	8/9 Dot Clocks

Bit	Descriptions
7:6	**Reserved.** Read as 0s.
5	**Screen Off.** The display and hardware cursor will be disabled by setting the Screen Off bit. However, the Overlay stream will continue to be displayed so it must be halted separately if you want to blank the screen. 0 = Normal Operation (default). 1 = Disables video output (blanks the screen) and turns off the picture-generating logic. This allows the full memory bandwidth to be available for processor accesses. Synchronization pulses to the display, however, are maintained. Setting this bit to 1 can be used as a way to more rapidly update the frame buffer.
4	**Shift 4.** 0 = Load video shift registers every 1 or 2 character clocks (depending on bit 2 of this register) (default). 1 = Load shift registers every 4th character clock.
3	**Dot Clock Divide.** Setting this bit to 1 divides the dot clock by two and stretches all timing periods. This bit is used in standard VGA 40-column text modes to stretch timings to create horizontal resolutions of either 320 or 360 pixels (as opposed to 640 or 720 pixels, normally used in standard VGA 80-column text modes). 0 = Sequencer master clock output on the PCLK pin (used for 640 (720) pixel modes); Pixel clock is left unaltered (default). 1 = Pixel clock divided by 2 output on the PCLK pin (used for 320 (360) pixel modes).
2	**Shift Load.** Bit 4 of this register must be 0 for this bit to be effective. 0 = Load video data shift registers every character clock (default). 1 = Load video data shift registers every other character clock.
1	**Reserved.** Read as 0s.
0	**8/9 Dot Clocks.** This bit determines whether a character clock is 8 or 9 dot clocks long. 0 = 9 dot clocks (9 horizontal pixels) per character in text modes with a horizontal resolution of 720 pixels (default). 1 = 8 dot clocks (8 horizontal pixels) per character in text modes with a horizontal resolution of 640 pixels.

9.2.4. SR02—Plane/Map Mask

I/O (and Memory Offset) Address: 3C5h (Index=02h)
Default: 00h
Attributes: Read/Write

7 4	3 0
Reserved	Memory Planes Processor Write Access Enable.

Bit	Descriptions
7:4	**Reserved.** Read as 0s.
3:0	**Memory Planes [3:0] Processor Write Access Enable.** In both the Odd/Even Mode and the Chain 4 Mode, these bits still control access to the corresponding color plane. 0 = Disable. 1 = Enable. **Note:** This register is referred to in the VGA standard as the Map Mask Register. However, the word “map” is used with multiple meanings in the VGA standard and was, therefore, considered too confusing; hence, the reason for calling it the Plane Mask Register.

9.2.5. SR03—Character Font

I/O (and Memory Offset) Address:	3C5h (index=03h)
Default:	00h
Attributes:	Read/Write

7 6	5	4	3 2	1 0
Reserved (00)	Char Map A Select (bit 0)	Char Map B Select (bit 0)	Character Map A Select (bits 2 and 1)	Character Map B Select (bits 2 and 1)

Bit	Descriptions
7:6	**Reserved.** Read as 0s.
3:2,5	**Character Map Select Bits for Character Map B.** These three bits are used to select the character map (character generator tables) to be used as the secondary character set (font). Note that the numbering of the maps is not sequential.

Bit [3:2, 5]	Map Number	Table Location
00,0	0	1st 8KB of plane 2 at offset 0 (default)
00,1	4	2nd 8KB of plane 2 at offset 8K
01,0	1	3rd 8KB of plane 2 at offset 16K
01,1	5	4th 8KB of plane 2 at offset 24K
10,0	2	5th 8KB of plane 2 at offset 32K
10,1	6	6th 8KB of plane 2 at offset 40K
11,0	3	7th 8KB of plane 2 at offset 48K
11,1	7	8th 8KB of plane 2 at offset 56K

Bit	Descriptions
1:0,4	**Character Map Select Bits for Character Map A.** These three bits are used to select the character map (character generator tables) to be used as the primary character set (font). Note that the numbering of the maps is not sequential.

Bit [1:0,4]	Map Number	Table Location
0,00	0	1st 8KB of plane 2 at offset 0 (default)
0,01	4	2nd 8KB of plane 2 at offset 8K
0,10	1	3rd 8KB of plane 2 at offset 16K
0,11	5	4th 8KB of plane 2 at offset 24K
1,00	2	5th 8KB of plane 2 at offset 32K
1,01	6	6th 8KB of plane 2 at offset 40K
1,10	3	7th 8KB of plane 2 at offset 48K
1,11	7	8th 8KB of plane 2 at offset 56K

NOTES:

1. In text modes, bit 3 of the video data's attribute byte normally controls the foreground intensity. This bit may be redefined to control switching between character sets. This latter function is enabled whenever there is a difference in the values of the Character Font Select A and the Character Font Select B bits. If the two values are the same, the character select function is disabled and attribute bit 3 controls the foreground intensity.
2. Bit 1 of the Memory Mode Register (SR04) must be set to 1 for the character font select function of this register to be active. Otherwise, only character maps 0 and 4 are available.

9.2.6. SR04—Memory Mode Register

I/O (and Memory Offset) Address:	3C5h (index=04h)
Default:	00h
Attributes:	Read/Write

7 4	3	2	1	0
Reserved (0000)	Chain 4	Odd/Even	Extended Memory	Reserved (0)

Bit	Description
7:4	**Reserved.** Read as 0s.
3	**Chain 4 Mode.** The selections made by this bit affect both processor Read and Write accesses to the frame buffer. 0 = The manner in which the frame buffer memory is mapped is determined by the setting of bit 2 of this register (default). 1 = The frame buffer memory is mapped in such a way that the function of address bits 0 and 1 are altered so that they select planes 0 through 3.
2	**Odd/Even Mode.** Bit 3 of this register must be set to 0 for this bit to be effective. The selections made by this bit affect only processor writes to the frame buffer. 0 = The frame buffer memory is mapped in such a way that the function of address bit 0 such that even addresses select planes 0 and 2 and odd addresses select planes 1 and 3 (default). 1 = Addresses sequentially access data within a bit map, and the choice of which map is accessed is made according to the value of the Plane Mask Register (SR02).
1	**Extended Memory Enable.** This bit must be set to 1 to enable the selection and use of character maps in plane 2 via the Character Map Select Register (SR03). 0 = Disable processor accesses to more than the first 64KB of VGA standard memory (default). 1 = Enable processor accesses to the rest of the 256KB total VGA memory beyond the first 64KB.
0	**Reserved.** Read as 0s.

9.2.7. SR07—Horizontal Character Counter Reset

I/O (and Memory Offset) Address: 3C5h (index=07h)
Default: 00h
Attributes: Read/Write

Writing this register with any data causes the horizontal character counter to be held in reset (the character counter output will remain 0) until a write occurs to any other sequencer register location with SRX set to an index of 0 through 6.

The vertical line counter is clocked by a signal derived from the horizontal display enable (which does not occur if the horizontal counter is held in reset). Therefore, if a write occurs to this register during the vertical retrace interval, both the horizontal and vertical counters will be set to 0. A write to any other sequencer register location (with SRX set to an index of 0 through 6) may then be used to start both counters with reasonable synchronization to an external event via software control.

This is a standard VGA register that was not documented by IBM.

Bit	Description
7:0	**Horizontal Character Counter.**

9.3. Graphics Controller Registers

The graphics controller registers are accessed via either I/O space or Memory space. To access the registers the VGA Graphics Controller Index Register at I/O address 3CEh (or memory address 3CEh) is written with the index of the desired register. Then the desired register is accessed through the data port for the graphics controller registers at I/O address 3CFh (or memory address 3CFh).

9.3.1. GRX—GRX Graphics Controller Index Register

I/O (and Memory Offset) Address: 3CEh
Default: 000UUUUUb (U=Undefined)
Attributes: Read/Write

7 4	3 0
Reserved (0000)	Graphics Controller Register Index

Bit	Description
7:4	**Reserved.** Read as 0s.
3:0	**Sequencer Register Index.** This field selects any one of the graphics controller registers (GR[00:1F]) to be accessed via the data port at I/O location 3CFh.

9.3.2. GR00—Set/Reset Register

I/O (and Memory Offset) Address: 3CFh (index=00h)
Default: 0Uh (U=Undefined)
Attributes: Read/Write

7 … 4	3	2	1	0
Reserved (0000)	Set/Reset Plane 3	Set/Reset Plane 2	Set/Reset Plane 1	Set/Reset Plane 0

Bit	Description
7:4	**Reserved.** Read as 0s.
3:0	**Set/Reset Plane [3:0].** When the Write Mode bits (bits 0 and 1) of the Graphics Mode Register (GR05) are set to select Write Mode 0, all 8 bits of each byte of each memory plane are set to either 1 or 0 as specified in the corresponding bit in this register, if the corresponding bit in the Enable Set/Reset Register (GR01) is set to 1. When the Write Mode bits (bits 0 and 1) of the Graphics Mode Register (GR05) are set to select Write Mode 3, all processor data written to the frame buffer is rotated, then logically ANDed with the contents of the Bit Mask Register (GR08), and then treated as the addressed data's bit mask, while value of these four bits of this register are treated as the color value.

9.3.3. GR01—Enable Set/Reset Register

I/O (and Memory Offset) Address: 3CFh (Index=01h)
Default: 0Uh (U=Undefined)
Attributes: Read/Write

7 … 4	3	2	1	0
Reserved (0000)	Enable Set/ Reset Pln 3	Enable Set/ Reset Pln 2	Enable Set/ Reset Pln 1	Enable Set/ Reset Pln 0

Bit	Description
7:4	**Reserved.** Read as 0s.
3:0	**Enable Set/Reset Plane [3:0].** This register works in conjunction with the Set/Reset Register (GR00). The Write Mode bits (bits 0 and 1) must be set for Write Mode 0 for this register to have any effect. 0 = The corresponding memory plane can be read from or written to by the processor without any special bitwise operations taking place. 1 = The corresponding memory plane is set to 0 or 1 as specified in the Set/Reset Register (GR00).

9.3.4. GR02—Color Compare Register

I/O (and Memory Offset) Address: 3CFh (Index=02h)
Default: 0Uh (U=Undefined)
Attributes: Read/Write

7 … 4	3	2	1	0
Reserved (0000)	Color Compare Plane 3	Color Compare Plane 2	Color Compare Plane 1	Color Compare Plane 0

Bit	Description
7:4	**Reserved.** Read as 0s.
3:0	**Color Compare Plane [3:0]**. When the Read Mode bit (bit 3) of the Graphics Mode Register (GR05) is set to select Read Mode 1, all 8 bits of each byte of each of the 4 memory planes of the frame buffer corresponding to the address from which a processor read access is being performed are compared to the corresponding bits in this register (if the corresponding bit in the Color Don't Care Register (GR07) is set to 1). The value that the processor receives from the read access is an 8-bit value that shows the result of this comparison, wherein value of 1 in a given bit position indicates that all of the corresponding bits in the bytes across all of the memory planes that were included in the comparison had the same value as their memory plane's respective bits in this register.

9.3.5. GR03—Data Rotate Register

I/O (and Memory Offset) Address: 3CFh (Index=03h)
Default: 0Uh (U=Undefined)
Attributes: Read/Write

7 … 5	4 … 3	2 … 0
Reserved	Function Select	Rotate Count

Bit	Description
7:5	**Reserved.** Read as 0s.
4:3	**Function Select.** These bits specify the logical function (if any) to be performed on data that is meant to be written to the frame buffer (using the contents of the memory read latch) just before it is actually stored in the frame buffer at the intended address location. 00 = Data being written to the frame buffer remains unchanged, and is simply stored in the frame buffer. 01 = Data being written to the frame buffer is logically ANDed with the data in the memory read latch before it is actually stored in the frame buffer. 10 = Data being written to the frame buffer is logically ORed with the data in the memory read latch before it is actually stored in the frame buffer. 11 = Data being written to the frame buffer is logically XORed with the data in the memory read latch before it is actually stored in the frame buffer.
2:0	**Rotate Count.** These bits specify the number of bits to the right to rotate any data that is meant to be written to the frame buffer just before it is actually stored in the frame buffer at the intended address location.

9.3.6. GR04—Read Plane Select Register

I/O (and Memory Offset) Address:	3CFh (Index=04h)
Default:	0Uh (U=Undefined)
Attributes:	Read/Write

7 2	1 0
Reserved (000000)	Read Plane Select

Bit	Description
7:2	**Reserved.** Read as 0s.
1:0	**Read Plane Select.** These two bits select the memory plane from which the processor reads data in Read Mode 0. In Odd/Even Mode, bit 0 of this register is ignored. In Chain 4 Mode, both bits 1 and 0 of this register are ignored. The four memory planes are selected as follows: 00 = Plane 0 01 = Plane 1 10 = Plane 2 11 = Plane 3 These two bits also select which of the four memory read latches may be read via the Memory read Latch Data Register (CR22). The choice of memory read latch corresponds to the choice of plane specified in the table above. The Memory Read Latch Data register and this additional function served by 2 bits are features of the VGA standard that were never documented by IBM.

9.3.7. GR05—Graphics Mode Register

I/O (and Memory Offset) Address:	3CFh (Index=05h)
Default:	0UUU U0UUb (U=Undefined)
Attributes:	Read/Write

7	6 5	4	3	2	1 0
Reserved (0)	Shift Register Control	Odd/Even	Read Mode	Reserved (0)	Write Mode

Bit	Description
7	**Reserved.** Read as 0s.

Bit	Description
6:5	**Shift Register Control.** In standard VGA modes, pixel data is transferred from the 4 graphics memory planes to the palette via a set of 4 serial output bits. These 2 bits of this register control the format in which data in the 4 memory planes is serialized for these transfers to the palette.

Bits [6:5]=00

One bit of data at a time from parallel bytes in each of the 4 memory planes is transferred to the palette via the 4 serial output bits, with 1 of each of the serial output bits corresponding to a memory plane. This provides a 4-bit value on each transfer for 1 pixel, making possible a choice of 1 of 16 colors per pixel.

Serial Out	1st Xfer	2nd Xfer	3rd Xfer	4th Xfer	5th Xfer	6th Xfer	7th Xfer	8th Xfer
Bit 3	plane 3 bit 7	plane 3 bit 6	plane 3 bit 5	plane 3 bit 4	plane 3 bit 3	plane 3 bit 2	plane 3 bit 1	plane 3 bit 0
Bit 2	plane 2 bit 7	plane 2 bit 6	plane 2 bit 5	plane 2 bit 4	plane 2 bit 3	plane 2 bit 2	plane 2 bit 1	plane 2 bit 0
Bit 1	plane 1 bit 7	plane 1 bit 6	plane 1 bit 5	plane 1 bit 4	plane 1 bit 3	plane 1 bit 2	plane 1 bit 1	plane 1 bit 0
Bit 0	plane 0 bit 7	plane 0 bit 6	plane 0 bit 5	plane 0 bit 4	plane 0 bit 3	plane 0 bit 2	plane 0 bit 1	plane 0 bit 0

Bits [6:5]=01

Two bits of data at a time from parallel bytes in each of the 4 memory planes are transferred to the palette in a pattern that alternates per byte between memory planes 0 and 2, and memory planes 1 and 3. First the even-numbered and odd-numbered bits of a byte in memory plane 0 are transferred via serial output bits 0 and 1, respectively, while the even-numbered and odd-numbered bits of a byte in memory plane 2 are transferred via serial output bits 2 and 3. Next, the even-numbered and odd-numbered bits of a byte in memory plane 1 are transferred via serial output bits 0 and 1, respectively, while the even-numbered and odd-numbered bits of memory plane 3 are transferred via serial out bits 1 and 3. This provides a pair of 2-bit values (one 2-bit value for each of 2 pixels) on each transfer, making possible a choice of 1 of 4 colors per pixel.

Serial Out	1st Xfer	2nd Xfer	3rd Xfer	4th Xfer	5th Xfer	6th Xfer	7th Xfer	8th Xfer
Bit 3	plane 2 bit 7	plane 2 bit 5	plane 2 bit 3	plane 2 bit 1	plane 3 bit 7	plane 3 bit 5	plane 3 bit 3	plane 3 bit 1
Bit 2	plane 2 bit 6	plane 2 bit 4	plane 2 bit 2	plane 2 bit 0	plane 3 bit 6	plane 3 bit 4	plane 3 bit 2	plane 3 bit 0
Bit 1	plane 0 bit 7	plane 0 bit 5	plane 0 bit 3	plane 0 bit 1	plane 1 bit 7	plane 1 bit 5	plane 1 bit 3	plane 1 bit 1
Bit 0	plane 0 bit 6	plane 0 bit 4	plane 0 bit 2	plane 0 bit 0	plane 1 bit 6	plane 1 bit 4	plane 1 bit 2	plane 1 bit 0

This alternating pattern is meant to accommodate the use of the Odd/Even mode of organizing the 4 memory planes, which is used by standard VGA modes 2h and 3h.

Bits [6:5]=1x

Four bits of data at a time from parallel bytes in each of the 4 memory planes are transferred to the palette in a pattern that iterates per byte through memory planes 0 through 3. First the 4 most significant bits of a byte in memory plane 0 are transferred via the 4 serial output bits, followed by the 4 least significant bits of the same byte. Next, the same transfers occur from the parallel byte in memory planes 1, 2 and lastly, 3. Each transfer provides either the upper or lower half of an 8 bit value for the color for each pixel, making possible a choice of 1 of 256 colors per pixel.

Serial Out	1st Xfer	2nd Xfer	3rd Xfer	4th Xfer	5th Xfer	6th Xfer	7th Xfer	8th Xfer
Bit 3	plane 0 bit 7	plane 0 bit 3	plane 1 bit 7	plane 1 bit 3	plane 2 bit 7	plane 2 bit 3	plane 3 bit 7	plane 3 bit 3
Bit 2	plane 0 bit 6	plane 0 bit 2	plane 1 bit 6	plane 1 bit 2	plane 2 bit 6	plane 2 bit 2	plane 3 bit 6	plane 3 bit 2
Bit 1	plane 0 bit 5	plane 0 bit 1	plane 1 bit 5	plane 1 bit 1	plane 2 bit 5	plane 2 bit 1	plane 3 bit 5	plane 3 bit 1
Bit 0	plane 0 bit 4	plane 0 bit 0	plane 1 bit 4	plane 1 bit 0	plane 2 bit 4	plane 2 bit 0	plane 3 bit 4	plane 3 bit 0

This pattern is meant to accommodate mode 13h, a standard VGA 256-color graphics mode.

Bit	Description
4	**Odd/Even Mode.** 0 = Addresses sequentially access data within a bit map, and the choice of which map is accessed is made according to the value of the Plane Mask Register (SR02). 1 = The frame buffer is mapped in such a way that the function of address bit 0 is such that even addresses select memory planes 0 and 2 and odd addresses select memory planes 1 and 3. **Note:** This works in a way that is the inverse of (and is normally set to be the opposite of) bit 2 of the Memory Mode Register (SR02).
3	**Read Mode.** 0 = During a processor read from the frame buffer, the value returned to the processor is data from the memory plane selected by bits 1 and 0 of the Read Plane Select Register (GR04). 1 = During a processor read from the frame buffer, all 8 bits of the byte in each of the 4 memory planes corresponding to the address from which a processor read access is being performed are compared to the corresponding bits in this register (if the corresponding bit in the Color Don't Care Register (GR07) is set to 1). The value that the processor receives from the read access is an 8-bit value that shows the result of this comparison. A value of 1 in a given bit position indicates that all of the corresponding bits in the bytes across all 4 of the memory planes that were included in the comparison had the same value as their memory plane's respective bits in this register.
2	**Reserved.** Read as 0s.
1:0	**Write Mode.** 00 = Write Mode 0 — During a processor write to the frame buffer, the addressed byte in each of the 4 memory planes is written with the processor write data after it has been rotated by the number of counts specified in the Data Rotate Register (GR03). If, however, the bit(s) in the Enable Set/Reset Register (GR01) corresponding to one or more of the memory planes is set to 1, then those memory planes will be written to with the data stored in the corresponding bits in the Set/Reset Register (GR00). 01 = Write Mode 1 — During a processor write to the frame buffer, the addressed byte in each of the 4 memory planes is written to with the data stored in the memory read latches. (The memory read latches stores an unaltered copy of the data last read from any location in the frame buffer.) 10 = Write Mode 2 — During a processor write to the frame buffer, the least significant 4 data bits of the processor write data is treated as the color value for the pixels in the addressed byte in all 4 memory planes. The 8 bits of the Bit Mask Register (GR08) are used to selectively enable or disable the ability to write to the corresponding bit in each of the 4 memory planes that correspond to a given pixel. A setting of 0 in a bit in the Bit Mask Register at a given bit position causes the bits in the corresponding bit positions in the addressed byte in all 4 memory planes to be written with value of their counterparts in the memory read latches. A setting of 1 in a Bit Mask Register at a given bit position causes the bits in the corresponding bit positions in the addressed byte in all 4 memory planes to be written with the 4 bits taken from the processor write data to thereby cause the pixel corresponding to these bits to be set to the color value. 11 = Write Mode 3 — During a processor write to the frame buffer, the processor write data is logically ANDed with the contents of the Bit Mask Register (GR08). The result of this ANDing is treated as the bit mask used in writing the contents of the Set/Reset Register (GR00) are written to addressed byte in all 4 memory planes.

9.3.8. GR06—Miscellaneous Register

I/O (and Memory Offset) Address:	3CFh (Index=06h)
Default:	0Uh (U=Undefined)
Attributes:	Read/Write

7 … 4	3 … 2	1	0
Reserved (0000)	Memory Map Mode	Chain Odd/Even	Graphics / Text Mode

Bit	Description
7:4	**Reserved.** Read as 0s.
3:2	**Memory Map Mode.** These 2 bits control the mapping of the frame buffer into the processor address space as follows: **Bit [3:2]** **Frame Buffer Address Range** 00 A0000h – BFFFFh 01 A0000h – AFFFFh 10 B0000h – B7FFFh 11 B8000h – BFFFFh **Notes:** 1. This function is both in standard VGA modes and in extended modes that do not provide linear frame buffer access. 2. Software must set to the proper value.
1	**Chain Odd/Even.** This bit provides the ability to alter the interpretation of address bit A0, so that it may be used in selecting between the odd-numbered memory planes (planes 1 and 3) and the even-numbered memory planes (planes 0 and 2). 0 = A0 functions normally. 1 = A0 is switched with a high order address bit, in terms of how it is used in address decoding. The result is that A0 is used to determine which memory plane is being accessed (A0=0 for planes 0 and 2 and A0=1 for planes 1 and 3).
0	**Graphics/Text Mode.** 0 = Text mode. 1 = Graphics mode.

9.3.9. GR07—Color Don't Care Register

I/O (and Memory Offset) Address: 3CFh (Index=07h)
Default: 0Uh (U=Undefined)
Attributes: Read/Write

7 4	3	2	1	0
Reserved (0000)	Ignore Color Plane 3	Ignore Color Plane 2	Ignore Color Plane 1	Ignore Color Plane 0

Bit	Description
7:4	**Reserved.** Read as 0s.
3:0	**Ignore Color Plane [3:0].** Note that these bits have effect only when bit 3 of the Graphics Mode Register (GR05) is set to 1 to select read mode 1. 0 = The corresponding bit in the Color Compare Register (GR02) is not used in color comparisons. 1 = The corresponding bit in the Color Compare Register (GR02) is used in color comparisons.

9.3.10. GR08—Bit Mask Register

I/O (and Memory Offset) Address: 3CFh (Index=08h)
Default: Undefined
Attributes: Read/Write

Bit	Description
7:0	**Bit Mask.** 0 = The corresponding bit in each of the 4 memory planes is written to with the corresponding bit in the memory read latches. 1 = Manipulation of the corresponding bit in each of the 4 memory planes via other mechanisms is enabled. **Notes:** 1. This bit mask applies to any writes to the addressed byte of any or all of the 4 memory planes, simultaneously. 2. This bit mask is applicable to any data written into the frame buffer by the processor, including data that is also subject to rotation, logical functions (AND, OR, XOR), and Set/Reset. To perform a proper read-modify-write cycle into frame buffer, each byte must first be read from the frame buffer by the processor (and this will cause it to be stored in the memory read latches). The Bit Mask Register must be set, and the new data then must be written into the frame buffer by the processor.

9.3.11. GR10—Address Mapping

I/O (and Memory Offset) Address:	3CFh (Index=10h)
Default:	00h
Attributes:	R/W

7 5	4	3	2	1	0
Reserved	Paging to LM	VGA Buffer / Memory Map	Packed Mode Enable	Linear Mapping	Page Mapping

Bit	Description
7:5	**Reserved.**
4	**Page to Local Memory Enable.** Used Only if GR10(0) = 1 {Paging enabled} and (GR10(1) = 1 or GR10(2) = 1) {Either packed mode or Linear mode is enabled) and GR10(3) = 0 {VGA Buffer selected} 0 = Page to VGA Buffer. 1 = Page to Physical Local Memory.
3	**VGA Buffer/Memory Map Select.** 0 = VGA Buffer (default) 1 = Memory Map.
2	**Packed Mode Enable.** 0 = Address and data translation are bused register settings (default) 1 = Forced extended pack pixel address translation. In page mapping mode, register GR06 selects the video memory address.
1	**Linear Mapping (PCI).** 0 = Disable (default) 1 = Enable
0	**Page Mapping Enable.** This mode allows the mapping of the VGA space allocated in main memory (non local video memory) mode or all of local memory space through the [A0000:AFFFF] window (Using bit 4 of this register), which is a 64KB page. An internal address is generated using GR11[6:0] as the address line [22:16] extension to A[15:2]. 0 = Disable (default) 1 = Enable

Table 9. VGA Address Range

GR10 [2]	GR10 [1]	GR10 [0]	Note 1	Address Range (see note 2)	
				A0000-AFFFF Range (No GTT)	B0, B8 Ranges (No GTT)
0	0	0	Std VGA xlations	VGA Controller, No Paging	VGA Controller, No Paging
0	0	1	Paging and VGA xlation **not supported**	NA	NA
0	1	0	No Paging, No VGA xlations	Bypass VGA, No Paging	Bypass VGA, No Paging
0	1	1	Paging, No VGA xlations	Bypass VGA, Paged by GR11	Bypass VGA, No Paging
1	0	0	No Paging, No VGA xlations	Bypass VGA, No Paging	Bypass VGA, No Paging
1	0	1	Paging, No VGA xlations	Bypass VGA, Paged by GR11	Bypass VGA, No Paging
1	1	0	No Paging, No VGA xlations	Bypass VGA, No Paging	Bypass VGA, No Paging
1	1	1	Paging, No VGA xlations	Bypass VGA Paged by GR11	Bypass VGA, No Paging

NOTES:

1. GR10[2:0] must **not** be programmed to "001" value because the GMCH hardware does not support the paging and VGA translation mode. Unpredictable hardware behavior will occur if GR10[2:0] were programmed to "001" value.
2. VGA Address Range, selected by GR06, Graphics range selected through Graphics base address register in configuration space. Access to VGA range does not require a translation table and VGA range paging allows access to all of local memory if it is setup with bit 4 of this register or to all the stolen for VGA main memory space. Access to graphics range requires GTT to be set up and will result in a prefetch unless prefetch is disabled. Access to VGA range will not result in prefetch.
3. BIOS should access local memory through the "back door" mechanism by setting GR10=17h, GR11=0, and GR6=0 only when local memory has been enabled (MMADR+3000h), else the system will hang in a snoop stall forever.

9.3.12. GR11—Page Selector

I/O (and Memory Offset) Address:	3CFh (Index=11h)
Default :	00h
Attributes:	R/W

Bit	Description
7:0	**Page Select.** Selects a 64KB window within VGA space in NLVM mode or all of local memory when Page Mapping is enabled (GR10[0]=1). In addition, this register is used for page selection of memory mapped register addresses.

9.3.13. GR[14:1F]—Software Flags

I/O (and Memory Offset) Address:	3CFh (Index=14h-1fh)
Default:	00
Attribute:	R/W

Bit	Description
7:0	**Software Flags.** Used as scratch pad space in BIOS. These have no effect on H/W.

9.4. Attribute Controller Registers

Unlike the other sets of indexed registers, the attribute controller registers are not accessed through a scheme employing entirely separate index and data ports. I/O address 3C0h (or memory address 3C0h) is used both as the read and write for the index register, and as the write address for the data port. I/O address 3C1h (or memory address 3C1h) is the read address for the data port.

To write to one of the attribute controller registers, the index of the desired register must be written to I/O address 3C0h (or memory address 3C0h), and then the data is written to the very same I/O (memory) address. A flip-flop alternates with each write to I/O address 3C0h (or memory address 3C0h) to change its function from writing the index to writing the actual data, and back again. This flip-flop may be deliberately set so that I/O address 3C0h (or memory address 3C0h) is set to write to the index (which provides a way to set it to a known state) by performing a read operation from Input Status Register 1 (ST01) at I/O address 3BAh (or memory address 3BAh) or 3DAh (or memory address 3DAh), depending on whether the graphics system has been set to emulate an MDA or a CGA as per MSR[0].

To read from one of the attribute controller registers, the index of the desired register must be written to I/O address 3C0h (or memory address 3C0h), and then the data is read from I/O address 3C1h (or memory address 3C1h). A read operation from I/O address 3C1h (or memory address 3C1h) does not reset the flip-flop to writing to the index. Only a write to 3C0h (or memory address 3C0h) or a read from 3BAh or 3DAh (or memory address 3BAh or 3DAh), as described above, will toggle the flip-flop back to writing to the index.

9.4.1. ARX—Attribute Controller Index Register

I/O (and Memory Offset) Address:	3C0h
Default:	00UU UUUUb (U=Undefined)
Attributes:	Read/Write

7 6	5	4 0
Reserved (00)	Video Enable	Attribute Controller Register Index

Bit	Description
7:6	**Reserved.** Read as 0s.
5	**Video Enable.** Note that In the VGA standard, this is called the "Palette Address Source" bit. 0 = Disable. Attribute controller color registers (AR[00:0F]) can be accessed by the processor. 1 = Enable. Attribute controller color registers (AR[00:0F]) are inaccessible by the processor.
4:0	**Attribute Controller Register Index.** These five bits are used to select any one of the attribute controller registers (AR[00:14]), to be accessed. **Note:** AR12 is referred to in the VGA standard as the Color Plane Enable Register. The words "plane," "color plane," "display memory plane," and "memory map" have been all been used in IBM* literature on the VGA standard to describe the four separate regions in the frame buffer where the pixel color or attribute information is split up and stored in standard VGA planar modes. This use of multiple terms for the same subject was deemed to be confusing, therefore, AR12 is called the Memory Plane Enable Register. Attribute Controller Register Index.

9.4.2. AR[00:0F]—Palette Registers [0:F]

I/O (and Memory Offset) Address:	Read at 3C1h and Write at 3C0h; (index=00h-0Fh)
Default:	00UU UUUUb (U=Undefined)
Attributes:	Read/Write

7 6	5 0
Reserved	Palette Bits P[5:0]

Bit	Description
7:6	**Reserved.** Read as 0s.
5:0	**Palette Bits P[5:0].** In each of these 16 registers, these are the lower 6 of 8 bits that are used to map either text attributes or pixel color input values (for modes that use 16 colors) to the 256 possible colors available to be selected in the palette. **Note:** Bits 3 and 2 of the Color Select Register (AR14) supply bits P7 and P6 for the values contained in all 16 of these registers. Bits 1 and 0 of the Color Select Register (AR14) can also replace bits P5 and P4 for the values contained in all 16 of these registers, if bit 7 of the Mode Control Register (AR10) is set to 1.

9.4.3. AR10—Mode Control Register

I/O (and Memory Offset) Address:	Read at 3C1h and Write at 3C0h; (index=10h)
Default:	UUh (U=Undefined)
Attributes:	Read/Write

7	6	5	4	3	2	1	0
Palette Bits P5, P4 Select	Pixel Width/Clk Select	Pixel Panning Compat	Reserved (0)	Enable Blink/ Select Bkgnd Int	Enable Line Graphics Char Code	Select Display Type	Graphics/ Alpha Mode

Bit	Description
7	**Palette Bits P5, P4 Select.** 0 = P5 and P4 for each of the 16 selected colors (for modes that use 16 colors) are individually provided by bits 5 and 4 of their corresponding Palette Registers (AR[00:0F]). 1 = P5 and P4 for all 16 of the selected colors (for modes that use 16 colors) are provided by bits 1 and 0 of Color Select Register (AR14).
6	**Pixel Width/Clock Select.** 0 = Six bits of video data (translated from 4 bits via the palette) are output every dot clock. 1 = Two sets of 4 bits of data are assembled to generate 8 bits of video data which is output every other dot clock, and the Palette Registers (AR[00:0F]) are bypassed. **Note:** This bit is set to 0 for all of the standard VGA modes, except mode 13h.

Bit	Description
5	**Pixel Panning Compatibility.** 0 = Scroll both the upper and lower screen regions horizontally as specified in the Pixel Panning Register (AR13). 1 = Scroll only the upper screen region horizontally as specified in the Pixel Panning Register (AR13). **Note:** This bit has application only when split-screen mode is being used, where the display area is divided into distinct upper and lower regions which function somewhat like separate displays.
4	**Reserved.** Read as 0s.
3	**Enable Blinking/Select Background Intensity.** 0 = Disables blinking in graphics modes, and for text modes, sets bit 7 of the character attribute bytes to control background intensity, instead of blinking. 1 = Enables blinking in graphics modes and for text modes, sets bit 7 of the character attribute bytes to control blinking, instead of background intensity. **Note:** The blinking rate is derived by dividing the VSYNC signal. The Blink Rate Control Register (FR19) defines the blinking rate.
2	**Enable Line Graphics Character Code.** 0 = Every 9th pixel of a horizontal line (i.e., the last pixel of each horizontal line of each 9-pixel wide character box) is assigned the same attributes as the background of the character of which the given pixel is a part. 1 = Every 9th pixel of a horizontal line (i.e., the last pixel of each horizontal line of each 9-pixel wide character box) is assigned the same attributes as the 8th pixel if the character of which the given pixel is a part. This setting is intended to accommodate the line-drawing characters of the PC's extended ASCII character set -- characters with an extended ASCII code in the range of B0h to DFh. **Note:** In IBM* literature describing the VGA standard, the range of extended ASCII codes that are said to include the line-drawing characters is mistakenly specified as C0h to DFh, rather than the correct range of B0h to DFh.
1	**Select Display Type.** 0 = Attribute bytes in text modes are interpreted as they would be for a color display. 1 = Attribute bytes in text modes are interpreted as they would be for a monochrome display.
0	**Graphics/Alphanumeric Mode.** 0 = Alphanumeric (text) mode. 1 = Graphics mode.

9.4.4. AR11—Overscan Color Register

I/O (and Memory Offset) Address:	Read at 3C1h and Write at 3C0h; (index=11h)
Default:	UUh (U=Undefined)
Attributes:	Read/Write

Bit	Description
7:0	**Overscan.** These 8 bits select the overscan (border) color. The border color is displayed during the blanking intervals. For monochrome displays, this value should be set to 00h.

9.4.5. AR12—Memory Plane Enable Register

I/O (and Memory Offset) Address:	Read at 3C1h and Write at 3C0h; (index=12h)
Default:	00UU UUUUb (U=Undefined)
Attributes:	Read/Write

7 6	5 4	3	2	1	0
Reserved (00)	Video Status Mux	Enable Plane 3	Enable Plane 2	Enable Plane 1	Enable Plane 0

Bit	Description
7:6	**Reserved.** Read as 0s.
5:4	**Video Status Mux.** These 2 bits are used to select 2 of the 8 possible palette bits (P7-P0) to be made available to be read via bits 5 and 4 of the Input Status Register 1 (ST01). The table below shows the possible choices. **Bit [5:4]** / **ST01 Bit 5** / **ST01 Bit 4** 00 / P2 (default) / P0 (default) 01 / P5 / P4 10 / P3 / P1 11 / P7 / P6 These bits are typically unused by current software; they are provided for EGA compatibility.
3:0	**Enable Plane [3:0].** These 4 bits individually enable the use of each of the 4 memory planes in providing 1 of the 4 bits used in video output to select 1 of 16 possible colors from the palette to be displayed. 0 = Disable the use of the corresponding memory plane in video output to select colors, forcing the bit that the corresponding memory plane would have provided to a value of 0. 1 = Enable the use of the corresponding memory plane in video output to select colors. **Note:** AR12 is referred to in the VGA standard as the Color Plane Enable Register. The words “plane,” “color plane,” “display memory plane,” and “memory map” have been all been used in IBM literature on the VGA standard to describe the 4 separate regions in the frame buffer that are amongst which pixel color or attributes information is split up and stored in standard VGA planar modes. This use of multiple terms for the same subject was considered confusing; therefore, AR12 is called the Memory Plane Enable Register.

9.4.6. AR13—Horizontal Pixel Panning Register

I/O (and Memory Offset) Address: Read at 3C1h and Write at 3C0h; (index=13h)
Default: 0Uh (U=Undefined)
Attributes: Read/Write

7 4	3 0
Reserved (0000)	Horizontal Pixel Shift

Bit	Description
7:4	**Reserved.**
3:0	**Horizontal Pixel Shift 3-0.** This field holds a 4-bit value that selects the number of pixels by which the image is shifted horizontally to the left. This function is available in both text and graphics modes. In text modes with a 9-pixel wide character box, the image can be shifted up to 9 pixels to the left. In text modes with an 8-pixel wide character box, and in graphics modes other than those with 256 colors, the image can be shifted u0p to 8 pixels to the left. In standard VGA mode 13h (where bit 6 of the Mode Control Register, AR10, is set to 1 to support 256 colors), bit 0 of this register must remain set to 0, and the image may be shifted up to only 4 pixels to the left. In this mode, the number of pixels by which the image is shifted can be further controlled using bits 6 and 5 of the Preset Row Scan Register (CR08). **Number of Pixels Shifted**

Bits [3:0]	**9 Pixel Text**	**8-Pixel Text & Graphics**	**256-Color Graphics**
0h	1	0	0
1h	2	1	Undefined
2h	3	2	1
3h	4	3	Undefined
4h	5	4	2
5h	6	5	Undefined
6h	7	6	3
7h	8	7	Undefined
8h	0	Undefined	Undefined

9.4.7. AR14—Color Select Register

I/O (and Memory Offset) Address: Read at 3C1h and Write at 3C0h; (index=14h)
Default: 0Uh (U=Undefined)
Attributes: Read/Write

7 … 4	3	2	1	0
Reserved (0000)	P7	P6	Alt P5	Alt P4

Bit	Description
7:4	**Reserved.**
3:2	**Palette Bits P[7:6].** These are the 2 upper-most of the 8 bits that are used to map either text attributes or pixel color input values (for modes that use 16 colors) to the 256 possible colors contained in the palette. These 2 bits are common to all 16 sets of bits P5 through P0 that are individually supplied by Palette Registers 0-F (AR[00:0F]).
1:0	**Alternate Palette Bits P[5:4].** These 2 bits can be used as an alternate version of palette bits P5 and P4. Unlike the P5 and P4 bits that are individually supplied by Palette Registers 0-F (AR[00:0F]), these 2 alternate palette bits are common to all 16 of Palette Registers. Bit 7 of the Mode Control Register (AR10) is used to select between the use of either the P5 and P4 bits that are individually supplied by the 16 Palette Registers or these 2 alternate palette bits.

9.5. VGA Color Palette Registers

The palette DAC has two main components: a palette in which a selection of 256 colors may be stored, and a set of three digital to analog (D-to-A) converters, one each for the red, green and blue components used to produce a color on a CRT display. The palette DAC is also frequently called the RAMDAC, to emphasize the presence of memory alongside the three D-to-A converters, and the palette, itself, is often referred to as the CLUT or color look-up table.

During normal use, the palette DAC is operated either in direct-color mode or indexed-color mode. Direct color mode is used with pixel depths of 15, 16, or 24 bits per pixel. In direct color mode, the pixel data received from the frame buffer, through the sequencer and the attribute controller, directly specifies the color for a given pixel. This pixel data is pre-formatted such that certain bits of the pixel data for each pixel are used to provide the red, green and blue output values for each of the three corresponding 8-bit D-to-A converters. Indexed-color mode is used with pixel depths of 8 bits per pixel or less. In indexed-color mode, the incoming pixel data for each pixel is actually an 8-bit index that is used to choose one of the 256 color data positions within the palette. Each color data position holds a 24-bit color value that specifies the actual 8-bit red, green, and blue values for each of the three corresponding 8-bit D-to-A converters. In essence, the colors for each pixel are specified indirectly, with the actual choice of colors taking place in the color data positions of the palette, while the incoming pixel data chooses from among these color data positions. This method allows the full range of over 16 million possible colors to be accessible in modes with only 8 or fewer bits per pixel.

The color data stored in these 256 color data positions can be accessed only through a complex sub-addressing scheme, using a data register and two index registers. The Palette Data Register at I/O address 3C9h (or memory address offset 3C1h) is the data port. The Palette Read Index Register at I/O address 3C7h (or memory address offset 3C7h) and the Palette Write Index Register at I/O address 3C8h (or memory address offset 3C8h) are the two index registers. The Palette Read Index Register is used to choose the color data position that is to be read from the data port. The Palette Write Index Register is used to choose the color data position that is to be written to through the same data port. This arrangement allows the same data port to be used for reading from and writing to two different color data positions. Reading and writing the color data at a color data position involves three successive reads or writes since the color data stored at each color data position consists of three bytes.

To read a color data position, the index of the desired color data position must first be written to the Palette Read Index Register. Then all three bytes of data in a given color data position may be read at the Palette Data Register. The first byte read from the Palette Data Register retrieves the 8-bit value specifying the intensity of the red color component, while the second and third bytes read are the corresponding 8-bit values for the green and blue color components, respectively. After completing the third read operation, the Palette Read Index Register is automatically incremented so that the data of the next color data position becomes accessible for being read. This allows the contents of all 256 color data positions of the palette to be read by specifying only the index of the 0th color data position in the Palette Read Index Register, and then simply performing 768 successive reads from the Palette Data Register.

Writing a color data position entails a very similar procedure. The index of the desired color data position must first be written to the Palette Write Index Register. Then all three bytes of data to specify a given color may be written to the Palette Data Register. The first byte written to the Palette Data Register specifies the intensity of the red color component, the second byte specifies the intensity for the green color component, and the third byte specifies the same for the blue color component. One important detail is that all three of these bytes must be written before the hardware will actually update these three values in the given color data position. When all three bytes have been written, the Palette Write Index Register is automatically incremented so that the data of the next color data position becomes accessible for being written. This allows the contents of all 256 color data positions of the palette to be written by specifying only the index of the 0th color data position in the Palette Write Index Register, and then simply performing 768 successive writes to the Palette Data Register.

In addition to the standard set of 256 color data positions of the palette, there is also an alternate set of 8 color data positions used to specify the colors used to draw the cursor, and these are also accessed using the very same sub-addressing scheme. A bit in the Pixel Pipeline Configuration Register (PIXCONF) determines whether the standard 256 color data positions or the alternate eight color data positions are to be accessed through this sub-addressing scheme.

9.5.1. DACMASK—Pixel Data Mask Register

I/O (and Memory Offset) Address:	3C6h
Default:	Undefined
Attributes:	Read/Write

Bit	Description
7:0	**Pixel Data Mask.** In indexed-color mode, the 8 bits of this register are logically ANDed with the 8 bits of pixel data received from the frame buffer for each pixel. The result of this ANDing process becomes the actual index used to select color data positions within the palette. This has the effect of limiting the choice of color data positions that may be specified by the incoming 8-bit data. 0 = Corresponding bit in the resulting 8-bit index being forced to 0. 1 = Allows the corresponding bit in the resulting index to reflect the actual value of the corresponding bit in the incoming 8-bit pixel data. In direct-color mode, the palette is not used, and the data in this register is ignored.

9.5.2. DACSTATE—DAC State Register

I/O (and Memory Offset) Address:	3C7h
Default:	00h
Attributes:	Read Only

7 … 2	1 … 0
Reserved (000000)	DAC State

Bit	Description
7:2	**Reserved.** Read as 0s.
1:0	**DAC State.** This field indicates which of the two index registers was most recently written. **Bits [1:0]** **Index Register Indicated** 00 Palette Write Index Register at I/O Address 3C7h (default) 01 Reserved 10 Reserved 11 Palette Read Index Register at I/O Address 3C8h

9.5.3. DACRX—Palette Read Index Register

I/O (and Memory Offset) Address:	3C7h
Default:	00h
Attributes:	Write Only

Bit	Description
7:0	**Palette Read Index.** The 8-bit index value programmed into this register chooses which of 256 standard color data positions within the palette (or which of 8 alternate color data positions, depending on the state of a bit in the Pixel Pipeline Control 0 Register) are to be made accessible for being read via the Palette Data Register (DACDATA). The index value held in this register is automatically incremented when all three bytes of the color data position selected by the current index have been read.

9.5.4. DACWX—Palette Write Index Register

I/O (and Memory Offset) Address:	3C8h
Default:	00h
Attributes:	Write Only

Bit	Description
7:0	**Palette Write Index.** The 8-bit index value programmed into this register chooses which of 256 standard color data positions within the palette (or which of 8 alternate color data positions, depending on the state of a bit in the Pixel Pipeline Control 0 Register) are to be made accessible for being written via the Palette Data Register (DACDATA). The index value held in this register is automatically incremented when all three bytes of the color data position selected by the current index have been written.

9.5.5. DACDATA—Palette Data Register

I/O (and Memory Offset) Address:	3C9h
Default:	Undefined
Attributes:	Read/Write

Bit	Description
7:0	**Palette Data.** This byte-wide data port provides read or write access to the three bytes of data of each color data position selected using the Palette Read Index Register (DACRX) or the Palette Write Index Register (DACWX). The three bytes in each color data position are read or written in three successive read or write operations. The first byte read or written specifies the intensity of the red component of the color specified in the selected color data position. The second byte is for the green component, and the third byte is for the blue component. When writing data to a color data position, all three bytes must be written before the hardware will actually update the three bytes of the selected color data position. When reading or writing to a color data position, ensure that neither the Palette Read Index Register (DACRX) or the Palette Write Index Register (DACWX) are written to before all three bytes are read or written. A write to either of these two registers causes the circuitry that automatically cycles through providing access to the bytes for red, green and blue components to be reset such that the byte for the red component is the one that will be accessed by the next read or write operation via this register.

9.6. CRT Controller Register

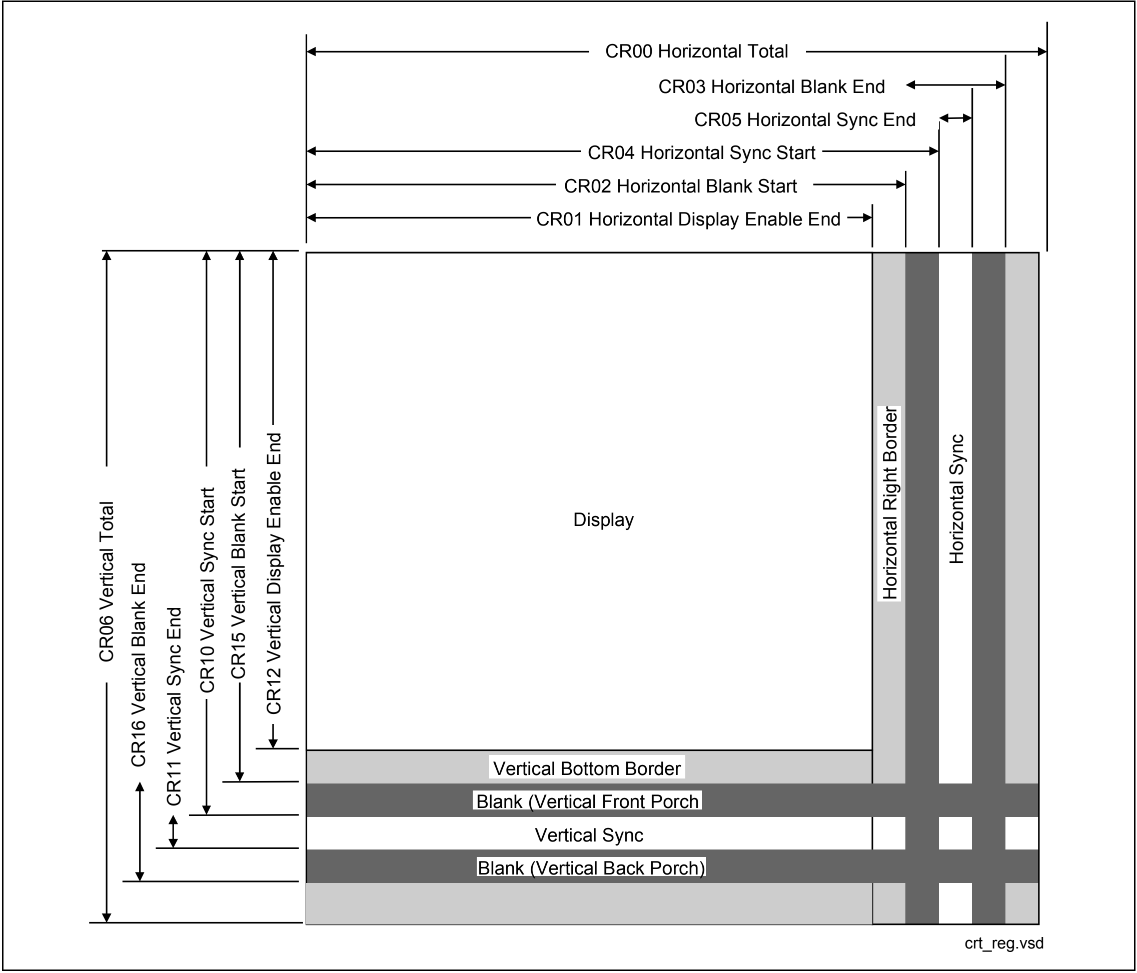
The CRT controller registers are accessed by writing the index of the desired register into the CRT Controller Index Register at I/O address 3B4h or 3D4h, depending on whether the graphics system is configured for MDA or CGA emulation. The desired register is then accessed through the data port for the CRT controller registers located at I/O address 3B5h or 3D5h, again depending upon the choice of MDA or CGA emulation as per MSR[0]. For memory mapped accesses, the Index register is at 3B4h (MDA mode) or 3D3h (CGA mode) and the data port is accessed at 3B5h (MDA mode) or 3D5h (CGA mode).

Notes:

1. Register CR80 enables / disables the CRTC Extensions, except for register CR41.
2. **Group 0 Protection:** In the original IBM VGA, CR[0:7] could be write-protected by CR11[7]. In BIOS code, this write protection is set following each mode change. Other protection groups have no current use, and would not be used going forward by the BIOS or by drivers. They are the result of an industry trend some years ago to attempt to write protect other groups of registers; however, all such schemes were chip specific. Only the IBM compatible write protection method (Group 0 Protection) is currently supported.

The following figure shows display fields and dimensions and the particular CRxx register that provides the control.

Figure 25 Display Fields and Dimensions CRxx Control Registers



9.6.1. CRX—CRT Controller Index Register

I/O (and Memory Offset) Address:	3B4h/3D4h
Default:	0Uh (U=Undefined)
Attributes:	Read/Write

The value which can be supported at CRX is likely to be impacted by the X09 register which modifies the meaning of the extended CRxx registers.

Bit	Description
7:0	**CRT Controller Register Index.** These 8 bits are used to select any one of the CRT controller registers to be accessed via the data port at I/O location 3B5h or 3D5h, depending upon whether the graphics system is configured for MDA or CGA emulation. The data port memory address offsets are 3B5h/3D5h.

9.6.2. CR00—Horizontal Total Register

I/O (and Memory Offset) Address:	3B5h/3D5h (index=00h)
Default:	00h
Attributes:	Read/Write (Group 0 Protection)

This register is used to specify the total length of each scan line. This encompasses both the part of the scan line that is within the active display area and the part that is outside of it. This register is extended to cover 16x12 resolution using CR35 and CR39.

Bit	Description
7:0	**Horizontal Total.** This field should be programmed with a value equal to the total number of character clocks within the entire length of a scan line, minus 5.

9.6.3. CR01—Horizontal Display Enable End Register

I/O (and Memory Offset) Address:	3B5h/3D5h (index=01h)
Default:	Undefined
Attributes:	Read/Write (Group 0 Protection)

This register is used to specify the end of the part of the scan line that is within the active display area relative to its beginning. In other words, this is the horizontal width of the active display area.

Bit	Description
7:0	**Horizontal Display Enable End.** This field should be programmed with a value equal to the number of character clocks that occur within the part of a scan line that is within the active display area, minus 1.

9.6.4. CR02—Horizontal Blanking Start Register

I/O (and Memory Offset) Address:	3B5h/3D5h (index=02h)
Default:	Undefined
Attributes:	Read/Write (Group 0 Protection)

This register is used to specify the beginning of the horizontal blanking period relative to the beginning of the active display area of a scan line.

Bit	Description
7:0	**Horizontal Blanking Start.** This field should be programmed with a value equal to the number of character clocks that occur on a scan line from the beginning of the active display area to the beginning of the horizontal blanking.

9.6.5. CR03—Horizontal Blanking End Register

I/O (and Memory Offset) Address: 3B5h/3D5h (index=03h)
Default: 1UUU UUUUb (U=Undefined)
Attributes: Read/Write (Group 0 Protection)

7	6 5	4 0
Reserved (0)	Display Enable Skew Control	Horizontal Blanking End Bits 4:0

Bit	Description
7	**Reserved.** Values written to this bit are ignored, and to maintain consistency with the VGA standard, a value of 1 is returned when this bit is read. At one time, this bit was used to enable access to certain light pen registers. At that time, setting this bit to 0 provided this access, but setting this bit to 1 was necessary for normal operation.
6:5	**Display Enable Skew Control.** Defines the degree to which the start and end of the active display area are delayed along the length of a scan line to compensate for internal pipeline delays. These 2 bits describe the delay in terms of a number character clocks. **Bit [6:5]** **Amount of Delay** 00 no delay 01 delayed by 1 character clock 10 delayed by 2 character clocks 11 delayed by 3 character clocks
4:0	**Horizontal Blanking End Bits [4:0].** This field provides the 5 least significant bits of a 6-bit value that specifies the end of the blanking period relative to its beginning on a single scan line. Bit 7 of the Horizontal Sync End Register (CR05) supplies the most significant bit. This 6-bit value should be programmed to be equal to the least significant 6 bits of the result of adding the length of the blanking period in terms of character clocks to the value specified in the Horizontal Blanking Start Register (CR02).

9.6.6. CR04—Horizontal Sync Start Register

I/O (and Memory Offset) Address: 3B5h/3D5h (index=04h)
Default: Undefined
Attributes: Read/Write (Group 0 Protection)

This register is used to specify the beginning of the horizontal sync pulse relative to the beginning of the active display area on a scan line.

Bit	Description
7:0	**Horizontal Sync Start.** This field should be set equal to the number of character clocks that occur from beginning of the active display area to the beginning of the horizontal sync pulse on a single scan line.

9.6.7. CR05—Horizontal Sync End Register

I/O (and Memory Offset) Address:	3B5h/3D5h (index=05h)
Default:	00h
Attributes:	Read/Write (Group 0 Protection)

7	6 5	4 0
Hor Blank End Bit 5	Horizontal Sync Delay	Horizontal Sync End

Bit	Description
7	**Horizontal Blanking End Bit 5.** This bit provides the most significant bit of a 6-bit value that specifies the end of the horizontal blanking period relative to its beginning. Bits [4:0] of Horizontal Blanking End Register (CR03) supplies the 5 least significant bits. See CR03[4:0] for further details. This 6-bit value should be set to the least significant 6 bits of the result of adding the length of the blanking period in terms of character clocks to the value specified in the Horizontal Blanking Start Register (CR02).
6:5	**Horizontal Sync Delay.** This field defines the degree to which the start and end of the horizontal sync pulse are delayed to compensate for internal pipeline delays. This capability is supplied to implement VGA compatibility. These field describes the delay in terms of a number character clocks. **Bit [6:5]** **Amount of Delay** 00 no delay 01 delayed by 1 character clock 10 delayed by 2 character clocks 11 delayed by 3 character clocks
4:0	**Horizontal Sync End.** This field provides the 5 least significant bits of a 5-bit value that specifies the end of the horizontal sync pulse relative to its beginning. A value equal to the 5 least significant bits of the horizontal character counter value at which time the horizontal retrace signal becomes inactive (logical 0). Thus, this 5-bit value specifies the width of the horizontal sync pulse. To obtain a retrace signal of W, the following algorithm is used: Value of Horizontal Sync start Register (CR04) + width of horizontal retrace signal in character clock units = 5 bit result to be programmed in this field

9.6.8. CR06—Vertical Total Register

I/O (and Memory Offset) Address:	3B5h/3D5h (index=06h)
Default:	00h
Attributes:	Read/Write (Group 0 Protection)

Bit	Description
7:0	**Vertical Total Bits [7:0].** This field provides the 8 least significant bits of either a 10-bit or 12-bit value that specifies the total number of scan lines. This includes the scan lines both inside and outside of the active display area. In standard VGA modes, where bit 0 of the I/O Control Register (CR80) is set to 0, the vertical total is specified with a 10-bit value. The 8 least significant bits of this value are supplied by these 8 bits of this register, and the 2 most significant bits are supplied by bits 5 and 0 of the Overflow Register (CR07). In extended modes, where bit 0 of the I/O Control Register (CR80) is set to 1, the vertical total is specified with a 12-bit value. The 8 least significant bits of this value are supplied by these 8 bits of this register, and the 4 most significant bits are supplied by bits [3:0] of the Extended Vertical Total Register (CR30). This 10-bit or 12-bit value should be programmed to equal the total number of scan lines, minus 2.

9.6.9. CR07—Overflow Register

I/O (and Memory Offset) Address:	3B5h/3D5h (index=07h)
Default:	UU0U UUU0b (U=Undefined)
Attributes:	Read/Write (Group 0 Protection on bits [7:5, 3:0])

7	6	5	4	3	2	1	0
Vert Sync Start Bit 9	Vert Disp En Bit 9	Vert Total Bit 9	Line Cmp Bit 8	Vert Blnk Start Bit 8	Vert Sync Start Bit 8	Vert Disp En Bit 8	Vert Total Bit 8

Bit	Description
7	**Vertical Sync Start Bit 9.** The vertical sync start is a 10-bit or 12-bit value that specifies the beginning of the vertical sync pulse relative to the beginning of the active display area. In standard VGA modes, where bit 0 of the I/O Control Register (CR80) is set to 0, the vertical sync start is specified with a 10-bit value. The 8 least significant bits of this value are supplied by bits [7:0] of the Vertical Sync Start Register (CR10), and the most and second-most significant bits are supplied by this bit and bit 2, respectively, of this register. In extended modes, where bit 0 of the I/O Control Register (CR80) is set to 1, the vertical display end is specified with a 12-bit value. The 8 least significant bits of this value are supplied by bits [7:0] of the Vertical Sync Start Register (CR10), and the 4 most significant bits are supplied by bits [3:0] of the Extended Vertical Sync Start Register (CR32) register. In extended modes, neither this bit, nor bit 2 of this register are used. This 10-bit or 12-bit value should be programmed to be equal to the number of scan lines from the beginning of the active display area to the start of the vertical sync pulse. Since the active display area always starts on the 0th scan line, this number should be equal to the number of the scan line on which the vertical sync pulse begins.

Bit	Description
6	**Vertical Display Enable End Bit 9.** The vertical display enable end is a 10-bit or 12-bit value that specifies the number of the last scan line within the active display area. In standard VGA modes, where bit 0 of the I/O Control Register (CR80) is set to 0, the vertical display enable end is specified with a 10-bit value. The 8 least significant bits of this value are supplied by bits [7:0] of the Vertical Display Enable End Register (CR12), and the most and second-most significant bits are supplied by this bit and bit 1, respectively, of this register. In extended modes, where bit 0 of the I/O Control Register (CR80) is set to 1, the vertical display enable end is specified with a 12-bit value. The 8 least significant bits of this value are supplied by bits [7:0] of the Vertical Display Enable End Register (CR12), and the 4 most significant bits are supplied by bits [3:0] of the Extended Vertical Display End Enable Register (CR31). In extended modes, neither this bit, nor bit 1 of this register are used. This 10-bit or 12-bit value should be programmed to be equal to the number of the last scan line within in the active display area. Since the active display area always starts on the 0th scan line, this number should be equal to the total number of scan lines within the active display area, minus 1.
5	**Vertical Total Bit 9.** The vertical total is a 10-bit or 12-bit value that specifies the total number of scan lines. This includes the scan lines both inside and outside of the active display area. In standard VGA modes, where bit 0 of the I/O Control Register (CR80) is set to 0, the vertical total is specified with a 10-bit value. The 8 least significant bits of this value are supplied by bits [7:0] of the Vertical Total Register (CR06), and the most and second-most significant bits are supplied by this bit and bit 0, respectively, of this register. In extended modes, where bit 0 of the I/O Control Register (CR80) is set to 1, the vertical total is specified with a 12-bit value. The 8 least significant bits of this value are supplied by bits [7:0] of the Vertical Total Register (CR06), and the 4 most significant bits are supplied by [3:0] bits of the Extended Vertical Total Register (CR30). In extended modes, neither this bit, nor bit 0 of this register are used. This 10-bit or 12-bit value should be programmed equal to the total number of scan lines, minus 2.
4	**Line Compare Bit 8.** This bit provides the second most significant bit of a 10-bit value that specifies the scan line at which the memory address counter restarts at the value of 0. Bit 6 of the Maximum Scan Line Register (CR09) supplies the most significant bit, and bits 7-0 of the Line Compare Register (CR18) supply the 8 least significant bits. Normally, this 10-bit value is set to specify a scan line after the last scan line of the active display area. When this 10-bit value is set to specify a scan line within the active display area, it causes that scan line and all subsequent scan lines in the active display area to display video data starting at the very first byte of the frame buffer. The result is what appears to be a screen split into a top and bottom part, with the image in the top part being repeated in the bottom part. When used in cooperation with the Start Address High Register (CR0C) and the Start Address Low Register (CR0D), it is possible to create a split display, as described earlier, but with the top and bottom parts displaying different data. The top part will display what data exists in the frame buffer starting at the address specified in the two aforementioned start address registers, while the bottom part will display what data exists in the frame buffer starting at the first byte of the frame buffer.

Bit	Description
3	**Vertical Blanking Start Bit 8.** The vertical blanking start is a 10-bit or 12-bit value that specifies the beginning of the vertical blanking period relative to the beginning of the active display area. In standard VGA modes, where bit 0 of the I/O Control Register (CR80) is set to 0, the vertical blanking start is specified with a 10-bit value. The 8 least significant bits of this value are supplied by bits [7:0] of the Vertical Blanking Start Register (CR15), and the most and second-most significant bits are supplied by bit 5 of the Maximum Scan Line Register (CR09) and this bit of this register, respectively. In extended modes, where bit 0 of the I/O Control Register (CR80) is set to 1, the vertical blanking start is specified with a 12-bit value. The 8 least significant bits of this value are supplied by bits [7:0] of the Vertical Blanking Start Register (CR15), and the 4 most significant bits are supplied by bits [3:0] of the Extended Vertical Blanking Start Register (CR33). In extended modes, neither this bit, nor bit 5 of the Maximum Scan Line Register (CR09) are used. This 10-bit or 12-bit value should be programmed to be equal to the number of scan line from the beginning of the active display area to the beginning of the blanking period. Since the active display area always starts on the 0th scan line, this number should be equal to the number of the scan line on which the vertical blanking period begins.
2	**Vertical Sync Start Bit 8.** The vertical sync start is a 10-bit or 12-bit value that specifies the beginning of the vertical sync pulse relative to the beginning of the active display area. In standard VGA modes, where bit 0 of the I/O Control Register (CR80) is set to 0, the vertical sync start is specified with a 10-bit value. The 8 least significant bits of this value are supplied by bits [7:0] of the Vertical Sync Start Register (CR10), and the most and second-most significant bits are supplied by bit 7 and this bit, respectively, of this register. In extended modes, where bit 0 of the I/O Control Register (CR80) is set to 1, the vertical display end is specified with a 12-bit value. The 8 least significant bits of this value are supplied by bits [7:0] of the Vertical Sync Start Register (CR10), and the 4 most significant bits are supplied by bits [3:0] of the Extended Vertical Sync Start Register (CR32) register. In extended modes, neither this bit, nor bit 7 of this register are used. This 10-bit or 12-bit value should be programmed to be equal to the number of scan lines from the beginning of the active display area to the start of the vertical sync pulse. Since the active display area always starts on the 0th scan line, this number should be equal to the number of the scan line on which the vertical sync pulse begins.
1	**Vertical Display Enable End Bit 8.** The vertical display enable end is a 10-bit or 12-bit value that specifies the number of the last scan line within the active display area. In standard VGA modes, where bit 0 of the I/O Control Register (CR80) is set to 0, the vertical display enable end is specified with a 10-bit value. The 8 least significant bits of this value are supplied by bits [7:0] of the Vertical Display Enable End Register (CR12), and the most and second-most significant bits are supplied by bit 6 and this bit, respectively, of this register. In extended modes, where bit 0 of the I/O Control Register (CR80) is set to 1, the vertical display enable end is specified with a 12-bit value. The 8 least significant bits of this value are supplied by bits [7:0] of the Vertical Display Enable End Register (CR12), and the 4 most significant bits are supplied by bits [3:0] of the Extended Vertical Display End Enable Register (CR31). In extended modes, neither this bit, nor bit 6 of this register are used. This 10-bit or 12-bit value should be programmed to be equal to the number of the last scan line within in the active display area. Since the active display area always starts on the 0th scan line, this number should be equal to the total number of scan lines within the active display area, minus 1.

Bit	Description
0	**Vertical Total Bit 8.** The vertical total is a 10-bit or 12-bit value that specifies the total number of scan lines. This includes the scan lines both inside and outside of the active display area. In standard VGA modes, where bit 0 of the I/O Control Register (CR80) is set to 0, the vertical total is specified with a 10-bit value. The 8 least significant bits of this value are supplied by bits [7:0] of the Vertical Total Register (CR06), and the most and second-most significant bits are supplied by bit 5 and this bit, respectively, of this register. In extended modes, where bit 0 of the I/O Control Register (CR80) is set to 1, the vertical total is specified with a 12-bit value. The 8 least significant bits of this value are supplied by bits [7:0] of the Vertical Total Register (CR06), and the 4 most significant bits are supplied by 3-0 bits of the Extended Vertical Total Register (CR30). In extended modes, neither this bit, nor bit 5 of this register are used. This 10-bit or 12-bit value should be programmed to be equal to the total number of scan lines, minus 2.

9.6.10. CR08—Preset Row Scan Register

I/O (and Memory Offset) Address:	3B5h/3D5h (index=08h)
Default:	0UUU UUUUb (U=Undefined)
Attributes:	Read/Write

7	6 5	4 0
Reserved (0)	Byte Panning	Starting Row Scan Count

Bit	Description
7	**Reserved.** Read as 0s.
6:5	**Byte Panning.** This field holds a 2-bit value that selects number of bytes (up to 3) by which the image is shifted horizontally to the left on the screen. This function is available in both text and graphics modes. In text modes with a 9-pixel wide character box, the image can be shifted up to 27 pixels to the left, in increments of 9 pixels. In text modes with an 8-pixel wide character box, and in all standard VGA graphics modes, the image can be shifted up to 24 pixels to the left, in increments of 8 pixels. The image can be shifted still further, in increments of individual pixels, through the use of bits [3:0] of the Horizontal Pixel Panning Register (AR13). **Number of Pixels Shifted** **Bit [6:5]** / **9-Pixel Text** / **8-Pixel Text & Graphics** 00 / 0 / 0 01 / 9 / 8 10 / 18 / 16 11 / 27 / 24
4:0	**Starting Row Scan Count.** This field specifies which horizontal line of pixels within the character boxes of the characters used on the top-most row of text on the display will be used as the top-most scan line. The horizontal lines of pixels of a character box are numbered from top to bottom, with the top-most line of pixels being number 0. If a horizontal line of the these character boxes other than the top-most line is specified, then the horizontal lines of the character box above the specified line of the character box will not be displayed as part of the top-most row of text characters on the display. Normally, the value specified by these 5 bits should be 0, so that all of the horizontal lines of pixels within these character boxes will be displayed in the top-most row of text, ensuring that the characters in the top-most row of text do not look as though they have been cut off at the top.

9.6.11. CR09—Maximum Scan Line Register

I/O (and Memory Offset) Address:	3B5h/3D5h (index=09h)
Default:	00h
Attributes:	Read/Write

7	6	5	4 ... 0
Double Scanning	Line Cmp Bit 9	Vert Blnk Start Bit 9	Starting Row Scan Count

Bit	Description
7	**Double Scanning Enable.** 0 = Disable. When disabled, the clock to the row scan counter is equal to the horizontal scan rate. This is the normal setting for many of the standard VGA modes and all of the extended modes. 1 = Enable. When enabled, the clock to the row scan counter is divided by 2. This is normally used to allow CGA-compatible modes that have only 200 scan lines of active video data to be displayed as 400 scan lines (each scan line is displayed twice).
6	**Line Compare Bit 9.** This bit provides the most significant bit of a 10-bit value that specifies the scan line at which the memory address counter restarts at the value of 0. Bit 4 of the Overflow Register (CR07) supplies the second most significant bit, and bits 7-0 of the Line Compare Register (CR18) supply the 8 least significant bits. Normally, this 10-bit value is set to specify a scan line after the last scan line of the active display area. When this 10-bit value is set to specify a scan line within the active display area, it causes that scan line and all subsequent scan lines in the active display area to display video data starting at the very first byte of the frame buffer. The result is what appears to be a screen split into a top and bottom part, with the image in the top part being repeated in the bottom part. When used in cooperation with the Start Address High Register (CR0C) and the Start Address Low Register (CR0D), it is possible to create a split display, as described earlier, but with the top and bottom parts displaying different data. The top part will display whatever data exists in the frame buffer starting at the address specified in the two aforementioned start address registers, while the bottom part will display whatever data exists in the frame buffer starting at the first byte of the frame buffer.
5	**Vertical Blanking Start Bit 9.** The vertical blanking start is a 10-bit or 12-bit value that specifies the beginning of the vertical blanking period relative to the beginning of the active display area. In standard VGA modes, where bit 0 of the I/O Control Register (CR80) is set to 0, the vertical blanking start is specified with a 10-bit value. The 8 least significant bits of this value are supplied by bits [7:0] of the Vertical Blanking Start Register (CR15), and the most and second-most significant bits are supplied by this bit and bit 3 of the Overflow Register (CR09), respectively. In extended modes, where bit 0 of the I/O Control Register (CR80) is set to 1, the vertical blanking start is specified with a 12-bit value. The 8 least significant bits of this value are supplied by bits [7:0] of the Vertical Blanking Start Register (CR15), and the 4 most significant bits are supplied by bits [3:0] of the Extended Vertical Blanking Start Register (CR33). In extended modes, neither this bit, nor bit 3 of the Overflow Register (CR09) are used. This 10-bit or 12-bit value should be programmed to be equal to the number of scan line from the beginning of the active display area to the beginning of the blanking period. Since the active display area always starts on the 0th scan line, this number should be equal to the number of the scan line on which the vertical blanking period begins.
4:0	**Starting Row Scan Count.** This field provides all 5 bits of a 5-bit value that specifies the number of scan lines in a horizontal row of text. This value should be programmed to be equal to the number of scan lines in a horizontal row of text, minus 1.

9.6.12. CR0A—Text Cursor Start Register

I/O (and Memory Offset) Address:	3B5h/3D5h (index=0Ah)
Default:	00UU UUUUb (U=Undefined)
Attributes:	Read/Write

This cursor is the text cursor that is part of the VGA standard, and should not be confused with the hardware cursor and popup (a.k.a., cursor and cursor 2), which are intended to be used in graphics modes. This text cursor exists only in text modes, and thus, this register is entirely ignored in graphics modes.

7 6	5	4 0
Reserved (00)	Text Cursor Off	Text Cursor Start

Bit	Description
7:6	**Reserved.** Read as 0s.
5	**Text Cursor Off.** 0 = Enables the text cursor. 1 = Disables the text cursor.
4:0	**Text Cursor Start.** This field specifies which horizontal line of pixels in a character box is to be used to display the first horizontal line of the cursor in text mode. The horizontal lines of pixels in a character box are numbered from top to bottom, with the top-most line being number 0. The value specified by these 5 bits should be the number of the first horizontal line of pixels on which the cursor is to be shown.

9.6.13. CR0B—Text Cursor End Register

I/O (and Memory Offset) Address:	3B5h/3D5h (index=0Bh)
Default:	0UUU UUUUb (U=Undefined)
Attributes:	Read/Write

This cursor is the text cursor that is part of the VGA standard, and should not be confused with the hardware cursor and popup (a.k.a., cursor and cursor 2), which are intended to be used in graphics modes. This text cursor exists only in text modes, and so this register is entirely ignored in graphics modes.

7	6 5	4 0
Reserved	Text Cursor Skew	Text Cursor End

Bit	Description
7	**Reserved.** Read as 0s.
6:5	**Text Cursor Skew.** This field specifies the degree to which the start and end of each horizontal line of pixels making up the cursor is delayed to compensate for internal pipeline delays. These 2 bits describe the delay in terms of a number character clocks. 00 = No delay 01 = Delayed by 1 character clock 10 = Delayed by 2 character clocks 11 = Delayed by 3 character clocks
4:0	**Text Cursor End.** This field specifies which horizontal line of pixels in a character box is to be used to display the last horizontal line of the cursor in text mode. The horizontal lines of pixels in a character box are numbered from top to bottom, with the top-most line being number 0. The value specified by these 5 bits should be the number of the last horizontal line of pixels on which the cursor is to be shown.

9.6.14. CR0C—Start Address High Register

I/O (and Memory Offset) Address:	3B5h/3D5h (index=0Ch)
Default:	Undefined
Attributes:	Read/Write

Bit	Description
7:0	**Start Address Bits [15:8] or [17:10].** This register provides either bits 15 through 8 of a 16-bit value that specifies the memory address offset from the beginning of the frame buffer, or bits 17 through 10 of a 32-bit buffer address at which the data to be shown in the active display area begins. (default is 0) In standard VGA modes, where bit 0 of the I/O Control Register (CR80) is set to 0, the start address is specified with a 16-bit value. The eight bits of this register provide the eight most significant bits of this value, while the eight bits of the Start Address Low Register (CR0D) provide the eight least significant bits. In extended modes, where bit 0 of the I/O Control Register (CR80) is set to 1, the start address is specified with a 32-bit value. Bits 31 through 24 of this value are provided by the Extended Start Address High Register (CR42). Bits 23 through 18 of this value are provided by bits 5 through 0 of the Extended Start Address Register (CR40). Bits 17 through 10 of this value are provided by this register. Bits 9 through 2 of this value are provided by the Start Address Low Register (CR0D). Bits 1 and 0 of this value are always 0, and therefore not provided. It should be further noted that, in extended modes, these 32 bits from these four registers are double-buffered and synchronized to VSYNC to ensure that changes occurring on the screen as a result of changes in the start address always have a smooth or instantaneous appearance. To change the start address in extended modes, all four registers must be set for the new value, and then bit 7 of the Extended Start Address Register (CR40) must be set to 1. Only if this is done, will the hardware update the start address on the next VSYNC. When this update has been performed, the hardware will set bit 7 of the Extended Start Address Register (CR40) back to 0.

9.6.15. CR0D—Start Address Low Register

I/O (and Memory Offset) Address: 3B5h/3D5h (index=0Dh)
Default: Undefined
Attributes: Read/Write

Bit	Description
7:0	**Start Address Bits [7:0] or [9:2].** This register provides either bits 7 through 0 of a 16 bit value that specifies the memory address offset from the beginning of the frame buffer, or bits 9 through 2 of a 32 bit buffer address at which the data to be shown in the active display area begins. (default is 0) In standard VGA modes, where bit 0 of the I/O Control Register (CR80) is set to 0, the start address is specified with a 16-bit value. The eight bits of the Start Address High Register (CR0C) provide the eight most significant bits of this value, while the eight bits of this register provide the eight least significant bits. In extended modes, where bit 0 of the I/O Control Register (CR80) is set to 1, the start address is specified with a 32-bit value. Bits 31 through 24 of this value are provided by the Extended Start Address High Register (CR42). Bits 23 through 18 of this value are provided by bits 5 through 0 of the Extended Start Address Register (CR40). Bits 17 through 10 of this value are provided by the Start Address High Register (CR0C). Bits 9 through 2 of this value are provided by this register. Bits 1 and 0 of this value are always 0, and therefore not provided. It should be further noted that, in extended modes, these 32 bits from these four registers are double-buffered and synchronized to VSYNC to ensure that changes occurring on the screen as a result of changes in the start address always have a smooth or instantaneous appearance. To change the start address in extended modes, all three registers must be set for the new value, and then bit 7 of the Extended Start Address Register (CR40) must be set to 1. Only if this is done, will the hardware update the start address on the next VSYNC. When this update has been performed, the hardware will set bit 7 of the Extended Start Address Register (CR40) back to 0.

9.6.16. CR0E—Text Cursor Location High Register

I/O (and Memory Offset) Address: 3B5h/3D5h (index=0Eh)
Default: Undefined
Attributes: Read/Write

This cursor is the text cursor that is part of the VGA standard, and should not be confused with the hardware cursor and popup (a.k.a., cursor and cursor 2), which are intended to be used in graphics modes. This text cursor exists only in text modes, and so this register is entirely ignored in graphics modes.

Bit	Description
7:0	**Text Cursor Location Bits [15:8].** This field provides the 8 most significant bits of a 16-bit value that specifies the address offset from the beginning of the frame buffer at which the text cursor is located. Bits 7:0 of the Text Cursor Location Low Register (CR0F) provide the 8 least significant bits.

9.6.17. CR0F—Text Cursor Location Low Register

I/O (and Memory Offset) Address:	3B5h/3D5h (index=0Fh)
Default:	Undefined
Attributes:	Read/Write

This cursor is the text cursor that is part of the VGA standard, and should not be confused with the hardware cursor and popup (a.k.a., cursor and cursor 2), which are intended to be used in graphics modes. This text cursor exists only in text modes, and so this register is entirely ignored in graphics modes.

Bit	Description
7:0	**Text Cursor Location Bits [7:0].** This field provides the 8 least significant bits of a 16-bit value that specifies the address offset from the beginning of the frame buffer at which the text cursor is located. Bits 7:0 of the Text Cursor Location High Register (CR0D) provide the 8 most significant bits.

9.6.18. CR10—Vertical Sync Start Register

I/O (and Memory Offset) Address:	3B5h/3D5h (index=10h)
Default:	Undefined
Attributes:	Read/Write

Bit	Description
7:0	**Vertical Sync Start Bits [7:0].** This register provides the 8 least significant bits of either a 10-bit or 12-bit value that specifies the beginning of the vertical sync pulse relative to the beginning of the active display area of a screen. In standard VGA modes, where bit 0 of the I/O Control Register (CR80) is set to 0, this value is described in 10 bits with bits [7,2] of the Overflow Register (CR07) supplying the 2 most significant bits. In extended modes, where bit 0 of the I/O Control Register (CR80) is set to 1, this value is described in 12 bits with bits [3:0] of the Extended Vertical Sync Start Register (CR32) supplying the 4 most significant bits. This 10-bit or 12-bit value should equal the vertical sync start in terms of the number of scan lines from the beginning of the active display area to the beginning of the vertical sync pulse. Since the active display area always starts on the 0th scan line, this number should be equal to the number of the scan line on which the vertical sync pulse begins.

9.6.19. CR11—Vertical Sync End Register

I/O (and Memory Offset) Address:	3B5h/3D5h (index=11h)
Default:	0U00 UUUUb (U=Undefined)
Attributes:	Read/Write

7	6	5	4	3	0
Protect Regs 0:7	Reserved	Vert Int Enable	Vert Int Clear	Vertical Sync End	

Bit	Description
7	**Protect Registers [0:7].** Note that the ability to write to Bit 4 of the Overflow Register (CR07) is not affected by this bit (i.e., bit 4 of the Overflow Register is always writeable). 0 = Enable writes to registers CR[00:07]. (default) 1 = Disable writes to registers CR[00:07].
6	**Reserved.** In the VGA standard, this bit was used to switch between 3 and 5 frame buffer refresh cycles during the time required to draw each horizontal line.
5	**Vertical Interrupt Enable.** Note that the graphics controller does not provide an interrupt signal which would be connected to an input of the system's interrupt controller. Bit 7 of Input Status Register 0 (ST00) indicates the status of the vertical retrace interrupt, and can be polled by software to determine if a vertical retrace interrupt has taken place. Bit 4 of this register can be used to clear a pending vertical retrace interrupt. 0 = Enable the generation of an interrupt at the beginning of each vertical retrace period. 1 = Disable the generation of an interrupt at the beginning of each vertical retrace period.
4	**Vertical Interrupt Clear.** Note that the graphics controller does not provide an interrupt signal which would be connected to an input of the system's interrupt controller. Bit 7 of Input Status Register 0 (ST00) indicates the status of the vertical retrace interrupt, and can be polled by software to determine if a vertical retrace interrupt has taken place. Bit 5 of this register can be used to enable or disable the generation of vertical retrace interrupts. 0 = Setting this bit to 0 clears a pending vertical retrace interrupt. This bit must be set back to 1 to enable the generation of another vertical retrace interrupt.
3:0	**Vertical Sync End.** This 4-bit field provides a 4-bit value that specifies the end of the vertical sync pulse relative to its beginning. This 4-bit value should be set to the least significant 4 bits of the result of adding the length of the vertical sync pulse in terms of the number of scan lines that occur within the length of the vertical sync pulse to the value that specifies the beginning of the vertical sync pulse (see the description of the Vertical Sync Start Register for more details).

9.6.20. CR12—Vertical Display Enable End Register

I/O (and Memory Offset) Address:	3B5h/3D5h (index=12h)
Default:	Undefined
Attributes:	Read/Write

Bit	Description
7:0	**Vertical Display Enable End Bits [7:0].** This register provides the 8 least significant bits of either a 10-bit or 12-bit value that specifies the number of the last scan line within the active display area. In standard VGA modes, where bit 0 of the I/O Control Register (CR80) is set to 0, this value is described in 10 bits with bits [6,1] of the Overflow Register (CR07) supplying the 2 most significant bits. In extended modes, where bit 0 of the I/O Control Register (CR80) is set to 1, this value is described in 12 bits with bits [3:0] of the Extended Vertical Display Enable End Register (CR31) supplying the 4 most significant bits. This 10-bit or 12-bit value should be programmed to be equal to the number of the last scan line within in the active display area. Since the active display area always starts on the 0th scan line, this number should be equal to the total number of scan lines within the active display area, minus 1.

9.6.21. CR13—Offset Register

I/O (and Memory Offset) Address:	3B5h/3D5h (index=13h)
Default:	Undefined
Attributes:	Read/Write

In High Resolution modes the Start Addresses in CR0C, CR0D, CR40, and CR42 must be programmed before CR13. This is not a requirement in VGA mode.

Bit	Description
7:0	**Offset Bits [7:0] of a 12-bit value.** This register provides the 8 least significant bits of a 12-bit value that specifies the number of words or Dwords of frame buffer memory occupied by each horizontal row of characters. Whether this value is interpreted as the number of words or Dwords is determined by the settings of the bits in the Clocking Mode Register (SR01.) This 12-bit value should be programmed to be equal to either the number of words or Dwords (depending on the setting of the SR01 register) of frame buffer memory that is occupied by each horizontal row of characters. The companion extended offset register **CR41[3:0**] specifies the 4 most significant bits of the 12-bit value. It is required of software to write both CR41[3:0] and CR13[7:0] to the desired 12-bit offset value for correct hardware operation. Where an 8-bit value is desired, for example in standard VGA mode, software must write “0000” to CR41[3:0], and the desired 8-bit value to CR13[7:0]. Note that unlike the operation of the other CRTC extension registers, CR80[0] – CRT Controller Interpretation Enable bit has no effect on CR41 and CR13.

9.6.22. CR14—Underline Location Register

I/O (and Memory Offset) Address:	3B5h/3D5h (index=14h)
Default:	0UUU UUUUb (U=Undefined)
Attributes:	Read/Write

7	6	5	4 0
Reserved (0)	DWord Mode	Count By 4	Underline Location

Bit	Description
7	**Reserved.** Read as 0s.
6	**DWord Mode.** 0 = Frame buffer addresses are interpreted by the frame buffer address decoder as being either byte addresses or word addresses, depending on the setting of bit 6 of the CRT Mode Control Register (CR17). 1 = Frame buffer addresses are interpreted by the frame buffer address decoder as being DWord addresses, regardless of the setting of bit 6 of the CRT Mode Control Register (CR17). Note that this bit is used in conjunction with bits 6 and 5 of the CRT Mode Control Register (CR17) to select how frame buffer addresses from the processor are interpreted by the frame buffer address decoder as shown below: **CR14[6]** | **CR17[6]** | **Addressing Mode** 0 | 0 | Word Mode 0 | 1 | Byte Mode 1 | 0 | DWord Mode 1 | 1 | DWord Mode
5	**Count By 4.** 0 = The memory address counter is incremented either every character clock or every other character clock, depending upon the setting of bit 3 of the CRT Mode Control Register. 1 = The memory address counter is incremented either every 4 character clocks or every 2 character clocks, depending upon the setting of bit 3 of the CRT Mode Control Register. Note that this bit is used in conjunction with bit 3 of the CRT Mode Control Register (CR17) to select the number of character clocks are required to cause the memory address counter to be incremented as shown, below: **CR14[5]** | **CR17[3]** | **Address Incrementing Interval** 0 | 0 | every character clock 0 | 1 | every 2 character clocks 1 | 0 | every 4 character clocks 1 | 1 | every 2 character clocks
4:0	**Underline Location.** This field specifies which horizontal line of pixels in a character box is to be used to display a character underline in text mode. The horizontal lines of pixels in a character box are numbered from top to bottom, with the top-most line being number 0. The value specified by these 5 bits should be the number of the horizontal line on which the character underline mark is to be shown.

9.6.23. CR15—Vertical Blanking Start Register

I/O (and Memory Offset) Address:	3B5h/3D5h (index=15h)
Default:	Undefined
Attributes:	Read/Write

Bit	Description
7:0	**Vertical Blanking Start Bits [7:0].** This register provides the 8 least significant bits of either a 10-bit or 12-bit value that specifies the beginning of the vertical blanking period relative to the beginning of the active display area of the screen. Whether this value is described in 10 or 12 bits depends on the setting of bit 0 of the I/O Control Register (CR80). In standard VGA modes, where bit 0 of the I/O Control Register (CR80) is set to 0, the vertical blanking start is specified with a 10-bit value. The most and second-most significant bits of this value are supplied by bit 5 of the Maximum Scan Line Register (CR09) and bit 3 of the Overflow Register (CR07), respectively. In extended modes, where bit 0 of the I/O Control Register (CR80) is set to 1, the vertical blanking start is specified with a 12-bit value. The 4 most significant bits of this value are supplied by bits [3:0] of the Extended Vertical Blanking Start Register (CR33). This 10-bit or 12-bit value should be programmed to be equal the number of scan lines from the beginning of the active display area to the beginning of the vertical blanking period. Since the active display area always starts on the 0th scan line, this number should be equal to the number of the scan line on which vertical blanking begins.

9.6.24. CR16—Vertical Blanking End Register

I/O (and Memory Offset) Address:	3B5h/3D5h (index=16h)
Default:	Undefined
Attributes:	Read/Write

This register provides an 8-bit value that specifies the end of the vertical blanking period relative to its beginning.

Bit	Description
7:0	**Vertical Blanking End Bits [7:0].** This 8-bit value should be set equal to the least significant 8 bits of the result of adding the length of the vertical blanking period in terms of the number of scan lines that occur within the length of the vertical blanking period to the value that specifies the beginning of the vertical blanking period (see the description of the Vertical Blanking Start Register for details).

9.6.25. CR17—CRT Mode Control

I/O (and Memory Offset) Address:	3B5h/3D5h (index=17h)
Default:	0UU0 UUUUb (U=Undefined)
Attributes:	Read/Write

7	6	5	4	3	2	1	0
CRT Ctrl Reset	Word or Byte Mode	Address Wrap	Reserved (0)	Count By 2	Horizontal Retrace Select	Select Row Scan Cntr	Compat Mode Support

Bit	Description
7	**CRT Controller Reset.** 0 = Forces horizontal and vertical sync signals to be inactive. No other registers or outputs are affected. 1 = Permits normal operation.
6	**Word Mode or Byte Mode.** 0 = The memory address counter's output bits are shifted by 1 bit position before being passed on to the frame buffer address decoder such that they are made into word-aligned addresses when bit 6 of the Underline Location Register (CR17) is set to 0. 1 = The memory address counter's output bits remain unshifted before being passed on to the frame buffer address decoder such that they remain byte-aligned addresses when bit 6 of the Underline Location Register (CR17) is set to 0. Note that this bit is used in conjunction with bits 6 and 5 of the CRT Mode Control Register (CR17) to control how frame buffer addresses from the memory address counter are interpreted by the frame buffer address decoder as shown below: **CR14[6]** **CR17[6]** **Addressing Mode** 0 0 Word Mode—Addresses from the memory address counter are shifted once to become word-aligned 0 1 Byte Mode—Addresses from the memory address counter are not shifted 1 0 DWord Mode—Addresses from the memory address counter are shifted twice to become DWord-aligned 1 1 DWord Mode—Addresses from the memory address counter are shifted twice to become DWord-aligned
5	**Address Wrap.** Note that this bit is only effective when word mode is made active by setting bit 6 in both the Underline Location Register and this register to 0. 0 = Wrap frame buffer address at 16 KB. This is used in CGA-compatible modes. 1 = No wrapping of frame buffer addresses.
4	**Reserved.** Read as 0s.

Bit	Description
3	**Count By 2.** This bit is used in conjunction with bit 5 of the Underline Location Register (CR14) to select the number of character clocks are required to cause the memory address counter to be incremented. 0 = The memory address counter is incremented either every character clock or every 4 character clocks, depending upon the setting of bit 5 of the Underline Location Register. 1 = The memory address counter is incremented either every other clock. **CR14[5]** **CR17[3]** **Address Incrementing Interval** 0 0 every character clock 0 1 every 2 character clocks 1 0 every 4 character clocks 1 1 every 2 character clocks
2	**Horizontal Retrace Select.** This bit provides a way of effectively doubling the vertical resolution by allowing the vertical timing counter to be clocked by the horizontal retrace clock divided by 2 (usually, it would be undivided). 0 = The vertical timing counter is clocked by the horizontal retrace clock. 1 = The vertical timing counter is clocked by the horizontal retrace clock divided by 2.
1	**Select Row Scan Counter.** 0 = A substitution takes place, where bit 14 of the 16-bit memory address generated of the memory address counter (after the stage at which these 16 bits may have already been shifted to accommodate word or DWord addressing) is replaced with bit 1 of the row scan counter at a stage just before this address is presented to the frame buffer address decoder. 1 = No substitution takes place. See following tables.
0	**Compatibility Mode Support.** 0 = A substitution takes place, where bit 13 of the 16-bit memory address generated of the memory address counter (after the stage at which these 16 bits may have already been shifted to accommodate word or DWord addressing) is replaced with bit 0 of the row scan counter at a stage just before this address is presented to the frame buffer address decoder. 1 = No substitution takes place. See following tables.

The following tables show the possible ways in which the address bits from the memory address counter can be shifted and/or reorganized before being presented to the frame buffer address decoder. First, the address bits generated by the memory address counter are reorganized, if need be, to accommodate byte, word or DWord modes. The resulting reorganized outputs (MAOut15-MAOut0) from the memory address counter may also be further manipulated with the substitution of bits from the row scan counter (RSOut1 and RSOut0) before finally being presented to the input bits of the frame buffer address decoder (FBIn15-FBIn0).

Table 10. Memory Address Counter Address Bits [15:0]

	Byte Mode **CR14 bit 6=0** **CR17 bit 6=1** **CR17 bit 5=X**	**Word Mode** **CR14 bit 6=0** **CR17 bit 6=0** **CR17 bit 5=1**	**Word Mode** **CR14 bit 6=0** **CR17 bit 6=0** **CR17 bit 5=0**	**DWord Mode** **CR14 bit 6=1** **CR17 bit 6=X** **CR17 bit 5=X**
MAOut0	0	15	13	12
MAOut1	1	0	0	13
MAOut2	2	1	1	0
MAOut3	3	2	2	1
MAOut4	4	3	3	2
MAOut5	5	4	4	3
MAOut6	6	5	5	4
MAOut7	7	6	6	5
MAOut8	8	7	7	6
MAOut9	9	8	8	7
MAOut10	10	9	9	8
MAOut11	11	10	10	9
MAOut12	12	11	11	10
MAOut13	13	12	12	11
MAOut14	14	13	13	12
MAOut15	15	14	14	13

X = Don’t Care

Table 11. Frame Buffer Address Decoder

	CR17 bit 1=1	CR17 bit 1=1	CR17 bit 1=0	CR17 bit 1=0
	CR17 bit 0=1	CR17 bit 0=0	CR17 bit 0=1	CR17 bit 0=0
FBIn0	MAOut0	MAOut0	MAOut0	MAOut0
FBIn1	MAOut1	MAOut1	MAOut1	MAOut1
FBIn2	MAOut2	MAOut2	MAOut2	MAOut2
FBIn3	MAOut3	MAOut3	MAOut3	MAOut3
FBIn4	MAOut4	MAOut4	MAOut4	MAOut4
FBIn5	MAOut5	MAOut5	MAOut5	MAOut5
FBIn6	MAOut6	MAOut6	MAOut6	MAOut6
FBIn7	MAOut7	MAOut7	MAOut7	MAOut7
FBIn8	MAOut8	MAOut8	MAOut8	MAOut8
FBIn9	MAOut9	MAOut9	MAOut9	MAOut9
FBIn10	MAOut10	MAOut10	MAOut10	MAOut10
FBIn11	MAOut11	MAOut11	MAOut11	MAOut11
FBIn12	MAOut12	MAOut12	MAOut12	MAOut12
FBIn13	MAOut13	MAOut13	RSOut0	RSOut0
FBIn14	MAOut14	RSOut1	MAOut14	RSOut1
FBIn15	MAOut15	MAOut15	MAOut15	MAOut15

9.6.26. CR18—Line Compare Register

I/O (and Memory Offset) Address:	3B5h/3D5h (index=18h)
Default:	Undefined
Attributes:	Read/Write

Bit	Description
7:0	**Line Compare Bits [7:0].** This register provides the 8 least significant bits of a 10-bit value that specifies the scan line at which the memory address counter restarts at the value of 0. Bit 6 of the Maximum Scan Line Register (CR09) supplies the most significant bit, and bit 4 of the Overflow Register (CR07) supplies the second most significant bit. Normally, this 10-bit value is set to specify a scan line after the last scan line of the active display area. When this 10-bit value is set to specify a scan line within the active display area, it causes that scan line and all subsequent scan lines in the active display area to display video data starting at the very first byte of the frame buffer. The result is what appears to be a screen split into a top and bottom part, with the image in the top part being repeated in the bottom part. (This register is only used in split screening modes, and this is not a problem because split screening is not actually used for extended modes. As a result, there is no benefit to extending the existing overflow bits for higher resolutions.) When used in cooperation with the Start Address High Register (CR0C) and the Start Address Low Register (CR0D), it is possible to create a split display, as described earlier, but with the top and bottom parts displaying different data. The top part will display whatever data exists in the frame buffer starting at the address specified in the two aforementioned start address registers, while the bottom part will display whatever data exists in the frame buffer starting at the first byte of the frame buffer.

9.6.27. CR22—Memory Read Latch Data Register

I/O (and Memory Offset) Address:	3B5h/3D5h (index=22h)
Default:	00h
Attributes:	Read Only

Bit	Description
7:0	**Memory Read Latch Data.** This field provides the value currently stored in 1 of the 4 memory read latches. Bits 1 and 0 of the Read Map Select Register (GR04) select which of the 4 memory read latches may be read via this register.

9.6.28. CR24— Test Register for Toggle State of Attribute Controller Register

I/O (and Memory Offset) Address:	3B5h/3D5h (index=24h)
Default:	00h
Attributes:	Read Only

7	6 … 0
Toggle Status	Reserved (0000000)

Bit	Description
7	**Toggle Status.** Last write to attribute register was to: 0 = index port 1 = data port
6:0	**Reserved.** Read as 0s.

9.6.29. CR30—Extended Vertical Total Register

I/O (and Memory Offset) Address:	3B5h/3D5h (index=30h)
Default:	00h
Attributes:	Read/Write

7 … 4	3 … 0
Reserved (0000)	Vertical Total Bits 11:8

Bit	Description
7:4	**Reserved.** Read as 0s. This field must be 0s when this register is written.
3:0	**Vertical Total Bits [11:8].** The vertical total is a 10-bit or 12-bit value that specifies the total number of scan lines. This includes the scan lines both inside and outside of the active display area. In standard VGA modes, where bit 0 of the I/O Control Register (CR80) is set to 0, the vertical total is specified with a 10-bit value. The 8 least significant bits of this value are supplied by bits [7:0] of the Vertical Total Register (CR06), and the 2 most significant bits are supplied by bits 5 and 0 of the Overflow Register (CR07). In standard VGA modes, these 4 bits of this register are not used. In extended modes, where bit 0 of the I/O Control Register (CR80) is set to 1, the vertical total is specified with a 12-bit value. The 8 least significant bits of this value are supplied by bits [7:0] of the Vertical Total Register (CR06), and the 4 most significant bits are supplied by these 4 bits of this register. This 10-bit or 12-bit value should be programmed to be equal to the total number of scan lines, minus 2.

9.6.30. CR31—Extended Vertical Display End Register

I/O (and Memory Offset) Address:	3B5h/3D5h (index=31h)
Default:	00h
Attributes:	Read/Write

7 4	3 0
Reserved (0000)	Vertical Display End Bits 11:8

Bit	Description
7:4	**Reserved.** Read as 0s. This field must be 0s when this register is written.
3:0	**Vertical Display End Bits [11:8].** The vertical display enable end is a 10-bit or 12-bit value that specifies the number of the last scan line within the active display area. In standard VGA modes, where bit 0 of the I/O Control Register (CR80) is set to 0, the vertical display enable end is specified with a 10-bit value. The 8 least significant bits of this value are supplied by bits [7:0] of the Vertical Display Enable End Register (CR12), and the 2 most significant bits are supplied by bits 6 and 1 of the Overflow Register (CR07). In standard VGA modes these 4 bits of this register are not used. In extended modes, where bit 0 of the I/O Control Register (CR80) is set to 1, the vertical display enable end is specified with a 12-bit value. The 8 least significant bits of this value are supplied by bits [7:0] of the Vertical Display Enable End Register (CR12), and the 4 most significant bits are supplied by these 4 bits of this register. This 10-bit or 12-bit value should be programmed to be equal to the number of the last scan line within in the active display area. Since the active display area always starts on the 0th scan line, this number should be equal to the total number of scan lines within the active display area, minus 1.

9.6.31. CR32—Extended Vertical Sync Start Register

I/O (and Memory Offset) Address:	3B5h/3D5h (index=32h)
Default:	00h
Attributes:	Read/Write

7 4	3 0
Reserved (0000)	Vertical Sync Start Bits 11:8

Bit	Description
7:4	**Reserved.** Read as 0s. This field must be 0s when this register is written.
3:0	**Vertical Sync Start Bits [11:8].** The vertical sync start is a 10-bit or 12-bit value that specifies the beginning of the vertical sync pulse relative to the beginning of the active display area. In standard VGA modes, where bit 0 of the I/O Control Register (CR80) is set to 0, the vertical sync start is specified with a 10-bit value. The 8 least significant bits of this value are supplied by bits [7:0] of the Vertical Sync Start Register (CR10), and the 2 most significant bits are supplied by bits 7 and 2 of the Overflow Register (CR07). In standard VGA modes, these 4 bits of this register are not used. In extended modes, where bit 0 of the I/O Control Register (CR80) is set to 1, the vertical display end is specified with a 12-bit value. The 8 least significant bits of this value are supplied by bits [7:0] of the Vertical Sync Start Register (CR10), and the 4 most significant bits are supplied by these 4 bits of this register. This 10-bit or 12-bit value should be programmed to be equal to the number of scan lines from the beginning of the active display area to the start of the vertical sync pulse. Since the active display area always starts on the 0th scan line, this number should be equal to the number of the scan line on which the vertical sync pulse begins.

9.6.32. CR33—Extended Vertical Blanking Start Register

I/O (and Memory Offset) Address:	3B5h/3D5h (index=33h)
Default:	00h
Attributes:	Read/Write

7 ... 4	3 ... 0
Reserved (0000)	Vertical Blanking Start Bits 11:8

Bit	Description
7:4	**Reserved.** Read as 0s. This field must be 0s when this register is written.
3:0	**Vertical Blanking Start Bits [11:8].** The vertical blanking start is a 10-bit or 12-bit value that specifies the beginning of the vertical blanking period relative to the beginning of the active display area. In standard VGA modes, where bit 0 of the I/O Control Register (CR80) is set to 0, the vertical blanking start is specified with a 10-bit value. The 8 least significant bits of this value are supplied by bits [7:0] of the Vertical Blanking Start Register (CR15), and the most and second-most significant bits are supplied by bit 5 of the Maximum Scan Line Register (CR09) and bit 3 of the Overflow Register (CR07), respectively. In standard VGA modes, these four bits are not used. In extended modes, where bit 0 of the I/O Control Register (CR80) is set to 1, the vertical blanking start is specified with a 12-bit value. The 8 least significant bits of this value are supplied by bits {7:0] of the Vertical Blanking Start Register (CR15), and the 4 most significant bits are supplied by these 4 bits of this register. This 10-bit or 12-bit value should be programmed to be equal to the number of scan line from the beginning of the active display area to the beginning of the blanking period. Since the active display area always starts on the 0th scan line, this number should be equal to the number of the scan line on which the vertical blanking period begins.

9.6.33. CR35— Extended Horizontal Total Time Register

I/O (and Memory Offset) Address:	3B5h/3D5h (index=35h)
Default:	00h
Attributes:	Read/Write

7 … 1	0
Reserved (0000000)	Ext Horiz Total

Bit	Description
7:1	**Reserved.** When this register is written to, these bits should be set to 0.
0	**Extended Horizontal Total (MSB that extends CR00).**

9.6.34. CR39—Extended Horizontal Blank Time Register

I/O (and Memory Offset) Address:	3B5h/3D5h (index=39h)
Default:	00h
Attributes:	Read/Write

7 … 1	0
Reserved (0000000)	Ext Horiz Total

Bit	Description
7:1	**Reserved.**
0	**Extended Horizontal Total (MSB that extends CR5[7], CR3[4:0]).**

9.6.35. CR40—Extended Start Address Register

I/O (and Memory Offset) Address: 3B5h/3D5h (index=40h)
Default: 00h
Attributes: Read/Write

7	6	5 … 0
Start Addr Enable	Reserved (0)	Start Address Bits 23:18

Bit	Description
7	**Extended Mode Start Address Enable.** This bit is used only in extended modes, where bit 0 of the I/O Control Register (CR80) is set to 1, to signal the graphics controller to update the start address. In extended modes, the start address is specified with a 30 bit value. These 30 bits, which are provided by the Start Address Low Register (CR0D), the Start Address High Register (CR0C), the Extended Start Address High Register (CR42) and bits [5:0] of this register, are double-buffered and synchronized to VSYNC to ensure that changes occurring on the screen as a result of changes in the start address always have a smooth or instantaneous appearance. To change the start address in extended modes, all three registers must be set for the new value, and then this bit of this register must be set to 1. Only if this is done, will the graphics controller update the start address on the next VSYNC. When this update has been performed, the graphics controller will set bit 7 of this register back to 0.
6	**Reserved.** Read as 0s. This field must be 0s when this register is written.
5:0	**Start Address Bits [23:18].** This start address is a 16 bit value that specifies the memory address offset from the beginning of the frame buffer, or a 32 bit buffer address at which the data to be shown in the active display area begins. (default is 0) In standard VGA modes, where bit 0 of the I/O Control Register (CR80) is set to 0, the start address is specified with a 16-bit value. The eight bits of the Start Address High Register (CR0C) provide the eight most significant bits of this value, while the eight bits of the Start Address Low Register (CR0D) provide the eight least significant bits. In extended modes, where bit 0 of the I/O Control Register (CR80) is set to 1, the start address is specified with a 32-bit value. Bits 31:24 of this value are provided by the Extended Start Address High Register (CR42). Bits 23:18 of this value are provided by bits 5:0 of this register. Bits 17:10 of this value are provided by the Start Address High Register (CR0C). Bits 9:2 of this value are provided by the Start Address Low Register (CR0D). Bits 1:0 of this value are always 0, and therefore not provided. Note that, in extended modes, these 32 bits from these four registers are double-buffered and synchronized to VSYNC to ensure that changes occurring on the screen as a result of changes in the start address always have a smooth or instantaneous appearance. To change the start address in extended modes, all four registers must be set for the new value, and then bit 7 of this register must be set to 1. Only if this is done, will the graphics controller update the start address on the next VSYNC. When this update has been performed, the graphics controller will set bit 7 of this register back to 0.

9.6.36. CR41—Extended Offset Register

I/O (and Memory Offset) Address: 3B5h/3D5h (index=41h)
Default: 00h
Attributes: Read/Write

7 … 4	3 … 0
Reserved (0000)	Offset Bits 11:8

Bit	Description
7:4	**Reserved.** Read as 0's. This field must be 0's when this register is written.
3:0	**Offset Bits [11:8] of a 12-bit value.** This register provides the 4 most significant bits of a 12-bit value that specifies the number of words or Dwords of frame buffer memory occupied by each horizontal row of characters. Whether this value is interpreted as the number of words or Dwords is determined by the settings of the bits in the Clocking Mode Register (SR01.) This 12-bit value should be programmed to be equal to either the number of words or Dwords (depending on the setting of the SR01 register) of frame buffer memory that is occupied by each horizontal row of characters. The companion offset register **CR13[7:0**] specifies the 8 least significant bits of the 12-bit value. It is required of software to write both CR41[3:0] and CR13[7:0] to the desired 12-bit offset value for correct hardware operation. Where an 8-bit value is desired, for example in standard VGA mode, software must write "0000" to CR41[3:0], and the desired 8-bit value to CR13[7:0]. Note that unlike the operation of the other CRTC extension registers, CR80[0] – CRT Controller Interpretation Enable bit has no effect on CR41 and CR13.

9.6.37. CR42—Extended Start Address High Register

I/O (and Memory Offset) Address: 3B5h/3D5h (index=42h)
Default: 00h
Attributes: Read/Write

Bit	Description
7:0	**Start Address High Bits [31:24].** This register provides bits [31:24] of a 32-bit buffer address that the data to be shown in the active display area begins. (default is 0) In standard VGA modes, where bit 0 of the I/O Control Register (CR80) is set to 0, the start address is specified with a 16-bit value. The eight bits of the Start Address High Register (CR0C) provide the eight most significant bits of this value, while the eight bits of the CR0D register provide the eight least significant bits. In extended modes, where bit 0 of the I/O Control Register (CR80) is set to 1, the start address is specified with a 32-bit value. Bits [31:24] of this value are provided by this register. Bits [23:18] of this value are provided by bits [5:0] of the Extended Start Address Register (CR40). Bits [17:10] of this value are provided by the Start Address High Register (CR0C). Bits [9:2] of this value are provided by the Start Address Low Register (CR0D). Bits [1:0] of this value are always 0, and therefore not provided. It should be further noted that, in extended modes, the 30 bits from these four registers are double-buffered and synchronized to VSYNC to ensure that changes occurring on the screen as a result of changes in the start address always have a smooth or instantaneous appearance. To change the start address in extended modes, all four registers must be set for the new value, and then bit 7 of the Extended Start Address Register (CR40) must be set to 1. Only then will the graphics controller update the start address on the next VSYNC. When the update is done, the graphics controller sets bit 7 of the Extended Start Address Register (CR40) back to 0.

9.6.38. CR70—Interlace Control Register

I/O (and Memory Offset) Address:	3B5h/3D5h (index=70h)
Default:	00h
Attributes:	Read/Write

7	6 … 0
Interlace Enable	CRT Half-Line Value

Bit	Description
7	**Interlace Enable.** 0 = Selects non-interlaced CRT output (default). 1 = Selects interlaced CRT output.
6:0	**CRT Half-Line Value.** When interlaced CRT output has been selected, the value in this register specifies the position along the length of a scan line at which the half-line vertical sync pulse occurs for the odd frame. This half-line vertical sync pulse begins at a position between two horizontal sync pulses on the last scan line, rather than coincident with the beginning of a horizontal sync pulse at the end of a scan line.

9.6.39. CR80—I/O Control

I/O (and Memory Offset) Address:	3B5h/3D5h(index 80h)
Default:	00h
Attributes:	Read/Write

7 … 2	1	0
Reserved (000000)	Attr Cntl Ext Enbl	CRT Cntl Int Enbl

Bit	Description
7:2	**Reserved.**
1	**Attribute Controller Extensions Enable.** Controls whether the attribute registers are accessed with both index and data at 3C0h (strict VGA mode), or whether they are accessed with 3c0h as the index and 3C1h is data. It is possible that BIOS software or driver software might not use the non-VGA mode. Either method should work, but it needs to be the method the software is using. 0 = Disable, i.e., strictly VGA compatible mode (default) 1 = Enable Attribute Controller extensions Index and Data of the Attribute Controller registers are accessible at 3C0h in standard VGA. When the Attribute Controller Extensions are enabled, Index and Data are accessible at addresses 3C0h and 3C1h, respectively.
0	**CRT Controller Interpretation Enable.** This bit modifies responses/functionality to registers CR30 and beyond, except for CR41. 0 = Registers have strict VGA Interpretation (default) 1 = Registers have extended VGA Interpretation (i.e., access to 4G space)

9.6.40. CR81—Reserved

I/O (and Memory Offset) Address:	3B5h/3D5h(index 81h)
Default:	00h
Attributes	Read/Write

This register is not present in 2D.

7 … 0
Reserved (00000000)

Bit	Description
7:0	**Reserved.** (This register did something useful in '554, and was not completely removed from Portola database, even though it does nothing useful).

9.6.41. CR82—Blink Rate Control

I/O (and Memory Offset) Address:	3B5h/3D5h(index 82h)
Default:	00h
Default:	83h is the VGA default, but 88h is used instead because
that's a more visually	appealing value
Attributes	Read/Write

The h/w default for this register does not match the VGA compatibility requirements. BIOS must make sure to set this register to correct value.

7 … 6	5 … 0
Character Blink Duty Cycle	VSync Blink Rate

Bit	Description
7:6	**Character Blink Duty Cycle.** (Character Blink is also known as Attribute Blink). 00 = 50% duty cycle. 01 = 25% duty cycle. 10 = 50% duty cycle (power on default).
5:0	**VSync Blink Rate.** Controls the cursor blink rate in terms of the number of VSyncs on the following basis: the programmed value must be the (actual value/2)-1. (The default is 3).

10. Programming Interface

The Graphics Controller (GC) contains an extensive set of registers and instructions (also referred to as "Commands") for controlling 2D, 3D, and video operations. This section describes the programmer's interface to these registers and instructions.

10.1. Reserved Bits and Software Compatibility

In many registers, instruction and memory layout descriptions, certain bits are marked as "Reserved". When bits are marked as reserved, it is essential for compatibility with future devices that software treat these bits as having a future, though unknown, effect. The behavior of reserved bits should be regarded as not only undefined, but also unpredictable. Software should follow these guidelines in dealing with reserved bits:

Do not depend on the states of any reserved bits when testing values of registers that contain such bits. Mask out the reserved bits before testing. Do not depend on the states of any reserved bits when storing to instruction or to a register.

When loading a register or formatting an instruction, always load the reserved bits with the values indicated in the documentation or zero.

10.2. Overview

The GC is programmed via three basic mechanisms.

POST-Time Programming of PCI Configuration Registers

These registers are programmed once during POST of the video device. The PCI Configuration registers are not covered in this PRM. For details on accessing the graphics controller's PCI configuration space, refer to the *Intel® 815 Chipset: 82815 Graphics and Memory Controller Hub* EDS. For additional information on accessing non-PCI registers refer to the *System Address Map* chapter.

Direct (Physical I/O and/or Memory-Mapped I/O) Access of GC Registers

Various GC functions can only be controlled via direct register access. In addition, direct register access is required to initiate the (asynchronous) execution of GC instruction streams. This programming mechanism is "direct" and synchronous with software execution on the processor.

Instruction Stream DMA (via Instruction Ring Buffers)

This programming mechanism utilizes the indirect (and asynchronous) execution of GC instruction streams to control certain GC functions (e.g., all 2D and 3D drawing operations). Software writes instructions into an instruction buffer (either a Ring Buffer or Batch Buffer) and informs the GC that the instructions are ready for execution. The Instruction Parser will then (at some point) read the instructions via DMA and execute them.

10.3. GC Register Programming

All of the GC registers (except for the PCI Configuration registers) are memory mapped. The base address of this 512 KB memory block is programmed in the MMADR PCI Configuration register. Note that 2D control registers (VGA and Extended VGA registers) are also located at their standard I/O locations. For more information on the registers, refer to the *System Address Map* chapter.

10.4. GC Instruction Streams

This section describes how instruction streams can be used to control GC operation and perform GC operations.

10.4.1. Instruction Use

GC instructions are used to control drawing engines and various GC functional units:

- **3D Instructions.** 3D instructions are used to program the 3D pipeline state and perform 3D rendering operations, including "StretchBlt" operations.
- **Motion Compensation.** A special "GFXBLOCK" instruction is used to perform Motion Compensation operations via the Mapping Engine.
- **2D Instructions (BLT).** These instructions are used to perform BLT operations.
- **Instruction Parser (IP) Instructions.** The IP instructions can be used to control and synchronize the instruction stream as well as perform various GC auxiliary functions (e.g., define graphics buffer attributes, perform display/overlay flips, etc.)

10.4.2. Instruction Transport Overview

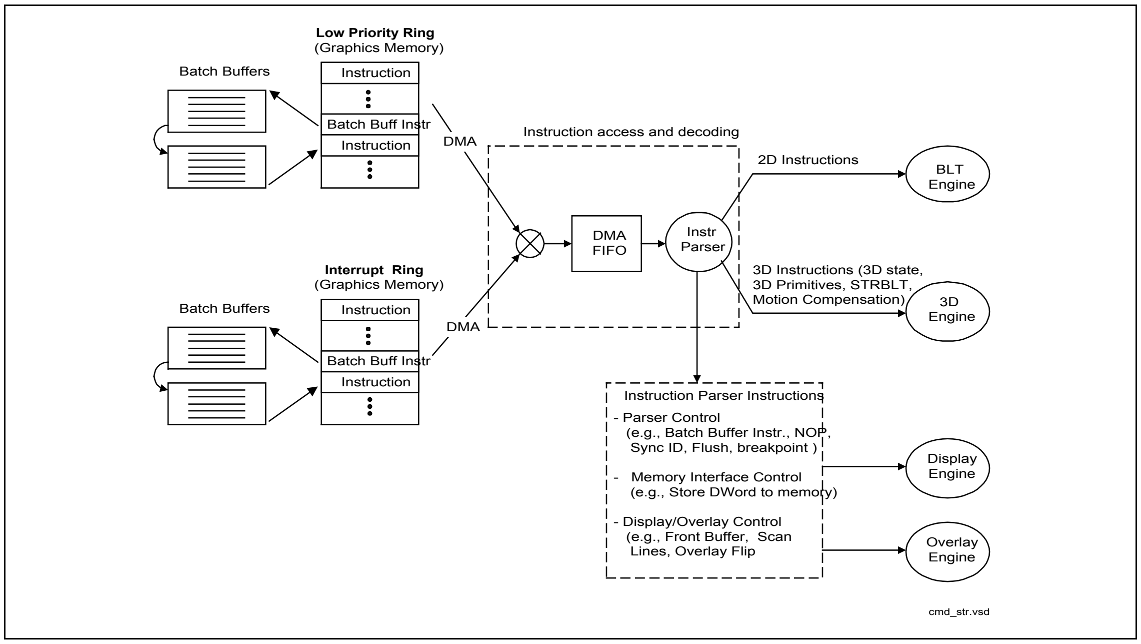
Instructions are not written directly to the GC; instead, they are placed in memory by software and later read (via DMA) by the GC's Instruction Parser unit. The primary mechanism used to transport instructions is through the use of two ring buffers (RB): Low Priority RB and Interrupt RB. A secondary mechanism for instruction transport is through the use of batch buffers. The IP uses a set of rules to determine the order in which instructions are executed. Following sections in this chapter provide descriptions of the ring buffers, batch buffers, and IP rules.

10.4.3. Instruction Parser

The following figure shows a high-level diagram of the GC instruction interface. The GC's Instruction Parser (IP) unit is responsible for:

- Detecting the presence of instructions (within the Ring Buffers)
- Arbitrating the execution of instruction streams
- Reading instructions from Ring Buffers and Batch Buffers via DMA
- Parsing the common "Client" (destination) field of instructions
- Execution of Instruction Parser instructions (which control IP functionality, provide synchronization functions as well as provide miscellaneous GC control functions)
- Redirection of 2D and 3D instructions to the appropriate destination while following drawing engine concurrency and coherency rules

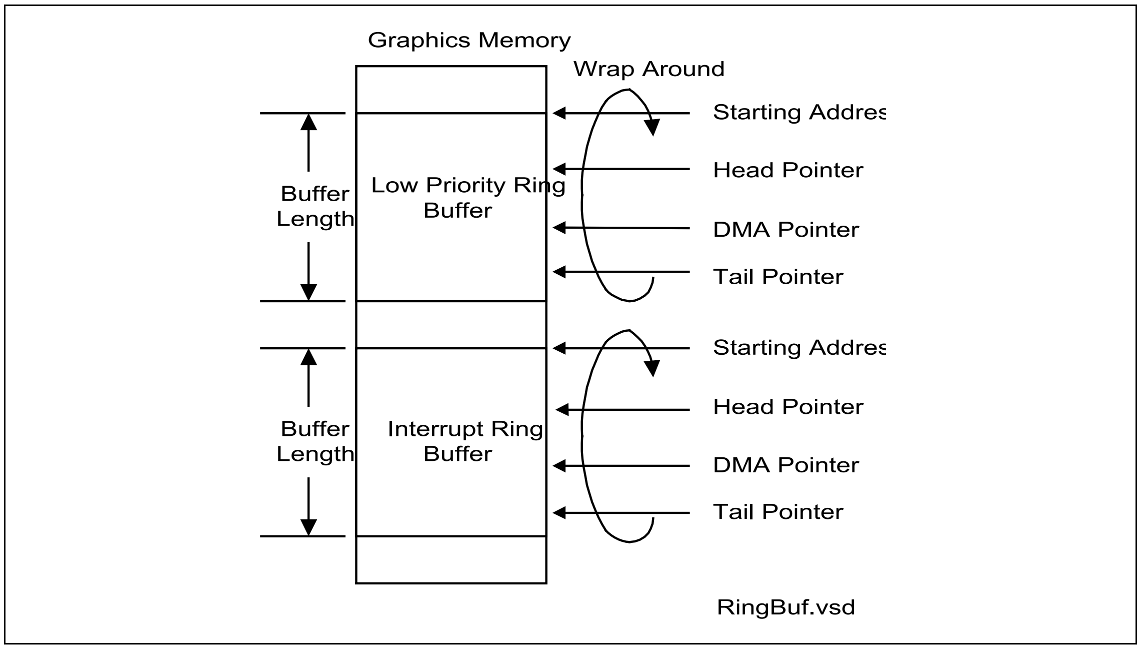
Figure 26. Graphics Controller Instruction Interface


Low Priority Ring
(Graphics Memory)
Batch Buffers
Instruction
Batch Buff Instr
Instruction
DMA
Instruction access and decoding
2D Instructions
BLT Engine
DMA FIFO
Instr Parser
Interrupt Ring
(Graphics Memory)
Batch Buffers
Instruction
Batch Buff Instr
Instruction
DMA
3D Instructions (3D state, 3D Primitives, STRBLT, Motion Compensation)
3D Engine
Instruction Parser Instructions
- Parser Control (e.g., Batch Buffer Instr., NOP, Sync ID, Flush, breakpoint)
- Memory Interface Control (e.g., Store DWord to memory)
- Display/Overlay Control (e.g., Front Buffer, Scan Lines, Overlay Flip
Display Engine
Overlay Engine
cmd_str.vsd

10.4.4. Ring Buffers (RB)

The GC provides two Ring Buffer (RB) mechanisms through which instructions can be passed to the Instruction Parser. They are referred to as the Interrupt and Low-Priority RBs, and are basically identical, except for differences in arbitration rules and priority.

Figure 27. Ring Buffers



10.4.4.1. Ring Buffer Registers

A Ring Buffer is defined by a set of two Ring Buffer registers—Low and High Priority. Before an RB can be used for instruction transport, software needs to program these registers. The fields contained within these registers are as follows:

- **Ring Buffer Valid**: This bit controls whether the RB is included in the instruction arbitration process. Software must program all other RB parameters before enabling an RB. An RB can be disabled and later re-enabled. Enabling or disabling an RB does not, of itself, change any other RB register fields.
- **Start Address:** This field points to a contiguous, 4 KB-aligned, linear (e.g., not tiled) memory address region which provides the actual instruction buffer area.
- **Buffer Length:** The size of the buffer, in 4KB increments, up to 2MB.
- **Head Offset:** This is the DWord offset (from Start Address) of the next instruction that the IP will execute. The IP will update this field as instructions are retired. (Note that, if instructions are pending execution, the IP will likely have fetched instructions past the Head Offset). As the GC does not "reset" the Head Offset when an RB is enabled, software must program the Head Offset field before enabling the Ring Buffer. Although this allows software to enable an RB with any valid values for Head/Tail (i.e., can enable or re-enable the RB with instructions already pending), it is anticipated that software will initialize the Head Offset to 0. Once the Head Offset reaches the Tail Offset (Head = Tail), the IP considers the RB empty.

- **Head Wrap Count:** This field is incremented by the IP every time the Head Offset wraps back to the start of the buffer. As it is included in the DWord written in the "report head" process, software can use this field to track IP progress as if the RB had a "virtual" length of 2048 times the size of the actual physical buffer.
- **Tail Offset:** This is the QWord offset (from Start Address) where software will write the next instruction. After writing instructions into the RB, software updates the Tail Offset field in order to submit the instructions for execution (by setting it to the QWord offset immediately following the last instruction to be submitted). The instructions submitted can wrap from the end of the buffer back to the top, in which case the Tail Offset written will be less than the previous value. Note that, since the RB empty condition is defined as "Head Offset == Tail Offset", software has to leave at least one QWord free at all times (i.e., the buffer is considered full when only one QWord is free).
- **Automatic Report Head Enable:** Software can request to have the hardware Head Pointer register contents written ("reported") to snooped system memory on a periodic basis. This is desirable as software needs to use the Head Offset to determine the amount of free space in the RB -- and having the Head Pointer periodically reported to system memory provides a fairly accurate Head Offset value automatically (i.e., without having to explicitly store a Head Offset value via an instruction). The Head Pointer register will be stored at an RB-specific displacement into the "hardware status page" (defined by the HWSTAM register).

Table 12. Ring Buffer Characteristics

Characteristic	Description
Alignment	4 KB page aligned
Max Size	2 MB
Length	Programmable in numbers of 4 KB pages
Start Pointer	Programmable page aligned address of the buffer
Head pointer	Programmable to initially setup ring Hardware maintained DWord Offset in the ring buffer. Pointer wraps
DMA pointer	Hardware maintained DMA request Double QWord Offset. Pointer wraps
Tail pointer	Programmable Double QWord Offset in the ring buffer.

10.4.4.2. Ring Buffer Initialization

Before initializing a RB, software must first allocate the desired number of 4 KB pages for use as buffer space. Then the RINGBUF registers associated with the RB are programmed. Once the Ring Buffer Valid bit is set, the RB will be considered for instruction arbitration, and the Head and Tail Offsets will either indicate an empty RB (i.e., Head == Tail), or will define some number of instructions to be executed.

10.4.4.3. Ring Buffer Use

Software can write new instructions into the "free space" of the RB, starting at the Tail Offset and up to (but not including) the QWord prior to the QWord indicated by the Head Offset. (Recall that software must leave at least one QWord empty in the RB at all times). Note that this "free space" may wrap from the end of the RB back to the start.

Software is required to use some mechanism to track instruction execution progress to determine the free space in the RB. This can be:

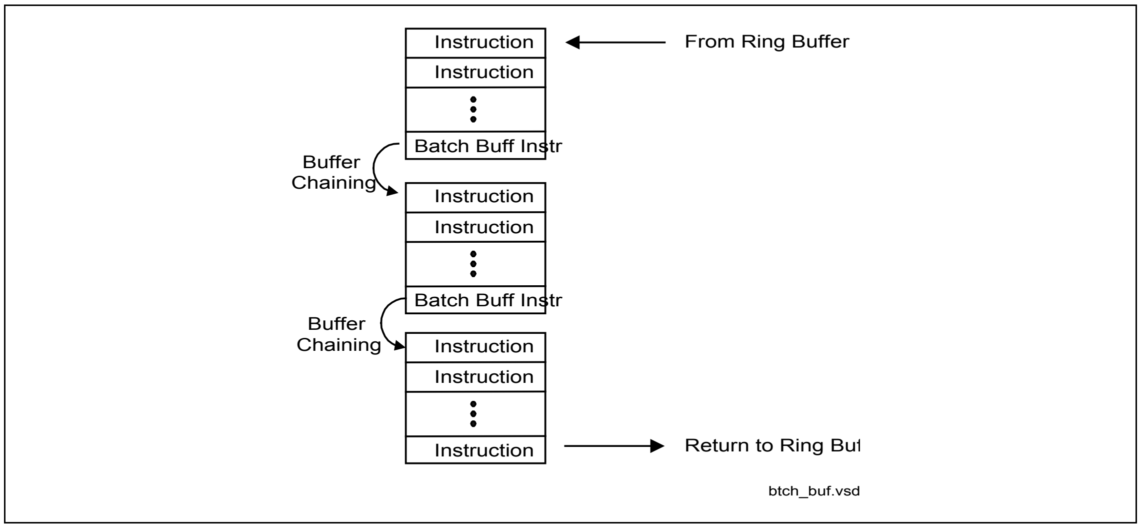
- A direct read of the Head Pointer register
- The automatic reporting of the Head Pointer register
- The explicit reporting of the Head Pointer register via the GFXCMDPARSER_REPORT_HEAD instruction
- Some other "implicit" means by which software can determine how far the IP has progressed in retiring instructions from an RB. This could include the use of "Store DWORD" instructions to write sequencing data to system memory.
- Once the instructions have been written (and padded out to a QWord, if necessary), software can write the Tail Pointer register to submit the new instructions for execution.

10.4.5. Batch Buffers

The GC provides for the execution of instruction sequences *external* to RBs. These sequences are called batch buffers, and are initiated through the use of GFXCMDPARSER_BATCH_BUFFER instructions that specify the starting address and length of the batch buffers. The arbitration rules used by the IP when executing batch buffers differ from those employed when executing RBs, and are described later in this chapter. When a batch buffer instruction is executed out of a RB, a batch buffer sequence is initiated where the GC reads the instructions sequentially (via DMA) from the batch buffer.

What happens when the end of the batch buffer is reached depends on the final instruction in the buffer. If the final instruction is a GFXCMDPARSER_BATCH_BUFFER instruction, another batch buffer sequence is initiated. This process, called "chaining", continues until a batch buffer terminates with an instruction other than GFXCMDPARSER_BATCH_BUFFER, at which point execution will resume in the RB at the instruction following the initial GFXCMDPARSER_BATCH_BUFFER.

Figure 28. Batch Buffer Sequence



10.4.6. Instruction Arbitration

The Instruction Parser supports up to four sources of pending instructions: two Ring Buffers and two Batch Buffer sequences (one batch buffer per ring buffer). The IP employs a set of rules to arbitrate among these instruction stream sources. This section describes these rules and discusses the reasoning behind them.

10.4.6.1. Arbitration Rationale

The Low Priority Ring Buffer (LPRB) is considered the primary mechanism by which drivers will pass instructions to the GC. However, the insertion of instruction sequences into the LPRB must be a synchronous operation, i.e., software must guarantee mutually exclusive access to the LPRB among contending sources (drivers). This ensures that one driver does not corrupt another driver's partially-completed instruction stream.

There are two general sources of requirements for asynchronous instruction generation and execution: interrupt handlers and contending drivers. An interrupt handler may be invoked when a driver is in the process of inserting instructions into the LPRB. To permit the interrupt handler to generate an instruction stream, the Interrupt Ring Buffer (IRB) is provided. The IRB instruction stream is considered higher priority than the LPRB stream; this priority is manifested in how the IRB is treated in the arbitration rules.

Note, however, that software retains some control over this arbitration process. The GFXCMDPARSER_ARB_ON_OFF instruction can be used from the LP instruction stream (RB or batch buffer) to temporarily disable the IRB from arbitration. This can be used to define uninterruptible critical sections in the LP stream (e.g., where some GC state needs to be protected from IRB instruction execution).

Another requirement for asynchronous instruction generation arises from competing (and asynchronous) drivers (e.g., user-mode driver libraries). In this case, the desire is to allow these entities to construct instruction sequences in an asynchronous fashion, via batch buffers. Synchronization is then only required to dispatch the batch buffers via GFXCMDPARSER_BATCH_BUFFER instructions inserted into the LPRB. Batch buffers can also be initiated from the IRB, though in this case the batch buffer is intended more for performance reasons -- where pre-generated (or "canned") instruction streams can be dispatched without having to copy them into the IRB.

LPRB batch buffers are considered non-interruptible for a number of reasons (IP complexity, GC context management, etc.). To provide some gross limits on IRB latency, batch buffers are interruptible at chain points (between the GFXCMDPARSER_BATCH_BUFFER which ends a batch and the start of the new batch). Using this mechanism, software can segment batched instruction sequences into chains of smaller batches. Batch buffers initiated from the IRB cannot, by definition, be interrupted so chaining them is of limited use.

10.4.6.2. Wait Instructions

The GFXCMDPARSER_WAIT_EVENT instruction is provided to allow instruction streams to be held pending until an asynchronous event occurs. When executed directly from a RB, the IP will treat the RB as if it were empty until the specific event occurs. This will temporarily remove that RB from arbitration. Therefore, a wait instruction placed in the IRB can allow LPRB activity to start/resume, even though there may be IRB instructions still pending.

When executed from a batch buffer, the IP will simply wait for the event to occur without performing rearbitration. As this basically halts the IP for a length of time, the use of wait instructions in batch buffers, and the impact on latency and performance, should be carefully considered by software developers.

10.4.6.3. Instruction Arbitration Points

The IP performs arbitration for instruction execution at the following points:

- Continuously when idle (i.e., no pending instructions)
- Between instructions in the LPRB
- After GFXCMDPARSER_BATCH_BUFFER instructions when executed from the LPRB or at the end of an LP batch buffer
- Upon execution of a wait instruction in the IRB (if a wait is required)

Software must consider the consequences of the IP redirecting instruction execution at these arbitration points. That is, software needs to coordinate the use and control of the instruction stream sources such that GC operations proceed in the intended order and with the intended GC state. For example, software must prevent the case where instructions placed in the IRB interrupt the LP instruction stream and invalidate a GC state required by the pending LP stream.

10.4.6.4. Instruction Arbitration Rules

At an arbitration point, the IP will consider the current state of instruction execution (i.e., Low Priority vs Interrupt, Ring vs. Batch) along with the current state of the RBs and possibly pending wait events. The IP will then determine how to proceed with execution, given that status information and the following instruction stream priorities (highest priority to lowest):

1. Next instruction in batch buffer (regardless of initiating ring buffer)
2. Next instruction pending in IRB (assuming IRB arbitration is enabled)
3. Initiation of LP batch buffer (including resumption of interrupted LP batch chain)
4. Next instruction pending in LPRB

10.4.6.5. Batch Buffer Protected Mode

To ensure that the graphics controller does not corrupt system memory or graphics memory through invalid instructions from a batch buffer sequence, the batch buffer instruction has a flag that can be set to indicate that it is from a non-trusted source.

When the IP processes a non-trusted batch buffer from one of the ring buffers, it does not allow any immediate store DWord instructions, because this instruction causes writes to system memory, not gathered through the GTT. The protection mode (Protected or Unprotected) is set in the batch buffer instruction that is in the ring buffer. The protection mode set persists throughout the batch buffer sequence; including batch buffers that are chained. Thus, a chained batch buffer cannot re-enable writes to system memory.

If the IP detects an instruction that is disallowed in protected mode, it stores the header of the instruction, the origin of the instruction, and an error code. In addition, such an event can generate an interrupt or a hardware write to system memory, if enabled and unmasked. At this point the IP, can only be reactivated by a **reset**.

10.5. Instruction Format

GC instructions are defined with various formats. The first DWord of all instructions is called the header DWord. The header contains the only field common to all instructions: the client field that determines the major GC unit that will process the instruction data. The Instruction Parser examines the client field of each instruction to condition the further processing of the instruction and route the instruction data accordingly. Valid client values are:

- Instruction Parser (00h)
- 2D Processor (02h)
- 3D Processor (03h)

GC instructions vary in length, though are always multiples of DWords. The length of an instruction is either:

- implied by the client/opcode
- fixed by the client/opcode yet included in a header field (so the IP explicitly knows how much data to copy/process)
- variable, with a field in the header indicating the total length of the instruction

Note that GC instruction *sequences* require QWord alignment and padding to QWord length to be placed in Ring and Batch Buffers.

The following subsections provide a brief overview of the GC instructions by client type. Figure 29 provides a diagram of the formats of the header DWords for all GC instructions. Table 13 provides a list of instruction mnemonics by client type.

10.5.1. Instruction Parser Instructions

Instruction Parser (IP) instructions are basically those instructions which do not require processing by the 2D or 3D Rendering/Mapping engines. The functions performed by these instructions include:

- Control of the instruction stream (e.g., Batch Buffer commands, breakpoints, ARB On/Off, etc.)
- Hardware synchronization (e.g., flush, wait-for-event)
- Software synchronization (e.g., Store DWORD, report head)
- Graphics buffer definition (e.g., Display buffer, Overlay buffer, 3D Destination and Z buffer)
- Miscellaneous functions

Refer to the *Instruction Parser Instructions* Chapter for a description of these instructions.

10.5.2. 2D Instructions

The 2D instructions include various flavors of Blt operations, along with instructions to set up the Blt engine state without actually performing a Blt. Most instructions are of fixed length, though there are a few instructions that include a variable amount of inline data at the end of the instruction. Refer to the *2D Instructions* Chapter for a description of these instructions.

10.5.3. 3D Instructions

The 3D instructions are used to program the 3D pipeline state and perform 3D, Stretch Blt, and MotionComp operations. All 3D state instructions are of fixed length, while the rendering instructions are all variable length. Refer to the *Rendering Engine Instruction* Chapter for a description of the 3D instructions.

Figure 29. Instruction Format For First DWord

TYPE	Bits				
	31:29	28:24	23	22	21:0
Parser	000	Opcode 00h – NOP 0Xh – Single DWord Packets 1Xh – Two DWord Packets 2Xh – Store DWord Packets 3Xh – Ring/Batch Buffer Pkts		Identification No./DWord Count Instruction Dependent Data 5:0 – DWord Count 5:0 – DWord Count 5:0 – DWord Count	
Rsvd.	001				
2D	010	Opcode			Instruction Dependent Data 4:0 – DWord Count
3DState24	011	Opcode – 00000–01111	Instruction Dependent Data 23:0 – 24 state and mask bits		
3DState24NP	011	Opcode – 10000–11000	Instruction Dependent Data 23:0 – 24 non-pipelined state and mask bits		
Rsvd	011	Opcode – 11001–11011			
3DState16	011	Opcode – 11100	23:19 Sub Opcode 00h–7Fh		18:16 – Texture Map. 15:0 – 16 state and mask bits
3DState16NP	011	Opcode – 11100	23:19 Sub Opcode 80h–FFh		18:16 – Scissor Rect. No. 15:0 – 16 state and mask bits
3DStateMW (Multiple DWord)	011	Opcode – 11101	23:16 Sub Opcode 00h–7Fh		15:0 – DWord Count
3DStateMWNP (Multiple DWord)	011	Opcode – 11101	23:16 Sub Opcode 80h–FFh		15:0 – DWord Count
3DBlock	011	Opcode – 11110	23:16 Sub Opcode		15:0 – DWord Count
3DPrim	011	Opcode – 11111	23:16 Sub Opcode		17:0 – DWord Count
Reserved	1XX				

NOTES:

1. SrcCopyImmBlt does not follow the 2D format.
2. The qualifier "NP" indicates that the state variable is non-pipelined and the render pipe is flushed before such a state variable is updated. All the other state variables are pipelined (default).

Table 13. Graphics Controller Instructions

Client	Instruction
00h–Command Parser	GFXCMDPARSER_NOP_IDENTIFICATION
	GFXCMDPARSER_BREAKPOINT_INTERRUPT
	GFXCMDPARSER_USER_INTERRUPT
	GFXCMDPARSER_WAIT_FOR_EVENT
	GFXCMDPARSER_FLUSH
	GFXCMDPARSER_CONTEXT_SEL
	GFXCMDPARSER_DEST_BUFFER_INFO
	GFXCMDPARSER _FRONT_BUFFER_INFO
	GFXCMDPARSER _Z_BUFFER_INFO
	GFXCMDPARSER_REPORT_HEAD
	GFXCMDPARSER_ARB_ON_OFF
	GFXCMDPARSER _DEST_BUFFER_INFO
	GFXCMDPARSER_OVERLAY_FLIP
	GFXCMDPARSER_LOAD_SCAN_LINES_INCL
	GFXCMDPARSER_LOAD_SCAN_LINES_EXCL
	GFXCMDPARSER_STORE_DWORD_IMM
	GFXCMDPARSER_STORE_DWORD_INDEX
	GFXCMDPARSER_BATCH_BUFFER
02h–2D Processor	SETUP_BLT
	SETUP_MONO_PATTERN_SL_BLT
	PIXEL_BLT
	SCANLINE_BLT
	TEXT_BLT
	TEXT_Immediate_BLT
	COLOR_BLT
	PAT_BLT
	MONO_PAT_BLT
	SRC_COPY_BLT
	MONO_SRC_COPY_BLT
	MONO_SRC_COPY_Immediate_BLT
	FULL_BLT
	FULL_MONO_SRC_BLT
	FULL_MONO_PATTERN_BLT
	FULL_MONO_PATTERN_MONO_SRC_BLT

Table 13. Graphics Controller Instructions

Client	Instruction
03h–Rendering Processor	GFXPRIMITIVE
	GFXBLOCK
	GFXRENDERSTATE_VERTEX_FORMAT
	GFXRENDERSTATE_MAP_TEXELS
	GFXRENDERSTATE_MAP_COORD_SETS
	GFXRENDERSTATE_MAP_INFO
	GFXRENDERSTATE_MAP_FILTER
	GFXRENDERSTATE_MAP_LOD_LIMITS
	GFXRENDERSTATE_MAP_LOD_CONTROL
	GFXRENDERSTATE_MAP_PALETTE_LOAD
	GFXRENDERSTATE_MAP_LOD_BIAS
	GFXRENDERSTATE_MAP_COLOR_BLEND_STAGES
	GFXRENDERSTATE_MAP_ALPHA_BLEND_STAGES
	GFXRENDERSTATE_COLOR_FACTOR
	GFXRENDERSTATE_COLOR_CHROMA_KEY
	GFXRENDERSTATE_SCR_DST_BLEND_MONO
	GFXRENDERSTATE_Z_BIAS_ALPHA_FUNC_REF
	GFXRENDERSTATE_LINE_WIDTH_CULL_SHADE_MODE
	GFXRENDERSTATE_BOOLEAN_ENA_1
	GFXRENDERSTATE_BOOLEAN_ENA_2
	GFXRENDERSTATE_FOG_COLOR
	GFXRENDERSTATE_DRAWING_RECTANGLE_INFO
	GFXRENDERSTATE_SCISSOR_ENABLE
	GFXRENDERSTATE_SCISSOR_RECTANGLE_INFO
	GFXRENDERSTATE_ANTI_ALIASING
	GFXRENDERSTATE_PROVOKING_VTX_PIXELIZATION_RULE
	GFXRENDERSTATE_DEST_BUFFER_VARIABLES

11. Instruction Parser Instructions

11.1. Introduction

The Graphics Controller (GC) contains an extensive set of instruction for controlling 2D and 3D operations. This section describes the programmer's interface to these instructions. The instructions can be categorized as follows:

- 3D Instructions. The 3D pipeline states and processing functions are controlled by a set of 3D instructions. (See the 3D Instruction section for detailed instruction descriptions).
- 2D Instructions. The 2D instructions are used to invoke BLT operations. (See the 2D Register and instruction section for detailed instruction descriptions).
- Instruction Parser Instructions. These instructions control various GC interface units (eg. Local Memory Interface, Display Interface), setting break points, etc.

11.2. Instruction Descriptions

11.2.1. GFXCMDPARSER_NOP_IDENTIFICATION

This instruction effectively provides a "no-operation" instruction that can be used to pad the instruction stream (e.g., in order to pad out a batch buffer to a QWord boundary). However, there is one operation this instruction can perform. If the Enable bit is set, the command parser will write the Identification Number field contents into the NOP Identification Register. This provides a general-purpose instruction stream tagging ("breadcrumb") mechanism.

One possible use would be for software to use a NOP_IDENTIFICATION instruction to tag a subsequent Breakpoint Interrupt event. The GFXCMDPARSER_NOP_IDENTIFICATION instruction format is:

DWord	Bit	Description
0	31:29	**Client:** 000 – Instruction Parser
	28:23	**Opcode:** 00h
	22	**Enable:** 1 = write the identification number. 0 = don't write the identification number.
	21:6	**Identification Number.**
	5:0	**Reserved.** MBZ

11.2.2. GFXCMDPARSER_BREAKPOINT_INTERRUPT

This instruction will generate a breakpoint interrupt and cause the parser to stop until the interrupt is cleared by writing the Interrupt Identity Register. If the interrupting event is masked through the IMR, the parser continues parsing. However, if the event is unmasked in the IMR and the interrupt is not enabled through the IER, the parser halts until the IIR is cleared. In this case the software has to poll the IIR to check for this condition and clear the IIR as an interrupt will not occur.

DWord	Bit	Description
0	31:29	**Client:** 000 – Instruction Parser
	28:23	**Opcode:** 01h
	22:0	**Reserved.** MBZ

11.2.3. GFXCMDPARSER_USER_INTERRUPT

This instruction will generate a user-defined interrupt if the interrupt is enabled and not masked. The parser continues parsing after processing this instruction. If a user interrupt is currently outstanding (not yet cleared in the IIR) this packet has no effect.

Dword	Bit	Description
0	31:29	**Client:** 000 – Instruction Parser
	28:23	**Opcode:** 02h
	22:0	**“Reserved MBZ”**

11.2.4. GFXCMDPARSER_WAIT_FOR_EVENT

This instruction can be used to pause instruction stream processing until a specific event occurs. Only one event can be specified -- specifying multiple events is UNDEFINED. The effect of the wait operation depends on the source of the instruction. If executed from a batch buffer, the instruction parser halts (and suspend instruction arbitration) until the event occurs. If executed from a ring buffer, further processing of that ring will be suspended, although instruction arbitration (from other rings) will continue.

DWord	Bit	Description
0	31:29	**Client:** 000 – Instruction Parser
	28:23	**Opcode:** 03h
	22:4	**Reserved:** 0000h
	3	**VBLANK:** If this bit is set, and this instruction is executed out of a Batch Buffer, the Parser halts when it parses this instruction until the beginning of the next display vertical blank. If executed out of a ring buffer, the Parser sets a flag that eliminates that ring from the arbitration until the flag is cleared. This flag is cleared by the appropriate edge detection of the Display Vertical Blank signal assertion.
	2	**DISPLAY FLIP PENDING:** If this bit is set, and this instruction is executed out of a Batch Buffer, the Parser halts when it parses this instruction until the flip event (new front buffer address has now been loaded into the active front buffer registers) . If executed out of a ring buffer, the Parser sets a flag that eliminates that ring from the arbitration till the flip event. If there is no flip pending, the parser just proceeds. Mechanism: The execution of a front buffer packet by the parser sets a flag (FlipPendingFlag) that is cleared by the flip event. If the FlipPendingFlag is set and this instruction shows up with this bit set, then the parser will wait for flip event just as described for the bit above. If FlipPendingFlag is not set and this instructions shows up with this bit set, the instruction has no effect and parser moves on. If the execution of this instruction happens to coincide with the flip event, the parser behaves as if the FlipPendingFlag is not set. The flip event can be a function of hsync or vsync as selected by the front buffer packet.
	1	**SCAN LINES:** This instruction with this bit set should be sent after the LOAD_SCAN_LINES instruction. If executed out of a batch buffer, this instruction will cause the parser to halt and wait if the scan_line_window indicator is asserted. If the scan line window indicator is de-asserted, the parser just moves on. If executed out of a ring buffer, the Parser sets a flag if the scan line window indicator is asserted. This flag eliminates that ring from the arbitration until the it is cleared. This flag is cleared by the trailing edge of the scan_line_window indicator. If the scan line window indicator is deasserted the flag is not set and the parser moves on. For more information look at the load_scan_line packet.
	0	**Reserved:** 0

11.2.5. GFXCMDPARSER_FLUSH

This instruction will flush all drawing engines and the frame buffer cache (a.k.a. local cache). In addition, it will conditionally invalidate the map cache. After this instruction is completed and followed by a store DWord, processor access to graphics memory will be coherent.

DWord	Bit	Description
0	31:29	**Client:** 000 – Instruction Parser
	28:23	**Opcode:** 04h
	22:2	**Reserved:** 000000h
	1	**Reserved.**
	0	**INVALIDATE_MAP_CACHE:** When this bit is set the parser will wait until the rendering pipeline is done and then invalidate the mapping engine cache.

11.2.6. GFXCMDPARSER_CONTEXT _SEL

The GFXCMDPARSER_CONTEXT_SEL instruction is used to inform the input interface to load or use one of two sets of state variables. The Pipeline supports the use of one set while the second set is being loaded. The driver has to ensure that the pipeline is flushed before it changes the USE address. The LOAD address does not have the same restriction.

Only YUV data is supported in Context 1. RGB is not supported.

DWord	Bits	Description
0	31:29	**Client:** 000 – Instruction Parser
	28:23	**Opcode:** 05h
	22:18	Reserved: 0h
	17	**Load Addr Enable:** 1 = Update Load Addr with the value in the Load Addr field 0 = Ignore Load Addr
	16	**Use Addr Enable:** 1 = Update Use Addr with the value in the Use Addr field 0 = Ignore Use Addr
	15:9	**Reserved. MBZ**
	8	**Load Addr:** address of the SV set to be loaded. 1 = State Variable Set 1 0 = State Variable Set 0
	7:1	**Reserved. MBZ**
	0	**Use Addr:** address of the SV set to be used. 1 = State Variable Set 1 0 = State Variable Set 0

11.2.7. GFXCMDPARSER _DEST_BUFFER_INFO

The GFXCMDPARSER DEST_BUFFER_INFO instruction is used to specify the information about the destination buffer. This is an "immediate" instruction and therefore software must ensure that the Rendering Engine and the local cache are flushed prior to modifying the destination buffer information. The format is:

Dword	Bit	Description
0	31:29	**Client:** 000 – Instruction Parser
	28:23	**Opcode:** 15h
	22:6	**Reserved:** 00000h
	5:0	**DWORD_LENGTH:** 00h
1	31:26	**Reserved**.
	25:12	**Base Address:** This is the base address of the rendered scene in linear space. The Memory Interface unit uses this address in conjunction with the Memory Fence Table Registers to determine the virtual (Tiled) address of the dest buffer. This surface address (linear) must be at least 4 KB aligned. in addition it must be 4 times pitch aligned aligned.
	11:2	**Reserved. MBZ**
	1:0	**Pitch:** This is the pitch of the Back Buffer. This is used by the TLB in computing the absolute address of the color requests. 00 = 512 bytes 01 = 1 KB 10 = 2 KB 11 = 4 KB

11.2.8. GFXCMDPARSER _FRONT_BUFFER_INFO

The GFXCMDPARSER _FRONT_BUFFER_INFO instruction is used to initialize the base address of the scene to be display by the Display Engine (DE) (a.k.a. flip). There are two choices for this instruction. In the first choice, the Instruction Parser sends the base address to the DE where its update is synchronized to the display syncs (sync flip). In the second choice, the DE's update is on the following hsync (async flip).

In the case of a double buffer swap operation requiring a flip between the display and render surface base addresses, in addition to the FRONT_BUFFER_INFO packet a DEST_BUFFER_INFO instruction will also need to be specified. Note that no special hardware is provided to synchronize these instructions together.

A bit of the Interrupt Status Register represents the status of the flip instruction. This flag is set when the Instruction Parser processes the GFXCMDPARSER_FRONT_BUFFER_INFO instruction. For the sync flip, the flag is cleared when the vertical sync occurs. For async flip the flag is cleared when the hardware determines all the information for the new buffer is acquired, this is estimated to be 32 scan lines.

Setting and clearing this flag generates a system memory write to the location stored in the Hardware Status Vector Address Register, if unmasked in the Hardware Status Mask Register. Clearing this flag will generate an external interrupt, if unmasked in the Interrupt Mask Register and enabled in the Interrupt Enable Register.

The flush instruction should be issued prior to the flip instruction; this is to ensure that hardware pipeline and cache structures are flushed and the rendered scene that is to be displayed next is in memory. The flush instruction waits for the blitter and the render/map pipeline to be Not Busy and the local cache to be coherent with memory before parsing continues. The format of the GFXCMDPARSER_FRONT_BUFFER_INFO instruction is:

DWord	Bits	Description
0	31:29	**Client:** 000 – Instruction Parser
	28:23	**Instruction Target:** 14h
	22:20	**Reserved. MBZ**
	19:8	**Front Buffer Pitch:** A 12-bit value that specifies the number of QWs of frame buffer memory occupied by each horizontal row of characters. This loading is similar to the loading of the offset registers CR41(3:0):: CR13(7:0), except that the pitch in this case is specified in QWs. Will not be loaded in case of Async flip. Example of pitch computation for 1024x768 @ 16 bpp: 1024 pixels * (2 B / 1 pixel) * (1 QW / 8 B) * (1 tile / 16 QW) = 16 tiles
	7	**Reserved. MBZ**
	6	**Flip type: "0": Synch flip, "1": Async flip**
	5:0	**Dword Length:** 00h
1	31:26	**Reserved. MBZ**
	25:3	**Front Buffer Base Address:** Virtual memory address bits 25:3 (max 64MB). The default value is 0. (unsigned int)
	2:0	**Reserved. MBZ**

11.2.9. GFXCMDPARSER _Z_BUFFER_INFO

This instruction is used to specify the base address and pitch of the Z buffer surface used by the 3D Rendering engine. This is an "immediate" command and therefore software must guarantee that the Rendering Engine and the local cache are flushed prior to modifying the z buffer information. The format of the GFXCMDPARSER _Z_BUFFER_INFO instruction is:

DWord	Bit	Description
0	31:29	**Client:** 000 **–** Rendering Processor
	28:23	**Opcode:** 16h
	22:6	**Reserved. MBZ**
	5:0	**Dword Length:** 00h
1	31:26	**Reserved**.
	25:12	**Base Address:** The base address of the Z buffer in linear space. The Memory Interface unit uses this address in conjunction with the Memory Fence Table Registers to determine the virtual (Tiled) address of the buffer . This surface address (linear) must be at least 4KB aligned. in addition it must be 4 times pitch aligned.
	11:2	**Reserved.**
	1:0	**Pitch:** This is the pitch of the Z Buffer. This is used by the TLB in computing the absolute address of Z requests. 00 = 512 bytes 01 = 1 KB 10 = 2 KB 11 = 4 KB

11.2.10. GFXCMDPARSER_REPORT_HEAD

These instruction causes the active ring buffer Head Pointer to be written to a cacheable (snooped) system memory location. The location written is relative to the address programmed in the Hardware Status Page Address Register, and depends on which ring buffer is active. (Refer to the description of the HSW_PGA register). The format is:

Dword	Bit	Description
0	31:29	**Client:** 000 – Instruction Parser
	28:23	**Opcode:** 07h
	22:0	**Reserved. MBZ**

11.2.11. GFXCMDPARSER_ARB_ON_OFF

The GFXCMDPARSER_ARB_ON_OFF instruction is used to inform the input interface to turn on/off all the rings except the ring that this instruction is executed from. It can be use from a batch buffer. This instruction can be used to prevent other ring buffers from interrupting an instruction sequence. The format is:

DWord	Bits	Description
0	31:29	**Client:** 000 – Instruction Parser
	28:23	**Opcode:** 08h
	22:1	**Reserved**.
	0	**Arbitration ON/OFF:** 1 = ON 0 = OFF

11.2.12. GFXCMDPARSER_OVERLAY_FLIP

DWord	Bits	Description
0	31:29	**Client:** 000 – Instruction Parser
	28:23	**Instruction Target:** 11h
	22:6	**Reserved.**
	5:0	**Dword Length:** 00h
1	31:29	**Reserved. MBZ**
	28:03	**Register Update Address:** This address is used by the Overlay at the next VBLANK event to start requesting data from memory.
	2:0	**Reserved. MBZ**

11.2.13. GFXCMDPARSER_LOAD_SCAN_LINES_INCL

This instruction is used to initialize the scan line window registers in the Display engine. If the display refresh is within this window the Display engine asserts a signal that is used by the instruction parser to process the WAIT_FOR_EVENT instruction. This instruction overrides any previous EXCL instruction. The format is:

DWord	Bits	Description
0	31:29	**Client:** 000 – Instruction Parser
	28:23	**Instruction Target:** 12h
	22:6	**Reserved.**
	5:0	**Dword Length:** 00h
1	31:16	**Start Scan Line Number.** (Line 0 is the first line of the display frame)
	15:0	**End Scan Line Number.** (Line 0 is the first line of the display frame)

11.2.14. GFXCMDPARSER_LOAD_SCAN_LINES_EXCL

This instruction is used to initialize the scan line window registers in the Display engine. If the display refresh is outside this window, the display engine asserts a signal that is used by the instruction parser to process the WAIT_FOR_EVENT instruction. This instruction overrides any previous INCL instruction. The format is:

Word	Bits	Description
0	31:29	**Client:** 000 – Instruction Parser
	28:23	**Instruction Target:** 13h
	22:6	**Reserved.**
	5:0	**DWord Length:** 00h
1	31:16	**Start Scan Line Number.** (Line 0 is the first line of the display frame)
	15:0	**End Scan Line Number.** (Line 0 is the first line of the display frame)

11.2.15. GFXCMDPARSER_STORE_DWORD_IMM

This instruction immediately causes a system write of the data word in the packet to the address which is also in the instruction packet. Note: All store DWords will invalidate the host - graphics pre-fetch cache. The GFXCMDPARSER_STORE_DWORD_IMM instruction format is:

DWord	Bit	Description
0	31:29	**Client:** 000 – Instruction Parser
	28:23	**Instruction Target:** 20h
	22:6	**Reserved.**
	5:0	**DWord Length:** 01h
1	31:2	**Address:** DWord aligned. The hardware only uses bits 31:2.
	1:0	**Reserved.** MBZ
2	31:0	**Data Word**

11.2.16. GFXCMDPARSER_STORE_DWORD_INDEX

This instruction immediately causes a system write of the data word in the packet to the address which is provided in the Hardware Status Page Address Register at offset specified.

Note: All store DWords will invalidate the host-graphics prefetch cache.

DWord	Bit	Description
0	31:29	**Client:** 000 – Instruction Parser
	28:23	**Instruction Target:** 21h
	22:6	**Reserved:** 0000h
	5:0	**Dword Length:** 01h
1	31:12	**Reserved.**
	11:2	**Dword Offset into a page:** A (11:2)
	1:0	**Reserved.**
2	31:0	**Data Word**

11.2.17. GFXCMDPARSER_BATCH_BUFFER

The GFXCMDPARSER_BATCH_BUFFER instruction is used to inform the input interface to parse an instruction buffer. The address on the instruction buffers is in graphics memory that should translate to a physical address in main memory. Note that in case of an AGP device batch buffers can be in AGP memory only.

The batch buffer instruction packet implements a protection ID. The protection ID is recorded as protection state by the parser when it starts a Batch buffer instruction executed from one of the rings. Chained Batch buffer instructions cannot change the protection state of the parser. If a batch buffer is pushed into the stack at a chain point, this state has to be stored as well. The protection state modifies the parser behavior as follows:

- In Unprotected Mode the parser generates an error if it parses a store DWORD immediate instruction. It is assumed that an unprotected batch buffer has not been verified by the driver and can have invalid store DWORD immediate instructions that write over protected areas in cacheable memory.
- In protected state the parser will allow all instructions to be parsed. It is assumed that a batch buffer identified as protected has been verified by the driver to ensure that store DWORD immediate instructions do not corrupt the operating system.

The Intel® 815 chipset hardware implementation has batch buffer size limit set at 512 KB - 8B.

The instruction needs a Buffer Start Address and a Buffer End Address that both must be QW aligned physical memory addresses. Using this start and end address, the batch buffer size definition is:

Batch Buffer Size = Buffer End Address - Buffer Start Address + 8 Bytes

This means the batch buffer does include valid commands up to and including the end address location.

```
@Start_Address          Valid Command \
@Start_Address + 8      Valid Command  \
... ...                                  > Total Size = End - Start + 8B
@End_Address - 8 Valid Command   /    must be limited to 512KB-8B
@End_Address            Valid Command  /
```

DWord	Bits	Description
0	31:29	**Client:** 000 – Instruction Parser
	28:23	**Instruction Target:** 30h
	22:6	**Reserved.**
	5:0	**Dword Length:** 01h
1	31:3	**Buffer Start Address:** Must be a QW aligned physical memory address.
	2:1	**Reserved.**
	0	**Protection id:** 1 = Un-Protected, 0 = Protected
2	31:3	**Buffer End Address:** Must be a QW aligned physical memory address.
	2:0	**Reserved.** MBZ

Helpful Hint: Use GFXCMDPARSER_NOP_IDENTIFICATION to pad buffer to a QW length.

12. 2D Instructions

This chapter contains the 2D graphics controller instructions. For each instruction the format specifies the functionality of a field. When an instruction does not require a field, it is ignored. All registers can only be written through instructions. No program I/O writing of the BLT registers is allowed.

12.1. BLTs To and From Cacheable Memory

The blitter can be used to transfer data from cacheable memory to graphics memory and vice versa using the blitter command packets. The source or destination operands in these packets can be steered towards cacheable memory.

Patterns may be used with the source. The driver is required to flush the drawing pipelines before and after each copy command targeting cacheable memory. In addition, the driver is required to turn arbitration off if this command is used from the low priority ring buffer. An example sequence from the low priority ring buffer is:

ARB_OFF, FLUSH, SCB, ...,FLUSH, SCB, FLUSH, ARB_ON. Where all Source copy blits (SCBs) use cacheable memory.

Either the source or destination surface can be in cacheable memory. Both Source and Dest surfaces in cacheable memory, as part of the same blit operation is not allowed. In either case the surface address programmed in this instruction must be in graphics address space. The GTT must be programmed to set up a scatter-gather translation from graphics memory pages to physical pages in cacheable memory. A surface that is being mapped to cacheable space must not be tiled or fenced.

A further restriction is that the source data must be the same pixel width as the destination. The only function, which is allowed, is color copies with a positive destination pitch and direction. The source operand can be either sign to allow mirroring in the vertical direction. An immediate source operand is not allowed.

12.2. BLT Engine Instructions

The following instructions are directed to the BLT Engine. The Instruction Target field is used as an opcode by the BLT Engine state machine to qualify the control bits, which are relevant for executing the instruction. The description for each DWord and bit field is contained in the *BLT Engine Instruction Definition* section. Each DWord in a packet has a corresponding Instruction Definition description where the details of the operation of the function are described.

Note: All reserved fields must be programmed to 0s.

12.2.1. SETUP_BLT

The setup instruction supplies common setup information including clipping coordinates used exclusively with the following 3 instructions:

1. PIXEL_BLT (PB) - 1 pixel write with the coordinate and solid pattern supplied for each pixel to be written. No non-solid patterns nor source operands are allowed.
2. SCANLINE_BLT (SLB) - 1 scan line of color or mono pattern and destination are the only operands allowed.
3. TEXT_BLT (TB) - linear monochrome source either through the instruction stream or from graphics memory (not cacheable) and destination are the only operands allowed. Source copy is the only supported operation. The raster operation field must = CCh.

Clipping addresses and coordinates are inclusive. (The BLT Engine performs a trivial reject for all 3 BLT instructions listed above before performing any accesses. If any pixels are included within the clipping rectangle, then it performs every access, but deasserts the byte enables for the pixels that are clipped.)

The source operand (TEXT_BLT only) never overlaps with the destination. Therefore, X and Y direction is always positive (left to right and top to bottom). Only a positive destination pitch is allowed.

All fields above indicate which instructions require valid state. These are the only instructions that require that state be saved between instructions. There are 4 dedicated registers to contain the state for these 3 instructions. All other BLTs use a temporary version of these. The 4 double word registers are: DW1 (Control), DW5 (Foreground color), DW6 (Background color), and DW7 (Pattern address). The ClipRect registers do not need to be saved since they are never used by the interrupt ring buffer.

Note: The *Mono source transparency mode* flag is maintained (although it is always set to one in current driver operation).

DWord	Bit	Description
0 = BR00	31:29	**Client** : 02h – 2D Processor
	28:22	**Instruction Target (Opcode) :** 00h
	21:05	Reserved. Must be Zero
0	04:00	**Dword Length :** 06h
1 = BR01	31	Reserved. Must be Zero
	30	Reserved. Must be Zero
	29	**Mono Source Transparency Mode:** (1 = transparency enabled; 0 = use background) - TB only
	28	Reserved. Must be Zero
	27	Reserved. Must be Zero
	26	**Must Be One ('1').**
		Color Depth: All 00 = 8 bit color 01 = 16 bit color 10 = 24 bit color 11 = reserved
	23:16	**Raster Operation:** (all; must = CC Hex for TB)
	15:00	**Destination Pitch (positive):** (13:00 are implemented in Intel® 810 chipset) (SLB & TB only)
2 = BR02	31:00	**ClipRect Y1 Address (Top):** (25:00 are implemented in Intel® 810 chipset) All
3 = BR03	31:00	**ClipRect Y2 Address (Bottom):** (25:00 are implemented in Intel® 810 chipset) All
4 = BR04	31:16	**ClipRect X2 Coordinate (Right):** All (27:16 = 12 bits are implemented in the Intel® 815 chipset)
	15:00	**ClipRect X1 Coordinate (Left):** All (11:00 = 12 bits are implemented in the Intel® 815 chipset)
5	31:24	Reserved. Must be Zero
5 = BR05	23:00	**Setup Background Color:** All
6	31:24	Reserved. Must be Zero
6 = BR06	23:00	**Setup Foreground Color:** (SLB & TB only)
7 = BR07	31:00	**Pattern Address for Color Pattern:** (25:06 are implemented in Intel® 810 chipset) (SLB only)

12.2.2. SETUP_MONO_PATTERN_SL_BLT

This setup instruction supplies common setup information including clipping coordinates used exclusively with the following instruction:

- SCANLINE_BLT (SLB) - 1 scan line of monochrome pattern and destination are the only operands allowed.

Clipping addresses and coordinates are inclusive. (The BLT Engine performs a trivial reject for this BLT before performing any accesses. If any pixels are included within the clipping rectangle, then it performs every access, but deasserts the byte enables for the pixels that are clipped.) These are the only instructions that require state. There are 4 dedicated registers to contain the state for this instruction. The 3 double word registers are: DW1 (Control), DW5 (Background color), and DW6 (Foreground color). The ClipRect registers do not need to be saved since they are never used by the interrupt ring buffer.

Only a positive destination pitch is allowed.

DWord	Bit	Description
0 = BR00	31:29	**Client** : 02h – 2D Processor
	28:22	**Instruction Target (Opcode) :** 10h
	21:05	Reserved. Must be Zero
	04:00	**Dword Length :** 07h
1 = BR01	31	**Solid Pattern Select:** (1 = solid pattern; 0 = no solid pattern) - (SLB & Pixel only)
	30:29	Reserved. Must be Zero
	28	**Mono Pattern Transparency Mode:** (1 = transparency enabled; 0 = use background)
	27	Reserved. Must be Zero
	26	**Must Be One ('1').**
	25:24	**Color Depth:** 00 = 8 bit color 01 = 16 bit color 10 = 24 bit color 11 = reserved
	23:16	**Raster Operation:**
	15:00	**Destination Pitch (positive):** (13:00 are implemented in Intel® 810 chipset)
2 = BR02	31:00	**ClipRect Y1 Address (Top):** (25:00 are implemented inIntel® 810 chipset)
3 = BR03	31:00	**ClipRect Y2 Address (Bottom):** (25:00 are implemented in Intel® 810 chipset)
4 = BR04	31:16	**ClipRect X2 coordinate (Right):** (27:16 = 12 bits are implemented in the Intel® 815 chipset)
	15:00	**ClipRect X1 Coordinate (Left):** (11:00 = 12 bits are implemented in the Intel® 815 chipset)
5	31:24	Reserved. Must be Zero
5 = BR05	23:00	**Setup Background Color:** All
6	31:24	Reserved. Must be Zero
6 = BR06	23:00	**Setup Foreground Color:** (SLB & TB only)
7 = BR20	31:00	**DW0 (least significant) for a Monochrome Pattern:**
8 = BR21	31:00	**DW1 (most significant) for a Monochrome Pattern:**

12.2.3. PIXEL_BLT

The Destination X coordinate and Destination Y Address is compared with the ClipRect registers. If it is within all 4 comparisons, then the pixel supplied in the SETUP_BLT instruction is written with the raster operation to (Destination Y Address + Destination X coordinate * bytes per pixel).

DWord	Bit	Description
0 = BR00	31:29	**Client** : 02h – 2D Processor
	28:22	**Instruction Target (Opcode) :** 20h
0 = BR08	21:06	**Destination X Coordinate:** (17:06 = 12 bits are implemented in the Intel® 815 chipset)
	05	Reserved. Must be Zero
0	04:00	**Dword Length :** 00h
1 = BR09	31:00	**Destination Y Address:** (address of the first pixel on a scan line) (25:00 are implemented in Intel® 810 chipset)

12.2.4. SCANLINE_BLT

The Destination Y Address is compared against the ClipRect Y address registers. If the address is within this comparison, then the pattern pixels dependent on the Destination X coordinates that fall within the ClipRect X comparisons are written using the raster operation to (Destination Y Address + Destination X coordinate * bytes per pixel). The horizontal alignment is relative to the destination from the lower bits of the destination address. The pattern vertical alignment indicates which pattern scan line is used (this is the least significant three bits of the destination vertical coordinate). With color patterns only 1 scan line should be read for this instruction.

- Solid pattern should use the SETUP_MONO_PATTERN_SL_BLT instruction.

DWord	Bit	Description
0 = BR00	31:29	**Client** : 02h – 2D Processor
	28:22	**Instruction Target (Opcode) :** 21h
	21:08	Reserved. Must be Zero
	07:05	**Pattern Vertical Alignment:** (which scan line of the 8x8 pattern to start on)
	04:00	**Dword Length :** 01h
1 = BR08	31:16	**Destination X2 Coordinate:** (Ending - Right) (Intel® 815 chipset implementation supports 27:16 = 12 bits)
	15:00	**Destination X1 Coordinate:** (Starting - Left) - X2 - X1 + 1 = width in pixels (Intel® 815 chipset implementation supports 11:00 = 12 bits)
2 = BR09	31:00	**Destination Y Address:** (address of the first pixel on a scan line) (25:00 is implemented inIntel® 810 chipset)

12.2.5. TEXT_BLT

All monochrome source scan lines and pixels that fall within the ClipRect Y addresses and X coordinates are written (ignoring the raster operation) to (Destination Y Address + Destination X coordinate * bytes per pixel). Source expansion color registers are always in the SETUP_BLT.

Note: All graphics controller BR03 fields (monochrome clipping parameters) are computed by the hardware while performing clipping.

DWord	Bit	Description
0 = BR00	31:29	**Client** : 02h – 2D Processor
	28:22	**Instruction Target (Opcode) :** 22h
	21:17	Reserved. Must be Zero
	16	**Bit (0) / Byte (1) packed:** Byte packing is used for the Windows* NT* driver
	04:00	**Dword Length :** 04h
1 = BR08	31:16	**Destination X2 Coordinate:** (Ending - Right) (27:16 = 12 bits are implemented in the Intel® 815 chipset)
	15:00	**Destination X1 Coordinate:** (Starting - Left) - X2 - X1 + 1 = width in pixels (11:00 = 12 bits are implemented in the Intel® 815 chipset)
2 = BR09	31:00	**Destination Y1 Address:** (address of the first pixel on the first scan line) (25:00 are implemented in Intel® 810 chipset)
3 = BR10	31:00	**Destination Y2 Address:** (address of the first pixel on the last scan line) (25:00 are implemented in Intel® 810 chipset)
4= BR11	31:16	Reserved. Must be Zero
	15:00	**Number of monochrome source Quadwords - 1:** (1 to 64k Quadwords = 64 to 4M bits)
5 = BR12	31:00	**Source Address:** (address of the first byte of the first pixel on the first scan line) (25:00 are implemented in Intel® 810 chipset)

12.2.6. TEXT_Immediate_BLT

This instruction allows the Driver to send data through the instruction stream, which eliminates the read latency of reading a source from memory. This allows graphics primitives such as Text to execute much faster. If an operand is in system cacheable memory and either is small or only accessed once, it can be copied directly to the instruction stream versus to graphics accessible memory.

- The IMMEDIATE_BLT data MUST transfer an even number of DWs. The BLT engine will hang if it does not get an even number of DWs.
- Monochrome source data is sent through the instruction stream.
- All monochrome source scan lines and pixels that fall within the ClipRect Y addresses and X coordinates are written (ignoring the raster operation) to (Destination Y Address + Destination X coordinate * bytes per pixel). Source expansion color registers are always in the SETUP_BLT.

DWord	Bit	Description
0 = BR00	31:29	**Client** : 02h – 2D Processor
	28:22	**Instruction Target (Opcode) :** 30h
	21:17	Reserved. Must be Zero
	16	**Bit (0) / Byte (1) packed:** Byte packing is for the Windows* NT* driver
	15:00	**Dword Length :** 02+ DWL = (Number of Immediate double words)h
1 = BR08	31:16	**Destination X2 Coordinate:** (Ending - Right) (27:16 = 12 bits are implemented in the Intel® 815 chipset)
	15:00	Destination X1 coordinate: (Starting - Left) - X2 - X1 + 1 = width in pixels (11:00 = 12 bits are implemented in the Intel® 815 chipset)
2 = BR09	31:00	**Destination Y1 Address:** (address of the first pixel on the first scan line) (25:00 are implemented in Intel® 810 chipset)
3 = BR10	31:00	**Destination Y2 Address:** (address of the first pixel on the last scan line) (25:00 are implemented in Intel® 810 chipset)
4	31:00	**Immediate Data DW 0:**
5	31:00	**Immediate Data DW 1:**
6 thru DWL+3	S	**Immediate Data DWs 2 through DWORD_LENGTH (DWL):**

12.2.7. COLOR_BLT

COLOR_BLT is the simplest BLT operation. It performs a color fill to the destination (with a possible ROP). The only operand is the destination operand, which is written dependent on the raster operation. The solid pattern color is stored in the pattern background register.

- This instruction is optimized to run at the maximum memory write bandwidth.
- Only a positive destination pitch is allowed.
- The typical Raster operation code = F0 which performs a copy of the pattern background register to the destination.

DWord	Bit	Description
0 = BR00	31:29	**Client** : 02h – 2D Processor
	28:22	**Instruction Target (Opcode) :** 40h
	21:05	Reserved. Must be Zero
	04:00	**Dword Length :** 03h
1 = BR13	31	**Solid pattern select:** (1 = solid pattern). Must be 1.
	30:27	Reserved. Must be Zero
	26	**Must Be One ('1').**
	25:24	**Color Depth:** 00 = 8 bit color 01 = 16 bit color 10 = 24 bit color 11 = Reserved
	23:16	**Raster Operation:**
	15:00	**Destination Pitch (positive):** (13:00 are implemented in Intel® 810 chipset)
2 = BR14	31:16	**Destination Height (in scan lines):** (28:16 are implemented in Intel® 810 chipset)
	15:00	**Destination Width (in bytes):** (12:00 are implemented in Intel® 810 chipset)
3 = BR09	31:00	**Destination Address:** Address of the first byte to be written (25:00 are implemented in Intel® 810 chipset)
4 = BR16	31:24	Reserved. Must be Zero
	23:00	**Solid Pattern Color:**

12.2.8. PAT_BLT

PAT_BLT is used when there is no source and the color pattern is not trivial (is not a solid color only).

The whole color pattern (8 x 8 pixels = 16, 32, or 64 DWs) is read at the beginning of the BLT and stored in the Texture Cache. The pattern vertical alignment specifies the first scan line of the pattern that is used. The horizontal alignment is relative to the destination from the lower bits of the destination address.

The only memory accesses required for the remainder of the BLT is destination accesses, which is dependent on the raster operation.

Only a positive destination pitch is allowed.

DWord	Bit	Description
0 = BR00	31:29	**Client** : 02h – 2D Processor
	28:22	**Instruction Target (Opcode) :** 41h
	21:08	Reserved. Must be Zero
	07:05	**Pattern Vertical Alignment:** (which scan line of the 8x8 pattern to start on)
	04:00	**Dword Length :** 03h
1 = BR13	31:28	Reserved. Must be Zero
	27	Reserved. Must be Zero
	26	**Must Be One (‘1’).**
	25:24	**Color Depth:** 00 = 8 bit color 01 = 16 bit color 10 = 24 bit color 11 = reserved
	23:16	**Raster Operation:**
	15:00	**Destination Pitch (positive):** (13:00 are implemented in Intel® 810 chipset)
2 = BR14	31:16	**Destination Height (in scan lines):** (28:16 are implemented in Intel® 810 chipset)
	15:00	**Destination Width (in bytes):** (12:00 are implemented in Intel® 810 chipset)
3 = BR09	31:00	**Destination Address:** Address of the first byte to be written (25:00 are implemented in Intel® 810 chipset)
4 = BR15	31:00	**Pattern Address:** (25:06 are implemented in Intel® 810 chipset)

12.2.9. MONO_PAT_BLT

MONO_PAT_BLT is used when there is no source and the monochrome pattern is not trivial (is not a solid color only). The monochrome pattern is loaded from the instruction stream and the only memory accesses are for the destination operand, which is dependent on the raster operation. The pattern vertical alignment indicates on which byte to start. The horizontal alignment is relative to the destination from the lower bits of the destination address. The monochrome pattern transparency mode indicates whether to use the pattern background color or deassert the write enables when the bit in the pattern is 0. When the pattern bit is 1, then the pattern foreground color is used in the ROP operation. The ROP value chosen should involve the Mono pattern data in the ROP operation.

- Only a positive destination pitch is allowed.

DWord	Bit	Description
0 = BR00	31:29	**Client** : 02h – 2D Processor
	28:22	**Instruction Target (Opcode) :** 42h
	21:08	Reserved. Must be Zero
	07:05	**Pattern Vertical Alignment:** (which scan line of the 8x8 pattern to start on)
	04:00	**Dword Length :** 06h
1 = BR13	31:29	Reserved. Must be Zero
	28	**Mono Pattern Transparency Mode:** (1 = transparency enabled; 0 = use background)
	27	Reserved. Must be Zero
	26	**Must Be One (‘1’).**
	25:24	**Color Depth:** 00 = 8 bit color 01 = 16 bit color 10 = 24 bit color 11 = reserved
	23:16	**Raster Operation:**
	15:00	**Destination Pitch (positive):** (13:00 are implemented in Intel® 810 chipset)
2 = BR14	31:16	**Destination Height (in scan lines):** (28:16 are implemented in Intel® 810 chipset)
	15:00	**Destination Width (in bytes):** (12:00 are implemented in Intel® 810 chipset)
3 = BR09	31:00	**Destination Address:** Address of the first byte to be written (25:00 are implemented in Intel® 810 chipset)
	31:24	Reserved. Must be Zero
4 = BR16	23:00	**Pattern Background Color:**
	31:24	Reserved. Must be Zero
5 = BR17	23:00	**Pattern Foreground Color:**
6 = BR20	31:00	**Pattern Data 0:** (least significant DW)
7 = BR21	31:00	**Pattern Data 1:** (most significant DW)

12.2.10. SRC_COPY_BLT

This BLT instruction performs a color source copy where the only operands involved is a color source and destination of the same bit width.

The source and destination operands may overlap, which means that the X and Y directions can be either forwards or backwards. The X direction field applies to both the destination and source operands. The source and destination pitches can be either sign.

Requirements:

- The Intel® 815 chipset hardware has a restriction that the BLT color source and destination operands must not co-exist on the same 32-Byte cacheline. To work-around this issue :
  - For BLTs that have sharing of a cacheline for a given scanline, the driver must treat this as an overlapping BLT case.
  - For linear memory, the surfaces allocated must be 32B-cacheline aligned.
- Three instructions are affected by this hardware restriction : SRC_COPY_BLT, FULL_BLT, and FULL_MONO_PATTERN_BLT. Essentially any instruction with Color Source & Destination will have the above restriction.

DWord	Bit	Description
0 = BR00	31:29	**Client** : 02h – 2D Processor
	28:22	**Instruction Target (Opcode) :** 43h
	21:05	Reserved. Must be Zero
	04:00	**Dword Length :** 04h
1 = BR13	31	Reserved. Must be Zero
	30	**X Direction** (1 = written from right to left (decrementing = backwards); 0 = incrementing)
	29:28	Reserved. Must be Zero
	27	Reserved. Must be Zero
	26	**Must Be One ('1').**
	25:24	**Color Depth:** 00 = 8 bit color 01 = 16 bit color 10 = 24 bit color 11 = reserved
	23:16	**Raster Operation:**
	15:00	**Destination Pitch (signed):** (13:00 are implemented in Intel® 810 chipset)
2 = BR14	31:16	**Destination Height (in scan lines):** (28:16 are implemented in Intel® 810 chipset)
	15:00	**Destination Width (in bytes):** (12:00 are implemented in Intel® 810 chipset)
3 = BR09	31:00	**Destination Address:** Address of the first byte to be written (25:00 are implemented in Intel® 810 chipset)
4 = BR11	31:14	Reserved. Must be Zero
	13:00	**Source Pitch (quadword aligned and signed):** (13:00 are implemented in Intel® 810 chipset)
5 = BR12	31:00	**Source Address:** (25:00 are implemented in Intel® 810 chipset)

12.2.11. MONO_SRC_COPY_BLT

This BLT instruction performs a monochrome source copy where the only operands involved is a monochrome source and destination. The source and destination operands cannot overlap, which means that the X direction must always be forward.

All non-text monochrome sources are word aligned. At the end of a scan line the monochrome source, the bits until the next word boundary must be ignored. The Monochrome source data bit position field <2:0> indicates which bit position within the first byte of the scan line should be used as the first source pixel.

The monochrome source transparency mode indicates whether to use the source background color or deassert the write enables when the bit in the source is 0. When the source bit is 1, the source foreground color is used in the ROP operation. The ROP value chosen should involve the Mono source data in the ROP operation.

The Destination pitch can either be positive or negative to allow mirroring in the Y direction.

DWord	Bit	Description
0 = BR00	31:29	**Client** : 02h – 2D Processor
	28:22	**Instruction Target (Opcode) :** 44h
	21:20	Reserved. Must be Zero
	19:17	**Monochrome source data bit position of the first pixel within a byte per scan line.**
	16:05	Reserved. Must be Zero
	04:00	**Dword Length :** 06h
1 = BR13	31:30	Reserved. Must be Zero
	29	**Mono Source Transparency Mode:** (1 = transparency enabled; 0 = use background)
	28	Reserved. Must be Zero
	27	Reserved : Must be Zero
	26	**Must Be One ('1').**
	25:24	**Color Depth:** 00 = 8 bit color 01 = 16 bit color 10 = 24 bit color 11 = reserved
	23:16	**Raster Operation:**
	15:00	**Destination Pitch (signed):** (13:00 are implemented in Intel® 810 chipset)
2 = BR14	31:16	**Destination Height (in scan lines):** (28:16 are implemented in Intel® 810 chipset)
	15:00	**Destination Width (in bytes):** (12:00 are implemented in Intel® 810 chipset)
3 = BR09	31:00	**Destination Address:** Address of the first byte to be written (25:00 are implemented in Intel® 810 chipset)

DWord	Bit	Description
4 4 = BR11	31:16	Reserved. Must be Zero
	15:00	**Number of Monochrome Source Quadwords - 1:** (1 to 64k Quadwords = 64 to 4M bits)
5 = BR12	31:00	**Source Address:** (address of the first byte of the first pixel on the first scan line) (25:00 are implemented in Intel® 810 chipset)
6 6 = BR18	31:24	Reserved. Must be Zero
	23:00	**Source Background Color:**
7 7 = BR19	31:24	Reserved. Must be Zero
	23:00	**Source Foreground Color:**

12.2.12. MONO_SRC_COPY_IMMEDIATE_BLT

This instruction allows the driver to send data through the instruction stream which eliminates the read latency of reading a source from memory. If an operand is in system cacheable memory and is either small or only accessed once, it can be copied directly to the instruction stream versus to graphics accessible memory.

The IMMEDIATE_BLT data MUST transfer an even number of DWs. The BLT engine will hang if it does not get an even number of DWs or the exact number of QWs required.

This BLT instruction performs a monochrome source copy where the only operands involved are a monochrome source and destination. The source and destination operands cannot overlap, which means that the X direction must always be forward.

Monochrome source data is sent through the instruction stream. BR11 = DW0[15:0] - 4

All non-text monochrome sources are word aligned. At the end of a scan line the monochrome source, the bits until the next word boundary must be ignored. The Monochrome source data bit position field [2:0] indicates which bit position within the first byte should be used as the first source pixel.

The monochrome source transparency mode indicates whether to use the source background color or deassert the write enables when the bit in the source is 0. When the source bit is 1, then the source foreground color is used in the ROP operation. The ROP value chosen should involve the Mono source data in the ROP operation.

The Destination pitch can either be positive or negative to allow mirroring in the Y direction.

DWord	Bit	Description
0 = BR00	31:29	**Client** : 02h – 2D Processor
	28:22	**Instruction Target (Opcode) :** 61h
	21:20	Reserved. Must be Zero
	19:17	**Monochrome source data bit position of the first pixel within a byte per scan line.**
	16	Reserved. Must be Zero
0 = BR11	15:00	**Dword Length :** 04+ DWL = (Number of Immediate double words)h
1 = BR13	31:30	Reserved. Must be Zero
	29	**Mono Source Transparency Mode:** (1 = transparency enabled; 0 = use background)
	28	Reserved. Must be Zero
	27	Reserved. Must be Zero
	26	**Must Be One (‘1’).**
	25:24	**Color Depth:** 00 = 8 bit color 01 = 16 bit color 10 = 24 bit color 11 = reserved
	23:16	**Raster Operation:**
	15:00	**Destination Pitch (signed):** (13:00 are implemented in Intel® 810 chipset)
2 = BR14	31:16	**Destination Height (in scan lines):** (28:16 are implemented in Intel® 810 chipset)
	15:00	**Destination Width (in bytes):** (12:00 are implemented in Intel® 810 chipset)
3 = BR09	31:00	**Destination Address:** Address of the first byte to be written (25:00 are implemented in Intel® 810 chipset)
	31:24	Reserved. Must be Zero
4 = BR18	23:00	**Source Background Color:**
5	31:24	Reserved. Must be Zero
5 = BR19	23:00	**Source Foreground Color:**
6	31:00	**Immediate Data DW 0:**
7	31:00	**Immediate Data DW 1:**
8 thru DWL+4	S	**Immediate Data DWs 2 through DWORD_LENGTH (DWL):**

12.2.13. FULL_BLT

The full BLT is the most comprehensive BLT instruction. It provides the ability to specify all 3 operands: destination, source, and pattern. The source and pattern operands are the same bit width as the destination operand.

The whole color pattern (8 x 8 pixels = 8, 16, or 64 DWs) is read at the beginning of the BLT and stored in the Texture Cache. The pattern vertical alignment specifies which scan line of the pattern is used first. The destination address specifies the horizontal alignment The only memory accesses required for the remainder of the BLT is source and destination accesses.

Both the source and destination pitches can be either sign. The pattern direction follows the destination operand.

The Intel® 815 chipset hardware has a restriction that the BLT color source and destination operands must not co-exist on the same 32-Byte cacheline. To work-around this issue :

- For BLTs that have sharing of a cacheline for a given scanline, the driver must treat this as an overlapping BLT case.
- For linear memory, the surfaces allocated must be 32B-cacheline aligned.

DWord	Bit	Description
0 = BR00	31:29	**Client** : 02h – 2D Processor
	28:22	**Instruction Target (Opcode) :** 45h
	21:11	Reserved. Must be Zero
	10:08	**Destination Transparency Mode:** See BR00 definition.
	07:05	**Pattern Vertical Alignment:** (which scan line of the 8x8 pattern to start on)
	04:00	**Dword Length :** 06h
1 = BR13	31	Reserved. Must be Zero
	30	**X Direction:** (1 = written from right to left (decrementing = backwards); 0 = incrementing)
	29:28	Reserved. Must be Zero
	27	Reserved. Must be Zero
	26	**Must Be One (‘1’).**
	25:24	**Color Depth:** 00 = 8 bit color 01 = 16 bit color 10 = 24 bit color 11 = reserved
	23:16	**Raster Operation:**
	15:00	**Destination Pitch (signed):** (13:00 are implemented in Intel® 810 chipset)
2 = BR14	31:16	**Destination Height (in scan lines):** (28:16 are implemented in Intel® 810 chipset)
	15:00	**Destination Width (in bytes):** (12:00 are implemented in Intel® 810 chipset)

DWord	Bit	Description
3 = BR09	31:00	**Destination Address:** Address of the first byte to be written (25:00 are implemented in Intel® 810 chipset)
4 = BR11	31:14	Reserved. Must be Zero
	13:00	**Source Pitch (quadword aligned and signed):** (13:00 are implemented in Intel® 810 chipset**)**
5 = BR12	31:00	**Source Address:** (25:00 are implemented in Intel® 810 chipset)
6 = BR18	31:24	Reserved. Must be Zero
	23:00	**Destination Transparency Color:**
7 = BR15	31:00	**Pattern Address:** (25:06 are implemented in Intel® 810 chipset)

12.2.14. FULL_MONO_SRC_BLT

The full BLT is the most comprehensive BLT instruction. It provides the ability to specify all 3 operands: destination, source, and pattern. The source operand is monochrome and the pattern operand is the same bit width as the destination operand.

The whole color pattern (8 x 8 pixels = 8, 16, or 48 DWs) is read at the beginning of the BLT and stored in the Texture Cache. The pattern vertical alignment specifies which scan line of the pattern is used first. The destination address specifies the horizontal alignment. The only memory accesses required for the remainder of the BLT is destination and sometimes monochrome source accesses, since the source is monochrome.

The monochrome source transparency mode indicates whether to use the source background color or deassert the write enables when the bit in the source is 0. When the source bit is 1, then the source foreground color is used in the ROP operation. The ROP value chosen should involve the mono source data in the ROP operation.

All non-text and non-immediate monochrome sources are word aligned. At the end of a scan line the monochrome source, the bits until the next word boundary must be ignored. The Monochrome source data bit position field [2:0] indicates which bit position within the first byte should be used as the first source pixel.

The destination pitch can be either sign to allow vertical mirroring of a monochrome source with a pattern which is accessed in the same direction as the destination operand.

DWord	Bit	Description
0 = BR00	31:29	**Client** : 02h – 2D Processor
	28:22	**Instruction Target (Opcode) :** 46h
	21:20	Reserved. Must be Zero
	19:17	**Monochrome source data bit position of the first pixel within a byte per scan line.**
	16:08	Reserved. Must be Zero
	07:05	**Pattern Vertical Alignment:** (which scan line of the 8x8 pattern to start on)
	04:00	**Dword Length :** 07h
1 = BR13	31:30	Reserved. Must be Zero
	29	**Mono Source Transparency Mode:** (1 = transparency enabled; 0 = use background)
	28	Reserved. Must be Zero
	27	Reserved. Must be Zero
	26	**Must Be One ('1').**
	25:24	**Color Depth:** 00 = 8 bit color 01 = 16 bit color 10 = 24 bit color 11 = reserved
	23:16	**Raster operation:**
	15:00	**Destination Pitch (signed):** (13:00 are implemented in Intel® 810 chipset)
2 = BR14	31:16	**Destination Height (in scan lines):** (28:16 are implemented in Intel® 810 chipset)
	15:00	**Destination Width (in bytes):** (12:00 are implemented in Intel® 810 chipset)
3 = BR09	31:00	**Destination Address:** Address of the first byte to be written (25:00 are implemented in Intel® 810 chipset)
4 = BR11	31:16	Reserved. Must be Zero
	15:00	**Number of Monochrome Source Quadwords - 1:** (1 to 64k Quadwords = 64 to 4M bits)
5 = BR12	31:00	**Source Address:** (address of the first byte of the first pixel on the first scan line) (25:00 are implemented in Intel® 810 chipset)
6 = BR18	31:24	Reserved. Must be Zero
	23:00	**Source Background Color:**
7 = BR19	31:24	Reserved. Must be Zero
	23:00	**Source Foreground Color:**
8 = BR15	31:00	**Pattern Address:** (25:06 are implemented in Intel® 810 chipset)

12.2.15. FULL_MONO_PATTERN_BLT

The full BLT is the most comprehensive BLT instruction. It provides the ability to specify all 3 operands: destination, source, and pattern. The pattern operand is monochrome and the source operand is the same bit width as the destination operand.

The monochrome pattern is loaded from the instruction stream. The pattern vertical alignment specifies which scan line of the pattern is used first. The destination address specifies the horizontal alignment The only operands accessed from memory are the source and destination operands.

The monochrome pattern transparency mode indicates whether to use the pattern background color or deassert the write enables when the bit in the source is 0. When the source bit is 1, then the pattern foreground color is used in the ROP operation. The ROP value chosen should involve the Mono pattern data in the ROP operation.

Both the source and destination pitches can be either sign. The pattern direction follows the destination operand.

The Intel® 815 chipset hardware has a restriction that the BLT color source and destination operands must not co-exist on the same 32-Byte cacheline. To work-around this issue

- For BLTs that have sharing of a cacheline for a given scanline, the driver must treat this as an overlapping BLT case.
- For linear memory, the surfaces allocated must be 32B-cacheline aligned.

DWord	Bit	Description
0 = BR00	31:29	**Client** : 02h – 2D Processor
	28:22	**Instruction Target (Opcode) :** 47h
	21:11	Reserved. Must be Zero
	10:08	**Destination Transparency Mode:** See BR00 definition.
	07:05	**Pattern Vertical Alignment:** (which scan line of the 8x8 pattern to start on)
	04:00	**Dword Length :** 09h
1 = BR13	31	**Solid Pattern Select:** (1 = solid pattern; 0 = no solid pattern)
	30	**X Direction:** (1 = written from right to left (decrementing = backwards; 0 = incrementing)
	29	Reserved. Must be Zero
	28	**Mono Pattern Transparency Mode:** (1 = transparency enabled; 0 = use background)
	27	Reserved. Must be Zero
	26	**Must Be One (‘1’).**
	25:24	**Color Depth:** 00 = 8 bit color 01 = 16 bit color 10 = 24 bit color 11 = reserved
	23:16	**Raster Operation:**
	15:00	**Destination Pitch (signed):** (13:00 are implemented in Intel® 810 chipset)

DWord	Bit	Description
2 = BR14	31:16	**Destination Height (in scan lines):** (28:16 are implemented in Intel® 810 chipset)
	15:00	**Destination Width (in bytes):** (12:00 are implemented in Intel® 810 chipset)
3 = BR09	31:00	**Destination Address:** Address of the first byte to be written (25:00 are implemented in Intel® 810 chipset)
4 = BR11	31:16	Reserved. Must be Zero
	15:00	**Source Pitch:** (quadword aligned and signed) (13:00 are implemented in Intel® 810 chipset)
5 = BR12	31:00	**Source Address:** (25:00 are implemented in Intel® 810 chipset)
6 = BR18	31:24	**Reserved**
	23:00	**Destination Transparency Color:**
7 = BR16	31:24	Reserved. Must be Zero
	23:00	**Pattern Background Color:**
	31:24	Reserved. Must be Zero
8 = BR17	23:00	**Pattern Foreground Color:**
9 = BR20	31:00	**Pattern Data 0:** (least significant DW)
A =BR21	31:00	**Pattern Data 1:** (most significant DW)

12.2.16. FULL_MONO_PATTERN_MONO_SRC_BLT

The full BLT provides the ability to specify all 3 operands: destination, source, and pattern. The pattern and source operands are monochrome.

The monochrome pattern is loaded from the instruction stream. The pattern vertical alignment specifies which scan line of the pattern is used first. The destination address specifies the horizontal alignment. The only memory accesses required for the remainder of the BLT is destination and monochrome source accesses.

The monochrome source transparency mode indicates whether to use the source background color or deassert the write enables when the bit in the source is 0. When the source bit is 1, the source foreground color is used in the ROP operation. The ROP value chosen should involve the Mono source and Mono pattern.

All non-text monochrome sources are word aligned. At the end of a scan line the monochrome source, the bits until the next word boundary must be ignored. The Monochrome source data bit position field [2:0] indicates which bit position within the first byte should be used as the first source pixel.

The monochrome pattern transparency mode indicates whether to use the pattern background color or deassert the write enables when the bit in the source is 0. When the source bit is 1, the pattern foreground color is used in the ROP operation. The monochrome source transparency mode works identical to the pattern transparency mode.

The destination pitches can be either sign. The pattern direction follows the destination operand.

DWord	Bit	Description
0 = BR00	31:29	**Client** : 02h – 2D Processor
	28:22	**Instruction Target (Opcode) :** 48h
	21:20	Reserved. Must be Zero
	19:17	**Monochrome source data bit position of the first pixel within a byte per scan line.**
	16:08	Reserved. Must be Zero
	07:05	**Pattern Vertical Alignment:** (which scan line of the 8x8 pattern to start on)
	04:00	**Dword Length :** 0Ah

DWord	Bit	Description
1 = BR13	31	**Solid Pattern Select:** (1 = solid pattern; 0 = no solid pattern)
	30	Reserved. Must be Zero
	29	**Mono Source Transparency Mode:** (1 = transparency enabled; 0 = use background)
	28	**Mono Pattern Transparency Mode:** (1 = transparency enabled; 0 = use background)
	27	Reserved. Must be Zero
	26	**Must Be One (‘1’).**
	25:24	**Color Depth:** 00 = 8 bit color 01 = 16 bit color 10 = 24 bit color 11 = reserved
	23:16	**Raster Operation:**
	15:00	**Destination Pitch (signed):** (13:00 are implemented in Intel® 810 chipset)
2 = BR14	31:16	**Destination Height (in scan lines):** (28:16 are implemented in Intel® 810 chipset)
	15:00	**Destination Width (in bytes):** (12:00 are implemented in Intel® 810 chipset)
3 = BR09	31:00	**Destination Address:** Address of the first byte to be written (25:00 are implemented in Intel® 810 chipset)
4 = BR11	31:16	Reserved. Must be Zero
	15:00	**Number of Monochrome Source Quadwords - 1:** (1 to 64k Quadwords = 64 to 4M bits)
5 = BR12	31:00	**Source Address:** (25:00 are implemented in Intel® 810 chipset)
6 = BR18	31:24	Reserved. Must be Zero
	23:00	**Source Background Color:**
7 = BR19	31:24	Reserved. Must be Zero
	23:00	**Source Foreground Color:**
8 = BR16	31:24	Reserved. Must be Zero
	23:00	**Pattern Background Color:**
9 = BR17	31:24	Reserved. Must be Zero
	23:00	**Pattern Foreground Color:**
A =BR20	31:00	**Pattern Data 0:** (least significant DW)
B =BR21	31:00	**Pattern Data 1:** (most significant DW)

12.3. BLT Engine Instruction Definitions

This section describes the BLT Engine instruction fields. These descriptions are in the format of register descriptions. For debug purposes, the read-only addresses indicated provide BLT Engine status.

12.3.1. BR00—BLT Opcode and Control

Memory Offset Address:	40000h
Default:	0000 0000
Attributes:	RO; DWord accessible

BR00 is the last executed instruction DWord 0. Bits [22:5] are written by every DW0 of every instruction. Bits [31:30] and [4:0] are status bits. Bits [28:27] are written from the DW0 [15:14] of a Setup instruction and Bit 29 is written with a 1 when a Setup instruction is written. Bit 29 is a decode of the Setup instruction Opcode.

31	30	29	28 22	21 20	19 17	16
BLTB SY	Clip Inst	Setup Mono Pattern	Instruction Target (Opcode)	Reserved. Must be Zero	Monochrome Source Start	Bit (0) / Byte (1) Packed

15 14	13	12	11	10 8	7 5	4	3	2	1	0
Reserved. Must be Zero	Text BLT	Scan Line BLT	Pixel BLT	Destination Transparency Mode	Pattern Vertical Alignment	DST RMW	Color Source	Mono Source	Color Pattern	Mono Pattern

Bit	Descriptions
31	**BLT Engine Busy.** This bit indicates whether the BLT Engine is busy (1) or idle (0). This bit is replicated in the SETUP BLT Opcode & Control register. 1 = Busy 0 = Idle
30	**Clip Instruction.** The current instruction performs clipping (1).
29	**Setup Monochrome Pattern.** This bit is decoded from the Setup instruction opcode to identify whether a color (0) or monochrome (1) pattern is used with the SCANLINE_BLT instruction. 1 = Monochrome 0 = Color
28:22	**Instruction Target (Opcode).** This is the contents of the Instruction Target field from the last BLT instruction. This field is used by the BLT Engine state machine to identify the BLT instruction it is to perform. The opcode specifies whether the source and pattern operands are color or monochrome.
21:20	**Reserved.** Must be Zero.
19:17	**Monochrome Source Start.** This field indicates the starting monochrome pixel bit position within a byte per scan line of the source operand. The monochrome source is word aligned which means that at the end of the scan line all bits should be discarded until the next word boundary.
16	**Bit/Byte Packed .** Byte packed is for the Windows* NT* driver 0 = Bit 1 = Byte
15:14	**Reserved.** Must be Zero.

Bit	Descriptions
13	**Text BLT.** Current Opcode is Text BLT.
12	**Scan Line BLT.** Current Opcode is Scan Line BLT.
11	**Pixel BLT.** Current Opcode is Pixel BLT.
10:8	**Destination Transparency Mode.** These bits control whether or not the byte(s) at the destination corresponding to a given pixel will be conditionally written, and what those conditions are. This feature can make it possible to perform various masking functions in order to selectively write or preserve graphics data already at the destination. All four of the sets of conditions that may be chosen as the controlling factor in performing color transparency involve comparing the color that has been specified to be used in the color expansion of any monochrome source data to other colors. This background color is in the Source Expansion Background Color Register. XX0 = No color transparency mode enabled. This causes normal operation with regard to writing data to the destination. 001 = The color specified as the background color for use in the color expansion of monochrome source data is compared to the color resulting from the bit-wise operation performed for each pixel. If these two colors are not equal, then the byte(s) at the destination corresponding to the current pixel are written with the result of the bit-wise operation. [Source color transparency]. 011 = The color specified as the background color for use in the color expansion of monochrome source data is compared to the color specified by the byte(s) at the destination corresponding to the current pixel. If these two colors are not equal, then the byte(s) at the destination corresponding to the current pixel are written with the result of the bit-wise operation. 101 = The color specified as the background color for use in the color expansion of monochrome source data is compared to the color resulting from the bit-wise operation performed for each pixel. If these two colors are equal, then the byte(s) at the destination corresponding to the current pixel are written with the result of the bit-wise operation. 111 = The color specified as the background color for use in the color expansion of monochrome source data is compared to the color specified by the byte(s) at the destination corresponding to the current pixel. If these two colors are equal, then the byte(s) at the destination corresponding to the current pixel are written with the result of the bit-wise operation. [Destination color transparency]
7:5	**Pattern Vertical Alignment.** Specifies which scan line's worth (which 1 of the 8 horizontal rows) of the 8x8 pattern will appear on the first scan line's worth of the data written to the destination. Depending upon the location of the destination, the upper left-hand corner of the upper left-hand tile of the pattern is usually aligned with the upper left-hand corner of the block of data written to the destination. The BLT Engine determines the horizontal alignment relative to the destination from the lower bits of the destination address, however, the vertical alignment relative to the destination must be supplied through these bits.
4	**Destination Read Modify Write.** This bit is decoded from the last instruction's opcode field and Destination Transparency Mode to identify whether a Destination read is needed.
3	**Color Source.** This bit is decoded from the last instructions opcode field to identify whether a color (1) source is used.
2	**Monochrome Source.** This bit is decoded from the last instructions opcode field to identify whether a monochrome (1) source is used.
1	**Color Pattern.** This bit is decoded from the last instructions opcode field to identify whether a color (1) pattern is used.
0	**Monochrome Pattern.** This bit is decoded from the last instructions opcode field to identify whether a monochrome (1) pattern is used.

12.3.2. BR01—Setup BLT Raster OP, Control, and Destination Offset

Memory Offset Address:	40004h
Default:	0000 xxxx
Attributes:	RO; DWord accessible

BR01 contains the contents of the last Setup instruction DWord 1. It is identical to the BLT Raster OP, Control, and Destination Offset definition, but it is used with the following instructions: PIXEL_BLT, SCANLINE_BLT, and TEXT_BLT.

31	30	29	28	27	26	25 24	23 16
SolPat	Rsvd	Mono Src Trans	Mono Pat Trans	Src Sel Mode	Dyn Color Depth Enable	Color Depth	Raster Operation

15 14	13 0
Rsvd	Destination Pitch (Offset)

Bit	Descriptions
31	**Solid Pattern Select.** This bit applies only when the pattern data is monochrome. This bit determines whether or not the BLT Engine actually performs read operations from the frame buffer in order to load the pattern data. Use of this feature to prevent these read operations can increase BLT Engine performance, if use of the pattern data is indeed not necessary. The BLT Engine is configured to accept either monochrome or color pattern data via the opcode field. 0 = This causes normal operation with regard to the use of the pattern data. The BLT Engine proceeds with the process of reading the pattern data, and the pattern data is used as the pattern operand for all bit-wise operations. 1 = The BLT Engine forgoes the process of reading the pattern data, the presumption is made that all of the bits of the pattern data are set to 0, and the pattern operand for all bit-wise operations is forced to the background color specified in the Color Expansion Background Color Register.
30	**Reserved.** Must be Zero.
29	**Monochrome Source Transparency Mode.** This bit applies only when the source data is in monochrome. This bit determines whether or not the byte(s) at the destination corresponding to the pixel to which a given bit of the source data also corresponds will actually be written if that source data bit has the value of 0. This feature can make it possible to use the source as a transparency mask. The BLT Engine is configured to accepted either monochrome or color source data via the opcode field. 0 = This causes normal operation with regard to the use of the source data. Wherever a bit in the source data has the value of 0, the color specified in the background color register is used as the source operand in the bit-wise operation for the pixel corresponding to the source data bit, and the bytes at the destination corresponding to that pixel are written with the result. 1 = Wher a bit in the source data has the value of 0, the byte(s) at the destination corresponding to the pixel to which the source data bit also corresponds are simply not written, and the data at those byte(s) at the destination are allowed to remain unchanged.

Bit	Descriptions
28	**Monochrome Pattern Transparency Mode.** This bit applies only when the pattern data is monochrome. This bit determines whether or not the byte(s) at the destination corresponding to the pixel to which a given bit of the pattern data also corresponds will actually be written if that pattern data bit has the value of 1. This feature can make it possible to use the pattern as a transparency mask. The BLT Engine is configured to accepted either monochrome or color pattern data via the opcode field. 0 = This causes normal operation with regard to the use of the pattern data. Wherever a bit in the pattern data has the value of 0, the color specified in the background color register is used as the pattern operand in the bit-wise operation for the pixel corresponding to the pattern data bit, and the bytes at the destination corresponding to that pixel are written with the result. 1 = Wher a bit in the pattern data has the value of 0, the byte(s) at the destination corresponding to the pixel to which the pattern data bit also corresponds are simply not written, and the data at those byte(s) at the destination are allowed to remain unchanged.
27	**Source Select Mode.** Configures the BLT Engine data source. 0 = The BLT Engine reads the source data from the frame buffer at the location specified in the Source Address Register. 1 = The BLT Engine accepts the source data from instruction stream controller through the IMMEDIATE_BLT instruction. The BLT Engine will hang if it doesn't get an even number of DWs
26	**Reserved.** Must Be One ('1').
25:24	**Dynamic Color Depth.** When a SETUP_BLT or SETUP_MONO_PATTERN_SL_BLT is parsed, the Color Depth field from these blit commands is reflected in these 2 bits. 00 = 8 Bit Color Depth 01 = 16 Bit Color Depth 10 = 24 Bit Color Depth 11 = Reserved
23:16	**Raster Operation Select.** These 8 bits are used to select which one of 256 possible raster operations is to be performed by the BLT Engine. The 8-bit values, and their corresponding raster operations, are intended to correspond to the 256 possible raster operations specified for graphics device drivers in the Microsoft Windows* environment. The opcode field must indicate a monochrome source if ROP = F0.
15:14	**Reserved.** Must be Zero.
13:0	**Destination Pitch (Offset).** These 14 bits store the signed memory address offset value by which the destination address originally specified in the Destination Address Register is incremented or decremented as each scan line's worth of destination data is written into the frame buffer by the BLT Engine, so that the destination address will point to the next memory address to which the next scan line's worth of destination data is to be written. If the intended destination of a BLT operation is within on-screen frame buffer memory, this offset is normally set so that each subsequent scan line's worth of destination data lines up vertically with the destination data in the scan line, above. However, if the intended destination of a BLT operation is within off-screen memory, this offset can be set so that each subsequent scan line's worth of destination data is stored at a location immediately after the location where the destination data for the last scan line ended, to create a single contiguous block of bytes of destination data at the destination.

12.3.3. BR02—Clip Rectangle Y1 Address

Memory Offset Address:	40008h
Default:	None
Attributes:	RO; DWord accessible

BR02 is loaded by either the SETUP_BLT or SETUP_MONO_PATTERN_SL_BLT instructions and is used with PIXEL_BLT, SCANLINE_BLT, or TEXT_BLT instructions.

31 26	25 0
Reserved. Must be Zero	Clip Rectangle Y1 Address Bits [25:0]

Bit	Descriptions
31:26	**Reserved.** Must be Zero. The maximum GC graphics address is 64 MB.
25:0	**Clip Rectangle Y1 Address Bits [25:0].** These 26 bits specify the top clipping address of the destination data. This clip compare is inclusive: draw if destination address is greater than or equal to this address. This address points to the first address of a scan line. The Clip Rectangle X registers take care of the pixel positions within a scan line.

12.3.4. BR03—Clip Rectangle Y2 Address

Memory Offset Address:	4000Ch
Default:	None
Attributes:	RO; DWord accessible

BR03 is loaded by either the SETUP_BLT or SETUP_MONO_PATTERN_SL_BLT instructions and is used with PIXEL_BLT, SCANLINE_BLT, or TEXT_BLT instructions.

31 26	25 0
Reserved. Must be Zero	Clip Rectangle Y2 Address Bits [25:0]

Bit	Descriptions
31:26	**Reserved.** Must be Zero. The maximum GC graphics address is 64 MBs. Debug implementation specific = Leftdiscard[11:06]
25:0	**Clip Rectangle Y2 Address Bits [25:0].** These 26 bits specify the bottom clipping address of the destination data. This clip compare is inclusive: draw if destination address is less than or equal to this address. This address points to the first byte of a scan line. The Clip Rectangle X registers take care of the pixel positions within a scan line.

12.3.5. BR04—Clip Rectangle X1 and X2

Memory Offset Address:	40010h
Default:	None
Attributes:	RO; DWord accessible

BR04 is loaded by either the SETUP_BLT or SETUP_MONO_PATTERN_SL_BLT instructions and is used with PIXEL_BLT, SCANLINE_BLT, or TEXT_BLT instructions.

31 28	27 16
Reserved. Must be Zero	Clip Rectangle X2 coordinate (right) [11:00]

15 12	11 0
Reserved. Must be Zero	Clip Rectangle X1 coordinate (left) [11:00]

Bit	Descriptions
31:28	**Reserved.** Must be Zero.
27:16	**Clip Rectangle X2 Coordinate.** These 12 bits specify the right most X coordinate which is written to the destination. The comparison is inclusive with a less than or equal. The byte address of this coordinate is: scan line address + X2 * bytes/pixel.
15:12	**Reserved.** Must be Zero.
11:0	**Clip Rectangle X1 Coordinate.** These 12 bits specify the left most X coordinate which is written to the destination. The comparison is inclusive with a greater than or equal. The byte address of this coordinate is: scan line address + X1 * bytes/pixel.

12.3.6. BR05—Setup Expansion Background Color

Memory Offset Address:	40014h
Default:	None
Attributes:	RO; DWord accessible

31 24	23 0
Reserved. Must be Zero	Setup Expansion Background Color Bits [23:0]

Bit	Descriptions
31:24	**Reserved.** Must be Zero.
23:0	**Setup Expansion Background Color Bits [23:0].** These bits provide the one, two, or three bytes worth of color data that select the background color to be used in the color expansion of monochrome pattern or source data for either the SCANLINE_BLT or TEXT_BLT instructions. BR05 is also used as the solid pattern for the PIXEL_BLT instruction. Whether one, two, or three bytes worth of color data is needed depends upon the color depth to which the BLT Engine has been set. For a color depth of 24bpp, 16bpp and 8bpp, bits [23:0], [15:0] and [7:0], respectively, are used.

12.3.7. BR06—Setup Expansion Foreground Color

Memory Offset Address:	40018h
Default:	None
Attributes:	RO; DWord accessible

31 24	23 0
Reserved. Must be Zero	Setup Expansion Foreground Color Bits [23:0]

Bit	Descriptions
31:24	**Reserved.** Must be Zero.
23:0	**Setup Expansion Foreground Color Bits [23:0].** These bits provide the one, two, or three bytes worth of color data that select the foreground color to be used in the color expansion of monochrome pattern or source data for either the SCANLINE_BLT or TEXT_BLT instructions. Whether one, two, or three bytes worth of color data is needed depends upon the color depth to which the BLT Engine has been set. For a color depth of 24bpp, 16bpp and 8bpp, bits [23:0], [15:0] and [7:0], respectively, are used.

12.3.8. BR07—Setup Color Pattern Address

Memory Offset Address:	4001Ch
Default:	None
Attributes:	RO; DWord accessible

31 — 26	25 — 16
Reserved. Must be Zero	Setup Color Pattern Address Bits [25:16]

15 — 6	5 — 0
Setup Color Pattern Address Bits [15:6]	Reserved. Must be Zero

Bit	Descriptions
31:26	**Reserved.** Must be Zero. The maximum GC graphics address is 64 MBs.
25:6	**Pattern Address.** These 20 bits specify the starting address of the color pattern from the SETUP_BLT instruction. This register works identically to the Pattern Address register, but this version is only used with the SCANLINE_BLT instruction execution. The pattern data must be located on a pattern-size boundary. The pattern is always of 8x8 pixels, and therefore, its size is dependent upon its pixel depth. The pixel depth may be 8, 16, or 24 bits per pixel if the pattern is in color (the pixel depth of a color pattern must match the pixel depth to which the graphics system has been set). Monochrome patterns require 8 bytes and is supplied through the instruction. Color patterns of 8, 16, and 24 bits per pixel color depth must start on 64-byte, 128-byte and 256-byte boundaries, respectively. **Note:** In the case of 24 bits per pixel, each scan line's worth (each row of 8 pixels) of pattern data takes up 24 consecutive bytes.
5:0	**Reserved.** Must be Zero. These bits always return 0 when read.

12.3.9. BR08—Destination X1 and X2

Memory Offset Address:	40020h
Default:	None
Attributes:	RO; DWord accessible

BR08 is loaded by either PIXEL_BLT, SCANLINE_BLT, or TEXT_BLT instructions. The PIXEL_BLT instruction only writes the destination X coordinate register.

31 28	27 16
Reserved. Must be Zero	Destination X2 coordinate (right) [11:00]

15 12	11 0
Reserved. Must be Zero	Destination X1 or X coordinate (left) [11:00]

Bit	Descriptions
31:28	**Reserved.** Must be Zero.
27:16	**Destination X2 coordinate.** These 12 bits specify the right most X coordinate which is written to the destination if not clipped. The comparison is inclusive with a less than or equal. The byte address of this coordinate is: scan line address + X2 * bytes/pixel.
15:12	**Reserved.** Must be Zero.
11:0	**Destination X1 or X coordinate (left).** These 12 bits specify the left most X coordinate which is written to the destination. This is also the working register, where it changes while the BLT Engine is working. The comparison is inclusive with a greater than or equal. The byte address of this coordinate is: scan line address + X1 * bytes/pixel. Note: Some instructions affect only one pixel (requiring only one coordinate); other instructions affect multiple pixels and need both coordinates.

12.3.10. BR09—Destination Address and Destination Y1 Address

Memory Offset Address:	40024h
Default:	None
Attributes:	RO; DWord accessible

31 26	25 0
Reserved. Must be Zero	Destination and Destination Y1 and Y Address Bits [25:0]

Bit	Descriptions
31:26	**Reserved.** Must be Zero.
25:0	**Destination and Destination Y1 and Y Address Bits.** These 26 bits specify the starting pixel address of the destination data. This register is also the working destination address register and changes as the BLT Engine performs the accesses. Used as the scan line address (Destination Y Address & Destination Y1 Address) for BLT instructions: PIXEL_BLT, SCANLINE_BLT, and TEXT_BLT. In this case the address points to the first pixel in a scan line and is compared with the ClipRect Y1 & Y2 address registers to determine whether the scan line should be written or not. The Destination Y1 address is the top scan line to be written for text. This register is always the last register written for a BLT drawing instruction. Writing BR09 starts the BLT engine execution. Note: Some instructions affect only one scan line (requiring only one coordinate); other instructions affect multiple scan lines and need both coordinates.

NOTES: This is a working register. If BR09 is read while the BLT engine is busy, the contents are unpredictable.

12.3.11. BR10—Destination Y2 Address

Memory Offset Address:	40028h
Default:	None
Attributes:	RO; DWord accessible

31 26	25 0
Reserved. Must be Zero	Destination Y2 Address Bits [25:0]

Bit	Descriptions
31:26	**Reserved.** Must be Zero. The maximum GC graphics address is 64 MBs. Debug implementation specific = bmtdpix[5:0]
25:0	**Destination Y2 Address.** Used as the scan line address (Destination Y2 Address) for BLT instruction: TEXT_BLT. The address points to the first pixel in a scan line and is compared with the ClipRect Y1 & Y2 address registers to determine whether the scan line should be written or not. The Destination Y2 address is the bottom scan line to be written for text.

12.3.12. BR11—BLT Source Pitch (Offset) or Monochrome Source Quadwords

Memory Offset Address:	4002Ch
Default:	None
Attributes:	RO; DWord accessible

31 … 14	13 … 0
Reserved	Source Pitch (Offset) or Monochrome Source Quadwords

Bit	Descriptions
31:14	**Reserved.**
13:0	**Source Pitch (Offset) or Monochrome Source Quadwords.** When the color source data is located within the frame buffer or AGP aperture, these signed 14 bits store the memory address offset (pitch) value by which the source address originally specified in the Source Address Register is incremented or decremented as each scan line's worth of source data is read from the frame buffer by the BLT Engine, so that the source address will point to the next memory address from which the next scan line's worth of source data is to be read. When the source data is provided by IMMEDIATE_BLT instruction, these 14 bits store the number of bytes to be counted from the beginning of a scan line's worth of data to where the next scan line's worth of data begins. Note that if the intended source of a BLT operation is within on-screen frame buffer memory, this offset is normally set to accommodate the fact that each subsequent scan line's worth of source data lines up vertically with the source data in the scan line, above. However, if the intended source of a BLT operation is within off-screen memory, this offset can be set to accommodate a situation in which the source data exists as a single contiguous block of bytes where in each subsequent scan line's worth of source data is stored at a location immediately after the location where the source data for the last scan line ended. When monochrome source data is being processed, this field is used for indicating the total number of Quadwords of data in the source data stream.

12.3.13. BR12—Source Address

Memory Offset Address:	40030h
Default:	None
Attributes:	RO; DWord accessible

31 26	25 0
Reserved. Must be Zero	Source Address Bits [25:0]

Bit	Descriptions
31:26	**Reserved.** Must be Zero. The maximum GC Graphics address is 64 MBs.
25:0	**Source Address Bits [25:0].** These 26 bits are used to specify the starting pixel address of the color source dataThe lower 3 bits are used to indicate the position of the first valid byte within the first Quadword of the source data.

NOTES: This is a working register. If BR12 is read while the BLT Engine is busy, the contents are unpredictable. The value contained in this register while the BLT Engine is busy has no relationship to the programmed value that can be read when the BLT Engine is idle.

12.3.14. BR13—BLT Raster OP, Control, and Destination Pitch

Memory Offset Address:	40034h
Default:	0000 xxxx
Attributes:	RO; DWord accessible

31	30	29	28	27	26	25 24	23 16
Sol Pat Sel	X Dir	Mon Src Trans	Mon Pat Trans	Src Sel Mode	Dyn Col En	Dynamic Color Depth	Raster Operation Select

15 14	13 0
Reserved. Must be Zero	Destination Pitch (Offset)

Bit	Descriptions
31	**Solid Pattern Select.** This bit applies only when the pattern data is monochrome. This bit determines whether or not the BLT Engine actually performs read operations from the frame buffer to load the pattern data. Use of this feature to prevent these read operations can increase BLT Engine performance, if use of the pattern data is indeed not necessary. The BLT Engine is configured to accept either monochrome or color pattern data via the opcode field. 0 = This causes normal operation with regard to the use of the pattern data. The BLT Engine proceeds with the process of reading the pattern data, and the pattern data is used as the pattern operand for all bit-wise operations. 1 = The BLT Engine forgoes the process of reading the pattern data, the presumption is made that all of the bits of the pattern data are set to 0, and the pattern operand for all bit-wise operations is forced to the background color specified in the Color Expansion Background Color Register.
30	**X Increment/Decrement Select.** 0 = The bytes corresponding to the pixels within each scan line of data are written to the destination starting with the bytes corresponding to the left-most pixel of each scan line at the destination being the first to be written, and then proceeding in a rightward direction toward the right-most pixel. 1 = The bytes corresponding to the pixels within each scan line of data are written to the destination starting with the bytes corresponding to the right-most pixel of each scan line at the destination being the first to be written, and then proceeding in a leftward direction toward the left-most pixel.
29	**Monochrome Source Transparency Mode.** This bit applies only when the source data is in monochrome. This bit determines whether or not the byte(s) at the destination corresponding to the pixel to which a given bit of the source data also corresponds will actually be written if that source data bit has the value of 0. This feature can make it possible to use the source as a transparency mask. The BLT Engine is configured to accepted either monochrome or color source data via the opcode field. 0 = This causes normal operation with regard to the use of the source data. Where a bit in the source data has the value of 0, the color specified in the background color register is used as the source operand in the bit-wise operation for the pixel corresponding to the source data bit, and the bytes at the destination corresponding to that pixel are written with the result. 1 = Where a bit in the source data has the value of 0, the byte(s) at the destination corresponding to the pixel to which the source data bit also corresponds are simply not written, and the data at those byte(s) at the destination are allowed to remain unchanged.

Bit	Descriptions
28	**Monochrome Pattern Transparency Mode.** This bit applies only when the pattern data is monochrome. This bit determines whether or not the byte(s) at the destination corresponding to the pixel to which a given bit of the pattern data also corresponds will actually be written if that pattern data bit has the value of 1. This feature can make it possible to use the pattern as a transparency mask. The BLT Engine is configured to accepted either monochrome or color pattern data via the opcode in the Opcode and Control register. 0 = This causes normal operation with regard to the use of the pattern data. Where a bit in the pattern data has the value of 0, the color specified in the background color register is used as the pattern operand in the bit-wise operation for the pixel corresponding to the pattern data bit, and the bytes at the destination corresponding to that pixel are written with the result. 1 = Wherever a bit in the pattern data has the value of 0, the byte(s) at the destination corresponding to the pixel to which the pattern data bit also corresponds are simply not written, and the data at those byte(s) at the destination are allowed to remain unchanged.
27	**Source Select Mode.** 0 = Configures the BLT Engine to read the source data from the frame buffer at the location specified in the Source Address Register. 1 = Configures the BLT Engine to accept the source data from the instruction stream controller through the IMMEDIATE_BLT instruction. The BLT Engine will hang if it does not get an even number of DWs.
26	**Reserved.** Must Be One (‘1’).
25:24	**Dynamic Color Depth.** 00 = 8 Bit Color Depth 01 = 16 Bit Color Depth 10 = 24 Bit Color Depth 11 = Reserved
23:16	**Raster Operation Select.** These 8 bits are used to select which one of 256 possible raster operations is to be performed by the BLT Engine. The 8-bit values, and their corresponding raster operations, are intended to correspond to the 256 possible raster operations specified for graphics device drivers in the Microsoft Windows* environment. The opcode must indicate a monochrome source operand if ROP = F0.
15:14	**Reserved.** Must be Zero.
13:0	**Destination Pitch (Offset).** These 14 bits store the signed memory address offset value by which the destination address originally specified in the Destination Address Register is incremented or decremented as each scan line’s worth of destination data is written into the frame buffer by the BLT Engine, so that the destination address will point to the next memory address to which the next scan line’s worth of destination data is to be written. If the intended destination of a BLT operation is within on-screen frame buffer memory, this offset is normally set so that each subsequent scan line’s worth of destination data lines up vertically with the destination data in the scan line, above. However, if the intended destination of a BLT operation is within off-screen memory, this offset can be set so that each subsequent scan line’s worth of destination data is stored at a location immediately after the location where the destination data for the last scan line ended, in order to create a single contiguous block of bytes of destination data at the destination.

12.3.15. BR14—Destination Width & Height

Memory Offset Address:	40038h
Default:	None
Attributes:	RO; DWord accessible

BR14 contains the values for the height and width of the data to be BLT. If these values are not correct, such that the BLT Engine is either expecting data it does not receive or receives data it did not expect, the system can hang.

31 : 29	28 : 16
Reserved. Must be Zero	Destination Height

15 : 13	12 : 0
Reserved. Must be Zero	Destination Byte Width

Bit	Descriptions
31:29	**Reserved.** Must be Zero.
28:16	**Destination Height.** These 13 bits specify the height of the destination data in terms of the number of scan lines. This is a working register.
15:13	**Reserved.** Must be Zero.
12:0	**Destination Byte Width.** These 13 bits specify the width of the destination data in terms of the number of bytes per scan line. The number of pixels per scan line into which this value translates depends upon the color depth to which the graphics system has been set.

NOTES: This is a working register. If BR14 is read while the BLT Engine is busy, the contents are unpredictable. The value contained in this register while the BLT Engine is busy has no relationship to the programmed value that can be read when the BLT Engine is idle.

12.3.16. BR15—Color Pattern Address

Memory Offset Address:	4003Ch
Default:	None
Attributes:	RO; DWord accessible

31 26	25 16
Reserved. Must be Zero	Color Pattern Address Bits [25:16]

15 6	5 0
Color Pattern Address Bits [15:6]	Reserved. Must be Zero

Bit	Descriptions
31:26	**Reserved.** Must be Zero. The maximum GC graphics address is 64 MBs.
25:6	**Color Pattern Address.** These 20 bits specify the starting address of the pattern. The pattern data must be located on a pattern-size boundary. The pattern is always of 8x8 pixels, and therefore, its size is dependent upon its pixel depth. The pixel depth may be 8, 16, or 24 bits per pixel if the pattern is in color (the pixel depth of a color pattern must match the pixel depth to which the graphics system has been set). Monochrome patterns require 8 bytes and is supplied through the instruction. Color patterns of 8, 16, and 24 bits per pixel color depth must start on 64-byte, 128-byte and 256-byte boundaries, respectively. **Note:** In the case of 24 bits per pixel, each scan line's worth (each row of 8 pixels) of pattern data takes up 32 consecutive bytes, not 24. It is formatted so that there is a contiguous block of 8 sets of 3 bytes, each corresponding to 1 of the 8 pixels, followed by a contiguous block of the 8 extra bytes. When the BLT Engine reads 24 bit-per-pixel pattern data, it will read only the first 24 bytes of each scan line's worth of data, picking up the 8 sets of 3 bytes for each of the 8 pixels, and entirely ignoring the remaining 8 bytes.
5:0	**Reserved.** Must be Zero. These bits always return 0 when read.

12.3.17. BR16—Pattern Expansion Background & Solid Pattern Color

Memory Offset Address:	40040h
Default:	None
Attributes:	RO; DWord accessible

31 24	23 0
Reserved. MBZ	Pattern Expansion Background Color Bits [23:0]

Bit	Descriptions
31:24	**Reserved.** Must be Zero.
23:0	**Pattern Expansion Background Color Bits [23:0].** These bits provide the one, two, three, or four bytes worth of color data that select the background color to be used in the color expansion of monochrome pattern data during BLT operations. Whether one, two, three, or four bytes worth of color data is needed depends upon the color depth to which the BLT Engine has been set. For a color depth of 24bpp, 16bpp and 8bpp, bits [23:0], [15:0] and [7:0], respectively, are used.

12.3.18. BR17—Pattern Expansion Foreground Color

Memory Offset Address:	40044h
Default:	None
Attributes:	RO; DWord accessible

31 24	23 0
Reserved. Must be Zero	Pattern Expansion Foreground Color Bits [23:0]

Bit	Descriptions
31:24	**Reserved.** Must be Zero.
23:0	**Pattern Expansion Foreground Color Bits [23:0].** These bits provide the one, two, three, or four bytes worth of color data that select the foreground color to be used in the color expansion of monochrome pattern data during BLT operations. Whether one, two, or three, or bytes worth of color data is needed depends upon the color depth to which the BLT Engine has been set. For a color depth of 24bpp, 16bpp and 8bpp, bits [23:0], [15:0] and [7:0], respectively, are used.

12.3.19. BR18—Source Expansion Background, and Destination Color

Memory Offset Address:	40048h
Default:	None
Attributes:	RO; DWord accessible

31 24	23 0
Reserved. Must be Zero	Source Expansion Background Color Bits [23:0]

Bit	Descriptions
31:24	**Reserved.** Must be Zero. Debug implementation specific [31:25] = bmnewcliplw[6:0].
23:0	**Source Expansion Background Color Bits [23:0].** These bits provide the one, two, or three bytes worth of color data that select the background color to be used in the color expansion of monochrome source data during BLT operations. This register is also used to support destination transparency mode and Solid color fill. Whether one, two, three, or four bytes worth of color data is needed depends upon the color depth to which the BLT Engine has been set. For a color depth of 24bpp, 16bpp and 8bpp, bits [23:0], [15:0] and [7:0], respectively, are used.

12.3.20. BR19—Source Expansion Foreground Color

Memory Offset Address:	4004Ch
Default:	None
Attributes:	RO; DWord accessible

31 24	23 0
Reserved. Must be Zero	Source Expansion Foreground Color Bits [23:0]

Bit	Descriptions
31:24	**Reserved.** Must be Zero. Debug implementation specific [31:26] = bmnewcliplw[12:07].
23:0	**Pattern/Source Expansion Foreground Color Bits [23:0].** These bits provide the one, two, or three bytes worth of color data that select the foreground color to be used in the color expansion of monochrome source data during BLT operations. Whether one, two, or three bytes worth of color data is needed depends upon the color depth to which the BLT Engine has been set. For a color depth of 24bpp, 16bpp and 8bpp, bits [23:0], [15:0] and [7:0], respectively, are used.

12.3.21. S_SLADD—Source Scan Line Address

Memory Offset Address:	40074h
Default:	None
Attributes:	RO; DWord accessible

31 26	25 0
Reserved. Must be Zero	Source Scan Line Address Bits [25:0]

Bit	Descriptions
31:26	**Reserved.** Must be Zero. The maximum GC graphics address is 64 MBs.
25:0	**Source Scan Line Address.** These 26 bits are used by the over scan line fetching state machine to address the source data. Source data is read when ever there is room in the internal memory buffer to avoid the read latency between scan lines.

NOTES: This is a working register. If this register is read while the BLT Engine is busy, the contents are unpredictable.

12.3.22. D_SLH—Destination Scan Line Height

Memory Offset Address:	40078h
Default:	None
Attributes:	RO; DWord accessible

D_SLH is a working register, which contains the current height that the over scan line state machine uses.

31 16	15 0
Destination Scan Line Height	Reserved. Must be Zero

Bit	Descriptions
31:16	**Destination Scan Line Height.** These 16 bits specify the height of the destination data in terms of the number of scan lines.
15:0	**Reserved.** Must be Zero.

NOTES: This is a working register. If this register is read while the BLT Engine is busy, the contents are unpredictable.

12.3.23. D_SLRADD—Destination Scan Line Read Address

Memory Offset Address:	4007Ch
Default:	None
Attributes:	RO; DWord accessible

31 26	25 0
Reserved. Must be Zero	Destination Scan Line Read Address Bits [25:0]

Bit	Descriptions
31:26	**Reserved.** Must be Zero. The maximum GC Graphics address is 64 MBs.
25:0	**Destination Address.** These 26 bits specify the scan line address of the destination data being read. This is a working register.

NOTES: This is a working register. If this register is read while the BLT Engine is busy, the contents are unpredictable.

This page is intentionally left blank.

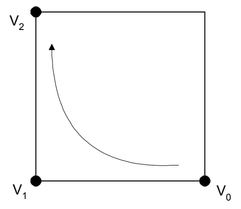
13. Rendering Engine Instructions

This chapter describes the 3D instructions and motion compensation instruction that controls the Graphics controller (GC) rendering engine. The GC Rendering Engine shall receive all software driver instructions through the instruction interface (ring buffers).

13.1. GFXPRIMITIVE

This instruction performs most of the rendering operations performed by the GC. Triangle and Triangle lists are rendered using this instruction. Triangle strips and fans reduce the number of vertices which must be delivered to the hardware for adjacent triangles and are also supported by this instruction. Lines and Line Lists continue to be supported. The GC also supports the rendering of axis aligned rectangles. The vertices of these primitives are may be of variable length. For example, one call to GFXPRIMITIVE may include only the X, Y values at the vertices, while another call may specify all 44 bytes of attribute information at each vertex. The following sections consider these primitives in detail.

Figure 30. Rectangle Vertices



The GC shall support both D3D* and OGL notations. The selection can be done in the GFXRENDERSTATE_RASTER_RULE state variable.

13.1.1. Axis Aligned Rectangles

Axis aligned rectangles are described by three vertices as shown in the adjacent figure. The vertices should always describe a right triangle where the base of the triangle is parallel to the x-axis and the vertical leg of the triangle is parallel to the y-axis.

13.1.2. Primitive Winding Order

The winding order of the vertices is considered if Primitive Culling is enabled. In the example above the winding order is clock-wise. In triangle strip, the winding order toggles on every triangle (e.g., CW, CCW, CW, CCW and so on). Therefore, hardware toggles the culling orientation on every triangle to match up with the strip sequence. If the primitive type is triangle strip, culling orientation is toggled on the 2nd, 4th, 6th triangle and so on. If the primitive type is triangle strip with reverse winding order, culling orientation is toggled on the 1st, 3rd, 5th triangles and so on.

13.1.3. Position Mask

In Variable vertex format, a position mask is sent to indicate the present of X, Y, Z, and 1/W parameters. If 1/W (RHW) is declared not present in the vertex packet, hardware forces 1/w equal to 1. In the case Z is declared not present, the Z value of the last triangle from the last vertex instruction packet will be used for the triangles in this packet.

13.1.4. Bias

A Z bias value can be included in the vertex instruction packet as a per polygon basis. This value is a floating point number that ranges from -1 to 1. The Z bias value is added to the Z value and then clamped to between 0 to 1 before primitive setup calculation. In triangle or line primitive type, Z bias of the last vertex (3rd in triangle, 2nd in line) will be used to add to all the vertices of the primitive and the Z bias value on the other vertices will be ignored. In triangle strip and fan, the Z bias value on the 3rd vertex will be used for the first triangle and the Z bias value on subsequent vertex which defined a triangle will be used for that triangle.

13.1.5. Primitive Rendering Instruction Format

DWord	Bits	Description
0	31:29	**Client :** 03h – Rendering Engine
	28:24	**Rendering Primitive :** 1Fh
	23	**Reserved:** 00h
	22:18	**Primitive Type:** 0h = Triangle or Triangle List 1h = Triangle Strip 2h = Triangle Strip with Reverse Winding Order 3h = Triangle Fan 4h = Polygon(Triangle Fan with Common Flat Shaded Vtx) 5h = Line or Line List 6h = Line Strip 7h = Rectangle or Rectangle List (must be axis aligned) 8h–1Fh = Reserved
	17:0	**DWord Count**
1	N/A	**Primitive[0] Vertex[0]:** GFXVERTEX
	N/A	**Primitive[0] Vertex[1]:** GFXVERTEX
	:	:
Variable	N/A	**Primitive[n] Vertex[2]:** GFXVERTEX Last DWORD of last vertex of last primitive

Refer to GFXVERTEX section for details on defining primitive vertex.

13.1.6. Variable Length Vertex Formats for Rendering Instructions

The GC supports variable length vertex formats. These formats are determined, by enable bits contained in the variable length vertex format (VLVF) instructions. The following table specifies the attributes associated with each vertex. The order of the attributes must be strictly observed.

For any attribute that is defined as not present, the value of the corresponding attribute from the last triangle of last packet will be used for the triangles in this packet. However, in the case of strip and fan primitives, the vertex numbers rotate from one primitive to the next primitive (e.g., the third vertex of one triangle will become the first vertex of the next triangle). Therefore, the re-used attribute value from the last primitive will not be guaranteed to be on the same vertex of the next primitive. In this case, undesired results may occur. It is recommended to turn off features that have the corresponding attributes disabled (e.g., if no texture coordinate is send, then turn off texture mapping feature).

Vertex Attribute	Comments
X Position	Required for every vertex
Triangle Edge V2–V0 Enable	Required only on the first vertex of a triangle and on each new vertex, which defines a triangle when processing strips and fans.
Triangle Edge V1–V2 Enable	Required only on the first vertex of a triangle and on each new vertex, which defines a triangle when processing strips and fans.
Triangle Edge V0–V1 Enable	Required only on the first vertex of a triangle and on each new vertex, which defines a triangle when processing strips and fans.
Y Position	Required for every vertex
Z Position	Optional
Z Bias	Optional
Reciprocal of W	Optional
Diffuse Color (Alpha, Red, Green and Blue)	Optional
Specular Color/Fog Factor	Optional
Texture Coordinate Set 0	Optional
Texture Coordinate Set 1	Optional

13.1.7. GFXVERTEX

Rendering Processor instructions include data per vertex. The vertex data format is the same for these instructions. GFXVERTEX is not an instruction, but a definition of this vertex data format. GFXVERTEX does not include an instruction header since it is not an instruction. The GFXVERTEX format is:

Dword	Bits	Description
0	31:4	**X position :** Relative to origin drawing rectangle or dest buffer. Valid data range is -383 to 1663. (IEEE float except 19 bit mantissa)
	3	**Reserved:** 00h
	2	**Triangle Edge V2–V0 Enable :** This flag enables anti-aliasing and wireframe capabilities for the triangle edge defined by the third and first vertices (1 = enable, 0 = disable). This flag field is valid only in the first vertex of the triangle definition. It is reserved otherwise.
	1	**Triangle Edge V1–V2 Enable :** This flag enables anti-aliasing and wireframe capabilities for the triangle edge defined by the second and third vertices (1 = enable, 0 = disable). This flag field is valid only in the first vertex of the triangle definition. It is reserved otherwise.
	0	**Triangle Edge V0–V1 Enable :** This flag enables anti-aliasing and wireframe capabilities for the triangle edge defined by the first and second vertices (1 = enable, 0 = disable). This flag field is valid only in the first vertex of the triangle definition. It is reserved otherwise.
1	31:0	**Y position :** Relative to origin of drawing rectangle or dest buffer. Valid data range is -383 to 1663. (IEEE float)
2	31:0	**Z position :** Normalized depth. Valid data range is 0 to 1. (IEEE float)
3	31:0	**Z Bias:** Valid data range is -1 to 1. (IEEE float)
4	31:0	**Reciprocal of W** : Valid data is any positive number.(IEEE float)
5	31:24	**Color Alpha** : Valid data range is 0 to 255. (unsigned int)
	23:16	**Color Red :** Valid data range is 0 to 255. (unsigned int)
	15:8	**Color Green :** Valid data range is 0 to 255. (unsigned int)
	7:0	**Color Blue :** Valid data range is 0 to 255. (unsigned int)
6	31:24	**Fog Factor :** Valid data range is 0 to 255. (unsigned int)
	23:16	**Specular Red :** Valid data range is 0 to 255. (unsigned int)
	15:8	**Specular Green :** Valid data range is 0 to 255. (unsigned int)
	7:0	**Specular Blue :** Valid data range is 0 to 255. (unsigned int)
7	31:0	**Tu 0 :** Texture coordinates. Data valid over entire IEEE floating point range. (IEEE float)
8	31:0	**Tv 0:** Texture coordinates. Data valid over entire IEEE floating point range. (IEEE float)
9	31:0	**Tu 1:** Texture coordinates. Data valid over entire IEEE floating point range. (IEEE float)
10	31:0	**Tv 1:** Texture coordinates. Data valid over entire IEEE floating point range. (IEEE float)

13.2. GFXRENDERSTATE_VERTEX_FORMAT

Flexible Vertex Format Packet

DWord	Bit	Description
0	31:29	**Client :** 03h – Render Processor
	28:24	**3DState24 :** 05h
	23:12	**Reserved :** 00h
	11:8	**Texture Coordinate Count:** This field identifies how many coordinates are present in the vertex. The valid range is 0–2.
	7	**Specular Color and Fog Factor Present**
	6	**Diffuse Color and Alpha Present**
	5	**Z-Offset Present**
	4	**Reserved:** 00h (DX6 Normal Vector Present Bit)
	3:1	**Position Mask:** 0h = Invalid 1h = XYZ Present, RHW not present 2h = XYZRHW Present 3h = XY Present, RHW and Z not present 4h = XYRHW Present, Z not present 5h–7h = Reserved
	0	**Reserved:** 00h

13.3. GFXBLOCK

The GFXBLOCK instruction is used with Motion Compensation. It is a variable length instruction, which contains intra-coded/correction data at the end of the instruction. The DWORD_LENGTH must correspond to the Block Pattern Bits that identify which quadrants of the block have associated correction data. When the horizontal or vertical block pattern are disabled the block pattern bits are assumed to be set. All of the block pattern bits must also be set when the Prediction Type is "intra-coded". Having no correction data is also valid to enable support for skipped blocks.

When the Block Pattern Format is disabled, bits [29:28]=0 and any Y, Cr, or Cb pattern bits, bits [27:22], are set, the error data must be set to 0. The alternative is to ensure that the Y, Cr, or Cb pattern bits, bits [27:22], are set to 0.

DWord	Bits	Description
0	31:29	**Client :** 03h – Rendering Engine
	28:24	**Rendering Block :** 1Eh
	23:16	**Opcode:** 00h
	15:0	**DWORD_LENGTH :** 5 + Intra-coded/Correction Data DWords
1	31:30	**Block Type:** 00 = Reserved (Macroblock – The Luminance and Chromanance blocks are all predicted from this instruction. The Height, Width and Motion Vectors are shifted right 1-bit when processing the chromanance blocks.) 01 = Luminance (Y) Block 10 = Chromanance Red (Cr) Block 11 = Chromanance Blue (Cb) Block
	29:28	**Block Pattern Format:** 00 = Disable Block Pattern Usage and Intra-coded/Correction data. The Height and Width are not constrained and need only be less than 1024. Intra-coded/Correction data is not allowed in this state. 01 = Single Block - Uses bit 27 for a Luminance block and bits 23,22 for the chromanance blocks respectively. If intra-coded/ correction data is present the order of the data is row-major for the entire width of the block. 10 = Halves: Left and Right - Uses bits 27, 26 for the two halves of the Luminance block respective. Intra-coded/ correction data is delivered in row-major order for the left half of the block, followed by the data for the right half of the block. Chromanance blocks are always considered to be Single blocks, using bits 23 and 22 for the Cr and Cb blocks, respectively. 11 = Quadrants: Upper-left, Upper-right, Lower-left, Lower-right - Uses bits 27:24 for the four quadrants of the Luminance block respective. Intra-coded/correction data is delivered in row-major order for the four quadrants of the block beginning with the upper-left quadrant and proceeding to the upper-right, lower-left and lower-right quadrants. Chromanance blocks are always considered to be Single blocks, using bits 23 and 22 for the Cr and Cb blocks respectively.
	27:24	**Y Block Pattern:** Enable/disable correction data for the Y block(s)
1	23	**Cr Block Pattern:** Enable/disable intra-coded/correction data for the Cr block

DWord	Bits	Description
1	22	**Cb Block Pattern:** Enable/disable intra-coded/correction data for the Cb block
	21:18	**Reserved:** 00h
	17:16	**Horizontal Motion Vector Precision:** 00 = 1/2 pixel 01 = 1/4 pixel (used for 2:1 down sampling) 10 = 1/8 pixel (used for 4:1 down sampling) 11 = Reserved
	15:14	**Vertical Motion Vector Precision:** 00 = 1/2 pixel 01 = 1/4 pixel (used for 2:1 down sampling) 10 = 1/8 pixel (used for 4:1 down sampling) 11 = Reserved
	13:12	**Prediction Type:** 00 = Intra-coded. (The motion vector fields are ignored. The Intra-coded values are each 8 bits wide with no padding.) 01 = Forward prediction (Only the forward motion vector is used. The correction data are signed 16 bit values) 10 = Backward prediction (Only the backward motion vector is used. The correction data are signed 16 bit values) 11 = Bi-directional Prediction (Both the forward and backward motion vectors are used. The correction data are signed 16 bit values)
	11:8	**Reserved:** 00h
	7:6	**Destination Picture Structure:** Specifies the structure of the destination surface for predicting this block. The overall structure of the picture may be different, as in field/frame prediction. This field overrides the vertical line stride and offset specified in the DEST_BUFFER_VARIABLES instruction. 00 = Frame (Vertical stride = 1 line) 01 = Reserved 10 = Top Field (Vertical stride = 2 lines) 11 = Bottom Field (Vertical stride = 2 lines)
	5	**Reserved:** 00h
	4:3	**Forward Reference Picture Structure:** This field overrides the vertical line stride and offset specified in the MAP_INFO instruction. 00 = Frame (Vertical stride = 1 line) 01 = Reserved 10 = Top Field (Vertical stride = 2 lines, offset = 0 lines) 11 = Bottom Field (Vertical stride = 2 lines, offset = 1 line)
	2	**Reserved:** 00h
1	1:0	**Backward Reference Picture Structure:** This field overrides the vertical line stride and offset specified in the MAP_INFO instruction. 00 = Frame (Vertical stride = 1 line) 01 = Reserved 10 = Top Field (Vertical stride = 2 lines, offset = 0 lines) 11 = Bottom Field (Vertical stride = 2 lines, offset = 1 line)

DWord	Bits	Description
2	31:26	**Reserved:** 00h
	25:16	**Horizontal Origin:** An unsigned integer specifying both the upper-left pixel of the destination block and the origin of the motion vectors in the reference frame(s). This value must be a multiple of the width. The valid range is 0 – 1023.
	15:10	**Reserved:** 00h
	9:0	**Vertical Origin:** An unsigned integer specifying both the upper-left pixel of the destination block and the origin of the motion vectors in the reference frame(s). This value must be a multiple of the height. The valid range is 0 – 1023.
3	31:26	**Reserved:** 00h
	25:16	**Height:** An unsigned integer specifying the height of the destination block and the source block(s), if required. The valid values for this field are dependant on the Block Pattern Format as follows: Disabled — Valid range: 1 – 1023 (used for skipped macroblocks) Single — Valid values are 2,4,8 Halves — Valid values are 2,4,8 Quadrants — Valid values are 4,8,16
	15:10	**Reserved:** 00h
	9:0	**Width:** An unsigned integer specifying the width of the destination block and the source block(s), if required. The valid values for this field are dependant on the Block Pattern Format as follows: Disabled — Valid range: 1 – 1023 (used for skipped macroblocks) Single — Valid values are 2,4,8 Halves — Valid values are 4,8,16 Quadrants — Valid values are 4,8,16
4	31:0	**Forward Motion Vector** (see the Motion Vector Format below)
5	31:0	**Backward Motion Vector** (see the Motion Vector Format below)
6 – …	N/A	**Block Intra-coded/Correction Data:** Each double word (32 bits) contains packed 8-bit data or 16-bit data depending on the Prediction Type: Intra-coded Block — Four 8-bit values packed in 32 bits. The least significant byte contains data for the left most pixel, spatially. Predicted Block — Two 16-bit values packed in 32 bits. The least significant word contains data for the left most pixel, spatially.
	N/A	…

13.3.1. Motion Vector Format

The motion vectors provided in the GFXBLOCK instruction have the following format.

DWord	Bits	Description
0	31:16	**Horizontal Motion Vector Value:** The value is signed 2's complement fixed point with the following formats, depending on the Motion Vector Precision bits. The range defines the clamp boundaries for the values. **Precision** / **Format** / **Range** $^1/_2$ pixel / S14.1 / [-1024.0–1023.5] $^1/_4$ pixel / S13.2 / [-1024.0–1023.75] $^1/_8$ pixel / S12.3 / [-1024.0–1023.875]
	15:0	**Vertical Motion Vector Value:** The value is signed 2's complement fixed point with the following formats, depending on the Motion Vector Precision bits. The range defines the clamp boundaries for the values. **Precision** / **Format** / **Range** $^1/_2$ pixel / S14.1 / [-1024.0–1023.5] $^1/_4$ pixel / S13.2 / [-1024.0–1023.75] $^1/_8$ pixel / S12.3 / [-1024.0–1023.875]

13.4. Non-pipelined State Variables

The GMCH does not optimize on state variables that change infrequently. ISVs are asked to group polygons for state changes for performance reasons. In the GMCH, the following state variables will not be pipelined:

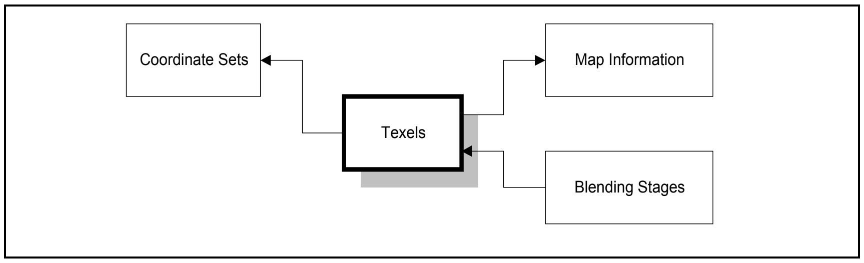
Z_BIAS[7:0], FOG_CLR[23:0], ALPHA_REF[7:0], ALPHA_FUNC[2:0], KILL_PIXEL, COLOR_KEYH[15:0], COLOR_KEYL[15:0], COLOR_INDEX[7:0], DEST_BUFFER_VARIABLES, DRAWING_RECTANGLE_INFO, SCISSOR_RECTANGLE_INFO.

These have been grouped appropriately into packets that tell the instruction parser that these are non-pipelined state variables. The 3D pipeline will be flushed up to and including the Color Calculator stage before these state variables are updated. The pixel cache or the streamers do not need to be flushed for state change. It is important for the application to group these changes into one pipeline to minimize performance impact. This optimization has no software impact other than a slight performance impact.

13.5. GFXRENDERSTATE_MAP_TEXELS

The Mapping Engine is capable of generating at most two texels per pixel. The texels may be obtained from two separate maps or the same map using different u,v coordinates. The binding between the texels which are generated by the engine and the coordinate set and the map information state is specified with this instruction. The texels are generated in the order specified by the Texel Index. If no texel is enabled, texture mapping is disabled and this is the default condition. If one of the two texels is enabled, one texture coordinate will be used for texture mapping. If both texels are enabled, both texture coordinates will be used for texture mapping.

Figure 31. State Variable Relationships



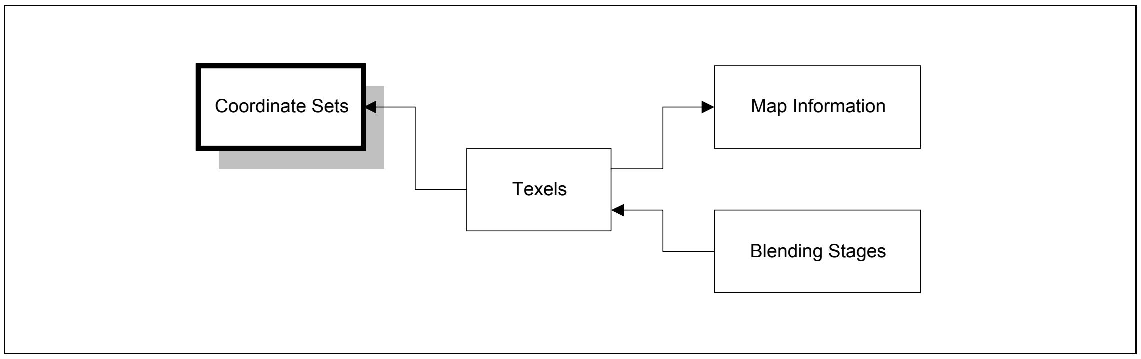
DWord	Bit	Description
0	31:29	**Client :** 03h – Rendering Engine
	28:24	**3DState16 :** 1Ch
	23:19	**Opcode :** 0h
	18:16	**Reserved:** 00h (Additional Texels)
	15	**Texel 1 State Variable Mask :** 0 = Do not update; 1 = Update
	14	**Texel 1 Enable :** 0 = Disable (default) 1 = Enable
	13:12	**Reserved:** 00h (Additional Coordinate Sets)
	11	**Texel 1 Coordinate Set Index :** Index to the Coordinate Set.
	10:9	**Reserved:** 00h (Addition Map Information)
	8	**Texel 1 Map Information Index :** Index to the Map Information.
	7	**Texel 0 State Variable Mask :** 0 = Do not update; 1 = Update
	6	**Texel 0 Enable :** 0 = Disable (default) 1 = Enable
	5:4	**Reserved:** 00h (Additional Coordinate Sets)
	3	**Texel 0 Coordinate Set Index :** Index to the Coordinate Set.
	2:1	**Reserved:** 00h (Addition Map Information)
0	0	**Texel 0 Map Information Index :** Index to the Map Information.

13.6. GFXRENDERSTATE_MAP_COORD_SETS

The Mapping Engine is capable of generating at most two map coordinate sets (u and v addresses) per pixel. Each output texel may be related to separate coordinate sets or to the same coordinate set, as shown below. The vertices of a GFXPRIMITIVE instruction may have 0, 1 or 2 map coordinate sets assigned with varying settings associated with each coordinate set.

A coordinate set may have normalized u/v coordinates or un-normalized coordinates. Normalized coordinates have been divided by the size of the map prior to delivery to the GC. This means that map addresses, which lie on the map, have a range of 0.0 to 1.0. 3D rendering utilized normalized coordinates. Un-normalized coordinates are in units of texels/pixels and have not been divided by the associated map's height or width. Un-normalized coordinates allow the Motion Compensation and Arithmetic Stretch Blitter operations to be specified in pixels avoiding floating-point issues. Un-normalized coordinates are not usually specified outside of the map extents. These two coordinate sets may also have different wrap, clamp, and mirror settings.

Figure 32. State Variable Relationships



DWord	Bit	Description
0	31:29	**Client :** 03h – Rendering Engine
	28:24	**3DState16 :** 1Ch
	23:19	**Opcode :** 1h
	18:17	**Reserved:** 00h (Additional Coordinate Sets)
	16	**Update Coordinate Set Index :** The valid range is 0–1.
	15	**Normalized Coordinate Set Mask :** 0 = Do not update; 1 = Update
	14	**Normalized Coordinate Set :** 0 = Coordinates are not normalized. 1 = Coordinates are normalized.
	13:12	**Reserved:** 00h
	11:8	**Reserved:** 00h (3 Dimensional Coordinates)
	7	**Address V State Variable Mask :** 0 = Do not update; 1 = Update
	6	**Reserved:** 00h
	5:4	**Address V :** Valid values are : 0h = Wrap 1h = Mirror 2h = Clamp 3h = Wrap Shortest
	3	**Address U State Variable Mask :** 0 = Do not update; 1 = Update
	2	**Reserved:** 00h
	1:0	**Address U :** Valid values are : 0h = Wrap 1h = Mirror 2h = Clamp 3h = Wrap Shortest

13.7. GFXRENDERSTATE_MAP_INFO

The Mapping Engine is capable of fetching texels from at most two maps per pixel. This instruction specifies the attributes relating to the location and format of the map. Two different texels can be fetched from the same map.

Figure 33. State Variable Relationships

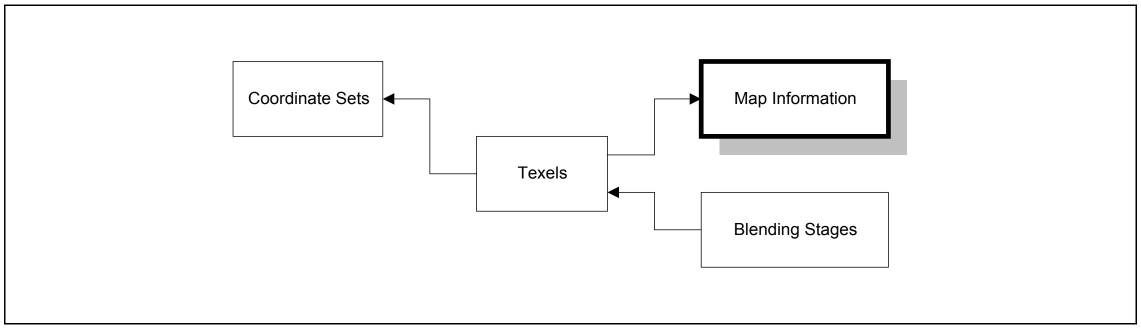


Table 14 identifies four classes of surface formats for the supported maps in the GC. The various attributes of the texels/pixels are placed on 4 abstract channels named F_0–F_3. These channels can be switched into 4 output channels, which are delivered by the Mapping Engine to the Color Calculator. The switching of these channels (ME_0 – ME_3) is controlled by the Output Channel Selection field described below.

Table 14. Summary of Source Surface Formats with Filter Output Channel Mappings

#	Source Format	Bpt	Attribute Types and Formats	Filter Channels			
				F_0	F_1	F_2	F_3
0	Arbitrary Attribute	8	Alpha, Intensity, Luminance, Chromanance, etc. (I: 8)	I	I	I	I
1	Alpha Attribute	16	Alpha and Luminance, Chromanance, etc. (AY: 88)	Y	Y	Y	A
2	Alpha Red Grn Blue	16	ARGB: 0565/1555/4444	R	G	B	A
3	4:2:2		YCrCb 4:2:2	Cr	Y	Cb	1's

Certain memory tiling properties are supported directly by this instruction. This functionality is provided primarily for discrete graphics devices and for surfaces that do not need transparent processor access (such as optimized texture maps). Bits [31:26] of the Base Address are used to identify memory access from discrete graphics devices, which are intended to bypass the Graphics Translation Table (GTT). The following table identifies which fields are used (✔) or ignored (⊗) for various permutations. The first two columns identify which cases (either discrete or integrated) graphics devices use. The Pitch field must be specified in all cases.

Discrete Device	Integrated Device	Base Address Bits [31:26]	Utilize Fence Registers	Fence Range Hit	Tiled Surface	Tile Walk	Surface
✔	✔	All Zeros	Yes	No	⊗	⊗	Linear
✔	✔	All Zeros	Yes	Yes	⊗	⊗	Tiled*
	✔	All Zeros	No	⊗	No	⊗	Linear
	✔	All Zeros	No	⊗	Yes	✔	Tiled
✔		Any Bit Set	⊗	⊗	No	⊗	Linear
✔		Any Bit Set	⊗	⊗	Yes	✔	Tiled

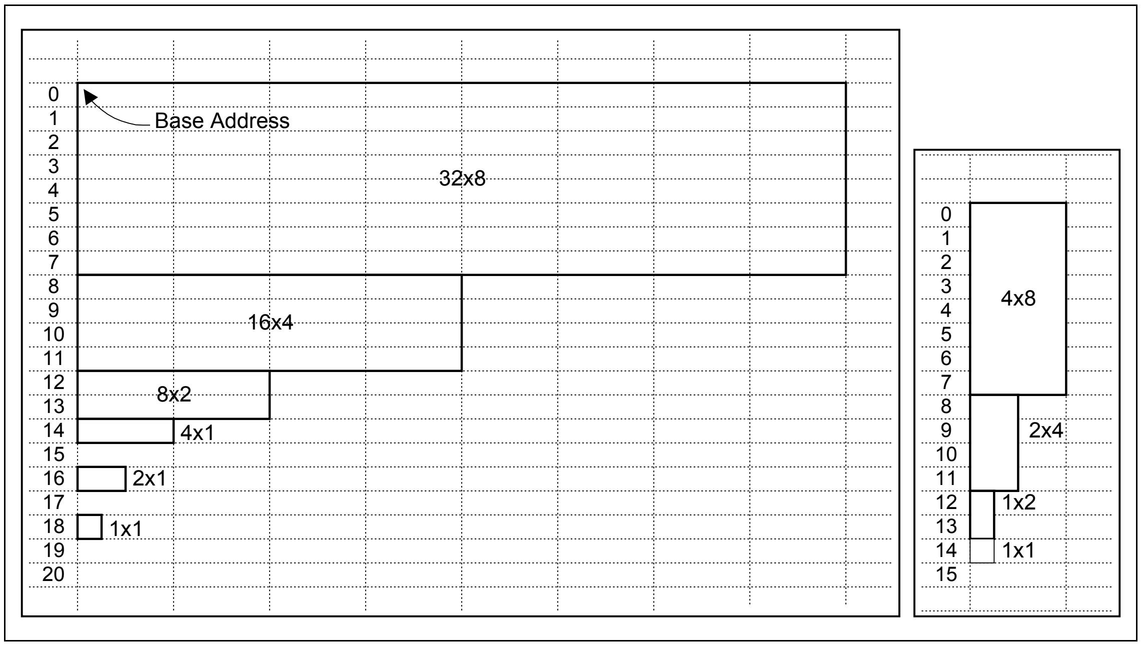
* The pitch specified in this instruction must be the same as the pitch in the corresponding fence register

Surfaces that contain mip-maps are located within a single rectangular area of memory identified by the base address of the upper left corner and a pitch. The pitch must be specified at least as large as the next power of two, equal to or greater than the widest mip-map. These surfaces may be overlapped in memory and must adhere to the following memory organization rules:

- The Base Address must be 4 KB aligned.
- Each successively smaller mip-map must lie vertically below and left aligned.
- Each mip-map must have its upper left corner vertically aligned to an even quadword address.

The following figures show an example of a 32x8 @ 16 bpt and a 4x8 @ 16 bpt map where the dashed lines identify quadwords.

Figure 34 Mip-map Surface Organization Example



DWord	Bit	Description
0	31:29	**Client** : 03h – Rendering Engine
	28:24	**3DstateMW** : 1Dh
	23:16	**Opcode** : 0h
	15:0	**DWORD_LENGTH** : 2h
1	31:29	**Reserved:** 00h
	28	**Update Map Index** : The valid range is 0–1.
	27	**Reserved:** 0h
	26:24	**Surface Format:** 0h = 8 bpt (Indexed) 1h = 8 bpt 2h = 16 bpt 3h–4h = Reserved 5h = 4:2:2 6h–7h = Reserved
1	23	**Reserved:** 0h
1	22:21	**Texel/Pixel Format:** 8 bpt (Indexed) Surface Format 00 = RGB 565 01 = ARGB 1555 10 = ARGB 4444 11 = AY 88 8 bpt Surface Format 00 = Reserved 01 = Reserved 10 = Reserved 11 = Reserved 16 bpt Surface Format 00 = RGB 565 01 = ARGB 1555 10 = ARGB 4444 11 = AY 88 4:2:2 00 = YCrCb, Swap Y format 01 = YCrCb, Normal 10 = YCrCb, UV Swap 11 = YCrCb, UV/Y Swap
1	20:19	**Output Channel Selection:** Selects the muxing for the possible 4 channels available in the surface (at the output of the filter) and the 4 output channels from the Mapping Engine. Mapping Engine Output Channels **ME_0 ME_1 ME_2 ME_3** 0h = F_0 F_1 F_2 F_3 1h = ⊗ F_0 ⊗ F_3 2h = ⊗ F_2 ⊗ F_3 3h = Reserved

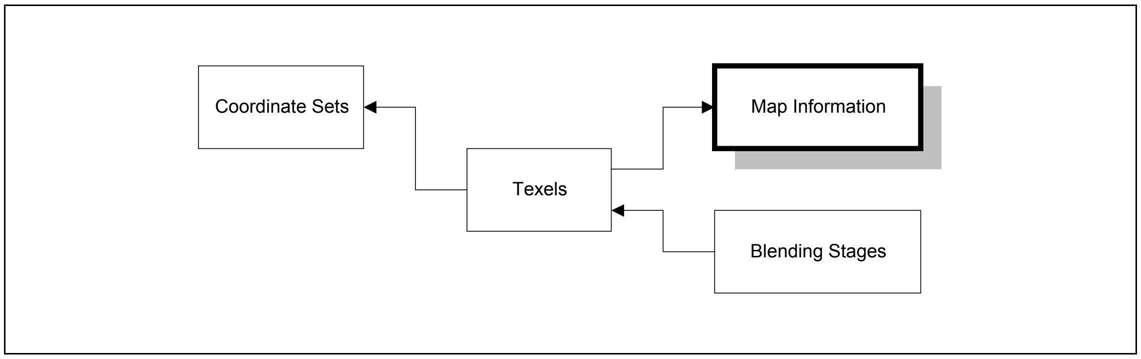
DWord	Bit	Description
1	18	**Color Space Conversion Enable:** 0 = Do not perform conversion. 1 = Perform color space conversion assuming biased chromanance values.
	17	**Vertical Line Stride:** The number of lines to skip between logically adjacent lines. The Forward/Backward Reference Picture Structure bits override this value when processing the GFXBLOCK instruction. 0 = Do not skip any lines. 1 = Skip 1 line. (provides support of Interleaved/field surfaces)
	16	**Vertical Line Stride Offset:** The number of lines to add as an initial offset when the Vertical Line Stride is 1. This value is overridden by the Forward/Backward Reference Picture Structure bits when processing the GFXBLOCK instruction. 0 = Add no offset. (top field) 1 = Add 1. (bottom field)
	15:11	**Reserved:** 00h
	10	**Utilize Fence Registers:** If enabled the Tiled Surface, Tile Height, Tile Walk and Pitch are ignored. All request addresses are compared with the fence registers and the parameters associated with any matching fence register are utilized in the tiler. Otherwise, the specified Tile Walk is utilized. 0 = Disable Fence Register Utilization 1 = Enable Fence Register Utilization
	9	**Tiled Surface:** Specifies the whether the surface is organized as rectangular memory or as tiled memory. This field is ignored when the fence registers are being utilized. 0 = Linear (Rectangular memory) 1 = Tiled
	8	**Tile Walk:** The direction of increasing sequential addresses. This field is ignored when the fence registers are being utilized or the surface is linear. 0 = X-Major 1 = Y-Major
	7:5	**Reserved:** 00h (For Tile Height)
	4	**Reserved:** 00h
1	3:0	**Pitch:** Log$_2$ of the surface pitch is specified in quadwords. If the surface resides in a fenced region, this value must correspond to the pitch specified when programming the fence registers.

DWord	Bit	Description
2	31	**Dimensions are Powers of 2:** This field specifies whether the following **Height** and **Width** fields of the map are specified as the $\log_2$ of the actual dimension or as the actual height or width. If the actual values are used, the coordinate set addresses must be non-normalized and are clamped (they can not be wrapped or mirrored). 0 = Height/Width are the actual dimensions of the map in pixels 1 = Height/Width are $\log_2$ of the actual dimensions of the base map
	30:26	**Reserved:** 00h (Additional Height)
	25:16	**Height:** If the dimensions are to be specified as powers of two (as determined by the Power of 2 field), this field must contain one of the following values: 0h = 1 texel high, 4h = 16 texels high, 8h = 256 texels high 1h = 2 texels high, 5h = 32 texels high, 9h = 512 texels high 2h = 4 texels high, 6h = 64 texels high, ah = 1024 texels high 3h = 8 texels high, 7h = 128 texels high If the dimensions are to be specified as the actual values, this field contains the surface <u>height -1</u> in units of texels. The range for this value is 0–1023.
	15:10	**Reserved:** 00h
	9:0	**Width:** If the dimensions are to be specified as powers of two (as determined by the Dimension Power of 2 field), this field must contain one of the following values: 0h = 1 texel wide, 4h = 16 texels wide, 8h = 256 texels wide 1h = 2 texels wide, 5h = 32 texels wide, 9h = 512 texels wide 2h = 4 texels wide, 6h = 64 texels wide, Ah = 1024 texels wide 3h = 8 texels wide, 7h = 128 texels wide If the dimensions are to be specified as the actual values, this field contains the surface <u>width -1</u> in units of texels. The valid range for this value is 0–1023 for all surface formats except 4:2:2. For 4:2:2 surfaces, this value may be even 8–1022.
3	31:4	**Map Base Address:** Virtual memory address, which is aligned on 16 byte boundaries. Bits [31:4] of a 32-bit byte address are specified.
	3:0	**Reserved:** 0h

13.8. GFXRENDERSTATE_MAP_FILTER

The Mapping Engine is capable of fetching texels/pixels from at most two maps per pixel. This instruction specifies the filter settings associated with specified map.

Figure 35. State Variable Relationships



Anisotropic filtering produces superior image quality with reduced performance. Adjusting the LOD bias can contain the aspect ratio of the filter. Generally the LOD bias should be set to -1.0 when anisotropic filtering is enabled. The Mip Mode, Min Mode and Mag Mode Filter state variables continue to have their meanings when anisotropic filtering is enabled.

The Mip Mode Filter state variable specifies whether mip-mapping is enabled. If so, it also identifies if the nearest mip map should be used for the texture data or texels between the two nearest maps should be dithered together when transitioning between maps. LOD dithering provides the ability to offset the map LOD based on the pixel x,y location. A 4x4 x,y matrix provides values which range from 0 to 15, which are scaled and then added to the LOD computation.

The Min Mode Filter state variable identifies the filtering operation to utilize when the map is minimized and a texel is smaller than a pixel. When this occurs the either Nearest or Linear filtering is performed. In Nearest filtering, the texel with coordinates nearest to the desired pixel value is used. In Linear filtering, a weighted average of a 2-by-2 area of texels surrounding the desired pixel is used.

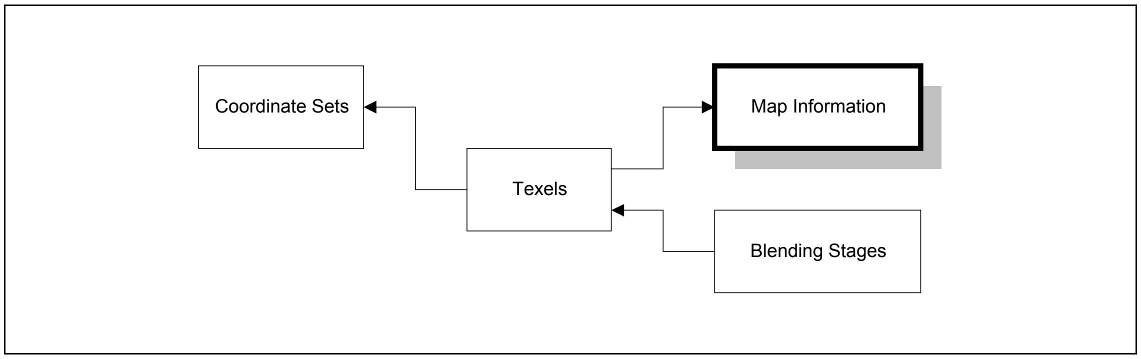
The Mag Mode s Filter state variable describes operations where the map is magnified and a texel is larger than one pixel. In this mode, the finest texture map (LOD 0) is addressed, and the state variable selects between Nearest and Linear filtering. In Nearest filtering, the texel with coordinates nearest to the desired pixel value is used. In Linear filtering, a weighted average of a 2-by-2 area of texels surrounding the desired pixel is used.

DWord	Bit	Description
0	31:29	**Client** : 03h – Rendering Engine
	28:24	**3DState16** : 1Ch
	23:19	**Opcode** : 2h
	18:17	**Reserved:** 00h (Additional Maps)
	16	**Update Map Index** : The valid range is 0–1.
	15:13	**Reserved:** 00h
	12	**Anisotropic Filtering Enabled Mask** : 0 = Do not update; 1 = Update
	11	**Reserved:** 00h
	10	**Anisotropic Filtering Enabled:** 0 = Disable Anisotropic Filtering 1 = Enable Anisotropic Filtering
	9	**Mip Mode Filter Mask** : 0 = Do not update; 1 = Update
	8	**Reserved:** 00h
	7:6	**Mip Mode Filter :** Valid values are : 00 = None, Disable mip mapping 01 = Nearest, Select the nearest mip map 02 = Dithered, Dither between nearest mip maps 03 = Linear, interpolate between nearest mip maps (combined with linear min/mag filters this is Trilinear filtering).
	5	**Mag Mode Filter Mask** : 0 = Do not update; 1 = Update
	4	**Reserved:** 00h
	3	**Mag Mode Filter :** Valid values are : 0 = Nearest 1 = Linear
	2	**Min Mode Filter Mask** : 0 = Do not update; 1 = Update
	1	**Reserved:** 00h
0	0	**Min Mode Filter :** Valid values are : 0 = Nearest 1 = Linear

13.9. GFXRENDERSTATE_MAP_LOD_LIMITS

The limits of the Level of Detail calculation can be controlled within GC for each Map. These values are specified in the following instruction as follows.

Figure 36. State Variable Relationships

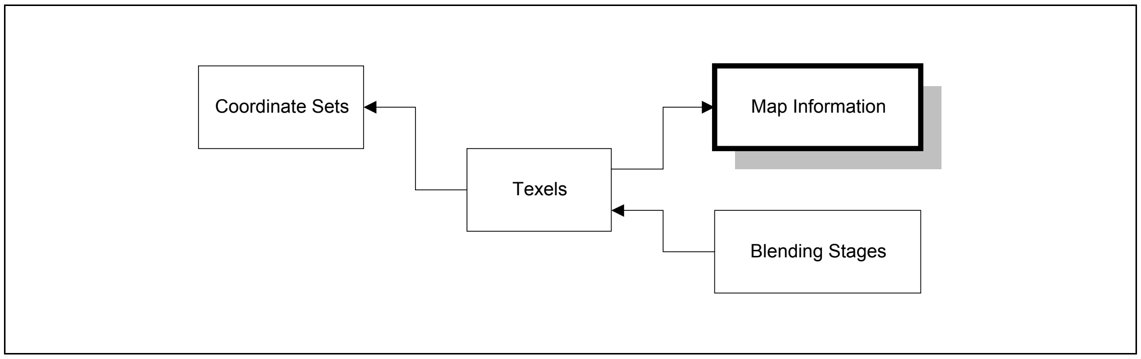


DWord	Bit	Description
0	31:29	**Client :** 03h – Rendering Engine
	28:24	**3DState16 :** 1Ch
	23:19	**Opcode :** 3h
	18:17	**Reserved:** 00h (Additional Maps)
	16	**Update Map Index :** The valid range is 0–1.
	15:14	**Reserved:** 00h
	13	**Maximum Mip Level Mask :** 0 = Do not update; 1 = Update
	12:5	**Maximum Mip Level :** This is an unsigned 4.4 bit value which defines the highest resolution map which may be accessed. A value of 0 equates to no limit on the mip map selection. Values 1 through 10 will force the mip map selection to be between the specified value and the highest defined mip map (the lowest resolution map). This value must be less than the Minimum Mip Level.
	4	**Minimum Mip Level Mask :** 0 = Do not update; 1 = Update
	3:0	**Minimum Mip Level :** This is an unsigned 4 bit value which defines the lowest resolution map which may be accessed. Values 0 through 10 will force the mip map selection to be between the maximum resolution mip map and the specified value. This value must be correctly identified for the current map (no assumptions are made about the number of mips associated with a particular map). This value must also be greater than the Maximum Mip Level.

13.10. GFXRENDERSTATE_MAP_LOD_CONTROL

The Level-of-Detail dither weight and bias can be associated with each map using the following instruction.

Figure 37. State Variable Relationships

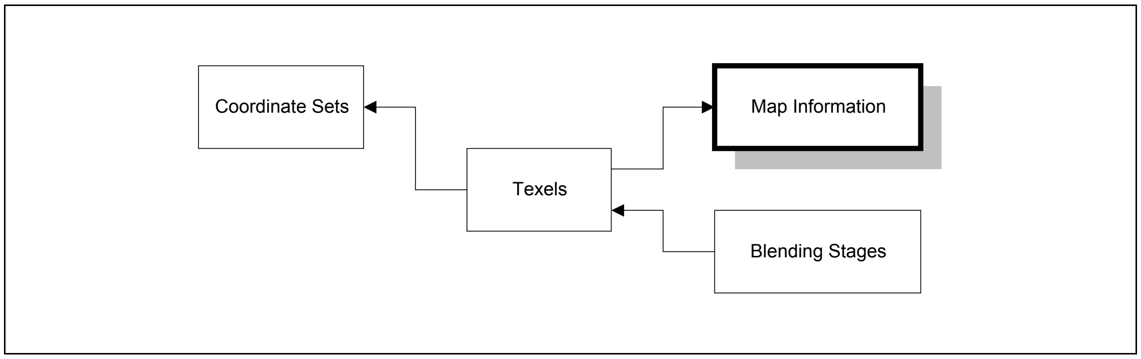


Dword	Bit	Description
0	31:29	**Client :** 03h – Rendering Engine
	28:24	**3DState16 :** 1Ch
	23:19	**Opcode :** 4h
	18:17	**Reserved:** 00h (Additional Maps)
	16	**Update Map Index :** The valid range is 0–1.
	15:11	**Reserved:** 00h
	10	**Texture LOD Dither Weight Mask :** 0 = Do not update; 1 = Update
	9:8	**Texture LOD Dither Weight:** A 4-bit dither value can be added to the computed and biased LOD which softens the transitions between mip-maps. The position or weight of the dither value is specified as follows: 00 = Full dither weight (the dither value has a range of 0.0 – 0.9375) 01 = ½ dither weight (the dither value has a range of 0.0 – 0.4375) 10 = ¼ dither weight (the dither value has a range of 0.0 – 0.1875) 11 = $^1/_8$ dither weight (the dither value has a range of 0.0 – 0.0625)
	7	**Texture LOD Bias Mask :** 0 = Do not update; 1 = Update
	6:0	**Texture LOD Bias :** This is a signed value that is added to the LOD when the LOD is determined at a textured pixel. This 2's complement fixed point value has 2 integer bits and 4 fractional bits and is signed extended before being added to the LOD. Valid data value range from -4.0 to 3.9375. (S2.4) The default value is 0.

13.11. GFXRENDERSTATE_MAP_PALETTE_LOAD

The Texture Palette is loaded using the following instruction. All 256 entries of the texture palette must be loaded every time this command packet is sent. In other words, even if only one entry of the texture palette needs to be updated, all 256 entries must be loaded and sent using this command packet.

Figure 38. State Variable Relationships

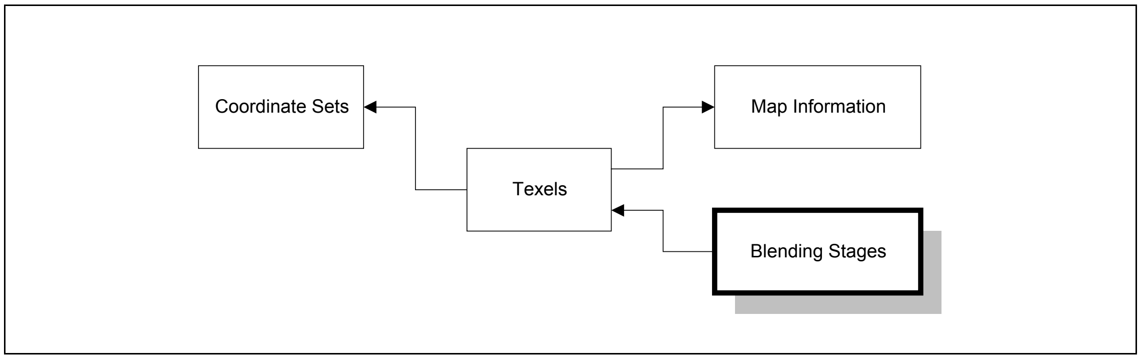


DWord	Bit	Description
0	31:29	**Client :** 03h – Rendering Engine
	28:24	**3DStateMWNP(Non-pipelined) :** 1Dh
	23:16	**Opcode :** 82h
	15:0	**DWORD_LENGTH :** 255
1	31:16	**Reserved:** 00h
	15:0	**16-bit Color[0]: Format:** RGB 565/1555/4444, YCrCb 565 or AI 88
2–255		…
256	31:16	**Reserved:** 00h
	15:0	**16-bit Color[255]:** Format: RGB 565/1555/4444, YCrCb 565 or AI 88

13.12. GFXRENDERSTATE_MAP_COLOR_BLEND_STAGES

The Rendering Engine supports three map color blend stages for the red, green, and blue channels. Any of these stages may perform an operation utilizing up to two texels from the Mapping Engine, the iterated face color, the iterated alpha and/or a constant color.

Figure 39. State Variable Relationships



The following equations are supported by each blending stage.

Blend Equation	Description
$Color_{out}$ = Arg1	Select Arg1
$Color_{out}$ = Arg2	Select Arg2
$Color_{out}$ = Arg1 * Arg2	Modulate
$Color_{out}$ = Arg1 * Arg2 * 2	Modulate and Multiply by 2
$Color_{out}$ = Arg1 * Arg2 * 4	Modulate and Multiply by 4
$Color_{out}$ = Arg1 + Arg2	Add
$Color_{out}$ = Arg1 - Arg2	Subtract
$Color_{out}$ = Arg1 + Arg2 - 0.5	Add Signed (Excess 128)
$Color_{out} = \alpha_{Iterated}$ * Arg1 + (1 - $\alpha_{Iterated}$) * Arg2	Linearly Blend using Iterated Alpha
$Color_{out} = \alpha_{Factor}$ * Arg1 + (1 - α_{Factor}) * Arg2	Linearly Blend using Alpha Factor
$Color_{out} = \alpha_{Texel0}$ * Arg1 + (1 - α_{Texel0}) * Arg2	Linearly Blend using Texel0's Alpha
$Color_{out} = \alpha_{Texel1}$ * Arg1 + (1 - α_{Texel1}) * Arg2	Linearly Blend using Texel1's Alpha
$Color_{out} = Color_{Texel0}$ * Arg1 + (1 - $Color_{Texel0}$) * Arg2	Linearly Blend using Texel0's Color
$Color_{out} = Color_{Texel1}$ * Arg1 + (1 - $Color_{Texel1}$) * Arg2	Linearly Blend using Texel1's Color

The settings for these stages are specified in the following instruction. When enable a stage for color operation, the corresponding alpha stage will also be enabled. The arguments and operation fields of the corresponding alpha stage must be programmed accordingly.

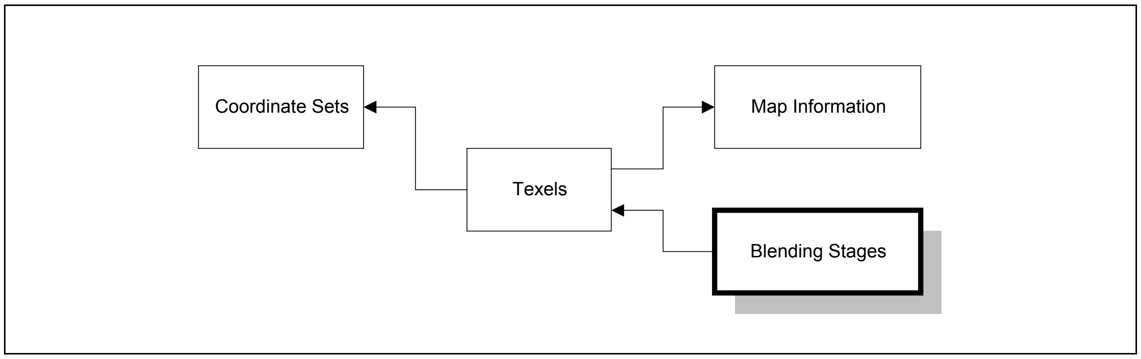
Dword	Bit	Description
0	31:29	**Client** : 03h – Rendering Engine
	28:24	**3DState24** : 00h
	23:22	**Reserved:** 00h (Additional Blending Stages)
	21:20	**Update Blending Stage Index** : The valid range is 0–2.
	19	**Current/Accumulator Select Mask** : 0 = Do not update; 1 = Update
	18	**Write result to Current Register or Accumulator Select :** 0 = Current Register (default) 1= Accumulator
	17	**Color Arg1 Mask** : 0 = Do not update; 1 = Update
	16:14	**Color Arg1:** Valid values are : 0h = All Ones 1h = Color Factor 2h = Accumulator Color (illegal setting for stage 0) 3h = Iterated Color 4h = Specular Color 5h = Current Color 6h = Texel0 Color 7h = Texel1 Color
	13	**Replicate Arg1 Alpha to Color Channels:** 0 = Do not Replicate Alpha to Color Channels 1 = Replicate Alpha to Color Channels
	12	**Invert Color Arg1:** 0 = Do not Invert Argument 1 = Invert Argument
	11	**Color Arg2 Mask** : 0 = Do not update; 1 = Update
	10:8	**Color Arg2 :** Valid values are : 0h = All Ones 1h = Color Factor 2h = Accumulator Color (illegal setting for stage 0) 3h = Iterated Color 4h = Specular Color 5h = Current Color 6h = Texel0 Color 7h = Texel1 Color
0	7	**Replicate Arg2 Alpha to Color Channels:** 0 = Do not Replicate Alpha to Color Channels 1 = Replicate Alpha to Color Channels

Dword	Bit	Description
0	6	**Invert Color Arg2:** 0 = Do not Invert Argument 1 = Invert Argument
	5	**Color Operation Mask** : 0 = Do not update; 1 = Update
	4:0	**Color Operation:** Valid values are : 00h = The user must explicitly disable each blend stage that is not used. Note: The user must ensure that if a blend stage is disabled, all higher number blend stages must be disabled too. E.g. It is invalid to disable stage 2 and use only stages 1 and 3. (This bit disables/enables both the color and the corresponding alpha stage) 01h = Select Arg1 02h = Select Arg2 03h = Modulate 04h = Modulate and Multiply by 2 05h = Modulate and Multiply by 4 06h = Add 07h = Add Signed 08h = Linearly Blend using the Iterated Alpha as the Blend Term 09h = Reserved 0Ah = Linearly Blend using the Alpha Factor as the Blend Term 0Bh = Reserved (Linearly Blend when a pre-multiplied map alpha is present) 0Ch = Reserved (Pre-modulate) 0Ch = Reserved (Bump/Environment Map) 0Dh = Reserved (Bump pre-modulate with luminance channel) 0Fh = Reserved 10h = Linearly Blend using Texel0's Alpha as the Blend Term 11h = Linearly Blend using Texel1's Alpha as the Blend Term 12h = Linearly Blend using Texel0's Color as the Blend Term 13h = Linearly Blend using Texel1's Color as the Blend Term 14h = Subtract (Arg1 - Arg2) 15h=1Fh – Reserved

13.13. GFXRENDERSTATE_MAP_ALPHA_BLEND_STAGES

The Rendering Engine supports three map color blend stages for the alpha channel. Any of these stages may perform an operation utilizing the alpha channel from up to two texels from the Mapping Engine, the iterated alpha and/or a constant color.

Figure 40. State Variable Relationships



The following equations are supported by each blending stage.

Blend Equation	Description
$\text{Alpha}_{\text{out}} = \text{Arg1}$	Select Arg1
$\text{Alpha}_{\text{out}} = \text{Arg2}$	Select Arg2
$\text{Alpha}_{\text{out}} = \text{Arg1} * \text{Arg2}$	Modulate
$\text{Alpha}_{\text{out}} = \text{Arg1} * \text{Arg2} * 2$	Modulate and Multiply by 2
$\text{Alpha}_{\text{out}} = \text{Arg1} * \text{Arg2} * 4$	Modulate and Multiply by 4
$\text{Alpha}_{\text{out}} = \text{Arg1} + \text{Arg2}$	Add
$\text{Alpha}_{\text{out}} = \text{Arg1} + \text{Arg2} - 0.5$	Add Signed (Excess 128)
$\text{Alpha}_{\text{out}} = \alpha_{\text{Iterated}} * \text{Arg1} + (1 - \alpha_{\text{Iterated}}) * \text{Arg2}$	Linearly Blend using Iterated Alpha
$\text{Alpha}_{\text{out}} = \alpha_{\text{Factor}} * \text{Arg1} + (1 - \alpha_{\text{Factor}}) * \text{Arg2}$	Linearly Blend using Alpha Factor
$\text{Alpha}_{\text{out}} = \alpha_{\text{Texel0}} * \text{Arg1} + (1 - \alpha_{\text{Texel0}}) * \text{Arg2}$	Linearly Blend using Texel0's Alpha
$\text{Alpha}_{\text{out}} = \alpha_{\text{Texel1}} * \text{Arg1} + (1 - \alpha_{\text{Texel1}}) * \text{Arg2}$	Linearly Blend using Texel1's Alpha

The settings for these stages are specified in the following instruction.

DWord	Bit	Description
0	31:29	**Client** : 03h – Rendering Engine
	28:24	**3DState24** : 01h
	23:22	**Reserved:** 00h (Additional Blending Stages)
	21:20	**Update Blending Stage Index** : The valid range is 0–2.
	19	**Reserved:** 00h
	18	**Alpha Arg1 Mask** : 0 = Do not update; 1 = Update
	17:15	**Alpha Arg1:** Valid values are : 0h = Reserved 1h = Alpha Factor 2h = Reserved 3h = Iterated Alpha 4h = Reserved 5h = Current Alpha 6h = Texel0 Alpha 7h = Texel1 Alpha
	14	**Reserved:** 00h
	13	**Invert Alpha Arg1:** 0 = Do not Invert Argument 1 = Invert Argument
	12	**Alpha Arg2 Mask** : 0 = Do not update; 1 = Update
	11	**Reserved:** 00h
	10:8	**Alpha Arg2 :** Valid values are : 0h = Reserved 1h = Alpha Factor 2h = Reserved 3h = Iterated Alpha 4h = Reserved 5h = Current Alpha 6h = Texel0 Alpha 7h = Texel1 Alpha
	7	**Reserved:** 00h
	6	**Invert Alpha Arg2:** 0 = Do not Invert Argument 1 = Invert Argument
0	5	**Alpha Operation Mask** : 0 = Do not update; 1 = Update

DWord	Bit	Description
0	4:0	**Alpha Operation:** Valid values are : 00h = Reserved (Enable/Disable of stage is programmed in the Color Operation field) 01h = Select Arg1 02h = Select Arg2 03h = Modulate 04h = Modulate and Multiply by 2 05h = Modulate and Multiply by 4 06h = Add 07h = Add Signed 08h = Linearly Blend using the Iterated Alpha as the Blend Term 09h = Reserved 0Ah = Linearly Blend using the Alpha Factor as the Blend Term 0Bh = Reserved (Linearly Blend when a pre-multiplied map alpha is present) 0Ch = Reserved (Pre-modulate) 0Ch = Reserved (Bump/Environment Map) 0Dh = Reserved (Bump pre-modulate with luminance channel) 0Fh = Reserved 10h = Linearly Blend using Texel0’s Alpha as the Blend Term 11h = Linearly Blend using Texel1’s Alpha as the Blend Term 12h–1Fh = Reserved

13.14. GFXRENDERSTATE_COLOR_FACTOR

This instruction specifies a constant color factor, which can be used in the blending stages.

DWord	Bit	Description
0	31:29	**Client :** 03h – Rendering Engine
	28:24	**3DStateMW :** 1Dh
	23:16	**Instruction :** 1h
	15:0	**DWORD_LENGTH :** 0
1	31:0	**Color Factor:** This is a 32 bit value in ARGB 8888 format. The top 8 bits used by the alpha blending stage and the lower 24 bits used by the color blending stage.

13.15. GFXRENDERSTATE_COLOR_CHROMA_KEY

ColorKey and ChromaKey are terms used to describe two methods of removing a specific color or range of colors from a map that is applied to a primitive.

When a color palette is used with indices to indicate a color in the palette, the indices can be compared against a state variable "**ColorKey** Index Value" and if a match occurs and ColorKey is enabled, then an action will be taken to remove this value's contribution to the resulting pixel color.

The **ChromaKey** mode refers to testing the RGB components to see if they fall between a high (Chroma_High_Value) and low (Chroma_Low_Value) state variable values. If the color of a texel contribution is in this range and ChromaKey is enabled, then an action will be taken to remove this contribution to the resulting pixel color.

When the color value of the nearest neighbor texel falls within the range of the ChromaKey values or the palletized texel index matches the ColorKey value, the pixel can be programmed by the **Kill Pixel** state variable to not be written back to the frame buffer. If neither "Key" mode applies, then a **blend** occurs as described below.

When multiple texture is enabled, only the color of texel 0 is used for ChromaKey or ColorKey. The color of texel 1 is not subject to texture color/chroma key. Filtering and use of texel 1 (if enabled) proceeds normally irrespective of color/chroma key.

Kill Pixel Mode

This is the default function for texture color/chroma key in D3D*. (Prior to DX7*, it was the *only* function). Pixels whose sampled texels are considered "matching" the color/chroma key are killed (discarded) and not drawn (regardless of any subsequent pipeline operations). The Intel® 810 chipset implementation of the "matching" criteria is somewhat different than the D3D* reference rasterizer behavior introduced in DX6.1. DX6.1 will kill the pixel if any "contributing" sampled texels match the key. (Here "contributing" is defined as having an non-zero filter (beta) weight – as opposed to non-contributing texels that may be sampled/fetched but effectively not used). This behavior tends to enlarge keyed-out areas depending on the size of the filter. The Intel® 810 chipset implementation only kills a pixel if the *nearest neighbor* sampled texel matches the key. Non-nearest neighbor texels matching the key are effectively replaced by the nearest neighbor's color/alpha and then used in the filter. This behavior avoids the enlargement of keyed-out areas.

The Intel® 815 chipset supports both Kill Pixel modes. The default is the DX6.1 implementation.

"Blend" Mode

This mode was intended to support the "BLENDCOLORKEY" D3D* mode (which will be introduced into D3D* in DX7*). Here color/chroma key does not discard pixels, it simply modifies the filtered texel color/alpha in a way which permits keyed-out areas of textures from not contributing to the pixel output. Here the application is responsible for programming the pixel pipeline (texture blend, alpha test, alpha blend, etc.) to process keyed/non-keyed texel/pixel results in the intended fashion (i.e., there is no automatic override of pixel pipeline controls). For example, the application may simply choose to use the filtered texel's alpha value in the Alpha Test stage to discard pixels with zero alpha (presumably introduced by the color/chroma key function) – this would effectively emulate the kill pixel mode. Another scenario might modulate the filtered texel's color by the filtered texel's alpha, and then add the result to the current pixel.

The Intel® 810 chipset implementation of Blend color/chroma key mode is, again, different than what Microsoft eventually defined for DX7*. DX7* specifies that any contributing texels which match the key must be replaced by black with zero alpha prior to application of the normal texture filter. The Intel® 810 chipset implements a different function, where the output texel has it's alpha value forced to zero if the nearest neighbor texel sample matches the key (the **color** associated with this keyed-out texel is UNDEFINED). If only non-nearest neighbor texels are found to match the key, those texels are effectively replaced by the nearest neighbor color/alpha prior to application of the filter.

The Intel® 815 chipset supports both blend modes. The default is the DX7* implementation.

Dword	Bit	Description
0	31:29	**Client :** 03h – Rendering Engine
	28:24	**3DStateMW :** 1Dh
	23:16	**Instruction :** 2h
	15:0	**DWORD_LENGTH :** 1
1	31	**Reserved:** 00h
	30	**KeyedPixelControl Write Mask:** 0 = Do not update; 1 = Update
	29	**KeyedPixelControl:** 0 = OLD Algorithms (Intel® 810 chipset Compatible) If '*killpix*' mode is enabled, then a pixel is killed if and only if the nearest texel is keyed out. If '*killpix*' mode is disabled, then for any pixel that has a keyed out nearest texel, the alpha value forced to 0. The contributed RGBA of any other texels that are keyed out are replaced by the RGBA of the nearest texel and blended as usual. 1 = NEW Algorithms (DX6.1 Kill Pixel, DX7* Blend). Default. If '*killpix*' mode is enabled, then a pixel is killed if any of the texels contributing to its color is keyed out. If '*killpix*' mode is disabled, then for any contributing texel which is keyed out, R,G,B,A are forced to 0. Any such texel will still be blended with the other texels as usual.
	28	**Kill Pixel Write Mask :** 0 = Do not update; 1 = Update
	27	**Kill Pixel Write When Match on Key :** 0 = Do not Kill Pixel; 1 = Kill Pixel Enabling the Kill Pixel Write state prevents the pixel from being written into the destination buffer independent of any alpha test.
	26	**Color Index Value Mask :** 0 = Do not update; 1 = Update
	25	**ChromaKey Low Value Mask :** 0 = Do not update; 1 = Update
	24 23:0	**ChromaKey High Value Mask :** 0 = Do not update; 1 = Update **ChromaKey Low Value:** This is a 24 bit value in RGB 888 format. Only the upper 5/6/5 bits of the RGB 888 color are compared against the texel colors.
2	31:24	**Color Index Value :** This is an 8 bit color index. The default value is 0.
	23:0	**ChromaKey High Value:** This is a 24 bit value in RGB 888 format. Only the upper 5/6/5 bits of the RGB 888 color are compared against the texel colors.

13.16. GFXRENDERSTATE_SRC_DST_BLEND_MONO

The state variable **Specular_RGB_Enable** is set in the graphics command GFXRENDERSTATE_SRC_DST_BLEND_MONO. It is also dependent on the value of the state variable **Specular_Enable,** which is set in the graphics command GFXRENDERSTATE_BOOLEAN_ENA_1.

Table 15. Selecting Specular Mode

		Specular RGB Enable Bits [15:14] of GFXRENDERSTATE_SRC_DST_BLEND_MONO	
		0	**1**
Specular Enable	**0**	No Specular	No Specular
Bits [9:8] of GFXRENDERSTATE_BOOLEAN_ENA_1	**1**	Monochrome Specular	Full Color (RGB) Specular

The GC 3D section of the graphics device supports alpha blend modes defined in the following state variables except where noted as "Not Implemented".

1. RENDERSTATE_SRC_DST_BLEND
2. RENDERSTATE_BLEND_ENABLE
3. RENDERSTATE_BLEND_COLOR (Texture compositing)

RENDERSTATE_BLEND_ENABLE

1. Blending OFF
2. Blending On

Additional Blending Modes

The color calculator super-unit of the graphics device shall support four new Blending operations. The blend factors for the source and destination colors are controlled by separate state variables and shall be defined as shown below. The RGBA values of the source and destination are indicated with the subscripts s and d respectively.

The Zero blend factor is :
  Rout = 0
  Gout = 0
  Bout = 0
  Aout = 0
The One blend factor is :
  Rout = Rin
  Gout = Gin
  Bout = Bin
  Aout = Ain
The Src_Color blend factor is :
  Rout = Rs * Rin
  Gout = Gs * Gin
  Bout = Bs * Bin
  Aout = As * Ain
The Inv_Src_Color blend factor is :
  Rout = (1 - Rs) * Rin
  Gout = (1 - Gs) * Gin
  Bout = (1 - Bs) * Bin
  Aout = (1 - As) * Ain
The Src_Alpha blend factor is :
  Rout = As * Rin
  Gout = As * Gin
  Bout = As * Bin
  Aout = As * Ain
The Inv_Src_Alpha blend factor is :
  Rout = (1 - As) * Rin
  Gout =(1 - As) * Gin
  Bout = (1 - As) * Bin
  Aout = (1 - As) * Ain
The Dst_Color blend factor is :
  Rout = Rd * Rin
  Gout = Gd * Gin
  Bout = Bd * Bin
  Aout = Ad * Ain
The Inv_Dst_Color blend factor is :
  Rout = (1 - Rd) * Rin
  Gout = (1 - Gd) * Gin
  Bout = (1 - Bd) * Bin
  Aout = (1 - Ad) * Ain
The Both_Src_Alpha blend factor is :
  Source :
    Rout = As * Rin
    Gout = As * Gin
    Bout = As * Bin
    Aout = As * Ain
  Destination :
    Rout = (1 - As) * Rin
    Gout = (1 - As) * Gin
    Bout = (1 - As) * Bin
    Aout = (1 - As) * Ain
The Both_Inv_Src_Alpha blend factor is :
  Source :
    Rout = (1 - As) * Rin
    Gout = (1 - As) * Gin
    Bout = (1 - As) * Bin
    Aout = (1 - As) * Ain
  Destination :
    Rout = As * Rin
    Gout = As * Gin
    Bout = As * Bin
    Aout = As * Ain

The blending factors shall be applied to the source and destination RGBA values and computed using one of the blending equations shown below. The equation is selected by a state variable.

Add: Cs * S + Cd * D

Where Cs is the source RGBA, Cd is the destination RGBA, S is the source blending factor, and D is the destination blending factor.

Bit	Description
31:29	**Client :** 03h – Rendering Processor
28:24	**3DState24 :** 08h
23:16	**Reserved :** 00h
15	**Specular RGB State Mask :** 0 = Do Not Update 1 = Update
14	**Specular RGB Enable :** 0 = Disable (default) 1 = Enable Controls specular color interpolation. When enabled, a full RGB specular color is interpolated, otherwise a monochrome specular value is interpolated.
13	**Color Monochrome (Diffuse) Enable State Mask :** 0 = Do Not Update 1 = Update
12	**Color Monochrome (Diffuse) Enable :** 0 = Disable (default) 1 = Enable
11	**Source Blend State Mask :** 1 = Update; 0 = Do Not Update
10:6	**Source Blend State :** (see table below)
5	**Destination Blend State Mask :** 1 = Update; 0 = Do Not Update
4:0	**Destination Blend State :** (see table below)

Opcode	Source / Destination Blend State
	Reserved (default)
01h	Zero
02h	One
03h	Src_Color
04h	Inv_Src_Color
05h	Src_Alpha
06h	Inv_Src_Alpha
07h	Reserved
08h	Reserved
09h	Dst_Color
0ah	Inv_Dst_Color
0bh	Reserved
0ch	Both_Src_Alpha
0dh	Both_Inv_Src_Alpha

In the case of the monochrome state variables, the rasterizer will shade primitives by interpolating the blue color and using the result for the red, green and blue color components for specular and diffuse colors.

13.17. GFXRENDERSTATE_Z_BIAS_ALPHA_FUNC_REF

DWord	Bit	Description
0	31:29	**Client :** 03h – Render Processor
	28:24	**3DState24NP (Non-pipelined) :** 14h
	23	**Reserved :** 0
	22	**Z Bias State Mask :** 1 = Update ; 0 = Do Not Update
	21:14	**Z Bias :** This is a signed value that is added to the source Z value prior to the Z compare function. This 2's complement fixed point value has 7 integer bits and 0 fractional bits and is sign extended before being added to the destination Z value. Valid data values range from -128 to 127. (S7) The default value is 0. The Z value written back to Z-buffer included this Z bias value.
	13	**Alpha Function State Mask :** 1 = Update; 0 = Do Not Update
	12:9	**Alpha Function :** Valid values are : 00h = Reserved 01h = Never (never pass) 02h = Less (pass if the source Alpha is less than the Alpha reference) 03h = Equal (pass if the source Alpha is equal to the source Alpha) 04h = Lequal (pass if the source Alpha is less than or equal to the source Alpha) 05h = Greater (pass if the source Alpha is greater than the source Alpha) 06h = Notequal (pass if the source Alpha is note equal to the source Alpha) 07h = Gequal (pass if the source Alpha is greater than or equal to the source Alpha) 08h = Always (always pass)
	8	**Alpha Reference State Mask :** 1 = Update; 0 = Do Not Update
	7:3	**Alpha Reference value for color calculation equation(s) :** This is bits 7:3 of an alpha reference value. This value specifies a reference alpha value which pixels are tested against when alpha-testing is enabled. The default value is 0. (unsigned int)
	2:0	**Reserved :** 000

The Z Bias state variable specifies a value with which to offset the source z value before performing the z compare test (refer to as backend Z bias). This is used for coplanar polygon priority. If two polygons are to be rendered which are coplanar, due to the inherent precision differences induced by unique x, y and z values, there is no guarantee which polygon will be closer or farther. By using the Z bias state variable, it is possible to offset the source z value (compare value) before comparing with the destination z value. The Z bias value is added at the LSBs of a 16 bit z representation. Using the Z bias state variable affects the source z value and the result of the z-bias addition is written to the z buffer.

Also, Z biasing is supported by adding to the source z coefficients (Co) after setup and then interpolation is performed (refer to as frontend Z bias). This is done through the vertex instruction packet by enabling z-bais in the control word and specifying a z-bias value with the vertex data. The result of the addition will be used in z function comparison. The result is written back to the Z buffer.

13.18. GFXRENDERSTATE_LINE_WIDTH_CULL_SHADE_ MODE

The provoking vertex refers to the vertex that selected the flat shaded color for the primitive. In the OpenGL* specification, the third/second (triangle/line) vertex should used. In D3D* specification, the first vertex should be used. There is an additional mode to allow the selection of the common vertex for triangle fan primitive and the third/second (triangle/line) vertex for other primitives.

Please note that this instruction packet affects the following selections:

- D3D* and OGL strip and fan notations selection
- D3D* and OGL vertex selection on flat shading color
- In flat shade mode the value of the first vertex in the primitive is used to determine the value of the face. In Gouraud shade mode the value of the face is determined by a linear interpolation between all of the primitive's vertices.

DWord	Bit	Description
0	31:29	**Client :** 03h **–** Render Processor
	28:24	**3DState24 :** 02h
	23	Reserved
	22:21	Reserved
	20	**Z Function State Mask :** 1 = Update ; 0 = Do Not Update
	19:16	**Z Function :** Valid values are : 00h = Reserved 01h = Never (never pass) 02h = Less (pass if the source Z is less than the destination Z) 03h = Equal (pass if the source Z is equal to the destination Z) 04h = Lequal (pass if the source Z is less than or equal to the destination Z) 05h = Greater (pass if the source Z is greater than the destination Z) 06h = Notequal (pass if the source Z is note equal to the destination Z) 07h = Gequal (pass if the source Z is greater than or equal to the destination Z) 08h = Always (always pass)
	15	**Line Width State Mask :** 1 = Update; 0 = Do Not Update
	14:12	**Line Width :** This value specifies the width of lines in pixels. The variable is represented as an unsigned value with 2 integer bits and 1 fractional bit. Valid values range from 0.0 to 3.5 in .5 increments. The default value is 0.
	11	**Alpha Shade Mode State Mask :** 1 = Update; 0 = Do Not Update
	10	**Alpha Shade Mode :** 1 = Flat, 0 = Gouraud (default)
	9	**Fog Shade Mode State Mask :** 1 = Update; 0 = Do Not Update
	8	**Fog Shade Mode :** 1 = Flat, 0 = Gouraud (default)
	7	**Specular Shade Mode State Mask :** 1 = Update; 0 = Do Not Update
	6	**Specular Shade Mode :** 1 = Flat, 0 = Gouraud (default)
	5	**Color Shade Mode State Mask :** 1 = Update; 0 = Do Not Update
	4	**Color Shade Mode :** 1 = Flat, 0 = Gouraud (default)
	3	**Cull Mode State Mask :** 1 = Update; 0 = Do Not Update
	2:0	**Cull Mode :** To support culling of back facing triangles 000 = Reserved 001 = Cull No Triangle (No back face removal) 010 = Cull CW Triangles (Allow CCW triangles) 011 = Cull CCW Triangles (Allow CW triangles) 100 = Cull Both (For performance experiments 101 to 111 = Reserved

13.19. GFXRENDERSTATE_BOOLEAN_ENA_1

Dword	Bit	Description
0	31:29	**Client :** 03h – Render Processor
	28:24	**3DState24 :** 03h
	23:20	**Reserved :** 0h
	19	**Specular Setup Enable Mask :** 1 = Update; 0 = Do Not Update
	18	**Specular Setup Enable :** 0 = Disable (default), 1 = Enable This SV enables specular gradient calculations in the Setup Unit (independent of other specular SVs). This bit is currently not used. This bit, along with it's corresponding mask bit, are tied to bits 8 and 9 of this same register.
	17	**Alpha Setup Enable Enable Mask :** 1 = Update; 0 = Do Not Update
	16	**Alpha Setup Enable :** 0 = Disable (default), 1 = Enable This SV enables alpha gradient calculations in the Setup Unit (independent of other alpha SVs).
	15	**Color Index Key Enable Mask :** 1 = Update; 0 = Do Not Update
	14	**Color Index Key Enable :** 0 = Disable (default), 1 = Enable
	13	**Chromakey Enable Mask :** 1 = Update; 0 = Do Not Update
	12	**Chromakey Enable :** 0 = Disable (default), 1 = Enable
	11	**Source Z Bias Enable Mask:** 1 = Update; 0 = Do Not Update
	10	**Source Z Bias Enable:** 1 = Enable, 0 = Disable (default)
	9	**Specular Enable State Mask :** 1 = Update; 0 = Do Not Update
	8	**Specular Enable :** 1 = Enable, 0 = Disable (default)
	7	**Fog Enable State Mask :** 1 = Update; 0 = Do Not Update
	6	**Fog Enable :** 1 = Enable, 0 = Disable (default)
	5	**Alpha Enable State Mask :** 1 = Update; 0 = Do Not Update
	4	**Alpha Test Enable :** 1 = Enable, 0 = Disable (default)
	3	**Blend Enable State Mask :** 1 = Update; 0 = Do Not Update
	2	**Blend Enable :** 1 = Enable, 0 = Disable (default)
	1	**Z Enable State Mask :** 1 = Update; 0 = Do Not Update
0	0	**Z Enable :** 1 = Enable, 0 = Disable (default)

13.20. GFXRENDERSTATE_BOOLEAN_ENA_2

DWord	Bit	Description
0	31:29	**Client :** 03h **–** Render Processor
	28:24	**3DState24 :** 04h
	23:18	**Reserved :** 00h
	17	**Mapping Cache Enable Mask :** 1 = Update; 0 = Do Not Update
	16	**Mapping Cache Enable :** 1 = Enable, 0 = Disable (default) Any change made to the Mapping Cache Enable state variable must be followed by a GFXCMDPARSER_FLUSH instruction to cause the hardware to flush the 3D pipeline.
	15	**Alpha Dither Enable Mask :** 1 = Update; 0 = Do Not Update
	14	**Alpha Dither Enable :** 1 = Enable, 0 = Disable (default)
	13	**Fog Dither Enable Mask :** 1 = Update; 0 = Do Not Update
	12	**Fog Dither Enable :** 1 = Enable, 0 = Disable (default)
	11	**Specular Dither Enable Mask :** 1 = Update; 0 = Do Not Update
	10	**Specular Dither Enable :** 1 = Enable, 0 = Disable (default)
	9	**Color Dither Enable Mask :** 1 = Update; 0 = Do Not Update
	8	**Color Dither Enable:** 1 = Enable, 0 = Disable (default)
	7:4	**Reserved:** 0h
	3	**Frame Buffer Write Enable Mask :** 1 = Update; 0 = Do Not Update
	2	**Frame Buffer Write Enable :** 1 = Enable, 0 = Disable (default)
	1	**Z Buffer Write Enable Mask :** 1 = Update; 0 = Do Not Update
	0	**Z Buffer Write Enable :** 1 = Enable, 0 = Disable (default)

The frame buffer write enable feature can be used to reliably render coplanar (layered) polygons. An example of this type of layering would be to render a window on a wall. With a 16-bit z buffer, there is not sufficient precision to render this type of scene with consistent priority. Some pixels of the window will be intermixed with some pixels of the wall, due to precision inaccuracies.

13.21. GFXRENDERSTATE_FOG_COLOR

The GFXRENDERSTATE_FOG_COLOR state instruction format is:

DWord	Bit	Description
0	31:29	**Client :** 03h – Render Processor
	28:24	**3DState24NP (Non-pipelined) :** 15h
	23:19	**Fog Color Red :** Bits 7:3 of the red fog color component. The default value is 0. (unsigned int)
	18:16	**Reserved :** 0h
	15:10	**Fog Color Green :** Bits 7:2 of the green fog color component. The default value is 0. (unsigned int)
	9:8	**Reserved :** 0h
	7:3	**Fog Color Blue :** Bits 7:3 of the blue fog color component. The default value is 0. (unsigned int)
	2:0	**Reserved :** 0h

13.22. GFXRENDERSTATE_DRAWING_RECTANGLE_INFO

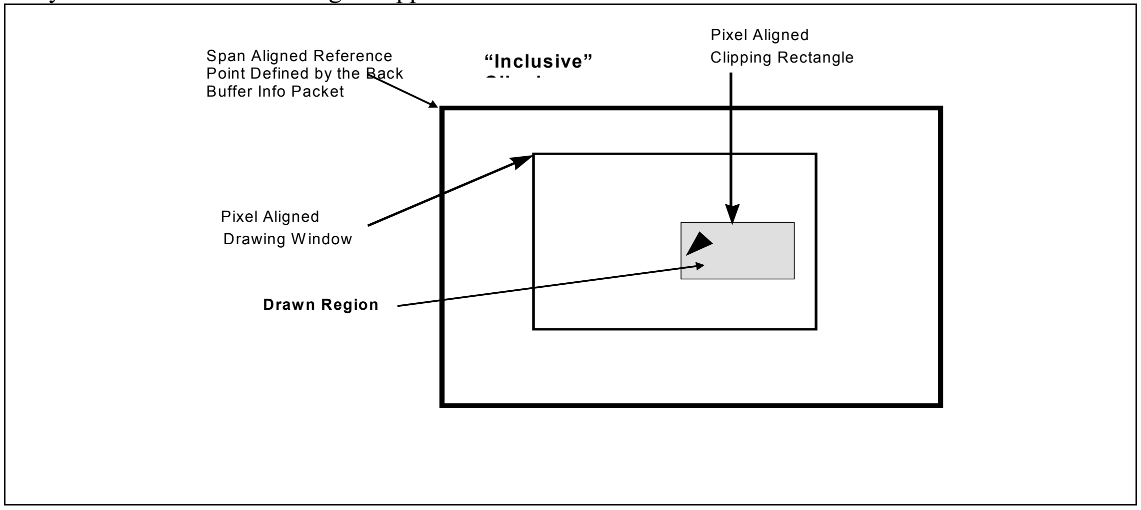
The values programmed in this packet are in screen coordinates. In full screen mode, the screen pitch and height minus one need to be programmed into this packet. For example, for screen size of 1280x1024, the values programmed should be Xmin=0, Ymin=0, Xmax=1279, Ymax=1023. When rendering rectangle primitives, the height or width of the drawing rectangle must be greater than 1 (e.g., (Xmax - Xmin) > 1 and (Ymax - Ymin) > 1).

The drawing rectangle coordinates need to be clipped to the screen boundary by software before send down to hardware.

DWord	Bits	Description
0	31:29	**Client** : 03h – Rendering Engine
	28:24	**3DStateMWNP (Non-pipelined)** : 1Dh
	23:16	**Opcode** : 80h
	15:0	**DWORD_LENGTH** : 3
1	31	**Drawing/Scissor Rectangle clipping Enable** (for validation purpose only). This bit should be set to enable in normal mode. 0 = Enable 1 = Disable
	30:28	**Reserved:** 00h
	27:26	**X Bias:** A 2-bit value of x bias for dithering compensation
	25:24	**Y Bias:** A 2-bit value of y bias for dithering compensation
	23:0	**Reserved:** 00h
2	31:16	**Clipped DR Ymin:** Y coordinate of upper left corner of drawing rectangle after being clipped to screen boundary. Only the lower 10 bits in U10 format is used by hardware. This value is a pixel-aligned, unsigned integer, and reference to the upper-left corner of the 2D buffer defined in the dest buffer info packet.
	15:0	**Clipped DR Xmin:** X coordinate of upper left corner rectangle after being clipped to screen boundary. Only the lower 11 bits in U11 format is used by hardware. This value is a pixel-aligned, unsigned integer, and reference to the upper-left corner of the 2D buffer defined in the dest buffer info packet.
3	31:16	**Clipped DR Ymax:** Y coordinate of lower right corner rectangle after being clipped to screen boundary. Only the lower 10 bits in U10 format is used by hardware. This value is a pixel-aligned, unsigned integer, and reference to the upper-left corner of the 2D buffer defined in the dest buffer info packet.
	15:0	**Clipped DR Xmax:** X coordinate of lower right corner rectangle after being clipped to screen boundary. Only the lower 11 bits in U11 format is used by hardware. This value is a pixel-aligned, unsigned integer, and reference to the upper-left corner of the 2D buffer defined in the dest buffer info packet.
4	31:27	**Reserved**
	26:16	**DR origin Y:** Origin of the Y coordinate of the drawing rectangle before being clipped to screen boundary. The value is S10 in 2's complement format. This value is a pixel-aligned, and reference to the upper-left corner of the 2D buffer defined in the dest buffer packet.
	15:12	**Reserved**
	11:0	**DR origin X:** Origin of the X coordinate of the drawing rectangle before being clipped to screen boundary. The value is S11 in 2's complement format. This value is a pixel-aligned, and reference to the upper-left corner of the 2D buffer defined in the dest buffer packet.

13.23. GFXRENDERSTATE_SCISSOR_ENABLE

Only inclusive mode scissoring is supported.


Span Aligned Reference Point Defined by the Back Buffer Info Packet
"Inclusive"
Pixel Aligned Clipping Rectangle
Pixel Aligned Drawing Window
Drawn Region

DWord	Bit	Description
0	31:29	**Client :** 03h – Rendering Engine
	28:24	**3DState16NP (Non-pipelined) :** 1Ch
	23:19	**Opcode :** 10h
	18:17	**Reserved:** 00h (For Future Additional Scissor Rectangles)
	16	**Scissor Rectangle ID :** This number must be 0 for this stepping since only one clipping rectangle is allowed. (More clipping rectangles can be allowed in future expansion)
	15:2	**Reserved:** 00h
	1	**Scissor Rectangle Enable Mask :** 1 = Update; 0 = Do Not Update
0	0	**Scissor Rectangle Enable :** 0 = Disable (default), 1 = Enable

13.24. GFXRENDERSTATE_SCISSOR_RECTANGLE_INFO

The coordinate in this instruction packet is relative to the origin (upper left corner, DWord #4) of the drawing rectangle defined in the GFXRENDERSTATE_DRAWING_RECTANGLE_INFO packet. The coordinates of the scissor rectangle do not need to be clipped to screen boundary before sent to hardware.

DWord	Bits	Description
0	31:29	**Client** : 03h – Rendering Engine
	28:24	**3DStateMWNP (Non-pipelined)** : 1Dh
	23:16	**Opcode** : 81h
	15:0	**DWORD_LENGTH** : 1
1	31:16	**Ymin:** Y coordinate of upper left corner. Only the lower 10 bits is used by hardware. This value is pixel-aligned, U10 format, and reference to the origin (unclipped upper-left corner) of the drawing rectangle.
	15:0	**Xmin:** X coordinate of upper left corner. Only the lower 11 bits is used by hardware. This value is pixel aligned, U11 format, and reference to the origin (unclipped upper-left corner) of the drawing rectangle.
2	31:16	**Ymax:** Y coordinate of lower right corner. Only the lower 10 bits is used by hardware. This value is pixel-aligned, U10 format, and reference to the origin (unclipped upper-left corner) of the drawing rectangle.
	15:0	**Xmax:** X coordinate of lower right corner. Only the lower 11 bits is used by hardware. This value is pixel-aligned, U11 format, and reference to the origin (unclipped upper-left corner) of the drawing rectangle.

13.25. Stipple Pattern

The stipple pattern is a 4x4 bit memory that serves as pixel write mask. When stipple is enabled, the frame buffer will only be updated with pixels that have 1's in their associated stipple pattern memory locations. The stipple pattern memory maps to a 4x4 pixel grid. The stipple pattern repeats for x and y coordinates on every span (4x4 pixels). The purpose of the stipple pattern is mostly for testing. This is a non-pipeline state variable. The stipple pattern pixel enable function can be represented as:

DWord	Bits	Description
0	31:29	**Client :** 03h – Rendering Engine
	28:24	**3DStateMWNP (Non-pipelined) :** 1Dh
	23:16	**Opcode :** 83h
	15:0	**DWORD_LENGTH :** 0
1	31:17	**Reserved**
	16	**Enable:** 1 = Enable, 0 = Disable
	15:0	**Stipple Pattern**

13.26. GFXRENDERSTATE_ANTI_ALIASING

The Anti-aliasing packet defines the anti-aliasing enable, bounding-box expansion, line anti-aliasing region, polygon anti-aliasing region, and edge flag enable State variables.

The *anti-aliasing enable* state variable turns on anti-aliasing of polygons and lines. The Bounding Box enclosing the triangle or line is expanded in all directions by the *bounding box expansion* value (in pixels) . The *line antialiasing region width* state variable allows for the specification of the width over which to blend for antialiasing of lines. Similarly, the *polygon antialiasing region width* state variable allows for the specification of the width over which to blend for antialiasing of polygon edges. The *edge flag enable* state variable enables all the edge flags of a triangle/line overriding the edge flags specified in the vertex format.

DWord	Bits	Description
0	31:29	**Client :** 03h – Rendering Engine
	28:24	**3DState24 :** 06h
	23:14	**Reserved:** 000h
	13	**Edge Flag Enable Mask:** 1 = Update; 0 = Do Not Update
	12	**Edge Flag Enable:** 1 = Enable; 0 = Disable
	11	**Polygon Antialiasing Region Width Enable Mask:** 1 = Update, 0 = Do Not Update
	10:9	**Polygon Antialiasing Region Width:** Valid values are: 0h = 0.5 pixels 1h = 1.0 pixels 2h = 2.0 pixels 3h = 4.0 pixels
	8	**Line Antialiasing Region Width Enable Mask:** 1 = Update, 0 = Do Not Update
	7:6	**Line Antialiasing Region Width:** Valid values are: 0h = 0.5 pixels 1h = 1.0 pixels 2h = 2.0 pixels 3h = 4.0 pixels
	5	**Bounding Box Expansion Mask:** 1 = Update, 0 = Do Not Update
	4:2	**Bounding Box Expansion:** U3.0 format(Unsigned) with valid values from 0 to 7pixels
	1	**Anti-aliasing Enable Mask:** 1 = Update, 0 = Do Not Update
0	0	**Anti-aliasing Enable:** 1 = Enable, 0 = Disable

13.27. GFXRENDERSTATE_PROVOKING_VTX_PIXELIZATION_RULE

The provoking vertex state variables provide the flexibility of selecting the flat-shaded vertex for the first triangle/line of a primitive packet. The subsequent flat-shaded vertices are in incrementing order, except in the case of triangle fans with common flat shaded vertex. Separate State variables are provided for triangles and lines. The following tables show the sequence of flat-shaded vertices for triangles and lines.

Primitive Type	Triangle Strip/List Flat Shading SV	Triangle Fan Flat Shading SV	Line Strip/List Flat Shading SV	Flat Shaded Vertex Sequence
Triangle/Rectangle List	00	XX	XX	0, 3, 6, 9, 12 ....
	01	XX	XX	1, 4, 7, 10, 13 ...
	10	XX	XX	2, 5, 8, 11, 14 ...
Triangle Strip/Triangle Strip Reverse Winding order	00	XX	XX	0, 1, 2, 3, 4 .. ....
	01	XX	XX	1, 2, 3, 4, 5. .. ....
	10	XX	XX	2, 3, 4, 5, 6 .. ....
Triangle Fan	XX	00	XX	0, 0,0,0,0,0 .. ....
"	XX	01	XX	1, 2, 3, 4, 5. .. ....
"	XX	10	XX	2, 3, 4, 5, 6 .. ....
Polygon(Triangle Fan With Common Flat Shaded Vertex)	XX	XX	XX	0,0,0,0,0,0 ....
Line List	XX	XX	X0	0, 2, 4, 6, 8 ....
	XX	XX	X1	1, 3, 5, 7, 9 ...
Line Strip	XX	XX	X0	0, 1, 2, 3, 4 .. ....
	XX	XX	X1	1, 2, 3, 4, 5. .. ....

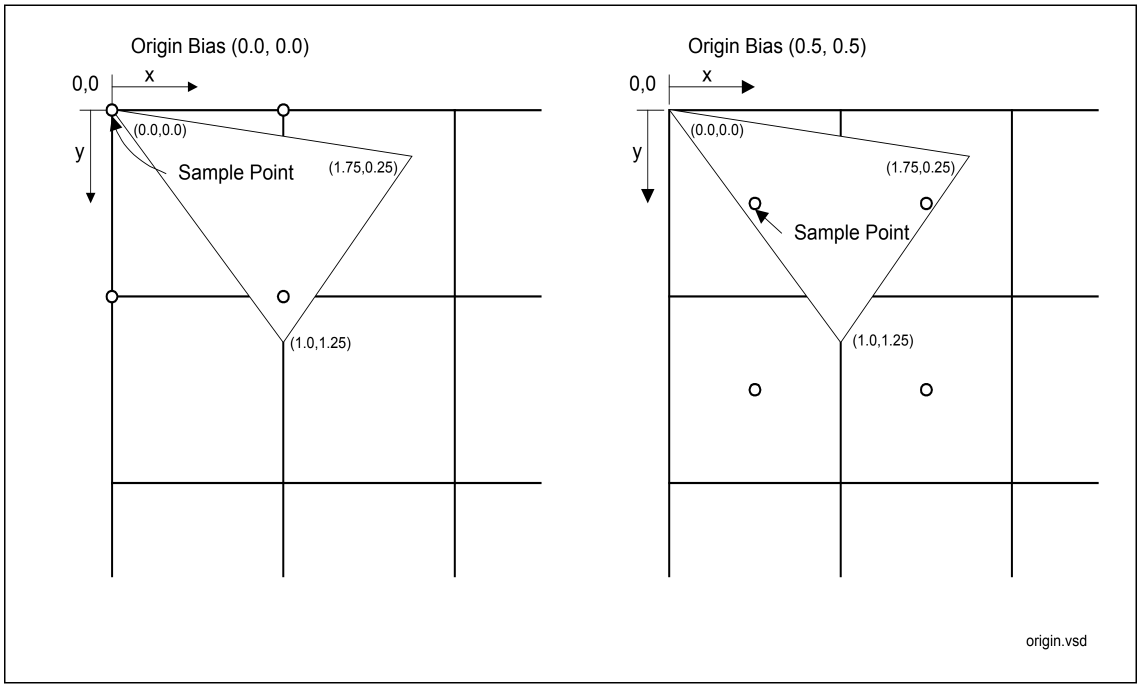
The *Pixelization Rule* state-variable sets up the hardware to follow correct pixelization rules. It should be set in Render Mode (i.e., for GFXPrimitive packet). The *Small Triangle Filter Enable*, enables filtering of small triangles. (*Small Triangles* are defined as triangles which do not light any pixels). It should be set only in Render Mode when Pixelization Rule is enabled, and the Horizontal & Vertical Bias Values are equal and are programmed to either 1000 or 0000. If other bias values are used then the results may be unpredictable. (The Horizontal & Vertical Bias values are programmed in the GFXRENDERSTATE_DEST_BUFFER_VARIABLES packet). The following table summarizes acceptable values of these SVs:

Packet	Horizontal Bias	Vertical Bias	Pixelization Rule	Small Triangle Filter
GFXPRIMITIVE	0000	0000	1	1
GFXPRIMITIVE	1000	1000	1	1
GFXPRIMITIVE	XXXX	XXXX	1	0
GFXPRIMITIVE	XXXX	XXXX	0	0
GFXBLOCK	XXXX	XXXX	0	0

DWord	Bits	Description
0	31:29	**Client** : 03h – Rendering Engine
	28:24	**3DState24** : 07h
	23:13	**Reserved:** 0000h
	12	**Small Triangle Filter Enable Mask :** 1 = Update ; 0 = Do Not Update
	11	**Small Triangle Filter Enable:** If Set then Small Triangles i.e triangles which donot cover any pixels, and are either vertical or horizontal will be filtered out. Note: This should be enabled only in GFXPrimitive Mode when Pixelization Rule is enabled, and the “Horizontal & Vertical Bias” values (in the Dest Buffer Packet) are both the same and programmed to 1000 or 0000.
	10	**Pixelization Rule Mask:** 1 = Update ; 0 = Do Not Update
	9	**Pixelization Rule:** This bit should be set in GFXPRIMITIVE MODE for following the correct Pixelization Rules.
	8	**Line Strip/List Flat Shaded Vertex Select Mask:** 1 = update, 0 = Do Not Update
	7:6	**Line Strip/List Flat Shaded Vertex Select:** 0h = Vertex 0 1h = Vertex 1 2h = Reserved 3h = Reserved
	5	**Triangle Fan Flat Shaded Vertex Select Enable Mask:** 1 = update, 0 = Do Not Update
	4:3	**Triangle Fan Flat Shaded Vertex Select:** 0h = Vertex 0 1h = Vertex 1 2h = Vertex 2 3h = Reserved
	2	**Triangle Strip/List Flat Shaded Vertex Select Enable Mask:** 1 = update, 0 = Do Not Update
0	1:0	**Triangle Strip/List Flat Shaded Vertex Select:** 0h = Vertex 0 1h = Vertex 1 2h = Vertex 2 3h = Reserved

13.28. GFXRENDERSTATE_DEST_BUFFER_VARIABLES

The GFXRENDERSTATE_DEST_BUFFER_VARIABLES instruction is used to specify the information about the destination buffer. This information is used to initialize rendering hardware parameters. The destination buffer contains the pixel color data for the scene being rasterized by the 3D Rendering Engine. This is a non-pipelined state variable. Two examples of how the Origin BIAS works are shown below.



The format is:

DWord	Bit	Description
0	31:29	**Client** : 03h – Rendering Engine
	28:24	**3DStateMWNP (Non-pipelined)** : 1Dh
	23:16	**Opcode** : 85h
	15:0	**DWORD_LENGTH** : 0

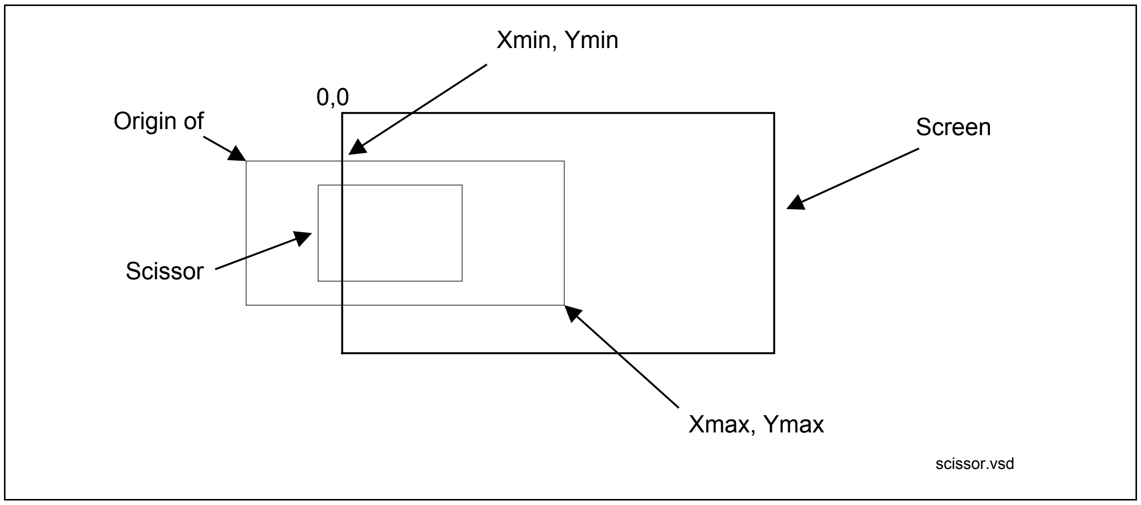
DWord	Bit	Description
1	31:24	**Reserved:** 00h
	23:20	**Destination Origin Horizontal Bias:** This is an unsigned value (0.4) that is used to bias the origin of the X values associated with the vertices of a primitive. The unbiased origin is located in the upper-left corner of a square pixel with the center located at 0.5, 0.5. A bias value of ½ (8h) would move the origin toward the right, such that an X value of 0.0 would specify the center of a pixel.
	19:16	**Destination Origin Vertical Bias:** This is an unsigned value (0.4) which is used to bias the origin of the Y values associated with the vertices of a primitive. The unbiased origin is located in the upper-left corner of a square pixel/texel with the center located at 0.5, 0.5. A bias value of ½ (8h) would move the origin toward the bottom, such that a Y value of 0.0 would specify the center of a pixel/texel.
	15:14	**Reserved:** 00h
	13:12	**4:2:2 Channel Write Select:** Select which channel(s) are written to the destination buffer when the surface format is 4:2:2 using byte masks. 00 = Write all channels (Y,Cr, and Cb) 01 = Write only the Y channel 10 = Write only the Cr channel 11 = Write only the Cb channel
	11	**Reserved** (Additional Buffer Formats)
	10:8	**Dest Buffer Format:** 0h = Any 8-bit Surface 1h = RGB 555 (16th bit is written as 0) 2h = RGB 565 3h = Reserved (ARGB 8888) 4h = 4:2:2 YCrCb (Y Swap format in memory) 5h = 4:2:2 YcrCb (Normal format in memory) 6h = 4:2:2 YcrCb (UV Swap format in memory) 7h = 4:2:2 YcrCb (UV/Y Swap format in memory)
	7:2	**Reserved:** 00h
	1	**Vertical Line Stride:** The number of lines to skip between logically adjacent lines. 0 = Do not skip any lines. 1 = Skip 1 line. (provides support for Interleaved/field surfaces)
	0	**Vertical Line Stride Offset:** The number of lines to add as an initial offset when the Vertical Line Stride is 1. 0 = Add no offset. (top field) 1 = Add 1. (bottom field)

13.29. Programming Hints/Rules

The following provides programming hints/rules for 3D, Motion Compensation, and Stretch Blitter operations.

Drawing and Scissor Rectangles

- Must declare a Drawing Rectangle packet first before Scissor Rectangle packet
- Coordinates of Drawing Rectangle must be $X_{max} > X_{min}$ and $Y_{max} > Y_{min}$
- Drawing Rectangle width and height must be greater than 1pixel for Rectangle primitives.
- Drawing Rectangle X & Y coordinates must be within the screen size. The drawing rectangle must be clipped to screen coordinates before send to hardware. However, the origin of the drawing rectangle can be negative since it is need not be clipped. It is the reference point (origin) of all the primitives and scissor rectangle that belong to the drawing rectangle.



Small Triangle Filter

- Small Triangle Filter can only be enabled if the horizontal and vertical bias values (defined in Dest Buffer variable packet) are programmed to (0000,0000) or (1000,1000). This is corresponds to the pixel center of (0,0) and (0.5,0.5), respectively.

Mapping Engine Cache (MEC)

- A Texture palette load must contain all 256 entries. Even if only one entry needs to be updated, all 256 entries in the Texture palette must be loaded using the texture palette load packet.
- Color Key and Chroma Key cannot be enabled at the same time.
- In mult-texture, Colorkey and Chromkey only compare with Map 0 texels and not on Map 1 texels.
- Any programming change made to the Mapping Engine Cache Enable state variable must be followed by a 3D pipeline flush, otherwise pixel corruption will occur. For example, software changes the state of the Mapping Engine cache by flipping the bit Mapping Cache Enable using the

3D instruction GFXRENDERSTATE_BOOLEAN_EN_2. To ensure proper hardware operation, software must follow up with the GFXCMDPARSER_FLUSH instruction to cause the hardware to flush the 3D pipeline.

Color Calculator

- Texture blend stage 0 must always be enabled. For untextured primitives, a select arg operation should be programmed. This is equivalent to a pass through operation to the next pipeline stage.
- When Stage N is enabled, Stages 0:(N-1) should also be enabled.
- Current (as opposed to accumulator) must be selected as output for the last enabled stage.
- Data must be written to Current output register before it is selected as an input argument for the next stage.
- Data must be written to Accumulator output register before it is selected as an input argument for a later stage.
- Replicate α in specular is not allowed.
- Current α cannot be used for blending.

Stretch BLT

- When doing stretch BLT in YUV 422 format, the origin of the source rectangle must be DWord aligned. In other words, the X coordinate of the origin must be even.

Texture/Color/Z Surface Pitch

- Texture surface pitch[1] == Pitch of tiled region (programmed in fence register).
- Texture surface pitch must be ≥ 4 QWs for linear organization of memory.

  i.e. MAP_INFO_DW1[3:0] ≥ 2h
- Texture surface pitch must be ≥ 16 QWs for Tiled organization of memory.

  i.e. MAP_INFO_DW1[3:0] ≥ 4h
- Texture surface pitch (in bytes) must be ≥ Map_Width (in bytes)

  i.e. $8 * 2^{\text{MAP\_INFO\_DW1[3:0]}} \geq 2^{\text{MAP\_INFO\_DW2[3:0]}}$ (pow2mapsize == 1)

  $8 * 2^{\text{MAP\_INFO\_DW1[3:0]}} \geq$ MAP_INFO_DW2[9:0] (pow2mapsize == 0)
- Texture base address (MAP_INFO_DW3[26:4]) must be QW aligned. Also,

  2n * surface pitch ≤ texture base address < (2n+1) surface pitch.
- Color Pitch and Z Pitch[2] == Pitch of tiled region.
- Only Bit(26:4) in the Texture Map Base Address are used in texture address calculation.

Context 1

- No state variable pipelining in Context 1. Change of state variables required a pipeline flush.
- No Multi-TX in context 1. Only texel 0 is available.

[1] The texture surface pitch (MAP_INFO_DW1[3:0]) is expressed in terms of log_2(pitch in QWs).

[2] The Color/Z pitch (DEST_BUFFER_INFO_DW2[1:0]/ Z_BUFFER_INFO_DW1[1:0]) is expressed in terms of log_2(pitch in QWs/64).

- Z buffering is not supported (Z_en is overloaded to 0).
- State variables that have a context1 are: blend_en, dest_blend(3:0), src_blend(3:0), xoffset(11:0), yoffset(11:0), all Dest Buffer format variable packet, all Map Info packet, Map Cache enable.
- Limit support on FVF packet format is supported: TX coordinate count =1, Specular_Fog_Present=0, Diffuse_Alpha_Present= programmable, Z_Offset_Present=0, Position Mask= programmable.
- Context 1 is intended to be used for StretchBLT.

In Context 1, Blending is not supported for Stretch-BLT. In this context, Blending is only supported for Sub-Picture.

This page is intentionally left blank.

14. Clock Control Registers

The clock control registers are accessed by writing to the memory mapped address offset.

The Intel® 815 chipset has three PLLs to generate all the clocks. The Host PLL generates the host clock whose frequency is controlled by an external strap. In addition, the Host PLL generates the system and local memory core clock and graphics core clock. The Hub PLL generates the clock for the Hub Interface unit. The Display PLL generates the display or LCD clock.

The display clock can be controlled by three blocks of registers: DCLK0, DCLK1, and DCLK2. Each display clock has its own Display Clock i Divisor registers for M, N and a byte of Display & LCD Clock Divisor Select Register within which are P (Divisor) values and can be programmed independently. DCLK0 and DCLK1 normally are programmed to 25.175 MHz and 28.322 MHz respectively (VGA compatible clocks), DCLK2 is used for non-VGA modes.

The Display Clock i Divisor register and the appropriate byte of Display & LCD Clock Divisor Select Register are programmed with the loop parameters to be loaded into the clock synthesizer. The MSR[3:2] register is used to select between DCLK0(default), DCLK1 and DCLK2. Writing to LCD / TV-Out Control [31] = 1 and [0] = 1 selects the LCD clock. The MSR[3:2] are ignored when this condition is true.

The data written to these registers are calculated based on the reference frequency, the desired output frequency, and characteristic VCO constraints as described in the Functional Description. From the calculation, the M,N and P values are obtained.

14.1. Programming Notes

Three blocks of registers exist to program up to three unique frequencies for the display clock. These registers are named DCLK0, DCLK1, and DCLK2. Each of these blocks can be programmed independently of each other. However, only one can be used to control the DPLL at any given time.

The MSR register (bits 3:2) is used to determine which of the DCLK0, 1, 2 registers groups will control the DPLL. Writing to the MSR register bits 3:2 also transfers the Display Clock Divisor and Display & LCD Clock Divisor Select Register contents to the VCO register file.

Example Programming Sequence (DCLK0)

1. Write the Display Clock 0 Divisor register with the M-REG value and N-REG value.
2. Write the clock 0 byte of the Display & LCD Clock Divisor Select Register with the P-REG value.
3. Write the MSR register, bits 3:2 = '00', to select DCLK0 (NOTE: This is the default value).

Example Programming Sequence (DCLK1)

1. Write the Display Clock 1 Divisor register with the M-REG value and N-REG value.
2. Write the clock 1 byte of the Display & LCD Clock Divisor Select Register with the P-REG value.
3. Write the MSR register, bits 3:2 = '01', to select DCLK1.

Example Programming Sequence (DCLK2)

1. Write the Display Clock 2 Divisor register with the M-REG value and N-REG value.
2. Write the clock 2 byte of the Display & LCD Clock Divisor Select Register with the P-REG value.
3. Write the MSR register, bit 3 = '1', to select DCLK2.

Example Programming Sequence (LCD CLK)

1. Write the LCD clock Divisor register with the M-REG value and N-REG value.
2. Write the LCD byte of the Display & LCD Clock divisor Select Register with the P-REG value.
3. Write the LCD / TV-Out Control[31] = 1 and [0] = 1; MSR[3:2] are ignored when this condition is true.

14.2. DCLK_0D—Display Clock 0 Divisor Register

Address Offset:	06000h–06003h
Default Value:	00030013h
Attribute:	R/W
Size:	32 bits

The **Display Clock 0 Divisor** register and the **Display & LCD Clock Divisor Select Register** are programmed with the loop parameters to be loaded into the clock synthesizer.

Data is written to the **Display Clock 0 Divisor** register followed by a write to the Clock 0 byte of the **Display & LCD Clock Divisor Select Register**. The completion of the write to the **Display & LCD Clock Divisor Select Register** causes data from both registers to transfer to the VCO register file simultaneously. This prevents wild fluctuations in the VCO output during intermediate stages of a clock programming sequence.

31 26	25 16	15 10	9 0
Reserved	VCO 0 N-Divisor	Reserved	VCO 0 M-Divisor

Bit	Description
31:26	**Reserved.**
25:16	**VCO 0 N-Divisor.** N-Divisor value calculated for the desired output frequency. (default = 03h)
15:10	**Reserved.**
9:0	**VCO 0 M-Divisor.** M-Divisor value calculated for the desired output frequency. (default = 13h)

14.3. DCLK_1D—Display Clock 1 Divisor Register

Address Offset:	06004h–06007h
Default Value:	00100053h
Attribute:	R/W
Size:	32 bits

The **Display Clock 1 Divisor** register and the **Display & LCD Clock Divisor Select Register** are programmed with the loop parameters to be loaded into the clock synthesizer.

Data is written to the **Display Clock 1 Divisor** register followed by a write to the Clock 1 byte of the **Display & LCD Clock Divisor Select Register**. The completion of the write to the **Display & LCD Clock Divisor Select Register** causes data from both registers to transfer to the VCO register file simultaneously. This prevents wild fluctuations in the VCO output during intermediate stages of a clock programming sequence.

31 26	25 16	15 10	9 0
Reserved	VCO 1 N-Divisor	Reserved	VCO 1 M-Divisor

Bit	Description
31:26	**Reserved**
25:16	**VCO 1 N-Divisor.** N-Divisor value calculated for the desired output frequency. (default = 10h)
15:10	**Reserved**
9:0	**VCO 1 M-Divisor.** M-Divisor value calculated for the desired output frequency. (default = 53h)

14.4. DCLK_2D—Display Clock 2 Divisor Register

Address Offset:	06008h–0600Bh
Default Value:	00030013h
Attribute:	R/W
Size:	32 bits

The **Display Clock 2 Divisor** register and **Display & LCD Clock Divisor Select Register** are programmed with the loop parameters to be loaded into the clock synthesizer.

Data is written to **Display Clock 2 Divisor** register followed by a write to the Clock 2 byte of the **Display & LCD Clock Divisor Select Register**. The completion of the write to the **Display & LCD Clock Divisor Select Register** causes data from both registers to transfer to the VCO register file simultaneously. This prevents wild fluctuations in the VCO output during intermediate stages of a clock programming sequence.

31 26	25 16	15 10	9 0
Reserved	VCO 2 N-Divisor	Reserved	VCO 2 M-Divisor

Bit	Description
31:26	**Reserved.**
25:16	**VCO 2 N-Divisor.** N-Divisor value calculated for the desired output frequency. (default = 03h)
15:10	**Reserved.**
9:0	**VCO 2 M-Divisor.** M-Divisor value calculated for the desired output frequency. (default = 13h)

14.5. LCD_CLKD—LCD Clock Divisor Register

Address Offset:	0600Ch–0600Fh
Default Value:	00030013h
Attribute:	R/W
Size:	32 bits

The **LCD Clock Divisor** register and **Display & LCD Clock Divisor Select Register** are programmed with the loop parameters to be loaded into the clock synthesizer.

Data is written to **LCD Clock Divisor** register followed by a write to the LCD Clock byte of the **Display & LCD Clock Divisor Select Register**. The completion of the write to the **Display & LCD Clock Divisor Select Register** causes data from both registers to transfer to the VCO register file simultaneously. This prevents wild fluctuations in the VCO output during intermediate stages of a clock programming sequence.

31 26	25 16	15 10	9 0
Reserved	VCO LCD N-Divisor	Reserved	VCO LCD M-Divisor

Bit	Description
31:6	**Reserved**.
25:16	**VCO LCD N-Divisor.** N-Divisor value calculated for the desired output frequency. (default = 03h)
15:10	**Reserved.**
9:0	**VCO LCD M-Divisor.** M-Divisor value calculated for the desired output frequency. (default = 13h)

14.6. DCLK_0DS—Display & LCD Clock Divisor Select Register

Address Offset:	06010h–06013h
Default Value:	40404040h
Attributes:	R/W
Size:	32 bits

Display clock i {i=0 to 2} becomes effective after programming the appropriate **byte** i {i = 0 to 2}in this register. LCD clock becomes effective after programming byte 3 in this register.

31	30 28	27	26	25 24
Reserved	Post Divisor Select LCD Clk	Reserved	VCO Loop Div LCD clk	Reserved

23	22 20	19	18	17 16
Reserved	Post Divisor Select Clk 2	Reserved	VCO Loop Div clk 2	Reserved

15	14 12	11	10	9 8
Reserved	Post Divisor Select Clk 1	Reserved	VCO Loop Div clk 1	Reserved

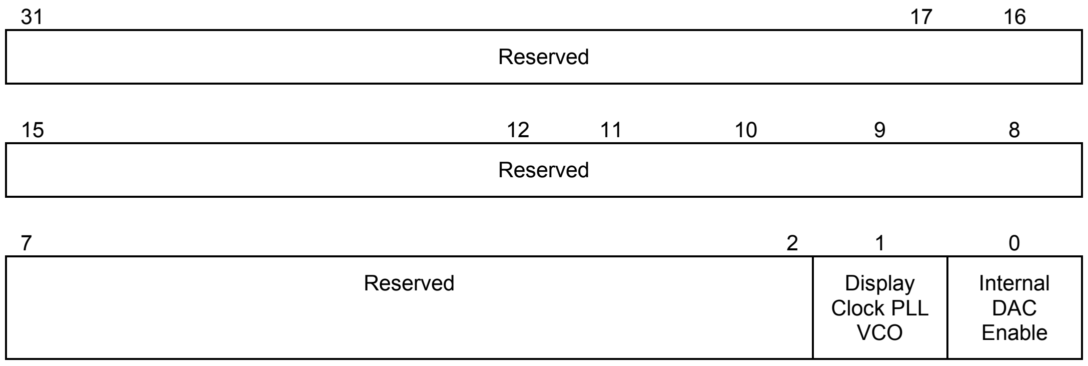
7	6 4	3	2	1	0
Reserved	Post Divisor Select Clk 0	Reserved	VCO Loop Div clk 0	Reserved	DPLL DSYNCON

Bit	Description
31	**Reserved.**
30:28	**Post Divisor Select LCD Clock.** 000 = Divide by 1 001 = Divide by 2 010 = Divide by 4 011 = Divide by 8 100 = Divide by 16 (default) 101 = Divide by 32 11x = Reserved
27	**Reserved.**
26	**VCO Loop Divide LCD Clock.** 0 = Divided by 4 (default) 1 = Divided by 16
25:23	**Reserved.**

Bit	Description
22:20	**Post Divisor Select clock 2.** 000 = Divide by 1 001 = Divide by 2 010 = Divide by 4 011 = Divide by 8 100 = Divide by 16 (default) 101 = Divide by 32 11x = Reserved
19	**Reserved.**
18	**VCO Loop Divide clock 2.** 0 = Divided by 4*M (default) 1 = Divided by 16*M
17:15	**Reserved.**
14:12	**Post Divisor Select clock 1.** 000 = Divide by 1 001 = Divide by 2 010 = Divide by 4 011 = Divide by 8 100 = Divide by 16 (default) 101 = Divide by 32 11x = Reserved
11	**Reserved.**
10	**VCO Loop Divide clock 1.** 0 = Divided by 4*M (default) 1 = Divided by 16*M
9:7	**Reserved.**
6:4	**Post Divisor Select clock 0.** 000 = Divide by 1 001 = Divide by 2 010 = Divide by 4 011 = Divide by 8 100 = Divide by 16 (default) 101 = Divide by 32 11x = Reserved
3	**Reserved.**
2	**VCO Loop Divide clock 0.** 0 = Divided by 4 (default) 1 = Divided by 16
1:0	**Reserved.**

14.7. PWR_CLKC—Power Management and Miscellaneous Clock Control

Address Offset:	6014h–06017h
Default Value:	0000 0101 h
Attribute:	R/W
Size:	32 bits

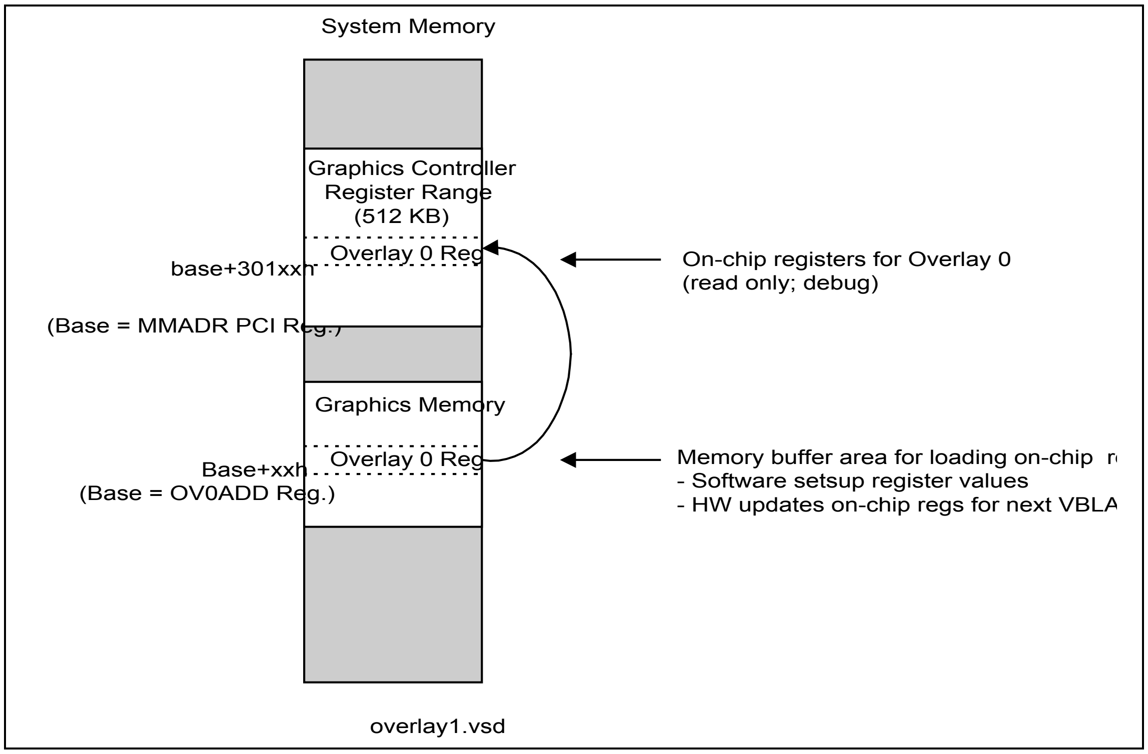


Bit	Description
31:2	**Reserved.**
1	**Display Clock PLL VCO** 0 = Enable (default) 1 = Disable
0	**Internal DAC Enable.** 0 = Disables the internal DAC (PowerDown). If HSYNC/VSYNCControl[0] = 0, disables HSYNC and VSYNC. 1 = Enables the internal DAC and does not allow disable of HSYNC and VSYNC via HSYNC/VSYNC Control[0]. (default)

15. Overlay Registers

This chapter contains the Overlay and Gamma Correction registers and an Overlay instruction. The current graphics controller implements one overlay that is referred to as Overlay 0. Note that the Overlay 0 control registers are indirectly written by first setting up a buffer in memory and then instructing the graphics controller to update the on-chip registers from this buffer. Software can invoke the update process by writing to the OV0ADD register or by issuing the Overlay Flip instruction. Note that the Gamma Correction registers are read/written directly. The register/instruction categories are listed in the Overlay Register/Instruction Categories table.

Note: The Intel® 815 chipset does not support overlay for low display resolution modes. Thus, the overlay is not supported for all resolution modes smaller than 640x480.


System Memory
Graphics Controller Register Range (512 KB)
Overlay 0 Reg
base+301xxh
(Base = MMADR PCI Reg.)
Graphics Memory
Overlay 0 Reg
Base+xxh
(Base = OV0ADD Reg.)
On-chip registers for Overlay 0
(read only; debug)
Memory buffer area for loading on-chip r
- Software setsup register values
- HW updates on-chip regs for next VBLA
overlay1.vsd

Do not use the “Wait for VBLANK” mechanism to force a sequence of overlay flips. Use the “Wait for Scan Lines” mechanism with the scan line set up to be at least 1 scan line after vertical blank start to force the loading of the next Overlay x Register Update Address which will take effect after the next displayed overlay frame.

Table 16. Overlay Register/Instruction Categories

Register/Instruction Category	Mem. Address Offset	Comment
Overlay 0 Register Update Address (OV0ADD)	30000h	• Used to update Overlay 0 registers. • Provides physical memory address of buffer area used for updating on-chip registers • Write of OV0ADD register causes hardware to update on-chip registers on next VBLANK.
Display/Overlay 0 Status Register (DOV0STA)	30008h	• Overlay Status bits
Gamma Correction (GAMMA[0:5])	30010h–30027h	• Not part of double-buffer scheme. Access registers directly.
Overlay Register Sets • Overlay Buffer Pointer • Overlay Stride • Overlay Initial Phase • Overlay Dest. Window Position/Size • Overlay Source Size • Overlay Scale Factor • Overlay Color Correction • Overlay Destination Color Key • Overlay Source Color Key • Overlay Configuration • Overlay Command	301xxh (on chip RO regs) Base+xxh (Mem Buffer regs)	• “xx” indicates a particular register address. • On-chip registers are not directly writeable. — Registers are double buffered (buffer in memory and on-chip registers). — Software sets up buffer in memory. — Software writes to OV0ADD Register which provides memory buffer address location and causes hardware to read memory buffer and update on-chip registers during next VBLANK. • On-chip registers can be read through Debug read path. See OV0ADD register description
Overlay Flip Instruction	NA	• This instruction invokes the Overlay register update.

15.1. OV0ADD—Overlay 0 Register Update Address Register

Memory Address Offset:	30000h–30003h
Default Value:	00000000
Access:	R/W
Size:	32 bits

This register provides a physical memory address that will be used on the next register update for Overlay 0. This register is double buffered to allow it to be updated during overlay display.

Updating Register Values

A write to this register sets an internal bit (readable through the status register) that causes all the register values that were written to the memory buffer area to be loaded into the corresponding on-chip registers and become active on the next VBLANK event. Overlay 0 Flip is asserted after all registers are updated from memory.

Debug Read Path For Overlay 0 Registers

The read path is the active register. The staging register is not readable.

Memory Address Offset: 30100h–301xxh (xx=register offset)

These registers will provide a debug read path for testing overlay internal register functionality. The address offsets correspond to the memory offsets used to load the registers.

31 29	28 0
Reserved	Register Update Address

Bit	Description
31:29	**Reserved.**
28:0	**Register Update Address.** Physical memory address that will be used on the next register update for Overlay 0..

15.2. DOV0STA—Display/Overlay 0 Status Register

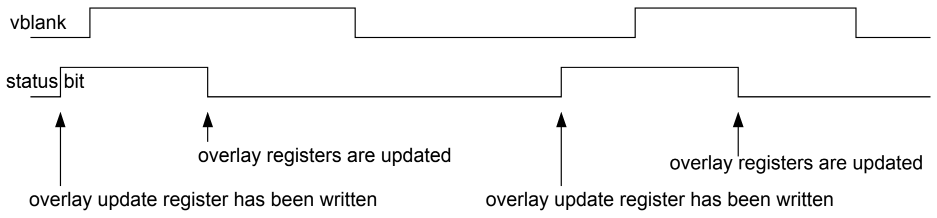
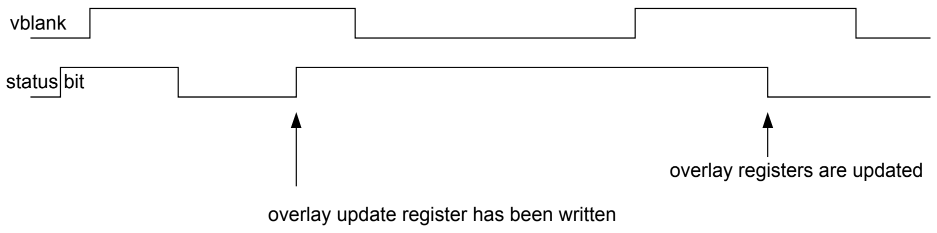
Memory Address Offset:	30008h–3000Bh
Default Value:	0000 5000h
Access:	RO
Size:	32 bits

This read-only register indicates the status for the overlay. References to display are either the primary timing generator or the secondary timing generator depending on which is currently being used.

31	30:24
Reg Update Status	Reserved

23:21	20:19	18:15
Reserved	Overlay 0 Current Buffer/Field.	Reserved

14	13	12	11	10:0
Not Active Pixel	Reserved	Not Active Video Scan Line	Reserved	Display Line Status.

Bit	Description
31	**Register Update Status.** All registers latched (Flip Pending = 0).  vblank status bit overlay registers are updated overlay update register has been written overlay registers are updated overlay update register has been written  vblank status bit overlay registers are updated overlay update register has been written **0** = overlay registers have been updated. **1** = overlay update register has been written, overlay registers have not been updated.
30:21	**Reserved.**

Bit	Description
20:19	**Overlay 0 Current Buffer/Field.** This field indicates the Current Buffer. Updated at display VBLANK before the interrupt, this field is only valid when in field (interlaced) mode. 00 = Buffer 0 Field 0 01 = Buffer 0 Field 1 10 = Buffer 1 Field 0 11 = Buffer 1 Field 1
18:15	**Reserved.**
14	**Not Active Pixel.** This bit indicates the Display Horizontal Blank Active state (includes Border). Updated real time, set by leading edge of Overlay's display HBLANK and cleared by the trailing edge of Overlay's HBLANK. 0 = HBlank inactive 1 = HBlank active (default).
13	**Reserved.**
12	**Not Active Video Scan Line.** This bit indicates the Display Vertical Blank Active state (includes Border). Updated real time, set by leading edge of Overlay's display VBLANK and cleared by the trailing edge of Overlay's VBLANK. 0 = VBlank inactive 1 = VBlank active (default).
11	**Reserved.**
10:0	**Display Line Status.** This field displays the line number. Reset to zero at the trailing edge of display VBLANK. Incremented at the trailing edge of Overlay's display HBLANK

15.3. Gamma Correction

Note that the registers in this section are read from or written to directly at the memory address offset location specified.

15.3.1.1. GAMC[5:0]—Gamma Correction Registers

Memory Address Offset:	30010h–30027h
	GAMC5 = 30010h
	GAMC4 = 30014h
	GAMC3 = 30018h
	GAMC2 = 3001Ch
	GAMC1 = 30020h
	GAMC0 = 30024h

Default Value:	GAMC5 = C0C0C0h
	GAMC4 = 808080h
	GAMC3 = 404040h
	GAMC2 = 202020h
	GAMC1 = 101010h
	GAMC0 = 080808h

Access:	R/W
Size:	32 bits

These registers determine the characteristics of the gamma correction for the overlay data. Each register is 32 bits wide. The registers are written to and read from together when accessed from the PCI.

The GAMCxx registers are not double buffered and should only be updated when the Overlay Engine is disabled. During operation, the bytes of these registers are read independently. These registers are dependent on the R, G, or B pixel values.

There are several restrictions when using gamma values. The restrictions are:

- Gamma curve should be always a rising curve
- GAMC0 cannot have value of 0
- Difference between the two adjacent gamma registers cannot be greater than 0x7F

31 24	23 16	15 8	7 0
Reserved	Red	Green	Blue

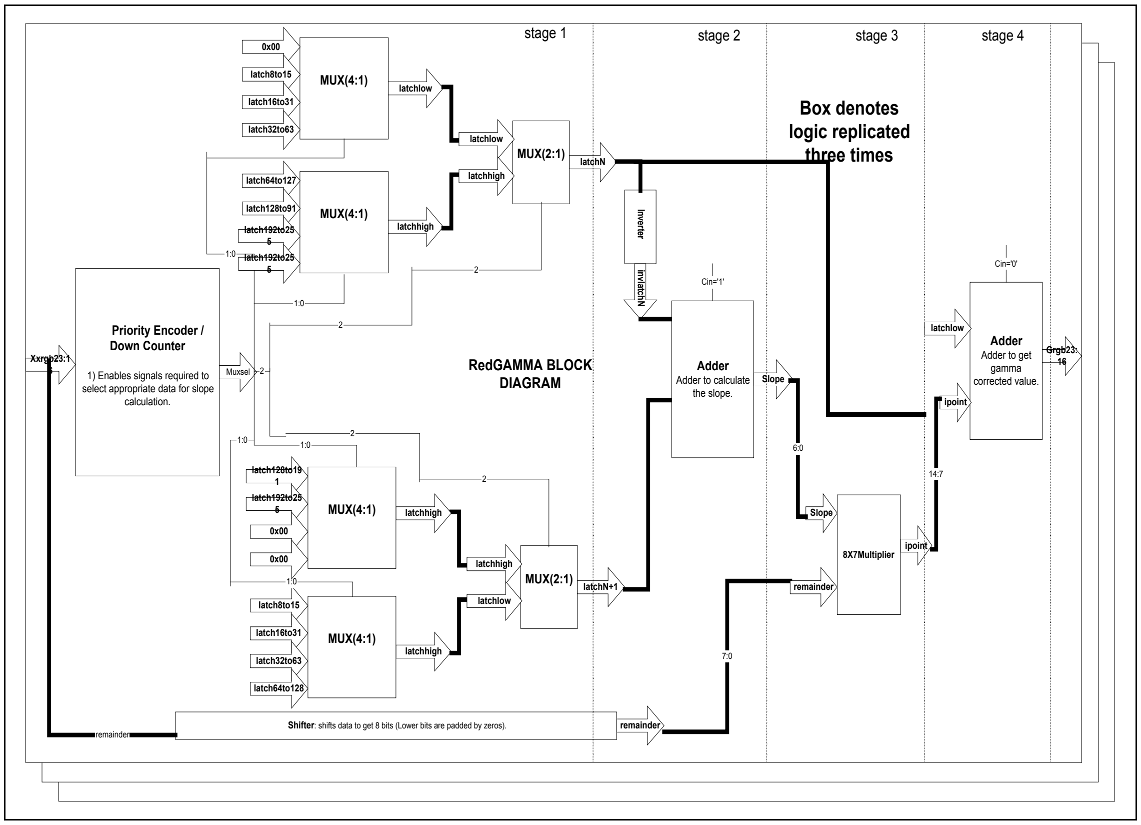
Bit	**Description**
31:24	**Reserved.**
23:16	**Red.**
15:8	**Green.**
7:0	**Blue.** GAMC5: Used of bit <7:6> = 11 GAMC4: Used of bits <7:6> = 10 GAMC3: Used of bits <7:6> = 01 GAMC2: Used of bits <7:5> = 001 GAMC1: Used of bits <7:4> = 0001 GAMC0: Used of bits <7:3> = 00001

15.3.1.2. Mathematical Gamma Correction For Overlay

Gamma correction is a function that corrects for non-linearity between display phosphor brightness as a function of electron beam current. Depending on the actual phosphor used, and/or whether the event that data streams have been pre-gamma corrected for different phosphor characteristics than the display device being used, the digital display stream must be corrected to achieve equally spaced increase in brightness on the display for equally spaced increases in color intensity values.

Gamma correction shall be chiefly implemented via a piecewise linear approximation of a curve. There are 6 individual breakpoint values, logarithmically spaced in the Color Intensity domain. Only one set of 8 values is provided and shared for each of the Red, Green, and Blue intensity components. These values shall be loadable via software control and are programmable. For each of Red, Green, and Blue, the appropriate Gamma breakpoint pairs are looked up and smoothly interpolated between to arrive at the final Red, Green, and Blue values that are output to the DAC.

Figure 41. Gamma Correction Unit Block Diagram



As shown, the inputs to the function are:

- PCI Register Bus: The chip internal PCI data bus and the appropriate register decodes for loading the Gamma breakpoint values

• Red In 2e07:00:	The Red color component of the Overlay Stream
• Green In 2e07:00:	The Red color component of the Overlay Stream
• Blue In 2e07:00:	The Red color component of the Overlay Stream

The output of the function is:

• Red Out:	The Gamma corrected Red color component.
• Green Out:	The Gamma corrected Green color component.
• Blue Out:	The Gamma corrected Blue color component.

Gamma Correction Theory Of Operation

The Gamma Correction function outlined above is implemented via piece wise linear interpolator that allows any arbitrary curve to be output as a function of any linear input. The function provides smooth (continuous) interpolations efficiently in hardware while taking little compromise in the preciseness of the output curve relative to the ideal. The breakpoints of the curve are defined via software and loaded via the PCI internal data bus.

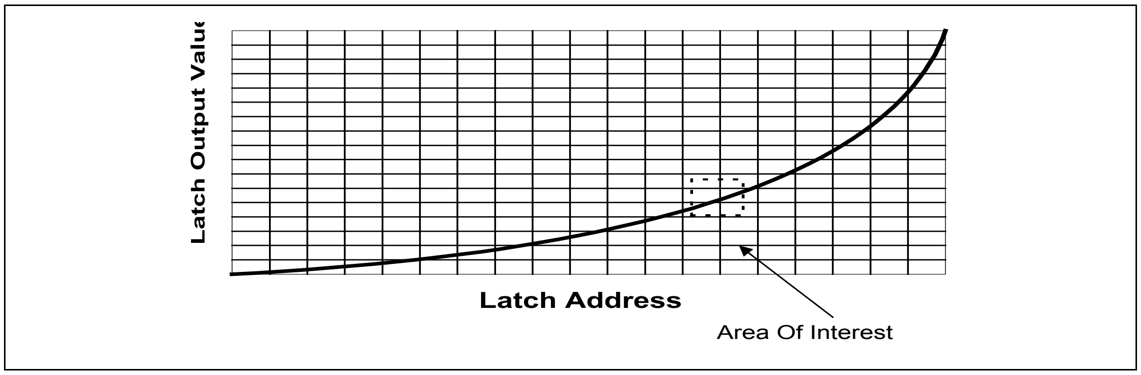
In general, the implementation outlined above allows for 7 logarithmically spaced breakpoints, of which the 5 internal breakpoints are software definable, the 2 endpoints are predefined at zero and 255 (decimal). The value at a given breakpoint, and the breakpoint +1 (using the MSB which is 1 of an 8 bit color are looked up in the table simultaneously). These two points define the exact desired values at those breakpoints. The values between the breakpoints are linearly interpolated using the remaining bits of the color. The output of the function therefore evaluates the equation:

Result = mX + B

Where:

m	= Slope
B	= Initial Value defined at table entry N

The slope, m, is derived by taking the difference between the breakpoint defined by [N] and the next successive breakpoint [N+1]. The value at [N+1] is simply derived via the input wiring to the mux. Since all slopes are defined to be positive for this function, the use of an unsigned multiplier is sufficient for this function.



The following gives a more detailed description of the algorithm. A line with two coordinate points as (x1, y1) and (x2, y2) can be defined by the following equation:

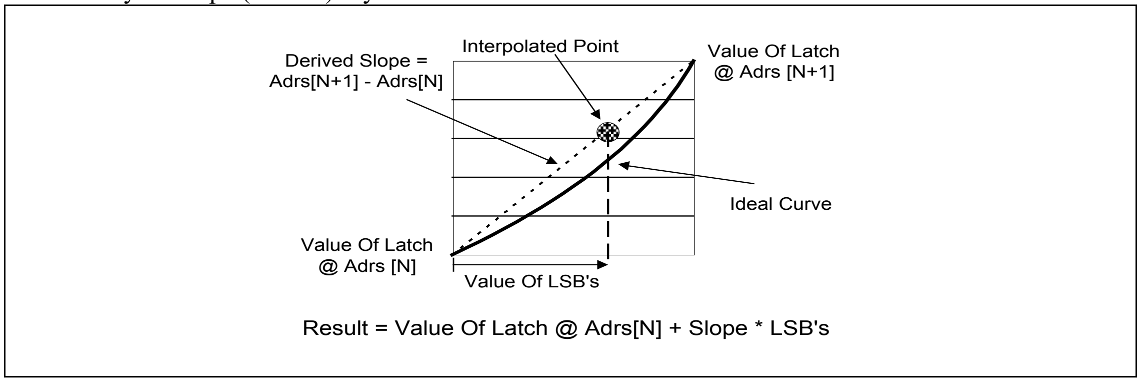
y2 = slope (x2 - x1) + y1

This implies slope of the line is:

slope = (y2 - y1) / x2 -x1

Another point with X = x3 on the same line can be defined as (x3, y3) where y3 is

y3 = slope (x3 - x1) + y1


Derived Slope = Adrs[N+1] - Adrs[N]
Interpolated Point
Value Of Latch @ Adrs [N+1]
Ideal Curve
Value Of Latch @ Adrs [N]
Value Of LSB's
Result = Value Of Latch @ Adrs[N] + Slope * LSB's

Gamma correction can be bypassed by programming the registers with data values corresponding to a linear curve with slope = 1. The register programming for gamma bypassing is as shown below.

Adrs[8] = 8	Adrs[64] = 64
Adrs[16] = 16	Adrs[128] = 128
Adrs[32] = 32	Adrs[192] = 192

When registers are programmed with the above values, output of the gamma unit is the same as the input, and no gamma correction occurs.

Gamma Hardware Implementation

```
Result = Adrs[192] + [[0x00      - Adrs[192]] * [X - 192] << 1] >> 7    192  <= X < 256
Result = Adrs[128] + [[Adrs[192] - Adrs[128]] * [X - 128] << 1] >> 7    128  <= X < 192
Result = Adrs[64  ] + [[Adrs[128] - Adrs[ 64 ]] * [[X - 64] << 1]] >> 7   64   <= X < 128
Result = Adrs[32  ] + [[Adrs[64  ] - Adrs[ 32 ]] * [[X - 32] << 2]] >> 7   32   <= X <  64
Result = Adrs[16  ] + [[Adrs[32  ] - Adrs[ 16 ]] * [[X - 16] << 3]] >> 7   16   <= X <  32
Result = Adrs[  8  ] + [[Adrs[16  ] - Adrs[  8  ]] * [[X -   8] << 4]] >> 7    8   <= X <  16
Result = 0x00       + [[Adrs[8    ] -    0x00  ]] * [[X - 0  ] << 4]] >> 7    0   <= X <   8
```

Consider Adrs[8] = 8, Adrs[16] = 16, Adrs[32] = 32, Adrs[64] = 64, Adrs[128] = 128, Adrs[192] = 192.

```
For X = 5, Result = 0 + [[8-0] * [5 << 4]] >> 7                  == 5
For X = 11, Result = 8 + [[16-8] * [3 << 4]] >> 7                == 11
For X = 24, Result = 16 + [[32-16] * [8 << 3]] >> 7              == 24
For X = 63, Result = 31 + [[64-32] * [31 << 2]] >> 7             == 63
For X = 100, Result = 64 + [[128-64] * [36 << 1]] >> 7           == 100
For X = 156, Result = 128 + [[192-128] * [28<< 1]] >> 7          == 156
For X = 200, Result = 192 + [[256-192] * [8 << 1]] >> 7          == 200
```

15.4. Memory Offset Registers

15.4.1. Overlay Buffer Pointer Registers

These registers provide address pointers into the system memory or Local memory buffer areas. The buffers must be QWord aligned. Pixel panning on a pixel basis is done using the byte addresses. Overlay buffers need to be QWord aligned and the stride should be a QWord multiple. Buffer pointers should always be aligned to the natural boundaries based on the data format. For planar formats (YUV410, YUV420), the Y and UV pointers should be naturally aligned to each other. Their natural alignment depends on the particular data format.

Format	Alignment	
	Pixels	Bytes
RGB packed	1	2
YUV 4:2:2 packed	2	4
YUV 4:1:1 packed	8	12
YUV Planar	1	1

Only the Register Update Address register should be written while the overlay is active. Otherwise, values will be loaded by writing the register image into memory and writing the command register with the address of the memory image in the Register Update address.

15.4.1.1. OBUF_0Y—Overlay Buffer 0 Y Pointer Register

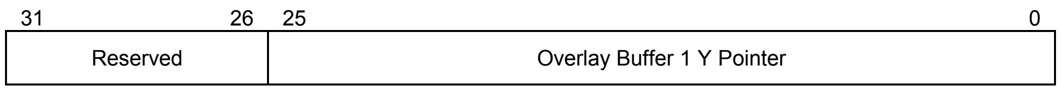
Memory Buffer Address Offset:	00h (R/W)
On-chip Reg. Mem Addr Offset:	30100h (RO; debug path)
Default Value:	00h
Access:	see address offset above
Size:	32 bits

31 26	25 0
Reserved	Overlay Buffer 0 Y Pointer

Bit	Description
31:26	**Reserved.**
25:0	**Overlay Buffer 0 Y Pointer.** For Y Planar or packed color data (byte address). Must be pixel aligned (low order bit zero for 16-bpp packed formats). When mirroring horizontally (X backwards), this points to the last byte of the line.

15.4.1.2. OBUF_1Y—Overlay Buffer 1 Y Pointer Register

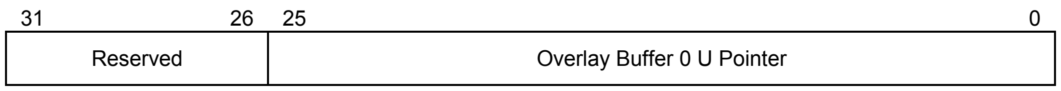
Memory Address Offset:	04h (R/W)
On-chip Reg. Mem Addr Offset:	30104h (RO; debug path)
Default Value:	00h
Access:	see address offset above
Size:	32 bits



Bit	Description
31:26	**Reserved.**
25:0	**Overlay Buffer 1 Y Pointer.** For Y Planar or packed color data (byte address). Must be pixel aligned (low order bit zero for 16-bpp packed formats). When mirroring horizontally (X backwards), this points to the last byte of the line.

15.4.1.3. OBUF_0U—Overlay Buffer 0 U Pointer Register

Memory Address Offset:	08h (R/W)
On-chip Reg. Mem Addr Offset:	30108h (RO; debug path)
Default Value:	00h
Access:	see address offset above
Size:	32 bits



Bit	Description
31:26	**Reserved.**
25:0	**Overlay Buffer 0 U Pointer.** This register is used for YUV Planar Modes only. It points to the start of the U addresses in the interleaved UV formats (byte address).

15.4.1.4. OBUF_0V—Overlay Buffer 0 V Pointer Register

Memory Address Offset:	0Ch (R/W)
On-chip Reg. Mem Addr Offset:	3010Ch (RO; debug path)
Default Value:	00h
Access:	see address offset above
Size:	32 bits

31 26	25 0
Reserved	Overlay Buffer 0 V Pointer

Bit	Description
31:26	**Reserved.**
25:0	**Overlay Buffer 0 V Pointer.** This register is used for YUV Planar Modes only. It points to the start of the V addresses in the interleaved UV formats (byte address).

15.4.1.5. OBUF_1U—Overlay Buffer 1 U Pointer Register

Memory Address Offset:	10h (R/W)
On-chip Reg. Mem Addr Offset:	30110h (RO; debug path)
Default Value:	00h
Access:	see address offset above
Size:	32 bits

31 26	25 0
Reserved	Overlay Buffer 1 U Pointer

Bit	Descriptiont
31:26	**Reserved.**
25:0	**Overlay Buffer 1 U Pointer.** This register is used for YUV Planar Modes only. It points to the start of the U addresses in the interleaved UV formats (byte address).

15.4.1.6. OBUF_1V—Overlay Buffer 1 V Pointer Register

Memory Address Offset:	14h (R/W)
On-chip Reg. Mem Addr Offset:	30114h (RO; debug path)
Default Value:	00h
Access:	see address offset above
Size:	32 bits

31 26	25 0
Reserved	Overlay Buffer 1 V Pointer

Bit	Description
31:26	**Reserved.**
25:0	**Overlay Buffer 1 V Pointer.** This register is used for YUV Planar Modes only. It points to the start of the V addresses in the interleaved UV formats (byte address).

15.4.2. Overlay Stride Registers

These values represent the width of the buffer that contains the overlay data. This is independent of the actual width displayed and is used to determine the line-to-line increment of the overlay buffer. In two line buffer mode, there is only room for 180 quadwords per scan line. If the source address is not quadword aligned, then for formats: YUV4:2:0, YUV4:2:2, and RGB, the source must be less then 720 pixels. The stride must be quadword aligned.

15.4.2.1. OV0STRIDE—Overlay 0 Stride Register

Memory Address Offset:	18h (R/W)
On-chip Reg. Mem Addr Offset:	30118h (RO; debug path)
Default Value:	00h
Access:	see address offset above
Size:	32 bits

31 26	28 16	15 13	12 0
Reserved	Overlay 0 UV planar Buffer Stride	Reserved	Overlay 0 Y planar or YUV/RGB Buffer Stride

Bit	Description
31:29	**Reserved.**
28:16	**Overlay 0 UV planar Buffer Stride.** Only used for YUV Planar formats and gives the U or V buffer stride in bytes. This is a two's complement number and is negative during Y mirroring. The low-order three bits are always zero forcing a QWord alignment. The range is ±4 KB.
15:13	**Reserved.**
12:0	**Overlay 0 Y planar or YUV/RGB Buffer Stride.** Buffer (Y planar or YUV/RGB packed) stride in bytes. This is a two's complement number and is negative during Y mirroring. The low-order three bits are always zero. The range is ±4 KB.

15.4.3. Overlay Initial Phase Registers

Provides a spatial sub-pixel accurate adjustment. This value is always a fractional positive number that when combined with the subtract one from initial phase bit, the possible range for initial phase becomes -1<phase<1. There are two separate vertical initial phase registers that are used in field based image processing.

15.4.3.1. YRGB_VPH—Y/RGB Vertical Phase Register

Memory Address Offset:	1Ch (R/W)
On-chip Reg. Mem Addr Offset:	3011Ch (RO; debug path)
Default Value:	00h
Access:	see address offset above
Size:	32 bits

31 20	19 16	15 4	3 0
Y/RGB Vertical Phase Buffer/Field 1	Reserved	Y/RGB Vertical Phase Buffer/Field 0	Reserved

Bit	Description
31:20	**Y/RGB Vertical Phase Buffer/Field 1.** This fractional value sets the initial vertical phase. In packed formats: YUV/RGB data buffer 1 when up scaling in frame mode YUV/RGB data field 1 when up scaling in field mode In planar YUV formats: Y data buffer 1 when up scaling in frame mode Y data field 1 when up scaling in field mode
19:16	**Reserved.**
15:4	**Y/RGB Vertical Phase Buffer/Field 0.** This fractional value sets the initial vertical phase. In packed formats: YUV/RGB data buffer 0 when up scaling in frame mode YUV/RGB data field 0 when up scaling in field mode. In planar YUV modes: Y data buffer 0 when up scaling in frame mode Y data field 0 when up scaling in field mode.
3:0	**Reserved.**

15.4.3.2. UV_VPH—UV Vertical Phase Register

Memory Address Offset:	20h (R/W)
On-chip Reg. Mem Addr Offset:	30120h (RO; debug path)
Default Value:	00h
Access:	see address offset above
Size:	32 bits

31 … 20	19 … 16	15 … 4	3 … 0
UV Vertical Phase 1	Reserved	UV Vertical Phase 0	Reserved

Bit	Description
31:20	**UV Vertical Phase 1.** This fractional value is only used in YUV planar formats where the UV plane may have a different vertical initial phase from the Y data. This field is used with Buffer 1 in frame mode or Field 1 in field mode.
19:16	**Reserved.**
15:4	**UV Vertical Phase 0.** This fractional value is only used in YUV planar formats where the UV plane may have a different vertical initial phase from the Y data. This field is used with Buffer 0 in frame mode or Field 0 in field mode.
3:0	**Reserved.**

15.4.3.3. HORZ_PH—Horizontal Phase Register

Memory Address Offset:	24h (R/W)
On-chip Reg. Mem Addr Offset:	30124h (RO; debug path)
Default Value:	00h
Access:	see address offset above
Size:	32 bits

31 … 20	19 … 16	15 … 4	3 … 0
UV Horizontal Phase	Reserved	Y/RGB Horizontal Phase	Reserved

Bit	Description
31:20	**UV Horizontal Phase.** Sets the initial horizontal phase for the UV data. Only used in YUV modes.
19:16	**Reserved.**
15:4	**Y/RGB Horizontal Phase.** Sets the initial horizontal phase for both buffers/fields. Unlike the vertical initial phases, this does not change buffer to buffer or field to field. YUV modes use a separate initial phase for Y and UV data. This value will either be the actual initial phase or the initial phase minus one.
3:0	**Reserved.**

15.4.3.4. INIT_PH—Initial Phase Register

Memory Address Offset:	28h (R/W)
On-chip Reg. Mem Addr Offset:	30128h (RO; debug path)
Default Value:	00h
Access:	see address offset above
Size:	32 bits

31 … 6	5 … 0
Reserved	Initial Phase minus one

Bit	Description
31:6	**Reserved.**
5:0	**Initial Phase minus one.** These bits provide a method of creating a negative initial phase. If the corresponding bit is set, the initial phase is the register value minus one. These bits should only be set in cases where the buffer pointer is pointing to the first pixel of the line or column because it will effectively cause the first pixel to be duplicated.
5	Y Vertical Buffer / Field 0
4	Y Vertical Buffer / Field 1
3	Y Horizontal
2	UV Vertical Buffer / Field 0
1	UV Vertical Buffer / Field 1
0	UV Horizontal

15.4.4. Overlay Destination Window Position/Size Registers

These registers allow for the positioning of the overlay data relative to the graphics display or the secondary display active region. It allows pixel accurate positioning. When using a secondary display, the area outside the overlay windows will be black RGB(0,0,0).

15.4.4.1. DWINPOS—Destination Window Position Register

Memory Address Offset:	2Ch (R/W)
On-chip Reg. Mem Addr Offset:	3012Ch (RO; debug path)
Default Value:	00h
Access:	see address offset above
Size:	32 bits

31 27	26 16	15 11	10 0
Reserved	Display Vertical top line	Reserved	Display Horizontal left pixel

Bit	Description
31:27	**Reserved.**
26:16	**Display Vertical top line.** Display vertical top in lines 0=begin at display first line
15:11	**Reserved.**
10:0	**Display Horizontal left pixel.** Determines where on the display screen coordinates the overlay display will begin. Display Horizontal Left in pixels 0=begin at display left edge

15.4.4.2. DWINSZ—Destination Window Size Register

Memory Address Offset:	30h (R/W)
On-chip Reg. Mem Addr Offset:	30130h (RO; debug path)
Default Value:	00h
Access:	see address offset above
Size:	32 bits

31 27	26 16	15 11	10 0
Reserved	Vertical Size	Reserved	HorizontalSize

Bit	Description
31:27	**Reserved.**
26:16	**Vertical Size.** Destination Vertical Size in lines (never specifies scan lines off the active display).
15:11	**Reserved.**
10:0	**Horizontal Size.** Destination Horizontal Size in pixels (never specifies pixels off the active display)

15.4.5. Overlay Source Size Registers

These registers provide information to the overlay engine on what data needs to be fetched from memory. If the overlay destination window is smaller than the result of the scaled up source, it will be clipped on the right and bottom of the overlay window. The source data is clipped within one QWord. Source width must not specify an additional QWord or more of data that is not required to satisfy the destination. If the scaled source is smaller than the destination window then the last pixel or last scan line accessed is used to fill the right and bottom areas. The minimum source width per operand (Y, U, and V for planar formats, packed formats have only one operand) is two QWords.

15.4.5.1. SWID—Source Width Register

Memory Address Offset:	34h (R/W)
On-chip Reg. Mem Addr Offset:	30134h (RO; debug path)
Default Value:	00h
Access:	see address offset above
Size:	32 bits

31 24	23 16	15 9	8 0
Reserved	UV Source Width	Reserved	Y/RGB Source Width

Bit	Description
31:24	**Reserved.**
23:16	**UV Source Width.** The number of bytes contained in a single line of planar UV source data. This is unused in packed modes and for planar modes it is used for the U and V source width (assumed to be the same) or in the interleaved mode (YI64) as TBD. When the last pixel is reached and the complete destination window has not been filled, this pixel will be repeated until the end of the destination window. When displaying YUV 4:1:1 data, this field contains the number of U bytes which identical to the number of V bytes and ¼ of the Y bytes.
15:9	**Reserved.**
8:0	**Y/RGB Source Width.** The number of bytes contained in a single line of source data. In planar modes, this is the Y source width. This should include all contributing pixel data. When the last pixel is reached and the complete destination window has not been filled, this pixel will be repeated until the end of the destination window. When displaying YUV 4:2:2 data, the atomic unit is a doubleword (2 Ys + U + V). When displaying YUV 4:1:1 data, the atomic unit is 3 doublewords (8 Ys + 2U + 2V). The starting offset within the buffer must reflect this restriction.

15.4.5.2. SWIDQW—Source Width In QWords Register

Memory Address Offset:	38h (R/W)
On-chip Reg. Mem Addr Offset:	30138h (RO; debug path)
Default Value:	00h
Access:	see address offset above
Size:	32 bits

31 … 24	23 … 16	15 … 9	8 … 0
Reserved	UV Source Width in QWs	Reserved	Y/RGB Source Width in QWs

Bit	Description
31:24	**Reserved.**
23:16	**UV Source Width in QWords.** The number of QWs contained in a single line of planar UV source data. Minimum size is two QWords.
15:9	**Reserved.**
8:0	**Y/RGB Source Width in QWords.** The number of QWs contained in a single line. In planar modes, this is the Y source width. Minimum size is two QWords.

15.4.5.3. SHEIGHT—Source Height Register

Memory Address Offset:	3Ch (R/W)
On-chip Reg. Mem Addr Offset:	3013Ch (RO; debug path)
Default Value:	00h
Access:	see address offset above
Size:	32 bits

31 26	25 16	15 11	10 0
Reserved	UV Source Height	Reserved	Y/RGB Source Height

Bit	Description
31:26	**Reserved.**
25:16	**UV Source Height.** In packed formats this is unused. For planar YUV formats it indicates the number of lines starting at and including the base address line contained in the UV source data. When the last line is reached and the complete destination window has not been filled, this line will be repeated until the end of the destination window. • A height must not be specified for lines that are not on the active display. • Minimum height is 2 lines or 4 lines for interlaced surfaces (Bob).
15:12	**Reserved.**
10:0	**Y/RGB Source Height.** In packed formats this indicates the number of lines starting at and including the base address line contained in the source data. In planar formats, it is the number of Y lines. This is used to determine where the end of the source is in the vertical direction to handle the edge effects related to the vertical filter. When the last line is reached and the complete destination window has not been filled, this line will be repeated until the end of the destination window. • A height must not be specified for lines that are not on the active display. • Minimum height is 2 lines or 4 lines for interlaced surfaces (Bob).

15.4.6. Overlay Scale Factor Registers

These registers provide the scaling information that is used to specify the amount of vertical and horizontal scaling. In the case of YUV formats, there are independent scale factors for Y and UV data to compensate for the various formats that include sub-sampled UV data. Up or down scaling is set using the bits of the command word. In the case of down scaling, the integer portion of the scale factor can be up to three in the vertical and is assumed to be one in the horizontal.

15.4.6.1. YRGBSCALE—Y/RGB Scale Factor Register

Memory Address Offset:	40h (R/W)
On-chip Reg. Mem Addr Offset:	30140h (RO; debug path)
Default Value:	00h
Access:	see address offset above
Size:	32 bits

31 20	19 17	16 15	14 3	2	1 0
Vertical Scale Factor Y/RGB	Reserved	Horizontal Scale Factor Y/RGB Integer	X Scale Factor Y/RGB	Re ser ved	Vertical Down Scale Integer

Bit	Description
31:20	**Vertical Scale Factor Y/RGB.** This is a fractional positive number that represents the scale factor to be used in vertical scaling. For packed formats, it applies to all color components and for planar YUV formats it is use for the Y component only. Always Bit 31:20 1 > scale ≥ 1/64 (the maximum vertical up zoom is 64X) (1/Scale for up scale fractional portion of the scale factor for down scaling)
19:17	**Reserved.**
16:15	**Horizontal Scale Factor Y/RGB Integer.** This field is used only for horizontal down scale. In the case of planar formats, it specifies the integer portion of the scale factor for Y data. Only Horizontal down scaling of 2 (or less) to 1 is supported.
14:3	**Horizontal Scale Factor Y/RGB.** This is a fractional positive number that represents the scale factor to be used in horizontal scaling. Always 1>scale≥0
2	**Reserved.**
1:0	**Vertical Down Scale Integer.** This field is used only for vertical down scale. In the case of planar formats, it specifies the integer portion of the scale factor for Y data.

15.4.6.2. UVSCALE—UV Scale Factor Register

Memory Address Offset:	44h (R/W)
On-chip Reg. Mem Addr Offset:	30144h (RO; debug path)
Default Value:	00h
Access:	see address offset above
Size:	32 bits

31 20	19 16
Vertical Scale Factor UV	Reserved

15	14 3	2	1 0
Reserved	Horizontal Scale Factor UV	Reserved	UV Down Scale Integer

Bit	Description
31:20	**Vertical Scale Factor UV.** Used only in YUV planar modes to set the scale factor for the UV data. This is different in formats where the UV data is vertically sub-sampled. Bit 31:20 1 > scale ≥ 1/64 (the maximum vertical up zoom is 64X)
19:15	**Reserved.**
14:3	**Horizontal Scale Factor UV.** Used only in YUV modes to set the scale factor for the UV data. This is different in formats where the UV data is sub-sampled.
2	**Reserved.**
1:0	**UV Down Scale Integer.** This field is used only for vertical down scale. It specifies the integer portion of the scale factor for UV data.

15.4.7. Overlay Color Correction Registers

Used for YUV sources only. Adjustments are made before the RGB conversion.

15.4.7.1. OV0CLRC0—Overlay 0 Color Correction 0 Register

Memory Address Offset:	48h (R/W)
On-chip Reg. Mem Addr Offset:	30148h (RO; debug path)
Default Value:	00h
Access:	see address offset above
Size:	32 bits

31 17	16 8	7 0
Reserved	Contrast	Brightness

Bit	Description
31:17	**Reserved.**
16:8	**Contrast.** - 7.3F (3.6 format) Bypassing Contrast, even for RGB, is accomplished by programming this field to 1.0
7:0	**Brightness.** Range ±127 a value of zero disables this adjustment affect. This value gets added to the Y value after contrast multiply and just before RGB conversion. Bypassing Brightness, even for RGB, is accomplished by programming this field to 0.

15.4.7.2. OV0CLRC1—Overlay 0 Color Correction 1 Register

Memory Address Offset:	4Ch (R/W)
On-chip Reg. Mem Addr Offset:	3014Ch (RO; debug path)
Default Value:	00h
Access:	see address offset above
Size:	32 bits

31 10	9 0
Reserved	Saturation/CrCb

Bit	Description
31:10	**Reserved.**
9:0	**Saturation/CrCb.** This operates in two modes. When YUV data is being operated on, the register contains the saturation value. When CbCr data is being used, it is the sum of the saturation multiplier value added to the CbCr scale factor (128/112). This unsigned fixed point number is in 3.7 format. Bypassing Saturation, even for RGB, is accomplished by programming this field to 1.0

15.4.8. Overlay Destination Color Key Registers

Used for YUV sources only. Adjustments are made before the RGB conversion.

15.4.8.1. DCLRKV—Destination Color Key Value Register

Memory Address Offset:	50h (R/W)
On-chip Reg. Mem Addr Offset:	30150h (RO; debug path)
Default Value:	00h
Access:	see address offset above
Size:	32 bits

31 24	23 0
Reserved	Destination Color Key Value

Bit	Description
31:24	**Reserved.**
23:0	**Destination Color Key Value.** In the format of the destination (screen). This can only be used when the screen is used as the destination (not TV). When the overlay is directed to TV output, this value replaces pixels that have been suppressed by the source color key.

15.4.8.2. DCLRKM—Destination Color Key Mask Register

Memory Address Offset:	54h (R/W)
On-chip Reg. Mem Addr Offset:	30154h (RO; debug path)
Default Value:	00h
Access:	see address offset above
Size:	32 bits

31	30	29	28 — 24	23 — 0
Dest Color Key En	Dest Const α Blend Enable	Always Const α Blend Enable	Reserved	Destination Color Key mask

Bit	Description
31	**Destination Color Key Enable.** 1 = destination color key is enabled 0 = destination color key is disabled
30	**Destination Constant Alpha Blend Enable** 1 = Enable Constant Alpha Blending between the overlay and the primary display when the Destination Color Key does not match and Destination Color Keying is enabled within the Alpha Blend Window. 0 = Disable Constant Alpha Blending
29	**Always Constant Alpha Blend Enable** 1 = Enable Constant Alpha Blending within the Alpha Blend Window 0 = Disable Constant Alpha Blending
28:24	**Reserved.**
23:0	**Destination Color Key Mask.** 0 = Bits that are active participants in the compare. 1 = Bits that are active participants in the compare. A mask of all ones will disable the color key (as if all colors match).

15.4.9. Overlay Source Color Key Registers

There is an overlay source key per overlay stream, which is used on a pixel basis. The Source comparison occurs after the horizontal zooming, but in the YUV formats before the color space conversion. If the source data (overlay) is within the range, then the primary display is selected.

If the overlay is in RGB mode, the most significant bits are duplicated on the least significant to form 8 bit channels:

R<4:0> -> R<4:0 ‘ 4:2>

G<5:0> -> G<5:0 ‘ 5:4>

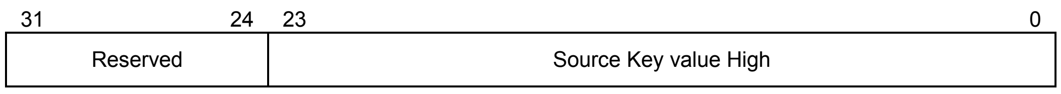
B<4:0> -> B<4:0 ‘ 4:2>

before the source key comparison is made.

Destination Color Key failing takes precedence over the Source Chroma Key failing. If both fail, then the primary display is selected.

15.4.9.1. SCLRKVH—Source Color Key Value High Register

Memory Address Offset:	58h (R/W)
On-chip Reg. Mem Addr Offset:	30158h (RO; debug path)
Default Value:	00h
Access:	see address offset above
Size:	32 bits



Bit	Description
31:24	**Reserved.**
23:0	**Source Key Value High.** This value is the high value that is compared with the overlay pixel per 8 bit channel. An overlay value greater than this field on any channel that is enabled fails the comparison and passes the overlay pixel.

15.4.9.2. SCLRKVL—Source Color Key Value Low Register

Memory Address Offset:	5Ch (R/W)
On-chip Reg. Mem Addr Offset:	3015Ch (RO; debug path)
Default Value:	00h
Access:	see address offset above
Size:	32 bits

31 : 24	23 : 0
Reserved	Source Key value Low

Bit	Description
31:24	**Reserved.**
23:0	**Source Key Value Low.** In the format of the source specifies the low end (greater than or equal) of the range of excluded source pixel data. Software sets bits that it does not want to be included within the comparison to 0. 1 = Included in comparison 0 = Not included in comparison

15.4.9.3. SCLRKM—Source Color Key Mask Register

Memory Address Offset:	60h (R/W)
On-chip Reg. Mem Addr Offset:	30160h (RO; debug path)
Default Value:	00h
Access:	see address offset above
Size:	32 bits

31	30 : 27	26 : 24
SourceConst α Blend Enable	Reserved	Source Key Mask Enables

23 : 16	15 : 8	7 : 0
Constant α Red<7:0>	Constant α Green <7:0>	Constant α Blue<7:0>

Bit	Description
31	**Source Constant Alpha Blend Enable.** 1 = Enable Source Alpha Blending when the logical OR of the Source Key Mask Enables are asserted within the Alpha Blend Window, and the comparison indicates that the overlay is to be displayed. 0 = Disable Source Alpha Blending.
30:27	**Reserved.**
26:24	**Source Key Mask Enables.** Each bit enables one channel. If the bit is a 1, the comparison result is used; otherwise, it is ignored. Bit 2 = Enables Comparison [23:16] Bit 1 = Enables Comparison [15:8] Bit 0 = Enables Comparison [7:0]
23:16	**Constant Alpha Red [7:0].** This involves three 4 x 8 bit alpha multipliers to be inserted in the Overlay and Primary Display merging logic. When the alpha blend is selected, the most significant 4 bits of each alpha channel is used as the alpha term. If the original 8 bits of the alpha channel = FF, the alpha channel value is treated as a 1. If alpha = 0, 1-0 must also be treated as a 1. This functionality only works for a primary display of 16 and 24 bits per pixel. pixel (R) = (alphaR * primary display (R)) + ((1-alphaR) * overlay (R))
15:8	**Constant Alpha Green [7:0].** This involves three 4 x 8 bit alpha multipliers to be inserted in the Overlay and Primary Display merging logic. When the alpha blend is selected, the most significant 4 bits of each alpha channel is used as the alpha term. If the original 8 bits of the alpha channel = FF, the alpha channel value is treated as a 1. If alpha = 0, 1-0 must also be treated as a 1. This functionality only works for a primary display of 16 and 24 bits per pixel. pixel (G) = (alphaG * primary display (G)) + ((1-alphaG) * overlay (G))
7:0	**Constant Alpha Blue [7:0].** This involves three 4 x 8 bit alpha multipliers to be inserted in the Overlay and Primary Display merging logic. When the alpha blend is selected, the most significant 4 bits of each alpha channel is used as the alpha term. If the original 8 bits of the alpha channel = FF, the alpha channel value is treated as a 1. If alpha = 0, 1-0 must also be treated as a 1. This functionality only works for a primary display of 16 and 24 bits per pixel. pixel (B) = (alphaB * primary display (B)) + ((1-alphaB) * overlay (B))

15.4.10. Overlay Configuration Registers

There is only 1 Overlay Configuration register which controls both overlay streams. It is read from memory with Overlay 0 register loads during Vertical Blank.

15.4.10.1. OV0CONF—Overlay Configuration Register

Memory Address Offset:	64h (R/W)
On-chip Reg. Mem Addr Offset:	30164h (RO; debug path)
Default Value:	00h
Access:	see address offset above
Size:	32 bits

31:8
Reserved

7	6	5	4	3:1	0
Reserved	Cr Cb Rdjust En (Reserved)	Reserved	YUV Conv En (Reserved)	Reserved	Line Buf Conf

Bit	Description
31:7	**Reserved.**
6	**Cr Cb Readjust Enable. (Reserved Field in Intel® 810 Chipset. Reserved for future implementations.)** 0 = Disable Cr Cb Readjust 1 = Enable Cr Cb Readjust (TV-Out mode = convert back to U,V = excess 128 and Y=Y+16)
5	**Reserved.**
4	**YUV Conversion Enable.** (**Reserved for future implementations.)** The conversion disabled setting should only be used when the overlay data is being directed out to the video side port. 0 = Enable YUV Conversion 1 = Disable YUV Conversion (TV-Out: YUV 4:4:4) mode
3:1	**Reserved.**
0	**Line Buffer Configuration.** Sets the line buffer configuration: 0 = 2 720 pixel line buffers 1 = 1 1440 pixel line buffer

15.4.11. OV0CMD—Overlay Command Register

Memory Address Offset:	68h (R/W)
On-chip Reg. Mem Addr Offset:	30168h (RO; debug path)
Default Value:	00h
Access:	see address offset above
Size:	32 bits

This register provides the data the overlay engine needs to begin work. A write to this register sets an internal bit (readable by the status) that will cause all the register values that were written to be internally latched and become active on the next VBLANK event.

31	30 28	27 25	24
Sel. Top OV (Rsvd)	Vertical Chrom Filter	Vertical Luminance Filter	Horiz Chrom Filter [24:22]

23 22	21 19	18 17	16
Horiz. Chrom Filter (cont)	Horizontal Luminance Filter	Mirroring	Y Adj

15 14	13 10	9	8
4:2:2: Byte Order	Source Format	Flip TV-Out Field Sel.	Flip Qual [8:7]

7	6	5	4	3	2 1	0
Flip Qual (Cont)	Vert. Initial Phase Sel.	Disp Flip Type	Ignore Buf and Field	Reserved	Buffer and Field	Overlay Enable

Bit	Description
31	**Select top overlay.** Reserved for future implemenations
30:28	**Vertical Chrominance Filter.** Vertical Chrominance Filter 000 = Scaling off (1:1) 001 = Line Replication 010 = Up Interpolation 011 = Reserved 100 = Reserved 101 = Pixel Dropping 110 = Down Interpolation 111 = Reserved
27:25	**Vertical Luminance Filter.** Vertical Luminance Filter 000 = Scaling off (1:1) 001 = Line Replication 010 = Up Interpolation 011 = Reserved 100 = Reserved 101 = Pixel Dropping 110 = Down Interpolation 111 = Reserved
24:22	**Horizontal Chrominance Filter.** Horizontal Chrominance Filter 000 = Scaling off (1:1) 001 = Line Replication 010 = Up Interpolation 011 = Reserved 100 = Reserved 101 = Pixel Dropping 110 = Down Interpolation 111 = Reserved
21:19	**Horizontal Luminance Filter.** Horizontal Luminance Filter also applies to RGB 000 = Scaling off (1:1) 001 = Line Replication 010 = Up Interpolation 011 = Reserved 100 = Reserved 101 = Pixel Dropping 110 = Down Interpolation 111 = Reserved
18:17	**Mirroring.** Mirroring affects the buffer address values used. See buffer address description for details. 00 = Normal 01 = Horizontal Mirroring 10 = Vertical Mirroring 11 = Both Horizontal and vertical
16	**Y adjust.** Defines the range of Y and UV values in the source data. This bit is only has an effect in YUV formats. 0 = Y 0-255 UV +-128 1 = Y 16-235 UV +-112 UV source values are always in excess 128 format

Bit	Description
15:14	**4:2:2 Byte Order.** Affects the byte order for 4:2:2 data. For other data formats these bits should be set to zero. 00 = Normal 01 = UV Swap 10 = Y Swap 11 = Y and UV swap
13:10	**Source Format.** 0000 = Reserved 0001 = Reserved 0010 = RGB 5:5:5 0011 = RGB 5:6:5 0100 = Reserved 0101 = Reserved 0110 = Reserved 0111 = Reserved 1000 = YUV 4:2:2 1001 = YUV 4:1:1 1010 = Reserved 1011 = Reserved 1100 = YUV 4:2:0 (MPEG-1 or 2) 1101 = Reserved 1110 = YUV 4:1:0 1111 = Reserved
9	**Flip TV-Out Field Select.** Selects which TV-Out field polarity for flips 0 = Between F0 and F1 1 = Between F1 and F0
8:7	**Flip Qualification.** Flip Qualification (& Display VBLANK). Flips can be caused automatically by the capture port logic on the completion of a field or frame capture. This event will be synchronized either with the display/overlay VBLANK event or the display/overlay VBLANK event combined with the current TV field. Manual flips can occur by an update of the overlay registers due to the writing of the Overlay Update Address Register when the register data specifies that the buffer/field should be changed or through a command packet that specifies a register update that changes the buffer/field. This can also be synchronized with the TV field. **Manual flip command** 00 = Flip (standard) 01 = Flip & TV-Out Field # **Automatic flipping** 10 = Capture Frame/field 11 = Capture Frame/field & TV-Out Field # 11 = is a Reserved encoding for Intel® 810 chipset
6	**Vertical Initial Phase Select.** Selects which initial vertical phase register to use. The choice is to always use the same register or alternate (for field processing) between the two registers. This bit will be overridden by the capture port when autoflipping. 0 = Use only field/buffer 0 initial vertical phase 1 = Use both initial vertical phase values

Bit	Description
5	**Display/Flip Type.** This bit affects the buffer addressing used for buffer display and the use of the initial vertical phase. Frame mode starts addressing at the value contained in the buffer address register and increments by stride as it increments from line to line. Initial phase selection is based on the buffer and the vertical Initial phase select bit. Field mode uses the field bit to determine if the start address should be the value in the start address register or the start address register plus stride. Field mode will increment the address by two times the stride as it increments from line to line. Initial phase selection is based on the field and the vertical Initial phase select. This bit will be overridden by the capture port when autoflipping. 0 = Frame Mode 1 = Field Mode
4	**Ignore Buffer and Field.** When set, don't update the buffer and field from command register. 0 = Use buffer and field data to update buffer/field 1 = Don't update buffer/field
2:1	**Buffer and field.** Selects on which buffer and field for display. This determines which buffer and field will be displayed when the overlay is enabled or when the ignore bit was clear. It would otherwise be ignored and the internal buffer/field values would be used. These are readable through the status register. 00 = Buffer 0 Field 0 01 = Buffer 0 Field 1 10 = Buffer 1 Field 0 11 = Buffer 1 Field 1
0	**Overlay Enable.** Changing this bit from a 0 to a 1 will cause the overlay to begin display after the next qualified flip event. A disable (1->0) will cause the overlay to stop displaying an image on the next display VBLANK. 0 = Disable (No display or memory fetches) 1 = Enable

15.4.12. Overlay Alpha Blend Window Position/Size Registers

These registers allow for the alpha blending of a subsection of the overlay window positioning of the overlay data relative to the graphics display or the secondary display active region. It allows pixel accurate positioning. The Overlay Alpha Blend Window must be programmed to be either equal or within the Overlay Window.

15.4.12.1. AWINPOS—Alpha Blend Window Position Register

Memory Address Offset:	70h (R/W)
On-chip Reg. Mem Addr Offset:	30170h (RO; debug path)
Default Value:	00h
Access:	see address offset above
Size:	32 bits

31 27	26 16	15 11	10 0
Reserved	Alpha Blend Vertical top line	Reserved	Alpha Blend Horizontal left pixel

Bit	Description
31:27	**Reserved.**
26:16	**Alpha Blend Vertical top line.** Determines where on the display screen coordinates the overlay display are alpha blended. Alpha Blend Vertical Top in lines 0 = begin at display first line
15:11	**Reserved.**
10:0	**Alpha Blend Horizontal left pixel.** Determines where on the display screen coordinates the overlay display are alpha blended. Alpha Blend Horizontal Left in pixels 0 = begin at display left edge

15.4.12.2. AWINSZ—Alpha Blend Window Size Register

Memory Address Offset:	74h (R/W)
On-chip Reg. Mem Addr Offset:	30174h (RO; debug path)
Default Value:	00h
Access:	see address offset above
Size:	32 bits

31 27	26 16	15 11	10 0
Reserved	Alpha Blend Vertical Size	Reserved	Alpha Blend Horizontal Size

Bit	Description
31:27	**Reserved.**
26:16	**Alpha Blend Vertial Size.** Determines where on the display screen coordinates the overlay display are alpha blended. Alpha Blend Vertical Size in lines (never specifies scan lines off the active display)
15:11	**Reserved.**
10:0	**Alpha Blend Horizontal Size.** Determines where on the display screen coordinates the overlay display are alpha blended. Alpha Blend Horizontal Size in pixels (never specifies pixels off the active display)

15.5. Overlay Flip Instruction

DWord	Bit	Description
0	31:29	**Client:** xxh
	28:23	**Function Index:**
	22:16	**Instruction Target**
1	31:0	**Register Update Address:**

Do not use the “Wait for VBLANK” mechanism to force a sequence of overlay flips. Use the “Wait for Scan Lines” mechanism with the scan line set up to be at least 1 scan line after vertical blank start to force the loading of the next Overlay x Register Update Address which will take effect after the next displayed overlay frame. For Intel® 810 chipset, the Overlay Update Address register can be loaded either before or after the “Wait for VBLANK” (primary display VBLANK). Future implementations should create a “Wait for Overlay Flip Not Pending” mechanism if this becomes an issue, especially since the Overlay may not use the primary display’s timing generator in the future.

This page is intentionally left blank.

16. Instruction, Memory, and Interrupt Control Registers

16.1. Instruction Control Registers

16.1.1. FENCE—Graphics Memory Fence Table Registers

Address Offset:	02000h – 0201Fh
Default Value:	00000000h
Access:	Read/32 bit Write only
Size:	8x 32 bits each

The Memory Hub (MH) performs address translation between linear space and tiled space. The intent of tiling is to locate graphics data that are close (in X and Y surface axes) in one memory page while still locating some amount of line oriented data sequentially in memory for display efficiency. All 3D rendering is done such that the QWords of any one span are all located in the same memory page, which improves rendering performance.

Tiled memory is supported for rendering surfaces located in graphics memory. A tiled memory surface is a surface, which has a secondary pitch and height, which are subsets of the surface's total pitch and height. The graphics controller maintains the constants required by the memory interface to perform the address translations for up to eight tiled regions (each with a different pitch and size).

The memory interface needs the surface pitch and tile height to perform the address translation. It uses the fence base address, size constants and the surface address to determine if the rendering surface is tiled and swizzle the bits as required by tiling. Since fence ranges are at least 512 KB and aligned, 7 address bits are required to specify the lower bound. In addition, the fence lower bound must be fence size aligned.

A Tile represents 2 KB of memory. Tile height is fixed at 16. Based on this the surface pitch has to be programmed in tiles. Eight Fence Table Registers occupy the address range specified above. Each Fence Table Register has the following format.

Note that X and Y major tiles are used for Host, texture, and Blitter source and destination surfaces. Display, Overlay, motion comp source, dest surfaces, color and Z surfaces if tiled must be X major.

31 26	25 19	18 16
Reserved	Fence Lower Bound	Reserved

15	14 13	12	11	10 8
Reserved	Reserved	Tile Walk	Reserved	Fence Size

7	6 4	3 1	0
Reserved	Fence Pitch	Reserved	Fence Valid

Bit	Description
31:26	**Reserved for address bits 31 downto 26**
25:19	**Fence Lower Bound :** Memory address bits 25 downto 19 (Must be size aligned)
18:15	**Reserved**
14:13	**Reserved: MBZ (“00”)**
12	**Tile walk** 0 = X Major 1= Y Major.
11	**Reserved**
10:8	**Fence size:** 000 = 512 KB 001 = 1 MB 010 = 2 MB 011 = 4 MB 100 = 8 MB 101 = 16 MB 110 = 32 MB 111 = Reserved.
7	**Reserved**
6:4	**Fence Pitch:** 000 = 1 Tile 001 = 2 Tiles 010 = 4 Tiles 011 = 8 Tiles 100 = 16 Tiles 101 = 32 Tiles 110 = Reserved 111 = Reserved
3:1	**Reserved**
0	**Fence Valid:** 0 = Unused 1 = Valid

16.1.2. PGTBL_CTL—Page Table Control Register

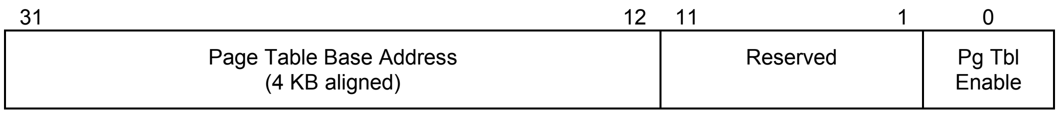
Address Offset:	02020h
Default Value:	00000000h
Access:	Read/Write
Size:	32 bits

This register enables/disables the page table mechanism and when enabled, also sets the base address of the 4 KB aligned page table.

The driver will write this register when it creates the page table at the start of virtual mode. A 64 KB physical contiguous region in memory-mapped space will be re-mapped to the page table located in local memory. Driver writes to this memory-mapped space will be offset into the page table as defined by the page table base. The physical address will be the combination of the page table base and page table offset derived from the memory mapped driver write. A translation resource access (TLB access) with the page table disabled results in an interrupt. This interrupt only occurs if there is a write through the TLB (the page table can be disabled and the display and overlay still runs with no interrupt or error reporting).

The page table can be disabled when there is no drawing engine (render-map/blitter) or high priority stream (overlay/display) active. The hardware incorporates several TLBs that cache page table entries. Disabling the page table through this register will invalidate all TLBs except the one serving display and overlay. The following table lists all the TLBs and their invalidation mechanism.

TLB	Normal Invalidation Mechanism	Note
Display	Refreshed on Vsync	Is not affected by bit 0
Overlay	Refreshed on Vsync	Is not affected by bit 0
Render/Blit	Flush	Is invalidated by page table disable
Host	Through a Page table write	Is invalidated by page table disable
Mapping	Through a Page table write	Is invalidated by page table disable
Command Stream	Through a Page table write	Is invalidated by page table disable



Bit	Description
31:12	**Page Table Base address 4 KB aligned.** Has to be within Main Memory. This is a physical address (no GTT translation).
11:1	**Reserved**
0	**Page Table Enable.** If the graphics memory range is accessed through the graphics translation table when this bit is not set asserts an interrupt event. All graphics streams other than cursor and VGA fall under this category. 0 = Disable 1 = Enable

16.1.3. PGTBL_ER—Page Table Error Register

Address Offset:	02024h (identical functionality in Device 0 at EC–EFh)
Default Value:	0000 0000h
Access:	Read Only
Size:	32 bits

This register stores information pertaining to page table error interrupts. Invalid physical address implies regions in main memory that have restricted accessibility such as PAM/SMM space etc.

The Display engine speculatively fetches data, and may cross 16 GTT mapped pages that are not intended for use. To prevent errors from occurring and hanging the display engine, the 16 GTT pages adjacent to the contiguous display region must be left unused but marked valid and mapped.

Once a page table error has been detected, the GMCH hardware will operate in an error state but continues to complete host cycles to memory in order to facilitate system debug (as opposed to hanging the system.) In the page table error detected state, further read will complete normally, whereas write will complete with all byte enables masked.

The page table error condition within the GMCH can be recovered by software writing a ‘1’ to bit 15 of the Interrupt Identity Register (IIR.) Note that this action is in addition to software clearing the page table error identification bit in the Error Identity Register (EIR[4].)

31 12	11 6	5 3	2 0
Physical Address 31:12	Reserved	Error ID	Error Type

Bit	Description
31:12	**Physical Address**. This field provides the page table access address for a page table Error Type 4, (i.e. bits 2:0 = “100”) indicating a translated address that points to PAM, SMM, and other restricted spaces in main memory. This field **is normally undefined and only registers a valid entry upon an occurrence of a page table error type 4**. Occurrence of a page table error type 4 is recorded in MMIO register 20B8h, Error Status Register, bit 4.
11:6	**Reserved**
5:3	**Error Identification:** Identifies the TLB that caused the error. After an error Render, Mapping and Blitter engines will stop execution. However, overlay, display and host operations will not stop. Each Source records the first error and ignores subsequent errors. See the Page Table Error Mask Register for more debug help. 000 = Buffer (BF) Unit 001 = Overlay 010 = Display 011 = Host 100 = Render 101 = Blitter 110 = Mapping 111 = Command Stream

Bit	Description
2:0	**Error Type:** 000 = Invalid Table 001 = Invalid Page table entry 010 = Incorrect target for Display surface (Request to lm if the surface started in mm or vice versa)/Overlay surface (Request to lm). 011 = Invalid Miss during Display/Overlay accesses 100 = Illegal translation data (translation is valid and address points to PAM, SMM, over top and other restricted spaces in main memory). 101 = Access to local memory when not present. 110 = Surface tiled in Y when not allowed (Render/Display/Overlay) 111 = Reserved

Notes:

Type 4 is reported by the main memory controller logic of the GMCH. Page Table Access done at the main memory controller is done on a QW basis. This logic area has no concept of DW accesses, i.e. the high or low DW page entry actually being requested. Error reporting for page entries pointing to invalid memory ranges is therefore done for both page entries within the QW (assuming both entries are marked valid.) If one page entry in the QW being accessed points to an illegal address, that entry is reported. If both entries in the QW being accessed point to invalid addresses, the hardware by convention reports the upper entry only.

When reporting page table error type 4 for addresses mapped to above top of physical memory, Page Table Entry:

```
31:29     Reserved              <= HW does NOT decode these bits.
28:12     Physical Address      <= HW is only sensitive to these bits.
11:3      Reserved
2:1       T1T0
0         Valid
```

This means any addresses mapped to range ≥ 512 MB and < 4 GB will be treated by the error detection logic as wrap around at the 512 MB boundary. Within the 512 MB block, if the address is at or above top of physical memory (or above top of physical memory minus TSEG range if TSEG is enabled), then a type 4 error will be generated.

16.1.4. PGTBL_ERRMSK—Page Table Error Mask Register

Address Offset:	02028h (identical functionality in Device 0 at F0–F3h)
Default Value:	0000 0000h
Access:	Read/Write
Size:	32 bits

This register is new to the Intel® 815 chipset (i.e., not in the Intel® 810 chipset)

The bits in this register mask out the corresponding page table error, preventing it from being reported in the Page Table Error Register. The Page Table Error Register can only show one page table error and associated error type at a time, and the error reported is one with the highest priority amongst all current page table errors. This Page Table Error Mask Register can be used to mask out higher priority errors, such that a lower priority error can be reported if currently asserted.

Page table errors are reported with the following priority (highest to lowest): OS, DS, host, render, blit, mapping engine, CS, and BF. If any page table error occurs, the highest priority error that is not masked with the associated bit in this Page Table Error Mask Register, will be reported in the Page Table Error Register, along with the associated error type. All lower priority page table errors will not be reported.

Procedure for determining if a particular type of low-priority page table error has occurred:

1. Bit 4 of the Error Status Register will be a 1 if any page table error has occurred.
2. Mask out all page table errors by writing 000000FFh to the Page Table Error Mask Register – this is done because the Error Status Register bit 4 will not be cleared if any page table error is asserted.
3. Unmask the target page table error by writing a 0 to the associated bit in the Page Table Error Mask Register.
4. Read bit 4 of the Error Status Register. If it is a 1, then the target page table error has occurred and the values reported in the Page Table Error Register are valid. Otherwise, the target page table error has not occurred.

31 … 9	8
Reserved	BF Error Mask

7	6	5	4	3	2	1	0
CS DMA Error Mask	OS Error Mask	DS Error Mask	Host Error Mask	Render Error Mask	TL Blit Error Mask	NC Blit Error Mask	ME Error Mask

Bit	Descriptions
31:9	**Reserved.**
8	**Buffer Unit Page Table Error Mask.** 0 = Not Masked (default) 1 = Masked
7	**Command Streamer DMA Page Table Error Mask.** 0 = Not Masked (default) 1 = Masked
6	**Overlay Page Table Error Mask.** 0 = Not Masked (default) 1 = Masked
5	**Display Page Table Error Mask.** 0 = Not Masked (default) 1 = Masked
4	**Host Page Table Error Mask.** 0 = Not Masked (default) 1 = Masked
3	**Render Operation Page Table Error Mask.** 0 = Not Masked (default) 1 = Masked
2	**Destination Blit Operation Page Table Error Mask.** 0 = Not Masked (default) 1 = Masked
1	**Source Blit Operation Page Table Error Mask.** 0 = Not Masked (default) 1 = Masked
0	**Mapping Engine Operation Page Table Error Mask.** 0 = Not Masked (default) 1 = Masked

16.1.5. RINGBUF—Ring Buffer Registers

Address Offset:	02030h – 0207Fh 02030h – 0203Fh: Low Priority Ring 02040h – 0204Fh: Interrupt Ring 02050h – 0205Fh: Reserved 02060h – 0207Fh: Reserved
Default Value:	00000000h
Access:	Read/32 bit Write Only
Size:	4 DWords

Each Ring buffer is defined by a four DWord register set, which includes starting address, length, head pointer, and tail pointer. The ring buffer can be disabled when empty. A driver uses two sets of ring buffers, low priority and interrupt.

Instruction ring buffers must be located in main memory. There is no hardware support for ring buffers that are mapped to local memory. There is neither a hardware error detection for intentional or inadvertent attempts to map the ring buffers to local memory. This requirement applies to both low priority and interrupt rings.

Intel® 810 Chipset and Intel®815 Chipset Errata

Address Offset: 2030h–207Fh

DWord Offset 0, 1, 2, and 3

Bit 2:1 Automatic Report Head pointer. Autoreport happens on a 64K or 128K boundary. When the head pointer **crosses the ring buffer size** on an autoreport boundary, the hardware erroneously reports the head pointer address == ring buffer size, as opposed to wrapping around and reporting 0. **This is an Intel® 810 chipset** and **Intel® 815** chipset **Silicon Errata**. Note that the size of the ring buffer is not restricted to a power of two.

A software work-around can be implemented as follows:

The reported head pointer should be bit-wise AND'd by the size of the ring buffer minus 1B.

For example,

ring buffer size = 128KB	
HW reported head pointer	= bbbbbbbb bbbbbbbb bbbbbbbb bbbbbbbb B
Mask (128KB-1B)	= 00000000 00000001 11111111 11111111 B
Corrected head pointer	= 00000000 0000000b bbbbbbbb bbbbbbbb B

Note that this error only occurs in autoreport head mode. There is not a problem with the instruction parser command GFXCMDPARSER_REPORT_HEAD.

Dword Offset	Bit	Description
0	31:21	**Reserved.**
	20:3	**Tail Pointer :** Programmable Qword Offset in the ring buffer (20:3 is used by the hardware).
	2:0	**Reserved.**
1	31:2	**Head pointer:** Hardware maintained DWord Offset in the ring buffer (20:2 is used by the hardware, Bits 31:21 are incremented whenever the head pointer wraps from the end to the start of the ring buffer <> Wrap Count).
	1:0	**Reserved.**
2	31:26	**Reserved**
	25:12	**Starting Address:** Programmable 4 KB page aligned address of the buffer. This is a physical address (no GTT translation).
	11:0	**Reserved**
3	31:21	**Reserved**
	20:12	**Buffer Length:** Programmable length of the ring buffer in 4 KB Pages (Maximum = 2 MB, x000 = One 4 KB Page)
	11:3	**Reserved**
	2:1	**Automatic Report Head pointer:** Report happens when the Ring DMA crosses 64 KB or 128 KB boundary. X0 = No report 01 = Report every 16 pages (64 KB) 11 = Report every 32 pages (128 KB).
	0	**Ring Buffer Valid:** 0 = Disabled 1 = Enabled.

16.1.6. HWS_PGA—Hardware Status Page Address Register

Address Offset:	02080h
Default Value:	1FFFF000h.
Access:	Read/Write
Size:	32 bits

Hardware status page physical address. The programmed address should be 4 KB aligned. Bits [11:0] are hardwired to 0. This Page will be used to report hardware status into system memory as follows.

Note: For the GMCH bits 31:29 must be “0”.

31 29	28 12	11 0
Must be 0s	Base Physical Address of Status Page [28:12]	0s (hardwired)

DWord Offset	Description
0	**Report Interrupt Status Register.**
1	**Report Low Priority Ring Head Pointer.**
2	**Report Interrupt Ring Head Pointer.**
3	**Reserved for other ring.**
(1K-1): 4	Can be Written through indexed store DWord instruction.

16.1.7. IPEIR—Instruction Parser Error Identification Register (debug)

Address Offset:	02088h
Default Value:	0000h
Access:	Read Only
Size:	32 bits

This register is used to help identify the instruction packet that generates an invalid instruction interrupt to the processor. The IPEIR will contain the origin of the offending packet. The Header will be stored in the error header register.

31 … 3	2	1 … 0
Reserved	BATCH / RING	RING ID

Bit	Description
31:3	Reserved
2	**BATCH / RING.** The invalid instruction came from either a batch buffer or instruction ring. 1 = Batch buffer 0 = Instruction RING
1:0	**RING ID.** The invalid instruction came from the Low Priority (00) or the Interrupt (01) ring. If the invalid instruction came from a batch buffer, this bit identifies which Instruction Ring the batch buffer instruction came from. 00 = Low Priority 01 = Intr ring 1X = Reserved

16.1.8. IPEHR—Instruction Parser Error Header Register (debug)

Address Offset:	0208Ch
Default Value:	0000h
Access:	Read Only
Size:	32 bits

This register is used to identify the instruction packet that generates an invalid instruction interrupt to the processor. The IPEHR will contain the header word of the offending packet. For debug purposes, the header of all instructions parsed will be written to this register. If an error occurs, the instruction parser halts. An interrupt indicating an error will be generated if it is unmasked.

Bit	Description
31:0	**Header.** This field will contain the instruction Header field of the instruction packet that generates an invalid instruction interrupt.

16.1.9. INSTDONE—Instruction Stream Interface Done Register

Address Offset:	02090h
Default Value:	FFFF FFFFh
Access:	Read only.
Size:	32 bits

This read only register reports engine done signals.

31–24
Reserved

23–19	18	17	16
Reserved	PC_Done	WM_Done	IT_Done

15	14	13	12	11	10	9	8
CC_DONE	MG_DON E	DG_DON E	QCC_DO NE)	FTCH_DO NE	MEC_DO NE	MECO_D ONE	CCMC_D ONE

7	6	5	4	3	2	1	0
Reserved	Blitter Done	Mapping Eng. Done	Render Eng. Done	Batch Done	Reserved	Intr. Ring Empty	Low Prior. Ring Emp.

Bit	Description
32:19	Reserved
18	Plane Converter Done (PC_DONE)
17	Window Mask Done (WM_DONE)
16	Interpolator Done (IT_DONE)
15	Color Calculator Cone (CC_DONE)
14	Mapping Engine Address Generator Done (MG_DONE)
13	Mapping Engine Dependency Address Done (DG_DONE)
12	Mapping Engine Cache Controller Done (QCC_DONE)
11	Mapping Fetch Unit Done (FTCH_DONE)
10	Mapping Engine Cache Done (MEC_DONE)
9	Mapping Engine Cache Output Done (MECO_DONE)
8	CCMC_DONE
7	Reserved.
6	Blitter Done.
5	Mapping Engine Done

Bit	Description
4	Render Engine Done
3	Batch Done
2	Reserved
1	Intr. Ring Empty or Disabled
0	Low Priority Ring Empty or Disabled

16.1.10. NOPID—NOP Identification Register

Address Offset:	02094h
Default Value:	00000000h
Access:	Read Only
Size:	32 bits

This register contains the value specified by the last GFXCMDPARSER_NOP_IDENTIFICATION instruction received.

31 22	21 0
Reserved	ID Number

Bit	Description
31:22	**Reserved**
21:0	**Identification Number.** Bits [5:0] of this identification number must be zero.

16.1.11. INSTPM—Instruction Parser Mode Register

Address Offset:	020C0h
Default Value:	00h
Access:	Read/Write
Size:	8 bits

The bits in this register control the operation of the Instruction Parser.

Note: If an instruction type is disabled, the parser will read it out of the instruction / batch FIFO but will not send it to its destination. Error checking will be performed.

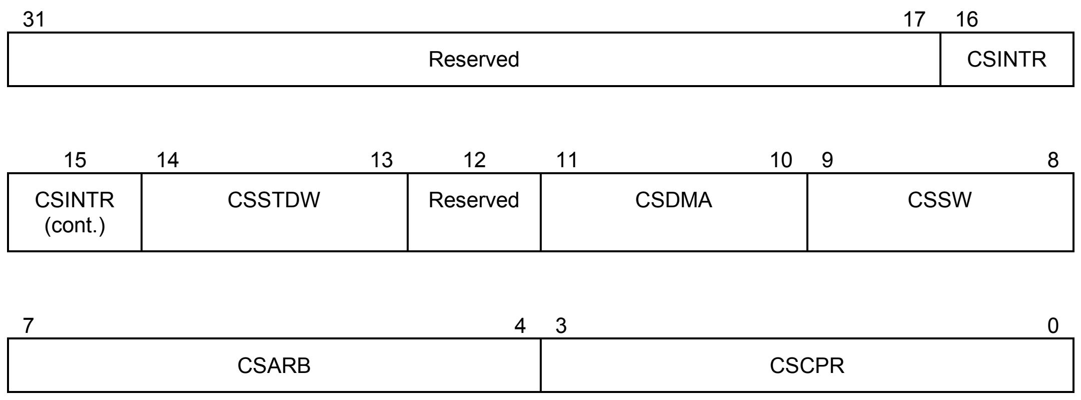
7	6	5	4	3	2	1	0
Reserved	Enable Sync Packet AGP Flush (Rsvd)	Enable Sync Packet Flush	Disable Mcomp Instrs	Disable GDI Blitter Instrs	Disable Render (3D/Stretc h) Instrs	Disable State Variable Updates	Disable Render Palette Updates

Bit	Description
7	**Reserved**
6	**Enable Sync AGP Flush.** Enable pipe and AGP flush. Set by software and cleared by the parser on detecting graphics pipe flushed before parsing a subsequent packet. 1 = Enable 0 = Cleared by graphics controller
5	**Enable Sync flush.** Enable pipe flush. Set by software and cleared by the parser on detecting graphics pipe flushed before parsing a subsequent packet. 1 = Enable 0 = Cleared by graphics controller
4	**Disable Mcomp Instructions.** Disable processing of mcomp instructions by parser. 1 = Disable 0 = Enable
3	**Disable GDI Blitter Instructions.** Disable processing of Blitter instructions 1 = Disable 0 = Enable
2	**Disable Render (3D/Stretch) Instructions.** Disable processing of 3D instructions (Client 00h) by parser. 1 = Disable 0 = Enable
1	**Disable State Variable Updates.** 1 = Disable 0 = Enable
0	**Disable Render Palette Updates.** 1 = Disable 0 = Enable

16.1.12. INSTPS—Instruction Parser State Register (debug)

Address Offset:	020C4h
Default Value:	0000h
Access:	Read Only
Size:	32 bits

This register contains the state code of the Instruction Parser in the CSI. Decoding the contents of this register will indicate what the Instruction Parser is currently doing.



Bit	Description
31:17	**Reserved**
16:15	**CSINTR State Machine:** Is responsible for the working the CS-PI Interrupt cycle generation interface. 00 = **INTRIDLE:** This is the reset state. It is also the idle state. The state machine comes back to this state on getting acknolwedged in either of the two request states. 01 = **INTREQON:** In this state the command stream turns on the interrupt flag to the HubLink unit and requests service 11 = **INTREQOFF:** In this state the command stream turns off the interrupt flag to the HubLink unit and requests service
14:13	**CSSTDW State Machine:** Is responsible for generation of the Store DWord snoop cycle to the PI unit on behalf of the command parser and the interrupt report meachnism. 000 = **STDWIDLE:** Waiting for next Store DWord Cycle 001 = **STDWINTR:** INT Report Store DWord Cycle in progress 010 = **STDWCP:** Command Parser Initiated Store DWord Cycle 100 = **STDWAUTO:** Auto report of head pointer 011 = **STDWINTD:** Request accepted, wait for HT to take data 101 = **STDWCPD:** Request accepted, wait for HT to take data 110 = **STDWAUTD:** Request accepted, wait for HT to take data
12	**Reserved**

Bit	Description
11:10	**CSDMA State Machine:** Is responsible for the control of the DMA FIFO. It get requests from the arbitration state machine. It manages the FIFO so that requests are only made when there is space in the FIFO. 00 = **DMAIDLE:** Reset State - No DMA Active 01 = **DMAREQ:** Request State – Generate request 10 = **DMAWT:** Wait State - Stay in this state till space becomes available 11 = **DMARWT:** Wait for Acknowledge State - Stay in this state till acknowledged. Hold request
9:8	**CSSW State Machine:** Main Arbiter between Low Priority Ring, Interrupt Priority Ring and Batch Buffer. 00 = **SWIDLE:** Reset State 01 = **SWBLOCK:** Assert Parser Block 10 = **SWSTOP:** Once Parser has gone to idle, request the dma engine to stop 11 = **SWPOP:** Once the DMA has gone to idle, pop the FIFO till the CSDMA indicates DMA done
7:4	**CSARB State Machine:** Blocks the Command Parser from parsing further and cleans up the FIFO. 0000 = **ARBIDLE:** Reset State – Waiting for next arbitration request 0001 = **ARBLOW:** Low Priority Ring Active 0010 = **ARBL2ISW:** Switch from Low Priority Ring to High Priority Ring. Initiated by the parser (by masking the valid bit) 0011 = **ARBLBAT:** In Batch Buffer initiated from Low Priority Ring 0100 = **ARBL2BSW:** Switch from Low Priority Ring to Batch Buffer. Initiated by the parser 0101 = **ARBINTR:** Interrupt Active Ring Active 0110 = **ARBI2BSW:** Switch from Interrupt Ring to Batch Buffer. Initiated by the parser 0111 = **ARBIBAT:** In Batch Buffer initiated from Interrupt Priority Ring 1000 = **ARBI2DSW:** Switch from Interrupt Ring to Idle, if a wait for event packet was received (parser masks the valid bit)
3:0	**CSCPR State Machine:** Command Parser. 0000 = **CPRIDLE:** Reset State. Parser is in Idle looking at the next header 0001 = **CPRSW:** When waiting for engines to go idle either for non-pipelined SVs or for engine switch 0010 = **CPRHDR:** Clock to get rid of header when not loaded externally 0011 = **CPRCMD:** Command State that have data/address to load 0100 = **CPREVT:** For some Parser events like WT4EVT packet, brekapoint interrupt 0101 = **CPRFLSH:** If all engines are done, request for Local Cache to flush itself 0110 = **CPRSTDA:** Store DWord Address (report head command also) 0111 = **CPRSTDW:** Store DWord Data 1000 = **CPRHLD1:** Hold address/data for external interface. For internal interface just mimic external interface 1001 = **CPRHLD2:** Hold Address and Data for one more clock 1010 = **CPRWT:** Wait State 1011 = **CPRPOPA:** If fast load 1100 = **CPRPOPB:** If fast load

16.1.13. BBP_PTR—Batch Buffer Parser Pointer Register (debug)

Address Offset:	020C8h
Default Value:	00000000h
Access:	Read Only
Size:	32 bits

This register contains the offset from the batch buffer start address of the DWord being parsed by the Instruction Parser.

31 19	18 2	1 0
Reserved	Address Offset	Reserved

Bit	Description
31:19	**Reserved**
18:2	**Batch Buffer Address Pointer Offset.**
1:0	**Reserved**

16.1.14. ABB_STR—Active Batch Buffer Start Address Register (debug)

Address Offset:	020CCh
Default Value:	00000000h
Access:	Read Only
Size:	32 bits

This register is loaded with the start address of the Batch Buffer request.

The ABB_STR and ABB_END Registers will not get loaded if they are popped off of the stack. This can occur if a low priority ring batch buffer is interrupted at a chain point by Interrupt Priority ring execution and then is later continued. The Start and End addresses for the low priority ring chain portion that was interrupted will not be stored in the ABB_STR and ABB_END Registers. This is an operational anomaly that will not be corrected.

31 26	25 3	2	1 0
Reserved	Batch Buffer Request Start Address	Reserved	Source of the Batch buffer

Bit	Description
31:26	**Reserved**
25:3	**Batch Buffer Request Start Address.**
2	**Reserved**
1:0	**Source of the Batch Buffer.** 00 = Low Priority Ring 01 = Interrrupt Ring 1X = Reserved

16.1.15. ABB_END—Active Batch Buffer End Address Register (debug)

Address Offset:	020D0h
Default Value:	00000000h
Access:	Read Only
Size:	32 bits

This register is loaded with the end address of the Batch Buffer request. The ABB_STR and ABB_END Registers will not get loaded if they are popped off of the stack. This can occur if a low priority ring batch buffer is interrupted at a chain point by Interrupt Priority ring execution and then is later continued. The Start and End addresses for the low priority ring chain portion that was interrupted will not be stored in the ABB_STR and ABB_END Registers. This is an operational anomaly that will not be corrected.

31 26	25 3	2	1 0
Reserved	Batch Buffer Request End Address	Reserved	Source of the Batch buffer

Bit	Description
31:26	**Reserved**
25:3	**Batch Buffer Request End Address**
2	**Reserved**
1:0	**Source of the Batch Buffer** 00 = Low Priority Ring 01 = Interrrupt Ring 1X = Reserved

16.1.16. DMA_FADD—DMA Engine Fetch Address (debug)

Address Offset:	020D4h
Default Value:	00000000h
Access:	Read Only
Size:	32 bits

This register contains the offset from the start address of the instruction being fetched by the DMA engine.

31 26	25 3	2	1 0
Reserved	Current Address of the DMA Pointer	Reserved	User of DMA Eng

Bit	Description
31:26	**Reserved.**
25:3	**Current DMA Address.**
2	**Reserved**
1:0	**User of the DMA Engine.** 00 = Low Priority Ring 01 = Interrupt Ring 10 = Batch Buffer from Low Priority Ring 11 = Batch Buffer from Interrupt Ring

16.1.17. MEM_MODE—Memory Interface Mode Register (debug)

Address Offset:	020DCh
Default Value:	00000000h
Access:	Read/Write
Size:	32 bits

15:8
Reserved

7:6	5	4:3	2	1	0
Other Local Cache Modes	Host Graphics Prefetch Enable	Reserved	Reserved	Graphics Address Translatio n Mode (Pg Tbl test mode)	Enable Instruction FIFO Debug Mode

Bit	Description
31:6	**Reserved.**
5	**Host Graphics Prefetch Mode,** 0 = Enable 1 = Disable
4:3	**Reserved**
2	**Reserved.** MUST BE SET TO 1
1	**Graphics Address Translation Mode.** 0 = Graphics Address translated through page table (Page Table test mode) 1 = Graphics address = Physical address in Local memory. (Not supported in the GMCH)
0	**Instruction FIFO Debug Mode.** 0 = Disable 1 = Enable

16.2. Interrupt Control Registers

The interrupt control registers described below all share the same bit definition. The bit definition is as follows:

15	14 13	12	11	10	9	8
HW Detect Error Master	Reserved	Sync Status Toggle	Pri Dply Flip Pending	Reserved	Overlay 0 Flip Pending	Rsvd

7	6	5	4	3	2	1	0
Pri Dply VBLANK.	Pri Dply Event	Reserved	Reserved	Reserved	Reserved	User Defined Interrupt	Breakpoint

Table 17. Bit Definition For Interrupt Control Registers

Bit	Description
15	**Hardware Detected Error Master.** When this status bit is set, it indicates that the hardware has detected an error. Set on an error condition and cleared by a processor write of a one to the appropriate bit contained in the error id register followed by a write of a one to this bit in the IIR. Further information on the source of the error comes from the “Error Status Register” which along with the “Error Mask Register” determine which error conditions will cause the error status bit and the interrupt to occur. The intent is to use the error bits to detect errors during debug and testing. Error conditions during normal operation should not occur. • **MM/LM Refresh timer error:** Indicates a refresh request buffer overrun. • **Page Table Error:** Indicates a page table error. • **Display or Overlay under run:** Set on either a display or overlay under run error. See display and overlay status registers to determine source of the error. • **Instruction Parser Error**: The Instruction Parser encounters an error while parsing an instruction.
14:13	**Reserved**
12	**Sync Status Toggle.** This bit is toggled when the instruction parser completes a flush with the sync enable bit active in the instruction parser mode register. The toggle event will happen after all the graphics engines are flushed. The store dword resulting from this toggle will cause the processor’s view of graphics memory to be coherent as well (invalidate the host graphics cache).
11	**Primary Display Flip Pending.** Status bit is set on a pending flip and cleared when the flip occurs, whereas IIR reflects Flip-Occurred# (which is contrary to the general definition of setting of IIR bits when interrupts occur). This is only used when the GFXCMDPARSER_FRONT_BUFFER _INFO packet is being used. See that instruction for additional information. To prevent race conditions, the status write occur before the STOREDWORD following the flip packet is written.
10	**Reserved**

Bit	Description
9	**Overlay 0 Flip Pending.** Status bit is set to reflect a pending flip when the parser parses a flip packet and cleared when the flip takes place (Display VBLANK), whereas IIR reflects Flip-Occurred# (which is contrary to the general definition of setting of IIR bits when interrupts occur). This is only affected by the use of flip packets not through the manual method or the capture auto flipping. To prevent race conditions, status register write must occur before the STOREDWORD following the flip packet is written.
8	**Reserved**
7	**Primary Display VBLANK.** Set at leading edge of display VBLANK. This is actually delayed to allow all internal hardware VBLANK events to occur before the interrupt is generated to eliminate race conditions. These events include the update of the display and overlay status bits and loading of the overlay registers.
6	**Primary Display Event.** Interrupt cause will be determined by reading the display status register and is one of the following: • Flat Panel Hot Plug Detect Interrupt • Display VSYNC • Display Line Compare On active going edge of OR of unmasked Display event bits Status - OR of unmasked Display event bits Note that the display line compare is also used through the instruction parser packet interface.
5	**Reserved**
4	**Reserved**
3	**Reserved**
2	**Reserved**
1	**User Defined Interrupt.** The Instruction Parser passed a “user-defined interrupt” packet. This is intended to be used with another mechanism (e.g., STOREDW instruction) to determine the source of the packet.
0	**Breakpoint.** The Instruction Parser parsed a “breakpoint” interrupt packet. The GFXCMDPARSER_BREAKPOINT_INTERRUPT packet can be used to generate a store dword cycle from the command streamer to main memory and halt the parsing of further commands. The HWSTAM (Hardware Status Mask Register- Offset 02098h) must have the break point bit unmasked to generate the store dword. In addition to the HWSTAM register, the corresponding bit in the IMR (Interrupt Mask Register - Offset 020A8h) must also be unmasked to halt the parsing of further commands. At the break point command packet , if both the HWSTAM and IMR are unmasked, the command parser will stop processing further commands until the breakpoint bit in the IIR (Interrupt Identity Register - Offset 020A4h) is cleared.

16.2.1. HWSTAM—Hardware Status Mask Register

Address Offset:	02098h
Default Value:	FFFFh
Access:	Read/Write
Size:	16 bits

This register has the same format as the Interrupt Control Registers. The corresponding bits are the mask bits that prevent that bit in the Interrupt Status Register from generating a PCI write cycle. Any unmasked interrupt bit (set to 0) will allow the Interrupt Status Register to be written to the address specified by the Hardware Status Vector Address Register when the Interrupt Status Register changes state.

15	14 13	12	11	10	9	8
HW Detect Error Master	Reserved	Sync Status Toggle	Pri Dply Flip Pending	Reserved	Overlay 0 Flip Pending	Reserved

7	6	5	4	3	2	1	0
Pri Dply VBLANK.	Pri Dply Event	Reserved	Reserved	Reserved	Reserved	User Defined Interrupt	Breakpoint

Bit	Description
15:0	**Interrupt Status Mask Bits**. 0 = Not Masked. 1 = Masked (prevents PCI write cycle.).

16.2.2. IER—Interrupt Enable Register

Address Offset:	020A0h
Default Value:	0000h
Access:	Read/Write
Size:	16 bits

Individual enables for each interrupt described above. A disabled interrupt will still appear in the Interrupt Identity Register to allow polling of interrupt sources.

15	14	13	12	11	10	9	8
HW Detect Error Master	Reserved		Sync Status Toggle	Pri Dply Flip Pending	Sec Dply Flip Pending (Rsvd in GMCH)	Overlay 0 Flip Pending	Overlay 1 Flip Pending (Rsvd in GMCH)

7	6	5	4	3	2	1	0
Pri Dply VBLANK.	Pri Dply Event	Reserved	Reserved	Reserved	Reserved	User Defined Interrupt	Breakpoint

Bit	Description
15:0	**Interrupt Enables.** (See Table 17.) 1 = Enable. 0 = Disable.

16.2.3. IIR—Interrupt Identity Register

Address Offset:	020A4h
Default Value:	0000h
Access:	Read/Write Clear
Size:	16 bits

The individual interrupt(s), which occurred, are determined via this register. The bit is set by the interrupt event and held until cleared by writing a '1' into the bit position.

15	14 13	12	11	10	9	8
HW Detect Error Master	Reserved	Sync Status Toggle	Pri Dply Flip Pending	Reserved	Overlay 0 Flip Pending	Reserved

7	6	5	4	3	2	1	0
Pri Dply VBLANK.	Pri Dply Event	Reserved	Reserved	Reserved	Reserved	User Defined Interrupt	Breakpoint

Bit	Description
15:0	**Interrupt Identity.** See. Table 17 1 = Interrupt occurred.

16.2.4. IMR—Interrupt Mask Register

Address Offset:	020A8h
Default Value:	FFFFh
Access:	Read/Write
Size:	16 bits

An interrupt that is masked by this register will not appear in the Interrupt Identity Register and will not generate an interrupt.

15	14 13	12	11	10	9	8
HW Detect Error Master	Reserved	Sync Status Toggle	Pri Dply Flip Pending	Reserved	Overlay 0 Flip Pending	Reserved

7	6	5	4	3	2	1	0
Pri Dply VBLANK.	Pri Dply Event	Reserved	Reserved	Reserved	Reserved	User Defined Interrupt	Breakpoint

Bit	Description
15:0	**Interrupt Mask Bits.** See. Table 17 0 = Not Masked 1 = Masked

16.2.5. ISR—Interrupt Status Register

Address Offset:	020ACh
Default Value:	0100h (probably still not quite correct value)
Access:	Read Only
Size:	16 bits

This register contains the non-persistent value of the signals causing each interrupt. These bits are not masked by the Interrupt Mask Register. The user interrupt and the breakpoint interrupt last for one clock pulse. The corresponding bits in this register will serve no practical purpose due to the short duration of the signal.

15	14 13	12	11	10	9	8
HW Detect Error Master	Reserved	Sync Status Toggle	Pri Dply Flip Pending	Reserved	Overlay 0 Flip Pending	Reserved

7	6	5	4	3	2	1	0
Pri Dply VBLANK.	Pri Dply Event	Reserved	Reserved	Reserved	Reserved	User Defined Interrupt	Breakpoint

Bit	Description
15:0	**Interrupt Status.** See Table 17. 1 = Signal caused interrupt.

16.2.6. Error Identity, Mask and Status Registers

The Error Identity, Mask, and Status registers have the following bit descriptions. The master error bit in the ISR and IIR register will be set when the OR of the unmasked (with a zero in the corresponding mask register bits) error status bits is true.

16.2.6.1. Page Table Error handling in Intel® 815 Chipset

Page table errors can be caused by accessing graphics aperture space when:

1. The page table is not enabled.
2. Attempting to access local memory in a UMA system.
3. Attempting to access a page table entry, which does not have the valid bit set.
4. The location of the page table is in restricted region (e.g., SMM space) in memory.

For cycles initiated from the graphics host, TLB error should not cause the system to hang.

TLB error is flagged **only** for a write. For the write cycle with the TLB error and for all subsequent write cycles, byte enables are masked. If local memory is accessed in UMA system, the cycle is forwarded to system memory and completed with masked byte enables. For a write TLB error, error registers should be appropriately set.

Graphics memory reads do not cause any side-effects. Hence, all reads with a TLB error are allowed to complete. Note that no error condition will be set in the Error registers for a read TLB error.

Irrespective of the cycle being a read or a write, if the location of page table is in a restricted region, a page table error will be set. This error is not resettable.

16.2.6.2. Resetting the Page Table Error

The page table error will be reset every time a write cycle is generated to bit 15 of the Interrupt Identity register (IIR), independent of the setting of the bit in the IMR or the IER. Resetting the page table error should cause the subsequent write cycles to be completed without masking of the byte enables. Note that a TLB error due to locating a page in system memory can never be cleared.

15 6	5	4	3	2	1	0
Reserved	MM/LM Refresh timer error	Page Table Error	Display or Overlay under run	Reserved	Reserved	Instruction Parser Error

Bit	Description
15:6	**Reserved**
5	**MM/LM Refresh Timer Error:** 1 = Indicates a refresh overrun.
4	**Page Table Error:** 1 = Page table error 0 = Cleared by a processor write of a one to the error identity bit.
3	**Display or Overlay Under run:** 1 = Display or overlay under run error 0 = Cleared by a processor write of a one to the Error identity bit. See display and overlay status registers to determine source of the error.
2	**Reserved**
1	**Reserved**
0	**Instruction Parser Error**: The Instruction Parser encounters an error while parsing an instruction. 1 = Error 0 = Cleared by a processor write of a one to the Error identity bit.

16.2.6.3. EIR—Error Identity Register

Address Offset:	020B0h
Default Value:	00h
Access:	Read/Write Clear
Size:	16 bits

15 6	5	4	3	2	1	0
Reserved	MM/LM Refresh timer error	Page Table Error	Display or Overlay under run	Reserved	Reserved	Instruction Parser Error

Bit	Description
15:0	**Error Identity Bits.** (See Error Identity, Mask, and Status Register Bit Definitions table.) 1 = Error occurred

16.2.6.4. EMR—Error Mask Register

Address Offset:	020B4h
Default Value:	FFh
Access:	Read/Write
Size:	16 bits

15 6	5	4	3	2	1	0
Reserved	MM/LM Refresh timer error	Page Table Error	Display or Overlay under run	Reserved	Reserved	Instruction Parser Error

Bit	Description
15:0	**Error Mask Bits.** (See Error Identity, Mask, and Status Register Bit Definitions table.) 0 = Not masked 1 = Masked

16.2.6.5. ESR—Error Status Register

Address Offset:	020B8h
Default Value:	FFh
Access:	Read Only
Size:	16 bits

15 6	5	4	3	2	1	0
Reserved	MM/LM Refresh timer error	Page Table Error	Display or Overlay under run	Reserved	Reserved	Instruction Parser Error

Bit	Description
15:0	**Error Status Bits.** (See Error Identity, Mask, and Status Register Bit Definitions table.)

16.3. Display Interface Control

16.3.1. FW_BLC—FIFO Watermark and Burst Length Control

Address Offset :	020D8h
Default Value:	22 31 73 17h
Access:	Read/Write
Size:	32 bits

These control values only apply to HIRes modes of operation. VGA modes ignore the settings of these registers in favor of fixed values.

For VGA Text mode, character buffer fetches are performed without regard to the space available in the FIFO (since this data is stored in the character buffer, not the FIFO). Character buffer fetches are performed as a single request of 8 QWs. Font data is fetched one QW at a time, and will begin when the FIFO has room for 8 character font QWs. VGA Graphics modes will perform requests one at a time so long as there is room for 1 QW in the FIFO.

1. FIFOs refer to ALL FIFOs in the DSI data path (i.e. The total FIFO space available is the sum of the DSI FIFO depth and the Display Engine FIFO depth). Currently, this depth is 48 QWs.
2. The h/w default is an invalid value: these quantities should never be programmed to zeros.
3. The hardware depends on these registers being set properly since it is possible to set the request length and watermarks to states which would cause the overflow of the sync FIFO. For example, assume a watermark is set to 33 QW and the request length is set to 32 QWs. When the first two requests have been completed, 64 QWs will have been written into the sync FIFO. During this time, only 16 QWs will be drained out of the FIFO to be written to the Display Engine FIFO. Since the sync FIFO in the DSI is only 32 QWs deep, this will result in (64-16-32) = 16 QW overflow of the FIFO.

31 28	27 24
Overlay Delay Timer1	Overlay Delay Timer0

23	22 20	19 18	17 12	11	10 8
Rsvd	MM Dply Burst Length	Reserved	MM Display FIFO Watermark	Rsvd	LM Display Burst Length

7 6	5 0
Overlay Watermark	Reserved

Bit	Description
31:28	**Overlay Delay Timer1.** Is used to insert waits states in between sets of YUVY requests to MM. The value in this register is multiplied by 16 to determine the wait state clock count.
27:24	**Overlay Delay Timer0.** Is used to insert waits states in between any two overlay streamer requests to MM except between sets of YUVY. The value in this register is multiplied by 16 to determine the wait state clock count.
23	**Reserved**
22:20	**MM Display Burst Length.** Size in QWs of individual requests issued to Memory. Use Multiples of 16 QWs for tiled memory. 000 = 8 Qws (N/A if Trickle Feed is on) 001 = 16 Qws, 010 = 24 Qws 011 = 32 Qws 100 = 40 Qws 101 = 48 Qws 110 = 56 Qws 111 = 64 QWs
19:18	**Reserved**
17:12	**MM Display FIFO Watermark.** Number of QWs stored in FIFOs, below which the DSI will generate requests to LMI (Value has to less than 32 and should be as recommended in the high priority bandwidth analysis spreadsheet).
11	**Reserved**
10:8	**LM Display Burst Length (Reserved in Intel® 815 chipset).** Size in QWs of individual requests issued to Memory. Use Multiples of 16 QWs for tiled memory. 000 = 8 QWs (N/A if Trickle Feed is on) 001 = 16 QWs 010 = 24 QWs 011 = 32 QWs 100 = 40 QWs 101 = 48 QWs 110 = 56 QWs 111 = 64 QWs.
7:6	**Overlay Watermark.** Number of QWs stored in the overlay FIFO below which a new overlay request will be generated. 00 = 24 QWs (default) 01 = 20 QWs 10 = 16 QWs 11 = 12 QWs
5:0	**LM Display FIFO Watermark (Reserved in Intel® 815 chipset).** Number of QWs stored in FIFOs, below which the DSI will generate requests to LMI (Value has to less than 32 and should be as recommended in the high priority bandwidth analysis spreadsheet).

17. LCD / TV-Out Register Description

During LCD or TV-Out mode, the BIOS will program the following LCD / TV-Out registers. These registers are 32-bit memory mapped. These registers are not double buffered and take effect when loaded. Further, this subsystem takes into account modified CR register values during vertical blank time for centering.

This subsystem allows the timing generator to be programmed to pixel granularity. The only exception is during VGA pixel doubling mode. During VGA pixel doubling, active pixel time must be a multiple of 4 pixels to account for centering with VGA pixel doubling and non-active times must be a multiple of 2 pixels clocks.

All fields are excess-0 encoded. This means that the hardware uses the value+1, where value is the entry in the field. Therefore if a 0 is programmed into a field the hardware uses the value 1 for that field.

17.1. HTOTAL—Horizontal Total Register

Address Offset:	60000h
Default Value:	00000000h
Access:	Read/Write
Size:	32 bits

31 28	27 16
Reserved	Horizontal Total Display Pixels

15 11	10 0
Reserved	Horizontal Active Display Pixels

Bit	Description
31:28	**Reserved.**
27:16	**Horizontal Total Display Pixels.** This 12-bit field provides a horizontal total up to 4096 pixels encompassing 2048 active display pixels, front/back border pixels and horizontal retrace period. Any pending event (HSYNC, VSYNC) is reset at htotal.
15:11	**Reserved.**
10:0	**Horizontal Active Display Pixels.** This 11-bit field provides horizontal active display resolutions up to 2048 pixels. Note that the first horizontal active display pixel always starts at 0.

17.2. HBLANK—Horizontal Blank Register

Address Offset:	60004h
Default Value:	00000000h
Access:	Read/Write
Size:	32 bits

31 28	27 16
Reserved	Horizontal Blank End

15 12	11 0
Reserved	Horizontal Blank Start

Bit	Description
31:28	**Reserved.**
27:16	**Horizontal Blank End.** Horizontal blank end expressed in terms of absolute pixel number relative to the horizontal active display start. Note: An asserted HBlank will be deasserted when HTotal occurs irrespective of what is programmed in HBlank end
15:12	**Reserved.**
11:0	**Horizontal Blank Start.** Horizontal blank start expressed in terms of absolute pixel number relative to the horizontal active display start.

17.3. HSYNC—Horizontal Sync Register

Address Offset:	60008h
Default Value:	00000000h
Access:	Read/Write
Size:	32 bits

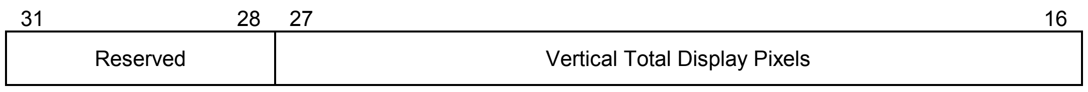
31 28	27 16
Reserved	Horizontal Sync End

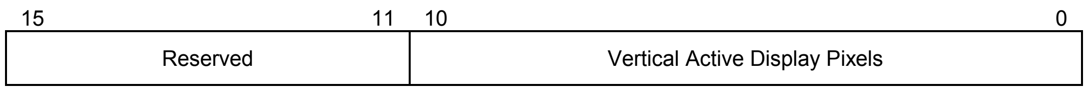
15 12	11 0
Reserved	Horizontal Sync Start

Bit	Description
31:28	**Reserved.**
27:16	**Horizontal Sync End.** Horizontal sync end expressed in terms of absolute pixel number relative to the horizontal active display start. Notes: 1. Minimum HSYNC width is 1 pixel clock. 1. An asserted HSYNC will be cleared as soon as HTOTAL end is reached, regardless of the value in the HSYNC End register.
15:12	**Reserved.**
11:0	**Horizontal Sync Start**. Horizontal sync start expressed in terms of absolute pixel number relative to the horizontal active display start. Note that when HSYNC Start is programmed equal to HBLANK Start, both HSYNC and HBLANK will be asserted on the same pixel clock.

17.4. VTOTAL—Vertical Total Register

Address Offset:	6000Ch
Default Value:	00000000h
Access:	Read/Write
Size:	32 bits





Bit	Description
31:28	**Reserved.**
27:16	**Vertical Total Display Pixels.** Total vertical display lines. This 12-bit field provides a vertical total up to 4096 lines encompassing 2048 active display lines, top/bottom border lines and vertical retrace period.
15:11	**Reserved.**
10:0	**Vertical Active Display Pixels.** Active vertical display lines. This 11-bit field provides vertical active display resolution up to 2048 lines. Note that the first vertical active display line always starts at 0.

17.5. VBLANK—Vertical Blank Register

Address Offset:	60010h
Default Value:	00000000h
Access:	Read/Write
Size:	32 bits

31 28	27 16
Reserved	Vertical Blank End

15 12	11 0
Reserved	Vertical Blank Start

Bit	Description
31:28	**Reserved.**
27:16	**Vertical Blank End.** Vertical blank end expressed in terms of absolute line number relative to the vertical active display start. Note: Vertical blank will be deasserted when vertical total occurs irrespective of what is programmed in vertical blank end.
15:12	**Reserved.**
11:0	**Vertical Blank Start.** Vertical blank start expressed in terms of absolute line number relative to the vertical active display start.

17.6. VSYNC—Vertical Sync Register

Address Offset:	60014h
Default Value:	00000000h
Access:	Read/Write
Size:	32 bits

31 28	27 16
Reserved	Vertical Sync End

15 12	11 0
Reserved	Vertical Sync Start

Bit	Description
31:28	**Reserved.**
27:16	**Vertical Sync End**. Vertical sync end expressed in terms of absolute line numbers relative to the vertical active display start. Notes: 1. When VSYNC Start is programmed equal to VBLNK Start, both VSYNC and VBLANK will be asserted on the same pixel clock. 2. VSYNC Start programmed beyond the VTOTAL end will prevent the VSYNC start and VSYNC end to occur..
15:12	**Reserved.**
11:0	**Vertical Sync Start.** Vertical sync start expressed in terms of absolute line number relative to the vertical active display start. Notes: 1. Minimum VSYNC width is 2 lines. A VSYNC programmed to 1 scan line does not generate the correct picture. 2. An asserted VSYNC will be cleared as soon as VTOTAL end is reached, regardless of the value in the VSYNC End register.

17.7. LCDTV_C—LCD/TV-Out Control Register

Address Offset:	60018h
Default Value:	00000000h
Access:	Read/Write
Size:	32 bits

31	30	29	28	27–24
LCD / TV-Out Enable	SYNC Polarity Control	Centering Enable	FP VESA VGA Mode	Reserved

23–16
Reserved

15	14	13	12	11	10	9	8
Reserved	FP / 740 Data Ordering	LCD Info. Data Enable	Reserved	VSYNC Control	HSYNC Control	VSYNC Output Control	HSYNC Output Control

7	6	5	4	3	2	1	0
Border Enable	Active Data ½ Pixel Order	Active Data Polarity	VSYNC Polarity Control	HSYNC Polarity Control	BLANK# Polarity Control	Dot Clock Source	Lock Dot Clock PLL N/M Regs

Bit	Description
31	**LCD / TV-Out Enable.** 1 = Enable. This bit enables the LCD / TV digital interface. The LCD / TV Timing Generator is jammed to Pixel 0 of Vertical Front Porch when this bit is a 0. The timing generator may be ignored depending on the LCD Timing Generator Bit (29). 0 = Disable and Tristate the whole interface: TVDATA[11:0], BLANK#, TVHSYNC, TVVSYNC, and TVCLK[1:0]. CLKIN is not disabled and can be used for Flat Panel Hot Plug detection.
30	**SYNC Polarity Control.** 0 = Source of TVHSYNC/TVVSYNC polarity is the LCDTV_C—LCD/TV-Out Control Register in multi-sync mode (default) 1 = Source of TVHSYNC/TVVSYNC polarity is the MSR—Miscellaneous Output Register in multi-sync mode.
29	**Centering Enable.** 0 = Disable. The LCD / TV timing generator controls all display timing when enabled by bit 31 above. 1 = Enable. Centers the VGA active image as defined in the VGA CRT registers within LCD/TV active image.

Bit	Description
28	**FP VESA VGA Mode** 0 = Disable. Use the LCD / TV Timing Generator. VGA Sync Polarity is ignored. FP Sync Polarity is used. Centering can be enabled for fixed resolution flat panels or TVs. The Flat Panel Dot clock PLL timing registers must be used for both flat panels and TVs. After these registers are written the Lock Dot Clock PLL N/M Registers must be set to 1 which makes the Dot Clock PLL only use the Flat Panel PLL registers. 1 = Enable. Use the VGA Timing Generator. VGA Sync polarity is passed though and FP Sync Polarity is ignored. Centering must be disabled. Also set bit 0 of this register, Lock Dot Clock PLL N/M Regs to a 0 which allows normal programming of the Dot Clock PLL registers. This bit should be disabled when driving a TV.
27:17	**Reserved.** MBZ
16:15	**Reserved.**
14	**FP / 740 Data Ordering** 0 = 740 Compliant Data Ordering: 1 = Flat Panel Data Ordering: R[7:0] ‘ G[7:4] followed by G[3:0] ‘ B[7:0].
13	**LCD Information Data Enable**. When enabled, transfers data from GC to the external device during Vertical sync. This transfer should be qualified by blank signal. 0 = Disable 1 = Enable. (Currently planned for Debug purposes)
12	**Reserved**.
11	**FPVSYNC Control.** 1 = FPVSYNC is disabled. •If in **FP VESA VGA Mode,** then this pin goes to the level of the VGA VSYNC when disabled. •If not in **FP VESA VGA Mode,** then this pin goes is in the deasserted state as specified by the VSYNC Polarity Control field. 0 = FPVSYNC is enabled. •When in **FP VESA VGA Mode,** then the VGA timing generator is the source of this signal •When not in **FP VESA VGA Mode,** then the source of this signal is this timing generator.
10	**FPHSYNC Control.** 1 = FPHSYNC is disabled. •If in **FP VESA VGA Mode,** then this pin goes to the level of the VGA HSYNC when disabled. •If not in **FP VESA VGA Mode,** then this pin goes is in the deasserted state as specified by the HSYNC Polarity Control field. 0 = FPHSYNC is enabled. •When in **FP VESA VGA Mode,** then the VGA timing generator is the source of this signal •When not in **FP VESA VGA Mode,** then the source of this signal is this timing generator.
9	**FPVSYNC Output Control.** 1 = Tristates the FPVSYNC pin. 0 = FPVSYNC is active unless LCD / TV Out Enable is deasserted. Though this bit is provided, the GC always use VSYNC as output.

Bit	Description
8	**FPHSYNC Output Control.** 1 = Tristates the FPHSYNC pin. 0 = FPHSYNC is active unless LCD / TV Out Enable is deasserted.
7	**Border Enable.** 1 = Border to the LCD / TV encoder is enabled. 0 = Border to the LCD / TV encoder is disabled.
6	**Active Data Order.** 1 = Reversed ½ pixel data ordering: G[3:0] ‘ B[7:0] followed by R[7:0] ‘ G[7:4]. 0 = Normal ½ pixel data ordering: R[7:0] ‘ G[7:4] followed by G[3:0] ‘ B[7:0].
5	**Active Data Polarity.** 1 = Inverted Pixel Data 0 = Normal Pixel Data
4	**VSYNC Polarity Control.** When the LCD / TV timing generator is disabled, the polarity is controlled by the VGA registers. 1 = Active high. 0 = Active low.
3	**HSYNC Polarity Control.** When the LCD / TV timing generator is disabled, the polarity is controlled by the VGA registers. 1 = Active high. 0 = Active low.
2	**BLANK# Polarity Control.** 1 = Active high. 0 = Active low.
1	**Dot Clock Source**. 1 = Dot Clock PLL Reference Source is External Pin = CLKIN 0 = Dot Clock PLL Reference Source is the default PLL source. The CLKIN pin can be used an Interrupt for FP hot plug detection. When the pin is used as a clock, the interrupt signal is forced to the deassertion level. The CLKIN / Interrupt pin is always an input. It is never disabled. An internal pull up is active when the pin is configured as an Interrupt. When configured as a clock the internal pull up is disabled.
0	**Lock Dot Clock PLL N/M Regs.** 1 = Dot Clock PLL N/M registers are locked. Use the LCD / TV PLL M/N registers and ignore the MSR register. 0 = Dot Clock PLL N/M registers are writeable. The MSR register controls which PLL M/N registers are used When either supporting a TV Encoder or a Flat Panel, but not in VESA VGA mode, the LCD / TV PLL M/N registers must be set up for the proper Dot clock frequency. Then this bit is written with a 1 to lock the registers. When written with a 1, forces the Dot Clock PLL to only look at the LCD / TV PLL M/N registers.

17.8. OVRACT—Overlay Active Register

Address Offset:	6001Ch
Default Value:	00000000h
Access:	Read/Write
Size:	32 bits

31 27	26 16
Reserved	Overlay Active End

15 12	11 0
Reserved	Overlay Active Start

Bit	Description
31:27	**Reserved.**
26:16	**Overlay Active End**. This filed takes into account the Overlay pipeline delays for turning off the overlay at the end of a scan line. When LCD / TV is enabled, then the overlay active end is controlled by the LCD / TV-Out Timing Generator and uses all the bits. When LCD / TV is disabled, then the overlay active end is controlled by the VGA Timing Generator and uses bits 15:3 for character clock resolution.
15:12	**Reserved.**
11:0	**Overlay Active Start.** This field takes into account the Overlay pipeline delays for lining up X=0 to the first active pixel. When LCD / TV is enabled, then the overlay active start is controlled by the LCD / TV-Out Timing Generator and all the bits are used. When LCD / TV is disabled, then the overlay active start is controlled by the VGA Timing Generator and uses bits 15:3 for character clock.

17.9. BCLRPAT— Border Color Pattern Register

Address Offset:	60020h
Default Value:	00000000h
Access:	Read/Write
Size:	32 bits

A border is sent if Border Enable is on. Also same color will be sent during pseudo border period in LCD no-scalar mode.

In VGA Centering mode, the VGA border color will be sent instead of the Flat Panel Border Color. HIRes Centering mode will still use the Flat Panel Border color.

31 25	24 16	15 8	7 0
Reserved	Red	Green	Blue

Bit	Description
31:25	**Reserved.**
24:16	**Red.**
15:8	**Green.**
7:0	**Blue.**

18. Local Memory Interface

18.1. DRT—DRAM Row Type

Address offset :	03000h
Default value :	00h
Access :	Read / write
Size :	8 bit

This 8-bit register identifies whether or not the local memory is populated.

7	1	0
Reserved		DRAM Populated

Bit	Description
7:1	**Reserved**
0	**DRAM Populated (DP).** This bit indicates whether or not the local memory is populated 0 = No local memory 1 = 4 MB local memory

18.2. DRAMCL—DRAM Control Low

Address offset :	03001h
Default value :	17h
Access :	Read / write
Size :	8 bit

7:5	4	3	2	1	0
Reserved	Paging Mode Control	RAS-to-CAS Override	CAS# Latency	RAS# Riming	RAS# Precharge Timing

Bit	Description
7:5	**Reserved**
4	**Paging Mode Control (PMC).** 0 = Page Open Mode. In this mode the GMCH memory controller tends to leave pages open. 1 = Page Close Mode. In this mode The GMCH memory controller tends to leave pages closed.
3	**RAS-to-CAS Override (RCO).** In units of local memory clock periods. (i.e., row activate command to read/write command) **Bit** **RAS#-to-CAS# delay** (t_{RCD}) 0 determined by CL bit (default) 1 2
2	**CAS# Latency (CL).** In units of local memory clock periods. **Bit** **CL** **RAS#-to-CAS# delay** (t_{RCD}) 0 2 2 1 3 3 (default)
1	**RAS# Riming (RT).** This bit controls RAS# active to precharge, and refresh to RAS# active delay (in local memory clocks). **Bit** **RAS# act. To precharge** (t_{RAS}) **Refresh to RAS# act.** (t_{RC}) 0 5 8 1 7 10 (default)
0	**RAS# Precharge Timing (RPT).** This bit controls RAS# precharge (in local memory clocks). **Bit** **RAS# Precharge** (t_{RP}) 0 2 1 3 (default)

18.3. DRAMCH—DRAM Control High

Address offset :	03002h
Default value :	08h
Access :	Read / write
Size :	8 bit

7 5	4 3	2 0
Reserved	DRAM Refresh Rate	Special Mode Select

Bit	Description
7:5	**Reserved**
4:3	**DRAM Refresh Rate (DRR).** DRAM refresh is controlled using this field. Disabling refresh results in the eventual loss of DRAM data, although refresh can be briefly disabled without data loss. The field must be set to normal refresh as soon as possible once DRAM testing is completed. 00 = Refresh Disabled 01 = Refresh Enabed (default) 11 = Reserved 10 = Reserved
2:0	**Special Mode Select (SMS).** These bits select special SDRAM modes used for testing and initialization. Note that the NOP command must be programmed first before any other command can be issued. 000 = Normal SDRAM mode (Normal, default). 001 = NOP Command Enable (NCE). This state forces cycles to DRAM to generate SDRAM NOP commands. 010 = All Banks Precharge Command Enable (ABPCE). This state forces cycles to DRAM to generate an all banks precharge command. 011 = Mode Register Command Enable (MRCE). This state forces all cycles to DRAM to be converted into MRS commands. The command is driven on the MA[11:0] lines. MA[2:0] correspond to the burst length, MA[3] corresponds to the wrap type, and MA[6:4] correspond to the latency mode. MA[11:7] are driven to 00000 by The GMCH, The BIOS must select an appropriate host address for each row of memory such that the right commands are generated on the MA[6:0] lines, taking into account the mapping of host addresses to local memory addresses. 100 = CBR Cycle Enable (CBRCE). This state forces cycles to DRAM to generate SDRAM CBR refresh cycles. 101 = Reserved. 11X = Reserved.

This page is intentionally left blank.

19. I/O Control Registers

19.1. HVSYNC—HSYNC/VSYNC Control Register

Address Offset:	05000h
Default Value:	00000000h
Size:	32 bits
Attribute:	R/W

Bits 19:16 are for DPMS and DDC Sync Select.

DPMS MODE	HSYNC/VSYNC Control[19:16]
Power On	0000 (i.e., pulse H and V)
StandBye	0010 (i.e., pulse V)
Suspend	1000 (i.e., pulse H)
Power Off	1010 (no pulse on H & V)

31 — 20	19	18	17	16
Reserved	VSYNC Control	VSYNC Data	HSYNC Control	HSYNC Data

15 — 1	0
Reserved	HSYNC/ VSYNC En

Bit	Description
31:20	**Reserved.**
19	**VSYNC Control.** Bit 19 (VSYNC Control) and bit 18 (VSYNC Data) are used by BIOS to take over the sync during DDC1 communication during POST. The BIOS can force the VSYNC data at the same time as VSYNC control enables this signal as an output, so that the VSYNC pulse occurs on every write by the BIOS. This is done to speed up some very slow DDC communications. 0 = Normal VSYNC output 1 = Contents of **VSYNC Data** will go out to VSYNC pin.
18	**VSYNC Data**
17	**HSYNC Control** 0 = Normal HSYNC output 1 = Contents of **HSYNC Data** will go out to HSYNC pin.
16	**HSYNC Data**.
15:1	**Reserved.**
0	**HSYNC/VSYNC Enable.** 0 = HSync and VSync are deactivated when the internal DAC is disabled. (default) 1 = HSync and VSync remain active when the internal DAC is disabled via the PWR_CLKC register.

19.2. GPIO Registers

19.2.1. GPIOA—General Purpose I/O Control Register A

Address offset :	05010h
Default value :	00h, 00h, 000U0000b, 000U0000b
Access :	Read / write
Size :	32 bit

This register controls the general purpose I/O pins DDCK and DDDA, which are used to create a Display Data Channel (DDC) serial bus for communication with an external analog display monitor.

31 — 16
Reserved

15 — 13	12	11	10	9	8
Reserved	DDDA Data In	DDDA Data value	DDDA Data Mask	DDDA Direction value	DDDA Direction Mask

7 — 5	4	3	2	1	0
Reserved	DDCK Data In	DDCK Data value	DDCK Data Mask	DDCK Direction value	DDCK Direction Mask

Bit	Description
31:16	**Reserved.**
15:13	**Reserved.**
12	**DDDA Data In—RO:** This is the value that is sampled on the DDDA pin as an input. The Data In bits [12], [4] of the GPIOA (GPIOB) register are read only, however, data is only latched into these bits when a write is done to the respective bytes of the GPIOA (GPIOB) register. Thus a read of the Data In bits must be preceded with a dummy write.
11	**DDDA Data Value—R/W:** This is the value that should be place on the DDDA pin as an output. This value is only written into the register if **DDDA DATA MASK** is also asserted. The value will appear on the pin if this data value is actually written to this register and the **DDDA DIRECTION VALUE** contains a value that will configure the pin as an output.
10	**DDDA Data Mask—R/W:** This is a mask bit to determine whether the **DDDA DATA VALUE** bit should be written into the register. 0 = Do NOT write DDDA Data Value bit (default). 1 = Write DDDA Data Value bit.
9	**DDDA Direction Value—R/W:** This is the value that should be used to define the ouput enable of the DDDA pin. This value is only written into the register if **DDDA DIRECTION MASK** is also asserted. The value that will appear on the pin is defined by what is in the register for the **DDDA DATA VALUE**.bit. 0 = Pin is configured as an input (default) 1 = Pin is configured as an output.

Bit	Description
8	**DDDA Direction Mask—R/W:** This is a mask bit to determine whether the **GPIO DIRECTION VALUE** bit should be written into the register. 0 = Do NOT write DDDA Direction Value bit (default). 1 = Write DDDA Direction Value bit.
7:5	**Reserved**
4	**DDCK Data In—RO:** This is the value that is sampled on the DDCK pin as an input. The Data In bits [12], [4] of the GPIOA register are read only, however, data is only latched into these bits when a write is done to the respective bytes of the GPIOA register. Thus a read of the Data In bits must be preceded with a dummy write.
3	**DDCK Data Value—R/W:** This is the value that should be place on the DDCK pin as an output. This value is only written into the register if **DDCK DATA MASK** is also asserted. The value will appear on the pin if this data value is actually written to this register and the **DDCK DIRECTION VALUE** contains a value that will configure the pin as an output.
2	**DDCK Data Mask—R/W:** This is a mask bit to determine whether the **DDCK DATA VALUE** bit should be written into the register. 0 = Do NOT write DDCK Data Value bit (default). 1 = Write DDCK Data Value bit.
1	**DDCK Direction Value—R/W:** This is the value that should be used to define the ouput enable of the GPIO0 pin. This value is only written into the register if **DDCK DIRECTION MASK** is also asserted. The value that will appear on the pin is defined by what is in the register for the **DDCK DATA VALUE**.bit. 0 = Pin is configured as an input (default) 1 = Pin is configured as an output.
0	**GPIO0 Direction Mask—R/W:** This is a mask bit to determine whether the **GPIO DIRECTION VALUE** bit should be written into the register. 0 = Do NOT write DDCK Direction Value bit (default). 1 = Write DDCK Direction Value bit.

19.2.2. GPIOB—General Purpose I/O Control Register B

Address offset :	05014h
Default value :	00h, 00h, 000U0000b, 000U0000b
Access :	Read / write
Size :	32 bit

This register controls the general purpose I/O pins LTVCK and LTVDA, which are used to create an I^2C connection to the external Digital Video Out controller.

31 ... 16
Reserved

15 ... 13	12	11	10	9	8
Reserved	LTVDA Data In	LTVDA Data value	LTVDA Data mask	LTVDA Direction value	LTVDA Direction Mask

7 ... 5	4	3	2	1	0
Reserved	LTVCK Data In	LTVCK Data value	LTVCK Data mask	LTVCK Direction value	LTVCK Direction Mask

Bit	Description
31:16	**Reserved.**
15:13	**Reserved.**
12	**LTVDA Data In—RO:** This is the value that is sampled on the LTVDA pin as an input. The Data In bits [12], [4] of the GPIOA (GPIOB) register are read only, however, data is only latched into these bits when a write is done to the respective bytes of the GPIOA (GPIOB) register. Thus a read of the Data In bits must be preceded with a dummy write.
11	**LTVDA Data Value—R/W:** This is the value that should be place on the LTVDA pin as an output. This value is only written into the register if **LTVDA DATA MASK** is also asserted. The value will appear on the pin if this data value is actually written to this register and the **LTVDA DIRECTION VALUE** contains a value that will configure the pin as an output.
10	**LTVDA Data Mask—R/W:** This is a mask bit to determine whether the **LTVDA DATA VALUE** bit should be written into the register. 0 = Do NOT write LTVDA Data Value bit (default). 1 = Write LTVDA Data Value bit.
9	**LTVDA Direction Value—R/W:** This is the value that should be used to define the ouput enable of the LTVDA pin. This value is only written into the register if **LTVDA DIRECTION MASK** is also asserted. The value that will appear on the pin is defined by what is in the register for the **LTVDA DATA VALUE**.bit. 0 = Pin is configured as an input (default) 1 = Pin is configured as an output.

Bit	Description
8	**LTVDA Direction Mask—R/W:** This is a mask bit to determine whether the **GPIO DIRECTION VALUE** bit should be written into the register. 0 = Do NOT write LTVDA Direction Value bit (default). 1 = Write LTVDA Direction Value bit.
7:5	**Reserved.**
4	**LTVCK Data In—RO:** This is the value that is sampled on the LTVCK pin as an input. The Data In bits [12], [4] of the GPIOA (GPIOB) register are read only, however, data is only latched into these bits when a write is done to the respective bytes of the GPIOA (GPIOB) register. Thus a read of the Data In bits must be preceded with a dummy write.
3	**LTVCK Data Value—R/W:** This is the value that should be place on the LTVCK pin as an output. This value is only written into the register if **LTVCK DATA MASK** is also asserted. The value will appear on the pin if this data value is actually written to this register and the **LTVCK DIRECTION VALUE** contains a value that will configure the pin as an output.
2	**GPIO2 Data Mask—R/W:** This is a mask bit to determine whether the **LTVCK DATA VALUE** bit should be written into the register. 0 = Do NOT write LTVCK Data Value bit (default). 1 = Write LTVCK Data Value bit.
1	**LTVCK Direction Value—R/W:** This is the value that should be used to define the ouput enable of the LTVCK pin. This value is only written into the register if **LTVCK DIRECTION MASK** is also asserted. The value that will appear on the pin is defined by what is in the register for the **LTVCK DATA VALUE**.bit. 0 = Pin is configured as an input (default) 1 = Pin is configured as an output.
0	**LTVCK Direction Mask—R/W:** This is a mask bit to determine whether the **GPIO DIRECTION VALUE** bit should be written into the register. 0 = Do NOT write LTVCK Direction Value bit (default). 1 = Write LTVCK Direction Value bit.

This page is intentionally left blank.

20. Display And Cursor Registers

The following are cursor, display, and pixel pipe registers in address range 70000h–7FFFFh.

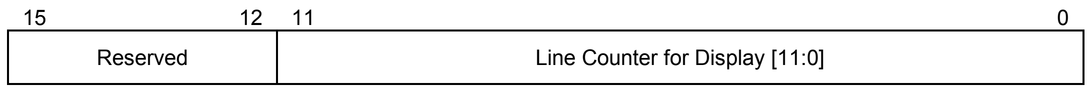
20.1. DISP_SL—Display Scan Line Count

Memory Offset Address:	70000h
Default:	0000h
Attributes:	Read only

This register enables the read back of the display vertical line counter. In interlaced display modes the line counter is initialized to the field and is incremented by two at each HSYNC.

The display line values are from CRTTG (the CRT timing generator) or the TV/FP timing generator, depending on whether the TV/FP timing generator is enabled and not in FP VESA VGA Mode (LCDTV_C[31]=1 AND LCDTV_C[28]=0) or not. The line counter changes at the leading edge of HSYNC, and can be safely read during display enable active time.

When in TV/FP centering mode, scan line 0 is the first active scan line of the TV/FP, not the first line of the centered active display.



Bit	Descriptions
15:12	**Reserved.**
11:0	**Line Counter for Display [11:00].**

20.2. DISP_SLC—Display Scan Line Count Range Compare

Memory Offset Address:	70004h
Default:	0000h
Attributes:	Read only

The Top and Bottom Line Count Compare registers are compared with the display line values from CRTTG (the CRT timing generator) or the TV/FP timing generator, depending upon whether the TV/FP timing generator is enabled and not in FP VESA VGA Mode (LCDTV_C[31]=1 AND LCDTV_C[28]=0) or not. The line counter changes at the leading edge of HSYNC. It can be safely read during the display enable active time. The Top compare register operator is a less than or equal, while the Bottom compare register operator is a greater than or equal. The results of these two comparisons are communicated to the command stream controller for generating interrupts, status, and command stream flow control (wait for scanline). These registers can be loaded from the command stream and read through the PCI.

When in TV/FP centering mode, scan line 0 is the first active scan line of the TV/FP, not the first line of the centered active display.

31	30 28	27 16
In/Ex	Reserved	Top Line Count Compare for Display SLC [11:00]

15 12	11 0
Reserved	Bottom Line Count Compare for Display SLC [11:00]

Bit	Descriptions
31	**Inclusive / Exclusive.** 1 = Inclusive: within the range. 0 = Exclusive: outside of the range.
30:28	**Reserved.**
27:16	**Top Line Count Compare for Display SLC [11:00].** This register is used as the top comparison (less than or equal to) value with the display vertical line counter.
15:12	**Reserved.**
11:0	**Bottom Line Count Compare for Display SLC [11:00].** This register is used as the bottom comparison (greater than or equal to) value with the display vertical line counter.

20.3. Pixel Pipeline Control

20.3.1. PIXCONF—Pixel Pipeline Configuration

Memory Offset Address:	70008h
Default:	00000000h
Attributes:	Read/Write

31:28	27	26	25	24
Reserved (0000)	Display Gamma Enable	Overlay Gamma Enable	Reserved	Reserved

23:21	20	19:16
Reserved (000)	CRT Control	Display Color Mode

15	14:13	12	11	10	9	8
8-Bit DAC Enable	Reserved	Cursor Display Enable	Extended Status Read	CRT Overscan Color	Reserved	Palette Addr

7:5	4	3	2	1	0
Reserved (000)	Reserved (0)	Reserved (*System Write 32*)	Reserved (0)	VGA Wrap	GUI Mode

Bit	Descriptions
31:28	**Reserved (0000).**
27	**Display path (Graphics) Gamma Enable.** (See note.) 0 = 16 and 24 BPP graphics data bypasses palette (default). 1 = 16 and 24 BPP graphics data goes through palette.
26	**Overlay path Gamma Enable.** (See note.) 0 = Video data bypasses palette (default). 1 = Video data goes through the palette. Useful when Alpha Blending the Overlay with the Primary Display to provide gamma correction for the display device. The Overlay Gamma Correction should be set up to un-gamma the overlay surface bringing it into the linear space before performing the alpha blending. Both the primary display (27 = 1) and the Overlay (26=1) should be passed through the palette after alpha blending to provide proper gamma correction for the display device.
25:21	**Reserved (00000).**
20	**CRT Control Signal Delay.** This bit affects CRT Display enable and CRT Blank signal delay with respect to CRT HSYNC and CRT VSYNC when the standard VGA pixel pipeline is used by CRT display engine. This bit has no effect on Flat Panel centering or optimized timing modes. 0 = CRT Display Enable and CRT Blank are delayed for standard VGA compatibility (default). 1 = CRT Display Enable and CRT Blank are not delayed.

Bit	Descriptions
19:16	**Display Color Mode.** 0000 = CRT standard VGA text and graphics mode and 1-bit/2-bit/4-bit packed graphics mode (Default). 0001 = Reserved. 0010 = CRT 8-bit packed extended graphics mode. 0011 = Reserved. 0100 = CRT 16-bit packed (5-5-5) extended graphics mode (Targa compatible). 0101 = CRT 16-bit packed (5-6-5) extended graphics mode (XGA compatible). 0110 = CRT 24-bit extended graphics mode compressed. 0111 = CRT 24-bit extended graphics mode uncompressed. In this mode, pixels are stored only on the lower three bytes (plane 0,1,2) of each double word and the most significant byte of each double word (plane 3) is not used.
15	**8-Bit DAC Enable.** 0 = 6-bit DAC (default). 1 = 8-bit DAC.
14:13	**Reserved.**
12	**Hardware Cursor Display Enable.** Software should always set this bit to 1. The setting of this bit to 1 should not do any harm even while in VGA mode. 0 = Disable (default). 1 = Enable.
11	**Enable Extended Status Read Mode.** 0 = Disable (default). 1 = With this bit enabled, the status of the internal state machines and values of the red and green data in the input holding register through the normal DAC register ports is available. The register ports are redefined as follows when this bit is set: DACMASK = Returns red input data holding value DACWX = Returns green input data holding value DACSTATE = Returns the status of the internal state machines in bits [7:2]
10	**CRT Overscan Color** 0 = Disable (default). 1 = Enable Protected CRT Overscan Color (Overscan[0]).
9	**Reserved.**
8	**Palette Addressing.** 0 = Disable (default). 1 = Enable Extended Palette Addressing (Enables access to all 8 locations).
7:2	**Reserved (000000).**
1	**VGA Wrap.** 0 = 256 KB wrap state (default) for memory starting at A0000. 1 = Do not wrap.

Bit	Descriptions
0	**GUI Mode.** 0 = Standard VGA and extended 4 bpp/16 color resolutions (default). Can still access memory in linear mode. 1 = High Resolution (i.e., not VGA or extended planar). **Transition from VGA modes to hires mode or opposite:** Software will turn the display engine off (screen off) using SR01[Screen Off] and wait for at least two HSYNC periods and no more than two VSYNC periods (since one of the isochronous streams is DRAM refresh (controlled by DRAMCXH[DRAM Refresh Status]), the wait should not be so long as to cause the DRAM content to degrade) before writing to PIXCONF[0] and turning the display on. This should ensure that all the data requested from the display engine will be out of the local memory interface before PIXCONF is touched. In addition, while switching from hi-res to VGA or VGA to hi-res, software will ensure that all the other isochronous streams are off before programming PIXCONF[0].

NOTES: Bits [27:24] are not normally used by the graphics BIOS or by the drivers because the gamma values are specific to a particular display device, and apply to two color or hi-color modes (16 and 24 bit). It is necessary to program the palette first with the gamma adjusted values. There is only one palette, so if both 3:2 are set, they have the same gamma adjustments. Typical code and typical drivers leave these bits as zero.

20.3.2. BLTCNTL—BLT Control

Memory Offset Address:	7000Ch
Default:	0000h
Attributes:	Read/Write

15 1	0
Reserved	BLT Status

Bit	Descriptions
15:1	**Reserved.**
0	**BLT Status – RO**. This read-only bit reflects the busy status of BLT Engine only. 0 = Idle (default) 1 = Busy

20.3.3. SWF[1:3]—Software Flag Registers

Memory Offset Address:	SWF1 =70014h
	SWF2 = 70018h
	SWF3 = 7001Ch

Default:	00000000h
Attributes:	Read/Write

These 32-bit registers are used as scratch pad space in BIOS, and have no effect on hardware.

20.3.4. DPLYBASE—Display Base Address Register

Memory Offset Address:	70020h
Default:	0000h
Attributes:	Read/Write

The display can be read from graphics memory. This register is the display staging register when written. The load register is transferred into the active register on the asserting edge of Vertical Sync. The read back register is from the active register.

31 26	25 3	2 0
Reserved	Display Base Address Bits	Reserved

Bit	Descriptions
31:26	**Reserved.** Bits <27:26> are implemented as scratch bits.
25:3	**Display Base Address Bits [25:03].** This is the base address of the display.
2:0	**Reserved.**

20.3.5. DPLYSTAS—Display Status Select Register

Memory Offset Address:	70024h
Default:	0000h
Attributes:	Read/Write

This register selects the proper events to be signaled to the Interrupt Control Register in the command stream. Status bits 0 and 1 are logically ORed to become the Vertical Blank status bit and status bits 8, 9, and 10 are logically ORed to become the Display Event status bit. These 2 sets of status bits are on separate bytes for future expansion purposes. The architecture allows each byte to be logically ORed. The enables, for each status bit is in the corresponding bit or the upper word. If a status enable bit is asserted, then the corresponding status bit is considered in the interrupt generation. The Status bits capture the event if enabled and are cleared by writing a 1 to the bit. This register has separate byte enables for writing.

31 27	26	25	24
Reserved	FP Hot Plug Enable	VSYNC En	Vert Line Comp En

23 18	17	16
Reserved	Verit Blank En	OVL Reg Upd En

15	14 11	10	9	8
FP Hot Plug Status	Reserved	FP Hot Plug Interrupt	VSYNC	Vert Line Comp

7 2	1	0
Reserved	Verit Blnk Dply Event Sel	OVL Reg Upd Vblnk Sel

Bit	Descriptions
31:27	**Reserved.**
26	**Flat Panel Hot Plug Detect Enable.** 0 = Flat Panel Hot Plug Detect Disabled 1 = Flat Panel Hot Plug Detect Enabled
25	**Vertical Sync Status Enable.** 0 = Vertical Sync Status Disabled 1 = Vertical Sync Status Enabled
24	**Display Line Compare Enable.** 0 = Display Line Compare Status Disabled 1 = Display Line Compare Status Enabled

Bit	Descriptions
23:18	**Reserved.**
17	**Vertical Blank Enable.** 0 = Vertical Blank Status Disabled 1 = Vertical Blank Status Enabled
16	**Overlay Registers Upated Enable.** 0 = Overlay Registers have been updated during Vertical Blank Status Disabled 1 = Overlay Registers have been updated during Vertical Blank Status Enabled
15	**Flat Panel Hot Plug Detect Status.** This bit is the state of the TVCLKIN pin of the TV/Flat Panel interface. This pin signals an interrupt when it is low. When an interrupt is asserted on the pin, this status bit reads back as a 1. This bit is forced low when the TVCLKIN pin is selected as a Clock Input reference to the Dot Clock PLL and NOT an Interrupt pin. 0 = Flat Panel Hot Plug Detect asserted 1 = Flat Panel Hot Plug Detect not asserted (forced to 1 when used as CLKIN)
14:11	**Reserved.**
10	**Flat Panel Hot Plug Detect (edge catcher).** This bit is the edge detector for bit 15 above. It is set to 1 when it detects a low to high transition. 0 = Flat Panel Hot Plug Detect not asserted; bit 15 has not transitioned from 1 to 0 (forced to 0 when used as CLKIN) 1 = Flat Panel Hot Plug Detect asserted; bit 15 has transitioned from 1 to 0.
9	**Vertical Sync Status.** 0 = Vertical Sync not asserted 1 = Vertical Sync asserted
8	**Display Line Compare Status.** 0 = Display Line Compare Status not asserted 1 = Display Line Compare Status asserted
7:2	**Reserved.**
1	**Vertical Blank Status.** 0 = Vertical Blank Status not asserted 1 = Vertical Blank Status asserted
0	**Overlay Registers Updated Status.** 0 = Overlay Registers have been updated during Vertical Blank Status not asserted 1 = Overlay Registers have been updated during Vertical Blank Status asserted

20.4. Hardware Cursor

The hardware cursor registers are memory mapped and accessible through 32 bit accesses.

20.4.1. CURCNTR—Cursor Control Register

Memory Offset Address:	70080h
Default:	0000h
Attributes:	Read/Write

This register is double buffered. The load register is transferred into the active register on the asserting edge of Vertical Sync.

15 … 5	4	3	2 … 0
Reserved	Coord Origin Select	Res	Cursor Mode Select

Bit	Descriptions
15:5	**Reserved.**
4	**Cursor Coordinate System Origin Select.** 0 = Selects the outermost upper left-hand corner of the screen border as the origin for the coordinate system used to position cursor. (default). 1 = Selects the upper left-hand corner of the active display area as the origin for the coordinate system used to position cursor.
3	**Reserved.**
2:0	**Cursor Mode Select.** These three bits select the mode for the cursor as follows: 000 = Cursor is disabled. This is the default after reset. 001 = 32x32 2bpp AND/XOR 2-plane mode 010 = Reserved 011 = Reserved 100 = 64x64 2bpp 3-color and transparency mode 101 = 64x64 2bpp AND/XOR 2-plane mode 110 = 64x64 2bpp 4-color mode 111 = Reserved

20.4.2. CURBASE—Cursor Base Address Register

Memory Offset Address:	70084h
Default:	0000h
Attributes:	Read/Write

The cursor can only be read from System memory. This register is double buffered. The load register is transferred into the active register on the asserting edge of Vertical Sync.

31 29	28 8	7 0
Reserved	Cursor Base Address Bits [28:08]	Reserved

Bit	Descriptions
31:29	**Reserved.**
28:8	**Cursor Base Address Bits [28:08].** These 21 bits provide most significant bits of a 29-bit physical address of the graphics non-cacheable system memory space where the 512 to 1 KB cursor data space for cursor is to be located. This is a physical address (No GTT translation).
7:0	**Reserved.**

20.4.3. CURPOS—Cursor Position Register

Memory Offset Address:	70088h
Default:	0000h
Attributes:	Read/Write

CURPOS is double buffered. The load register is transferred to the active register on the asserting edge of Vert Sync.

31	30 27	26 16
Y-Position Sign Bit	Reserved	Cursor Y-Position Magnitude Bits [10:0]

15	14 11	10 0
X-Position Sign Bit	Reserved	Cursor X-Position Magnitude Bits [10:0]

Bit	Descriptions
31	**Cursor Y-Position Sign Bit.** This bit provides the sign bit of a signed 12-bit value that specifies the horizontal position of cursor. (Default is 0).
30:27	**Reserved.**
26:16	**Cursor Y-Position Magnitude Bits 10:0.** This field provides the magnitude bits of a signed 12-bit value that specifies the vertical position of cursor. The sign bit of this value is provided by bit 31of this register. (Default is 0).
15	**Cursor X-Position Sign Bit.** This bit provides the sign bit of a signed 12-bit value that specifies the horizontal position of cursor. (Default is 0).
14:11	**Reserved.**
10:0	**Cursor X-Position Magnitude Bits 10:0.** These 11 bits provide the signed 12-bit value that specifies the horizontal position of cursor. The sign bit is provided by bit 15 of this register. (Default is 0).

21. Appendix A: Mode Parameters

This appendix contains the register programming information on a per-mode basis. Refer to the appropriate table for the specific values to use in order to correctly program the graphics adapter for the desired mode and frequency combination. Programming the graphics adapter with values not included in these tables may damage the adapter and any connected output devices.

Parameters for Screen Resolution/Refresh Rate:	**320x200_70Hz =**
Dot Clock Value	25 // 25.175-MHz Dot Clock
M Value	0x0013
N Value	0x0003
P Value	0x50
CR00	0x2E
CR01	0x27
CR02	0x28
CR03	0x90
CR04	0x2A
CR05	0x90
CR06	0xBF
CR07	0x1F
CR09	0xC0
CR10	0x9C
CR11	0x0E
CR12	0x8F
CR13	0x28
CR15	0x96
CR16	0xB9
CR30	0x01
CR31	0x01
CR32	0x01
CR33	0x01
CR35	0x00
CR39	0x00
Watermark for 8 bpp at 100-MHz local memory speed	0x0070C000
Watermark for 16 bpp at 100-MHz local memory speed	0x0020C000
Watermark for 24 bpp at 100-MHz local memory speed	0x0020C000
Watermark for 8 bpp at 133-MHz local memory speed	0x00008000
Watermark for 16 bpp at 133-MHz local memory speed	0x00008000
Watermark for 24 bpp at 133-MHz local memory speed	0x00008000
MSR Mask	0x40

Parameters for Screen Resolution/Refresh Rate:	**320x240_70Hz =**
Dot Clock Value	31 //31-MHz Dot Clock
M Value	0x0013
N Value	0x0002
P Value	0x50
CR00	0x31
CR01	0x27
CR02	0x28
CR03	0x92
CR04	0x2C
CR05	0x90
CR06	0x05
CR07	0x3E
CR09	0xC0
CR10	0xE9
CR11	0x0C
CR12	0xDF
CR13	0x28
CR15	0xE9
CR16	0xFC
CR30	0x02
CR31	0x01
CR32	0x01
CR33	0x01
CR35	0x00
CR39	0x00
Watermark for 8 bpp at 100-MHz local memory speed	0x0070C000
Watermark for 16 bpp at 100-MHz local memory speed	0x0020C000
Watermark for 24 bpp at 100-MHz local memory speed	0x0040A000
Watermark for 8 bpp at 133-MHz local memory speed	0x00008000
Watermark for 16 bpp at 133-MHz local memory speed	0x00008000
Watermark for 24 bpp at 133-MHz local memory speed	0x00008000
MSR Mask	0xC0

Parameters for Screen Resolution/Refresh Rate:	352X480_70Hz =
Dot Clock Value	15 //15.68-MHz Dot Clock
M Value	0x000D
N Value	0x0001
P Value	0x50
CR00	0x33
CR01	0x2B
CR02	0x2B
CR03	0x97
CR04	0x2D
CR05	0x91
CR06	0xF2
CR07	0x10
CR09	0x40
CR10	0xE0
CR11	0x03
CR12	0xDF
CR13	0x2C
CR15	0xDF
CR16	0xF3
CR30	0x01
CR31	0x01
CR32	0x01
CR33	0x01
CR35	0x00
CR39	0x00
Watermark for 8 bpp at 100-MHz local memory speed	0x0070C000
Watermark for 16 bpp at 100-MHz local memory speed	0x0020C000
Watermark for 24 bpp at 100-MHz local memory speed	0x00408000
Watermark for 8 bpp at 133-MHz local memory speed	0x00008000
Watermark for 16 bpp at 133-MHz local memory speed	0x00008000
Watermark for 24 bpp at 133-MHz local memory speed	0x00008000
MSR Mask	0xC0

Parameters for Screen Resolution/Refresh Rate:	352X576_70Hz =
Dot Clock Value	19 //19-MHz Dot Clock
M Value	0x0011
N Value	0x0004
P Value	0x40
CR00	0x35
CR01	0x2B
CR02	0x2B
CR03	0x99
CR04	0x2D
CR05	0x92
CR06	0x56
CR07	0x10
CR09	0x40
CR10	0x40
CR11	0x03
CR12	0x3F
CR13	0x2C
CR15	0x3F
CR16	0x57
CR30	0x02
CR31	0x02
CR32	0x02
CR33	0x02
CR35	0x00
CR39	0x00
Watermark for 8 bpp at 100-MHz local memory speed	0x0070C000
Watermark for 16 bpp at 100-MHz local memory speed	0x0020C000
Watermark for 24 bpp at 100-MHz local memory speed	0x00408000
Watermark for 8 bpp at 133-MHz local memory speed	0x00008000
Watermark for 16 bpp at 133-MHz local memory speed	0x00008000
Watermark for 24 bpp at 133-MHz local memory speed	0x00008000
MSR Mask	0xC0

Parameters for Screen Resolution/Refresh Rate:	**400x300_70Hz =**
Dot Clock Value	49 //49-MHz Dot Clock
M Value	0x001F
N Value	0x0006
P Value	0x40
CR00	0x3F
CR01	0x31
CR02	0x32
CR03	0x80
CR04	0x38
CR05	0x1D
CR06	0x86
CR07	0xF0
CR09	0xE0
CR10	0x63
CR11	0x06
CR12	0x57
CR13	0x32
CR15	0x63
CR16	0x7B
CR30	0x02
CR31	0x02
CR32	0x02
CR33	0x02
CR35	0x00
CR39	0x00
Watermark for 8 bpp at 100-MHz local memory speed	0x0070C000
Watermark for 16 bpp at 100-MHz local memory speed	0x00409000
Watermark for 24 bpp at 100-MHz local memory speed	0x00409000
Watermark for 8 bpp at 133-MHz local memory speed	0x00008000
Watermark for 16 bpp at 133-MHz local memory speed	0x00008000
Watermark for 24 bpp at 133-MHz local memory speed	0x00008000
MSR Mask	0x00

Parameters for Screen Resolution/Refresh Rate:	**512X384_70Hz =**
Dot Clock Value	82 //82-MHz Dot Clock
M Value	0x0027
N Value	0x000A
P Value	0x30
CR00	0x53
CR01	0x3F
CR02	0x40
CR03	0x94
CR04	0x47
CR05	0x0E
CR06	0x3B
CR07	0xFD
CR09	0xE0
CR10	0x0E
CR11	0x01
CR12	0xFF
CR13	0x40
CR15	0x0E
CR16	0x2D
CR30	0x03
CR31	0x02
CR32	0x03
CR33	0x03
CR35	0x00
CR39	0x00
Watermark for 8 bpp at 100-MHz local memory speed	0x0040A000
Watermark for 16 bpp at 100-MHz local memory speed	0x0040A000
Watermark for 24 bpp at 100-MHz local memory speed	0x0040A000
Watermark for 8 bpp at 133-MHz local memory speed	0x00008000
Watermark for 16 bpp at 133-MHz local memory speed	0x00008000
Watermark for 24 bpp at 133-MHz local memory speed	0x00008000
MSR Mask	0xC0

Parameters for Screen Resolution/Refresh Rate:	640x350_85Hz =
Dot Clock Value	31 //31.5-MHz Dot Clock
M Value	0x0013
N Value	0x0002
P Value	0x40
CR00	0x63
CR01	0x4F
CR02	0x4F
CR03	0x87
CR04	0x53
CR05	0x9B
CR06	0xBB
CR07	0x10
CR09	0x40
CR10	0x7D
CR11	0x00
CR12	0x5D
CR13	0x50
CR15	0x5D
CR16	0xBC
CR30	0x01
CR31	0x01
CR32	0x01
CR33	0x01
CR35	0x00
CR39	0x01
Watermark for 8 bpp at 100-MHz local memory speed	0x22003000
Watermark for 16 bpp at 100-MHz local memory speed	0x22005000
Watermark for 24 bpp at 100-MHz local memory speed	0x22007000
Watermark for 8 bpp at 133-MHz local memory speed	0x00008000
Watermark for 16 bpp at 133-MHz local memory speed	0x00008000
Watermark for 24 bpp at 133-MHz local memory speed	0x00008000
MSR Mask	0x80

Parameters for Screen Resolution/Refresh Rate:	**640x400_70Hz =**
Dot Clock Value	25 //25.175-MHz Dot Clock
M Value	0x0013
N Value	0x0003
P Value	0x40
CR00	0x5F
CR01	0x4F
CR02	0x50
CR03	0x82
CR04	0x54
CR05	0x80
CR06	0xBF
CR07	0x1F
CR09	0x40
CR10	0x9C
CR11	0x0E
CR12	0x8F
CR13	0x50
CR15	0x96
CR16	0xB9
CR30	0x01
CR31	0x01
CR32	0x01
CR33	0x01
CR35	0x00
CR39	0x00
Watermark for 8 bpp at 100-MHz local memory speed	0x00409000
Watermark for 16 bpp at 100-MHz local memory speed	0x0040A000
Watermark for 24 bpp at 100-MHz local memory speed	0x0040A000
Watermark for 8 bpp at 133-MHz local memory speed	0x00008000
Watermark for 16 bpp at 133-MHz local memory speed	0x00008000
Watermark for 24 bpp at 133-MHz local memory speed	0x00008000
MSR Mask	0x40

Parameters for Screen Resolution/Refresh Rate:	**640x400_85Hz =**
Dot Clock Value	31 //31.5-MHz Dot Clock
M Value	0x0013
N Value	0x0002
P Value	0x40
CR00	0x63
CR01	0x4F
CR02	0x4F
CR03	0x87
CR04	0x53
CR05	0x9B
CR06	0xBB
CR07	0x10
CR09	0x40
CR10	0x90
CR11	0x03
CR12	0x8F
CR13	0x50
CR15	0x8F
CR16	0xBC
CR30	0x01
CR31	0x01
CR32	0x01
CR33	0x01
CR35	0x00
CR39	0x01
Watermark for 8 bpp at 100-MHz local memory speed	0x22003000
Watermark for 16 bpp at 100-MHz local memory speed	0x22005000
Watermark for 24 bpp at 100-MHz local memory speed	0x22007000
Watermark for 8 bpp at 133-MHz local memory speed	0x00008000
Watermark for 16 bpp at 133-MHz local memory speed	0x00008000
Watermark for 24 bpp at 133-MHz local memory speed	0x00008000
MSR Mask	0x40

Parameters for Screen Resolution/Refresh Rate:	**640x480_60Hz =**
Dot Clock Value	25 //25.175-MHz Dot Clock
M Value	0x0013
N Value	0x0003
P Value	0x40
CR00	0x5F
CR01	0x4F
CR02	0x50
CR03	0x82
CR04	0x51
CR05	0x9D
CR06	0x0B
CR07	0x10
CR09	0x40
CR10	0xE9
CR11	0x0B
CR12	0xDF
CR13	0x50
CR15	0xE7
CR16	0x04
CR30	0x02
CR31	0x01
CR32	0x01
CR33	0x01
CR35	0x00
CR39	0x01
Watermark for 8 bpp at 100-MHz local memory speed	0x22002000
Watermark for 16 bpp at 100-MHz local memory speed	0x22004000
Watermark for 24 bpp at 100-MHz local memory speed	0x22006000
Watermark for 8 bpp at 133-MHz local memory speed	0x22008000
Watermark for 16 bpp at 133-MHz local memory speed	0x22008000
Watermark for 24 bpp at 133-MHz local memory speed	0x22108000
MSR Mask	0xC0

Parameters for Screen Resolution/Refresh Rate:	**640x480_70Hz =**
Dot Clock Value	28 //28-MHz Dot Clock
M Value	0x0053
N Value	0x0010
P Value	0x40
CR00	0x61
CR01	0x4F
CR02	0x4F
CR03	0x85
CR04	0x52
CR05	0x9A
CR06	0xF2
CR07	0x10
CR09	0x40
CR10	0xE0
CR11	0x03
CR12	0xDF
CR13	0x50
CR15	0xDF
CR16	0xF3
CR30	0x01
CR31	0x01
CR32	0x01
CR33	0x01
CR35	0x00
CR39	0x01
Watermark for 8 bpp at 100-MHz local memory speed	0x22002000
Watermark for 16 bpp at 100-MHz local memory speed	0x22004000
Watermark for 24 bpp at 100-MHz local memory speed	0x22005000
Watermark for 8 bpp at 133-MHz local memory speed	0x22008000
Watermark for 16 bpp at 133-MHz local memory speed	0x22008000
Watermark for 24 bpp at 133-MHz local memory speed	0x22108000
MSR Mask	0xC0

Parameters for Screen Resolution/Refresh Rate:	**640x480_72Hz =**
Dot Clock Value	31 //31.5-MHz Dot Clock
M Value	0x0013
N Value	0x0002
P Value	0x40
CR00	0x63
CR01	0x4F
CR02	0x4F
CR03	0x87
CR04	0x52
CR05	0x97
CR06	0x06
CR07	0x0F
CR09	0x40
CR10	0xE8
CR11	0x0B
CR12	0xDF
CR13	0x50
CR15	0xDF
CR16	0x07
CR30	0x02
CR31	0x01
CR32	0x01
CR33	0x01
CR35	0x00
CR39	0x01
Watermark for 8 bpp at 100-MHz local memory speed	0x22003000
Watermark for 16 bpp at 100-MHz local memory speed	0x22005000
Watermark for 24 bpp at 100-MHz local memory speed	0x22007000
Watermark for 8 bpp at 133-MHz local memory speed	0x22008000
Watermark for 16 bpp at 133-MHz local memory speed	0x22008000
Watermark for 24 bpp at 133-MHz local memory speed	0x22108000
MSR Mask	0xC0

Parameters for Screen Resolution/Refresh Rate:	**640x480_75Hz =**
Dot Clock Value	31 //31.5-MHz Dot Clock
M Value	0x0013
N Value	0x0002
P Value	0x40
CR00	0x64
CR01	0x4F
CR02	0x4F
CR03	0x88
CR04	0x51
CR05	0x99
CR06	0xF2
CR07	0x10
CR09	0x40
CR10	0xE0
CR11	0x03
CR12	0xDF
CR13	0x50
CR15	0xDF
CR16	0xF3
CR30	0x01
CR31	0x01
CR32	0x01
CR33	0x01
CR35	0x00
CR39	0x01
Watermark for 8 bpp at 100-MHz local memory speed	0x22003000
Watermark for 16 bpp at 100-MHz local memory speed	0x22005000
Watermark for 24 bpp at 100-MHz local memory speed	0x22007000
Watermark for 8 bpp at 133-MHz local memory speed	0x22008000
Watermark for 16 bpp at 133-MHz local memory speed	0x22008000
Watermark for 24 bpp at 133-MHz local memory speed	0x22108000
MSR Mask	0xC0

Parameters for Screen Resolution/Refresh Rate:	**640x480_85Hz =**
Dot Clock Value	36 //36-MHz Dot Clock
M Value	0x0010
N Value	0x0001
P Value	0x40
CR00	0x63
CR01	0x4F
CR02	0x4F
CR03	0x87
CR04	0x56
CR05	0x9D
CR06	0xFB
CR07	0x10
CR09	0x40
CR10	0xE0
CR11	0x03
CR12	0xDF
CR13	0x50
CR15	0xDF
CR16	0xFC
CR30	0x01
CR31	0x01
CR32	0x01
CR33	0x01
CR35	0x00
CR39	0x01
Watermark for 8 bpp at 100-MHz local memory speed	0x22003000
Watermark for 16 bpp at 100-MHz local memory speed	0x22005000
Watermark for 24 bpp at 100-MHz local memory speed	0x22107000
Watermark for 8 bpp at 133-MHz local memory speed	0x22008000
Watermark for 16 bpp at 133-MHz local memory speed	0x22008000
Watermark for 24 bpp at 133-MHz local memory speed	0x22108000
MSR Mask	0xC0

Parameters for Screen Resolution/Refresh Rate:	**720x400_85Hz =**
Dot Clock Value	35 //35.5-MHz Dot Clock
M Value	0x0045
N Value	0x000A
P Value	0x40
CR00	0x70
CR01	0x59
CR02	0x59
CR03	0x94
CR04	0x5D
CR05	0x86
CR06	0xBC
CR07	0x1F
CR09	0x40
CR10	0x90
CR11	0x03
CR12	0x8F
CR13	0x5A
CR15	0x8F
CR16	0xBD
CR30	0x01
CR31	0x01
CR32	0x01
CR33	0x01
CR35	0x00
CR39	0x01
Watermark for 8 bpp at 100-MHz local memory speed	0x22003000
Watermark for 16 bpp at 100-MHz local memory speed	0x22005000
Watermark for 24 bpp at 100-MHz local memory speed	0x22107000
Watermark for 8 bpp at 133-MHz local memory speed	0x00
Watermark for 16 bpp at 133-MHz local memory speed	0x00
Watermark for 24 bpp at 133-MHz local memory speed	0x00
MSR Mask	0x40

Parameters for Screen Resolution/Refresh Rate:	720x480_60Hz =
Dot Clock Value	28 //28.322-MHz Dot Clock
M Value	0x0053
N Value	0x0010
P Value	0x40
CR00	0x6B
CR01	0x59
CR02	0x59
CR03	0x8F
CR04	0x5B
CR05	0x84
CR06	0xEF
CR07	0x10
CR09	0x40
CR10	0xE0
CR11	0x03
CR12	0xE0
CR13	0x5A
CR15	0xDF
CR16	0xF0
CR30	0x01
CR31	0x01
CR32	0x01
CR33	0x01
CR35	0x00
CR39	0x00
Watermark for 8 bpp at 100-MHz local memory speed	0x22002000
Watermark for 16 bpp at 100-MHz local memory speed	0x22004000
Watermark for 24 bpp at 100-MHz local memory speed	0x22006000
Watermark for 8 bpp at 133-MHz local memory speed	0x22008000
Watermark for 16 bpp at 133-MHz local memory speed	0x22008000
Watermark for 24 bpp at 133-MHz local memory speed	0x22108000
MSR Mask	0x00

Parameters for Screen Resolution/Refresh Rate:	**720x480_75Hz =**
Dot Clock Value	35 //35-MHz Dot Clock
M Value	0x0021
N Value	0x0004
P Value	0x40
CR00	0x6F
CR01	0x59
CR02	0x5A
CR03	0x92
CR04	0x65
CR05	0x8E
CR06	0xF4
CR07	0x1F
CR09	0x40
CR10	0xE0
CR11	0x03
CR12	0xDF
CR13	0x5A
CR15	0xE0
CR16	0xF4
CR30	0x01
CR31	0x01
CR32	0x01
CR33	0x01
CR35	0x00
CR39	0x00
Watermark for 8 bpp at 100-MHz local memory speed	0x22003000
Watermark for 16 bpp at 100-MHz local memory speed	0x22004000
Watermark for 24 bpp at 100-MHz local memory speed	0x22008000
Watermark for 8 bpp at 133-MHz local memory speed	0x22008000
Watermark for 16 bpp at 133-MHz local memory speed	0x22008000
Watermark for 24 bpp at 133-MHz local memory speed	0x22108000
MSR Mask	0x00

Parameters for Screen Resolution/Refresh Rate:	**720x480_85Hz =**
Dot Clock Value	40 //40-MHz Dot Clock
M Value	0x0008
N Value	0x0001
P Value	0x30
CR00	0x6F
CR01	0x59
CR02	0x5A
CR03	0x92
CR04	0x65
CR05	0x8E
CR06	0xF7
CR07	0x1F
CR09	0x40
CR10	0xE0
CR11	0x03
CR12	0xDF
CR13	0x5A
CR15	0xE0
CR16	0xF7
CR30	0x01
CR31	0x01
CR32	0x01
CR33	0x01
CR35	0x00
CR39	0x00
Watermark for 8 bpp at 100-MHz local memory speed	0x22003000
Watermark for 16 bpp at 100-MHz local memory speed	0x22004000
Watermark for 24 bpp at 100-MHz local memory speed	0x22008000
Watermark for 8 bpp at 133-MHz local memory speed	0x22008000
Watermark for 16 bpp at 133-MHz local memory speed	0x22008000
Watermark for 24 bpp at 133-MHz local memory speed	0x2210C000
MSR Mask	0x00

Parameters for Screen Resolution/Refresh Rate:	720x576_60Hz =
Dot Clock Value	33 //33-MHz Dot Clock
M Value	0x0014
N Value	0x0002
P Value	0x40
CR00	0x6D
CR01	0x59
CR02	0x59
CR03	0x91
CR04	0x5C
CR05	0x85
CR06	0x53
CR07	0x10
CR09	0x40
CR10	0x40
CR11	0x03
CR12	0x3F
CR13	0x5A
CR15	0x3F
CR16	0x54
CR30	0x02
CR31	0x02
CR32	0x02
CR33	0x02
CR35	0x00
CR39	0x01
Watermark for 8 bpp at 100-MHz local memory speed	0x22002000
Watermark for 16 bpp at 100-MHz local memory speed	0x22005000
Watermark for 24 bpp at 100-MHz local memory speed	0x22006000
Watermark for 8 bpp at 133-MHz local memory speed	0x22008000
Watermark for 16 bpp at 133-MHz local memory speed	0x22008000
Watermark for 24 bpp at 133-MHz local memory speed	0x2210C000
MSR Mask	0x00

Parameters for Screen Resolution/Refresh Rate:	**720x576_75Hz =**
Dot Clock Value	43 //43-MHz Dot Clock
M Value	0x0029
N Value	0x000A
P Value	0x30
CR00	0x71
CR01	0x59
CR02	0x5A
CR03	0x94
CR04	0x65
CR05	0x8E
CR06	0x58
CR07	0xF0
CR09	0x60
CR10	0x40
CR11	0x03
CR12	0x3F
CR13	0x5A
CR15	0x40
CR16	0x58
CR30	0x02
CR31	0x02
CR32	0x02
CR33	0x02
CR35	0x00
CR39	0x00
Watermark for 8 bpp at 100-MHz local memory speed	0x22004000
Watermark for 16 bpp at 100-MHz local memory speed	0x22006000
Watermark for 24 bpp at 100-MHz local memory speed	0x22008000
Watermark for 8 bpp at 133-MHz local memory speed	0x22008000
Watermark for 16 bpp at 133-MHz local memory speed	0x22108000
Watermark for 24 bpp at 133-MHz local memory speed	0x2210C000
MSR Mask	0x00

Parameters for Screen Resolution/Refresh Rate:	**720x576_85Hz =**
Dot Clock Value	49 //49.5-MHz Dot Clock
M Value	0x001F
N Value	0x0006
P Value	0x30
CR00	0x71
CR01	0x59
CR02	0x5A
CR03	0x94
CR04	0x65
CR05	0x8E
CR06	0x5B
CR07	0xF0
CR09	0x60
CR10	0x40
CR11	0x03
CR12	0x3F
CR13	0x5A
CR15	0x40
CR16	0x5B
CR30	0x02
CR31	0x02
CR32	0x02
CR33	0x02
CR35	0x00
CR39	0x00
Watermark for 8 bpp at 100-MHz local memory speed	0x22004000
Watermark for 16 bpp at 100-MHz local memory speed	0x22006000
Watermark for 24 bpp at 100-MHz local memory speed	0x22009000
Watermark for 8 bpp at 133-MHz local memory speed	0x22008000
Watermark for 16 bpp at 133-MHz local memory speed	0x22108000
Watermark for 24 bpp at 133-MHz local memory speed	0x2210C000
MSR Mask	0x00

Parameters for Screen Resolution/Refresh Rate:	**800x600_56Hz =**
Dot Clock Value	36 //36-MHz Dot Clock
M Value	0x0010
N Value	0x0001
P Value	0x40
CR00	0x7B
CR01	0x63
CR02	0x63
CR03	0x9F
CR04	0x66
CR05	0x8F
CR06	0x6F
CR07	0x10
CR09	0x40
CR10	0x58
CR11	0x0A
CR12	0x57
CR13	0xC8
CR15	0x57
CR16	0x70
CR30	0x02
CR31	0x02
CR32	0x02
CR33	0x02
CR35	0x00
CR39	0x01
Watermark for 8 bpp at 100-MHz local memory speed	0x22003000
Watermark for 16 bpp at 100-MHz local memory speed	0x22005000
Watermark for 24 bpp at 100-MHz local memory speed	0x22107000
Watermark for 8 bpp at 133-MHz local memory speed	0x00
Watermark for 16 bpp at 133-MHz local memory speed	0x00
Watermark for 24 bpp at 133-MHz local memory speed	0x00
MSR Mask	0x00

Parameters for Screen Resolution/Refresh Rate:	800x600_60Hz =
Dot Clock Value	40 //40-MHz Dot Clock
M Value	0x0008
N Value	0x0001
P Value	0x30
CR00	0x7F
CR01	0x63
CR02	0x63
CR03	0x83
CR04	0x68
CR05	0x18
CR06	0x72
CR07	0x10
CR09	0x40
CR10	0x58
CR11	0x0C
CR12	0x57
CR13	0xC8
CR15	0x57
CR16	0x73
CR30	0x02
CR31	0x02
CR32	0x02
CR33	0x02
CR35	0x00
CR39	0x00
Watermark for 8 bpp at 100-MHz local memory speed	0x22003000
Watermark for 16 bpp at 100-MHz local memory speed	0x22006000
Watermark for 24 bpp at 100-MHz local memory speed	0x22108000
Watermark for 8 bpp at 133-MHz local memory speed	0x22008000
Watermark for 16 bpp at 133-MHz local memory speed	0x22008000
Watermark for 24 bpp at 133-MHz local memory speed	0x2210C000
MSR Mask	0x00

Parameters for Screen Resolution/Refresh Rate:	**800x600_70Hz =**
Dot Clock Value	45 //45-MHz Dot Clock
M Value	0x0054
N Value	0x0015
P Value	0x30
CR00	0x7D
CR01	0x63
CR02	0x63
CR03	0x81
CR04	0x68
CR05	0x12
CR06	0x6F
CR07	0x10
CR09	0x40
CR10	0x58
CR11	0x0B
CR12	0x57
CR13	0x64
CR15	0x57
CR16	0x70
CR30	0x02
CR31	0x02
CR32	0x02
CR33	0x02
CR35	0x00
CR39	0x00
Watermark for 8 bpp at 100-MHz local memory speed	0x22004000
Watermark for 16 bpp at 100-MHz local memory speed	0x22007000
Watermark for 24 bpp at 100-MHz local memory speed	0x2210A000
Watermark for 8 bpp at 133-MHz local memory speed	0x22008000
Watermark for 16 bpp at 133-MHz local memory speed	0x22108000
Watermark for 24 bpp at 133-MHz local memory speed	0x2210C000
MSR Mask	0x00

Parameters for Screen Resolution/Refresh Rate:	**800x600_72Hz =**
Dot Clock Value	50 //50-MHz Dot Clock
M Value	0x0017
N Value	0x0004
P Value	0x30
CR00	0x7D
CR01	0x63
CR02	0x63
CR03	0x81
CR04	0x6A
CR05	0x19
CR06	0x98
CR07	0x10
CR09	0x40
CR10	0x7C
CR11	0x02
CR12	0x57
CR13	0xC8
CR15	0x57
CR16	0x99
CR30	0x02
CR31	0x02
CR32	0x02
CR33	0x02
CR35	0x00
CR39	0x00
Watermark for 8 bpp at 100-MHz local memory speed	0x22004000
Watermark for 16 bpp at 100-MHz local memory speed	0x22007000
Watermark for 24 bpp at 100-MHz local memory speed	0x2210A000
Watermark for 8 bpp at 133-MHz local memory speed	0x22008000
Watermark for 16 bpp at 133-MHz local memory speed	0x22108000
Watermark for 24 bpp at 133-MHz local memory speed	0x2210C000
MSR Mask	0x00

Parameters for Screen Resolution/Refresh Rate:	**800x600_75Hz =**
Dot Clock Value	49 //49.5-MHz Dot Clock
M Value	0x001F
N Value	0x0006
P Value	0x30
CR00	0x7F
CR01	0x63
CR02	0x63
CR03	0x83
CR04	0x65
CR05	0x0F
CR06	0x6F
CR07	0x10
CR09	0x40
CR10	0x58
CR11	0x0B
CR12	0x57
CR13	0xC8
CR15	0x57
CR16	0x70
CR30	0x02
CR31	0x02
CR32	0x02
CR33	0x02
CR35	0x00
CR39	0x00
Watermark for 8 bpp at 100-MHz local memory speed	0x22006000
Watermark for 16 bpp at 100-MHz local memory speed	0x22007000
Watermark for 24 bpp at 100-MHz local memory speed	0x2210B000
Watermark for 8 bpp at 133-MHz local memory speed	0x22008000
Watermark for 16 bpp at 133-MHz local memory speed	0x22108000
Watermark for 24 bpp at 133-MHz local memory speed	0x2210C000
MSR Mask	0x00

Parameters for Screen Resolution/Refresh Rate:	**800x600_85Hz =**
Dot Clock Value	56 //56.25-MHz Dot Clock
M Value	0x0049
N Value	0x000E
P Value	0x30
CR00	0x7E
CR01	0x63
CR02	0x63
CR03	0x82
CR04	0x67
CR05	0x0F
CR06	0x75
CR07	0x10
CR09	0x40
CR10	0x58
CR11	0x0B
CR12	0x57
CR13	0xC8
CR15	0x57
CR16	0x76
CR30	0x02
CR31	0x02
CR32	0x02
CR33	0x02
CR35	0x00
CR39	0x00
Watermark for 8 bpp at 100-MHz local memory speed	0x22006000
Watermark for 16 bpp at 100-MHz local memory speed	0x22108000
Watermark for 24 bpp at 100-MHz local memory speed	0x2210B000
Watermark for 8 bpp at 133-MHz local memory speed	0x22008000
Watermark for 16 bpp at 133-MHz local memory speed	0x2210C000
Watermark for 24 bpp at 133-MHz local memory speed	0x22210000
MSR Mask	0x00

Parameters for Screen Resolution/Refresh Rate:	854X480_60Hz =
Dot Clock Value	43 //43-MHz Dot Clock
M Value	0x0029
N Value	0x000A
P Value	0x30
CR00	0x80
CR01	0x6A
CR02	0x6B
CR03	0x83
CR04	0x6F
CR05	0x1A
CR06	0xEF
CR07	0x1F
CR09	0x40
CR10	0xE0
CR11	0x03
CR12	0xDF
CR13	0x6B
CR15	0xE0
CR16	0xEF
CR30	0x01
CR31	0x02
CR32	0x02
CR33	0x02
CR35	0x00
CR39	0x00
Watermark for 8 bpp at 100-MHz local memory speed	0x00000000
Watermark for 16 bpp at 100-MHz local memory speed	0x00000000
Watermark for 24 bpp at 100-MHz local memory speed	0x00000000
Watermark for 8 bpp at 133-MHz local memory speed	0x00000000
Watermark for 16 bpp at 133-MHz local memory speed	0x00000000
Watermark for 24 bpp at 133-MHz local memory speed	0x00000000
MSR Mask	0x00

Parameters for Screen Resolution/Refresh Rate:	**854X480_75Hz =**
Dot Clock Value	41 //41.54-MHz Dot Clock
M Value	0x002B
N Value	0x000B
P Value	0x30
CR00	0x84
CR01	0x6A
CR02	0x6B
CR03	0x87
CR04	0x71
CR05	0x1C
CR06	0xF4
CR07	0x1F
CR09	0x40
CR10	0xE0
CR11	0x03
CR12	0xDF
CR13	0x6B
CR15	0xE0
CR16	0xF4
CR30	0x01
CR31	0x01
CR32	0x01
CR33	0x01
CR35	0x00
CR39	0x00
Watermark for 8 bpp at 100-MHz local memory speed	0x00000000
Watermark for 16 bpp at 100-MHz local memory speed	0x00000000
Watermark for 24 bpp at 100-MHz local memory speed	0x00000000
Watermark for 8 bpp at 133-MHz local memory speed	0x00000000
Watermark for 16 bpp at 133-MHz local memory speed	0x00000000
Watermark for 24 bpp at 133-MHz local memory speed	0x00000000
MSR Mask	0x00

Parameters for Screen Resolution/Refresh Rate:	**854X480_85Hz =**
Dot Clock Value	48 //Dot Clock
M Value	0x000A
N Value	0x0001
P Value	0x30
CR00	0x86
CR01	0x6A
CR02	0x6B
CR03	0x89
CR04	0x72
CR05	0x1D
CR06	0xF7
CR07	0x1F
CR09	0x40
CR10	0xE0
CR11	0x03
CR12	0xDF
CR13	0x6B
CR15	0xE0
CR16	0xF7
CR30	0x01
CR31	0x01
CR32	0x01
CR33	0x01
CR35	0x00
CR39	0x00
Watermark for 8 bpp at 100-MHz local memory speed	0x00000000
Watermark for 16 bpp at 100-MHz local memory speed	0x00000000
Watermark for 24 bpp at 100-MHz local memory speed	0x00000000
Watermark for 8 bpp at 133-MHz local memory speed	0x00000000
Watermark for 16 bpp at 133-MHz local memory speed	0x00000000
Watermark for 24 bpp at 133-MHz local memory speed	0x00000000
MSR Mask	0x00

Parameters for Screen Resolution/Refresh Rate:	**1024X768_60Hz =**
Dot Clock Value	65 //65-MHz Dot Clock
M Value	0x003F
N Value	0x000A
P Value	0x30
CR00	0xA3
CR01	0x7F
CR02	0x7F
CR03	0x87
CR04	0x82
CR05	0x93
CR06	0x24
CR07	0x10
CR09	0x40
CR10	0x02
CR11	0x08
CR12	0xFF
CR13	0x80
CR15	0xFF
CR16	0x25
CR30	0x03
CR31	0x02
CR32	0x03
CR33	0x02
CR35	0x00
CR39	0x00
Watermark for 8 bpp at 100-MHz local memory speed	0x22005000
Watermark for 16 bpp at 100-MHz local memory speed	0x22109000
Watermark for 24 bpp at 100-MHz local memory speed	0x2220D000
Watermark for 8 bpp at 133-MHz local memory speed	0x22008000
Watermark for 16 bpp at 133-MHz local memory speed	0x2210C000
Watermark for 24 bpp at 133-MHz local memory speed	0x22210000
MSR Mask	0xC0

Parameters for Screen Resolution/Refresh Rate:	1024X768_70Hz =
Dot Clock Value	75 //75-MHz Dot Clock
M Value	0x0017
N Value	0x0002
P Value	0x30
CR00	0xA1
CR01	0x7F
CR02	0x7F
CR03	0x85
CR04	0x82
CR05	0x93
CR06	0x24
CR07	0x10
CR09	0x40
CR10	0x02
CR11	0x08
CR12	0xFF
CR13	0x80
CR15	0xFF
CR16	0x25
CR30	0x03
CR31	0x02
CR32	0x03
CR33	0x02
CR35	0x00
CR39	0x00
Watermark for 8 bpp at 100-MHz local memory speed	0x22005000
Watermark for 16 bpp at 100-MHz local memory speed	0x2210A000
Watermark for 24 bpp at 100-MHz local memory speed	0x2220F000
Watermark for 8 bpp at 133-MHz local memory speed	0x22008000
Watermark for 16 bpp at 133-MHz local memory speed	0x2210C000
Watermark for 24 bpp at 133-MHz local memory speed	0x22214000
MSR Mask	0xC0

Parameters for Screen Resolution/Refresh Rate:	**1024X768_75Hz =**
Dot Clock Value	78 //78.75-MHz Dot Clock
M Value	0x0050
N Value	0x0017
P Value	0x20
CR00	0x9F
CR01	0x7F
CR02	0x7F
CR03	0x83
CR04	0x81
CR05	0x8D
CR06	0x1E
CR07	0x10
CR09	0x40
CR10	0x00
CR11	0x03
CR12	0xFF
CR13	0x80
CR15	0xFF
CR16	0x1F
CR30	0x03
CR31	0x02
CR32	0x03
CR33	0x02
CR35	0x00
CR39	0x00
Watermark for 8 bpp at 100-MHz local memory speed	0x22006000
Watermark for 16 bpp at 100-MHz local memory speed	0x2210B000
Watermark for 24 bpp at 100-MHz local memory speed	0x22210000
Watermark for 8 bpp at 133-MHz local memory speed	0x22008000
Watermark for 16 bpp at 133-MHz local memory speed	0x2210C000
Watermark for 24 bpp at 133-MHz local memory speed	0x22214000
MSR Mask	0x00

Parameters for Screen Resolution/Refresh Rate:	**1024X768_85Hz =**
Dot Clock Value	94 //94.5-MHz Dot Clock
M Value	0x003D
N Value	0x000E
P Value	0x20
CR00	0xA7
CR01	0x7F
CR02	0x7F
CR03	0x8B
CR04	0x85
CR05	0x91
CR06	0x26
CR07	0x10
CR09	0x40
CR10	0x00
CR11	0x03
CR12	0xFF
CR13	0x80
CR15	0xFF
CR16	0x27
CR30	0x03
CR31	0x02
CR32	0x03
CR33	0x02
CR35	0x00
CR39	0x00
Watermark for 8 bpp at 100-MHz local memory speed	0x22007000
Watermark for 16 bpp at 100-MHz local memory speed	0x2220E000
Watermark for 24 bpp at 100-MHz local memory speed	0x22212000
Watermark for 8 bpp at 133-MHz local memory speed	0x22108000
Watermark for 16 bpp at 133-MHz local memory speed	0x22210000
Watermark for 24 bpp at 133-MHz local memory speed	0x22314000
MSR Mask	0x00

Parameters for Screen Resolution/Refresh Rate:	**1152X864_60Hz =**
Dot Clock Value	80 //80-MHz Dot Clock
M Value	0x0008
N Value	0x0001
P Value	0x20
CR00	0xB3
CR01	0x8F
CR02	0x8F
CR03	0x97
CR04	0x93
CR05	0x9F
CR06	0x87
CR07	0x10
CR09	0x40
CR10	0x60
CR11	0x03
CR12	0x5F
CR13	0x90
CR15	0x5F
CR16	0x88
CR30	0x03
CR31	0x03
CR32	0x03
CR33	0x03
CR35	0x00
CR39	0x00
Watermark for 8 bpp at 100-MHz local memory speed	0x2220C000
Watermark for 16 bpp at 100-MHz local memory speed	0x22210000
Watermark for 24 bpp at 100-MHz local memory speed	0x22415000
Watermark for 8 bpp at 133-MHz local memory speed	0x22008000
Watermark for 16 bpp at 133-MHz local memory speed	0x2210C000
Watermark for 24 bpp at 133-MHz local memory speed	0x22214000
MSR Mask	0x00

Parameters for Screen Resolution/Refresh Rate:	**1152X864_70Hz =**
Dot Clock Value	96 //96-MHz Dot Clock
M Value	0x000A
N Value	0x0001
P Value	0x20
CR00	0xBB
CR01	0x8F
CR02	0x8F
CR03	0x9F
CR04	0x98
CR05	0x87
CR06	0x82
CR07	0x10
CR09	0x40
CR10	0x60
CR11	0x03
CR12	0x5F
CR13	0x90
CR15	0x5F
CR16	0x83
CR30	0x03
CR31	0x03
CR32	0x03
CR33	0x03
CR35	0x00
CR39	0x00
Watermark for 8 bpp at 100-MHz local memory speed	0x22107000
Watermark for 16 bpp at 100-MHz local memory speed	0x22210000
Watermark for 24 bpp at 100-MHz local memory speed	0x22415000
Watermark for 8 bpp at 133-MHz local memory speed	0x22108000
Watermark for 16 bpp at 133-MHz local memory speed	0x22210000
Watermark for 24 bpp at 133-MHz local memory speed	0x22314000
MSR Mask	0x00

Parameters for Screen Resolution/Refresh Rate:	1152X864_72Hz =
Dot Clock Value	99 //99-MHz Dot Clock
M Value	0x001F
N Value	0x0006
P Value	0x20
CR00	0xBB
CR01	0x8F
CR02	0x8F
CR03	0x9F
CR04	0x98
CR05	0x87
CR06	0x83
CR07	0x10
CR09	0x40
CR10	0x60
CR11	0x03
CR12	0x5F
CR13	0x90
CR15	0x5F
CR16	0x84
CR30	0x03
CR31	0x03
CR32	0x03
CR33	0x03
CR35	0x00
CR39	0x00
Watermark for 8 bpp at 100-MHz local memory speed	0x22107000
Watermark for 16 bpp at 100-MHz local memory speed	0x22210000
Watermark for 24 bpp at 100-MHz local memory speed	0x22415000
Watermark for 8 bpp at 133-MHz local memory speed	0x22108000
Watermark for 16 bpp at 133-MHz local memory speed	0x22210000
Watermark for 24 bpp at 133-MHz local memory speed	0x22314000
MSR Mask	0x00

Parameters for Screen Resolution/Refresh Rate:	**1152X864_75Hz =**
Dot Clock Value	108 //108-MHz Dot Clock
M Value	0x0010
N Value	0x0002
P Value	0x20
CR00	0xC3
CR01	0x8F
CR02	0x8F
CR03	0x87
CR04	0x97
CR05	0x07
CR06	0x82
CR07	0x10
CR09	0x40
CR10	0x60
CR11	0x03
CR12	0x5F
CR13	0x90
CR15	0x5F
CR16	0x83
CR30	0x03
CR31	0x03
CR32	0x03
CR33	0x03
CR35	0x00
CR39	0x01
Watermark for 8 bpp at 100-MHz local memory speed	0x22107000
Watermark for 16 bpp at 100-MHz local memory speed	0x22210000
Watermark for 24 bpp at 100-MHz local memory speed	0x22415000
Watermark for 8 bpp at 133-MHz local memory speed	0x2210C000
Watermark for 16 bpp at 133-MHz local memory speed	0x22210000
Watermark for 24 bpp at 133-MHz local memory speed	0x22318000
MSR Mask	0x00

Parameters for Screen Resolution/Refresh Rate:	**1152X864_85Hz =**
Dot Clock Value	121 //121-MHz Dot Clock
M Value	0x006D
N Value	0x0014
P Value	0x20
CR00	0xC0
CR01	0x8F
CR02	0x8F
CR03	0x84
CR04	0x97
CR05	0x07
CR06	0x93
CR07	0x10
CR09	0x40
CR10	0x60
CR11	0x03
CR12	0x5F
CR13	0x90
CR15	0x5F
CR16	0x94
CR30	0x03
CR31	0x03
CR32	0x03
CR33	0x03
CR35	0x00
CR39	0x01
Watermark for 8 bpp at 100-MHz local memory speed	0x2220C000
Watermark for 16 bpp at 100-MHz local memory speed	0x22210000
Watermark for 24 bpp at 100-MHz local memory speed	0x2241D000
Watermark for 8 bpp at 133-MHz local memory speed	0x2220C000
Watermark for 16 bpp at 133-MHz local memory speed	0x2210C000
Watermark for 24 bpp at 133-MHz local memory speed	0x22416000
MSR Mask	0x00

Parameters for Screen Resolution/Refresh Rate:	**1280x720_60Hz =**
Dot Clock Value	74 //74-MHz Dot Clock
M Value	0x0023
N Value	0x0004
P Value	0x30
CR00	0xCB
CR01	0x9F
CR02	0xA0
CR03	0x8E
CR04	0xB3
CR05	0x04
CR06	0xE8
CR07	0xF0
CR09	0x60
CR10	0xD0
CR11	0x03
CR12	0xCF
CR13	0xA0
CR15	0xD0
CR16	0xE8
CR30	0x02
CR31	0x02
CR32	0x02
CR33	0x02
CR35	0x00
CR39	0x00
Watermark for 8 bpp at 100-MHz local memory speed	0x2210A000
Watermark for 16 bpp at 100-MHz local memory speed	0x22210000
Watermark for 24 bpp at 100-MHz local memory speed	0x22415000
Watermark for 8 bpp at 133-MHz local memory speed	0x22008000
Watermark for 16 bpp at 133-MHz local memory speed	0x2210C000
Watermark for 24 bpp at 133-MHz local memory speed	0x22210000
MSR Mask	0x00

Parameters for Screen Resolution/Refresh Rate:	**1280x720_75Hz =**
Dot Clock Value	96 //96-MHz Dot Clock
M Value	0x000A
N Value	0x0001
P Value	0x20
CR00	0xCF
CR01	0x9F
CR02	0xA0
CR03	0x92
CR04	0xB3
CR05	0x04
CR06	0xEE
CR07	0xF0
CR09	0x60
CR10	0xD0
CR11	0x03
CR12	0xCF
CR13	0xA0
CR15	0xD0
CR16	0xEE
CR30	0x02
CR31	0x02
CR32	0x02
CR33	0x02
CR35	0x00
CR39	0x00
Watermark for 8 bpp at 100-MHz local memory speed	0x2210A000
Watermark for 16 bpp at 100-MHz local memory speed	0x22220000
Watermark for 24 bpp at 100-MHz local memory speed	0x22419000
Watermark for 8 bpp at 133-MHz local memory speed	0x22108000
Watermark for 16 bpp at 133-MHz local memory speed	0x22210000
Watermark for 24 bpp at 133-MHz local memory speed	0x22314000
MSR Mask	0x00

Parameters for Screen Resolution/Refresh Rate:	**1280x720_85Hz =**
Dot Clock Value	110 //110-MHz Dot Clock
M Value	0x0035
N Value	0x000A
P Value	0x20
CR00	0xD1
CR01	0x9F
CR02	0xA0
CR03	0x94
CR04	0xB3
CR05	0x04
CR06	0xF2
CR07	0xF0
CR09	0x60
CR10	0xD0
CR11	0x03
CR12	0xCF
CR13	0xA0
CR15	0xD0
CR16	0xF2
CR30	0x02
CR31	0x02
CR32	0x02
CR33	0x02
CR35	0x00
CR39	0x00
Watermark for 8 bpp at 100-MHz local memory speed	0x2210A000
Watermark for 16 bpp at 100-MHz local memory speed	0x22220000
Watermark for 24 bpp at 100-MHz local memory speed	0x2241B000
Watermark for 8 bpp at 133-MHz local memory speed	0x2210C000
Watermark for 16 bpp at 133-MHz local memory speed	0x22210000
Watermark for 24 bpp at 133-MHz local memory speed	0x22318000
MSR Mask	0x00

Parameters for Screen Resolution/Refresh Rate:	**1280x960_60Hz =**
Dot Clock Value	108 //108-MHz Dot Clock
M Value	0x0010
N Value	0x0002
P Value	0x20
CR00	0xDC
CR01	0x9F
CR02	0x9F
CR03	0x80
CR04	0xAB
CR05	0x99
CR06	0xE6
CR07	0x10
CR09	0x40
CR10	0xC0
CR11	0x03
CR12	0xBF
CR13	0xA0
CR15	0xBF
CR16	0xE7
CR30	0x03
CR31	0x03
CR32	0x03
CR33	0x03
CR35	0x00
CR39	0x01
Watermark for 8 bpp at 100-MHz local memory speed	0x2210A000
Watermark for 16 bpp at 100-MHz local memory speed	0x22210000
Watermark for 24 bpp at 100-MHz local memory speed	0x22415000
Watermark for 8 bpp at 133-MHz local memory speed	0x22108000
Watermark for 16 bpp at 133-MHz local memory speed	0x22210000
Watermark for 24 bpp at 133-MHz local memory speed	0x22314000
MSR Mask	0x00

Parameters for Screen Resolution/Refresh Rate:	**1280x960_75Hz =**
Dot Clock Value	129 //129-MHz Dot Clock
M Value	0x0029
N Value	0x0006
P Value	0x20
CR00	0xD3
CR01	0x9F
CR02	0x9F
CR03	0x97
CR04	0xAA
CR05	0x1B
CR06	0xE8
CR07	0x10
CR09	0x40
CR10	0xC0
CR11	0x03
CR12	0xBF
CR13	0xA0
CR15	0xBF
CR16	0xE9
CR30	0x03
CR31	0x03
CR32	0x03
CR33	0x03
CR35	0x00
CR39	0x01
Watermark for 8 bpp at 100-MHz local memory speed	0x2210A000
Watermark for 16 bpp at 100-MHz local memory speed	0x22210000
Watermark for 24 bpp at 100-MHz local memory speed	0x2241B000
Watermark for 8 bpp at 133-MHz local memory speed	0x2210C000
Watermark for 16 bpp at 133-MHz local memory speed	0x22214000
Watermark for 24 bpp at 133-MHz local memory speed	0x22417000
MSR Mask	0x00

Parameters for Screen Resolution/Refresh Rate:	**1280x960_85Hz =**
Dot Clock Value	148 //148.5-MHz Dot Clock
M Value	0x0042
N Value	0x0009
P Value	0x20
CR00	0xD3
CR01	0x9F
CR02	0x9F
CR03	0x97
CR04	0xA7
CR05	0x1B
CR06	0xF1
CR07	0x10
CR09	0x40
CR10	0xC0
CR11	0x03
CR12	0xBF
CR13	0xA0
CR15	0xBF
CR16	0xF2
CR30	0x03
CR31	0x03
CR32	0x03
CR33	0x03
CR35	0x00
CR39	0x01
Watermark for 8 bpp at 100-MHz local memory speed	0x2210A000
Watermark for 16 bpp at 100-MHz local memory speed	0x22220000
Watermark for 24 bpp at 100-MHz local memory speed	0x2241D000
Watermark for 8 bpp at 133-MHz local memory speed	0x2210C000
Watermark for 16 bpp at 133-MHz local memory speed	0x22314000
Watermark for 24 bpp at 133-MHz local memory speed	0x22417000
MSR Mask	0x00

Parameters for Screen Resolution/Refresh Rate:	1280x1024_60Hz =
Dot Clock Value	108 //108-MHz Dot Clock
M Value	0x0010
N Value	0x0002
P Value	0x20
CR00	0xCE
CR01	0x9F
CR02	0x9F
CR03	0x92
CR04	0xA5
CR05	0x13
CR06	0x28
CR07	0x10
CR09	0x40
CR10	0x00
CR11	0x03
CR12	0xFF
CR13	0xA0
CR15	0xFF
CR16	0x29
CR30	0x04
CR31	0x03
CR32	0x04
CR33	0x03
CR35	0x00
CR39	0x01
Watermark for 8 bpp at 100-MHz local memory speed	0x22107000
Watermark for 16 bpp at 100-MHz local memory speed	0x22210000
Watermark for 24 bpp at 100-MHz local memory speed	0x22415000
Watermark for 8 bpp at 133-MHz local memory speed	0x22110000
Watermark for 16 bpp at 133-MHz local memory speed	0x22110000
Watermark for 24 bpp at 133-MHz local memory speed	0x22318000
MSR Mask	0x00

Parameters for Screen Resolution/Refresh Rate:	1280x1024_70Hz =
Dot Clock Value	129 //128.89-MHz Dot Clock
M Value	0x0029
N Value	0x0006
P Value	0x20
CR00	0xD3
CR01	0x9F
CR02	0x9F
CR03	0x97
CR04	0xAA
CR05	0x1B
CR06	0x28
CR07	0x10
CR09	0x40
CR10	0x00
CR11	0x03
CR12	0xFF
CR13	0xA0
CR15	0xFF
CR16	0x29
CR30	0x04
CR31	0x03
CR32	0x04
CR33	0x03
CR35	0x00
CR39	0x01
Watermark for 8 bpp at 100-MHz local memory speed	0x22107000
Watermark for 16 bpp at 100-MHz local memory speed	0x22210000
Watermark for 24 bpp at 100-MHz local memory speed	0x22419000
Watermark for 8 bpp at 133-MHz local memory speed	0x22110000
Watermark for 16 bpp at 133-MHz local memory speed	0x22214000
Watermark for 24 bpp at 133-MHz local memory speed	0x22318000
MSR Mask	0x00

Parameters for Screen Resolution/Refresh Rate:	**1280x1024_72Hz =**
Dot Clock Value	132 //132-MHz Dot Clock //NOTINVESA
M Value	0x0014
N Value	0x0002
P Value	0x20
CR00	0xD3
CR01	0x9F
CR02	0x9F
CR03	0x97
CR04	0xAA
CR05	0x1B
CR06	0x29
CR07	0x10
CR09	0x40
CR10	0x00
CR11	0x03
CR12	0xFF
CR13	0xA0
CR15	0xFF
CR16	0x2A
CR30	0x04
CR31	0x03
CR32	0x04
CR33	0x03
CR35	0x00
CR39	0x01
Watermark for 8 bpp at 100-MHz local memory speed	0x22109000
Watermark for 16 bpp at 100-MHz local memory speed	0x22314000
Watermark for 24 bpp at 100-MHz local memory speed	0x22515000
Watermark for 8 bpp at 133-MHz local memory speed	0x22110000
Watermark for 16 bpp at 133-MHz local memory speed	0x22214000
Watermark for 24 bpp at 133-MHz local memory speed	0x22418000
MSR Mask	0x00

Parameters for Screen Resolution/Refresh Rate:	**1280x1024_75Hz =**
Dot Clock Value	135 //135-MHz Dot Clock
M Value	0x002B
N Value	0x0006
P Value	0x20
CR00	0xCE
CR01	0x9F
CR02	0x9F
CR03	0x92
CR04	0xA1
CR05	0x13
CR06	0x28
CR07	0x10
CR09	0x40
CR10	0x00
CR11	0x03
CR12	0xFF
CR13	0xA0
CR15	0xFF
CR16	0x29
CR30	0x04
CR31	0x03
CR32	0x04
CR33	0x03
CR35	0x00
CR39	0x01
Watermark for 8 bpp at 100-MHz local memory speed	0x22109000
Watermark for 16 bpp at 100-MHz local memory speed	0x22314000
Watermark for 24 bpp at 100-MHz local memory speed	0x2251D000
Watermark for 8 bpp at 133-MHz local memory speed	0x22110000
Watermark for 16 bpp at 133-MHz local memory speed	0x22314000
Watermark for 24 bpp at 133-MHz local memory speed	0x22416000
MSR Mask	0x00

Parameters for Screen Resolution/Refresh Rate:	**1280x1024_85Hz =**
Dot Clock Value	157 //157.5-MHz Dot Clock
M Value	0x0050
N Value	0x0017
P Value	0x10
CR00	0xD3
CR01	0x9F
CR02	0x9F
CR03	0x97
CR04	0xA7
CR05	0x1B
CR06	0x2E
CR07	0x10
CR09	0x40
CR10	0x00
CR11	0x03
CR12	0xFF
CR13	0xA0
CR15	0xFF
CR16	0x2F
CR30	0x04
CR31	0x03
CR32	0x04
CR33	0x03
CR35	0x00
CR39	0x01
Watermark for 8 bpp at 100-MHz local memory speed	0x2210B000
Watermark for 16 bpp at 100-MHz local memory speed	0x22415000
Watermark for 24 bpp at 100-MHz local memory speed	0x2251D000
Watermark for 8 bpp at 133-MHz local memory speed	0x22110000
Watermark for 16 bpp at 133-MHz local memory speed	0x22318000
Watermark for 24 bpp at 133-MHz local memory speed	0x22416000
MSR Mask	0x00

Parameters for Screen Resolution/Refresh Rate:	**1600x900_60Hz =**
Dot Clock Value	119 //Dot Clock
M Value	0x006B
N Value	0x0014
P Value	0x20
CR00	0x05
CR01	0xC7
CR02	0xC8
CR03	0x88
CR04	0xDF
CR05	0x14
CR06	0xA2
CR07	0xFF
CR09	0x60
CR10	0x84
CR11	0x07
CR12	0x83
CR13	0xC8
CR15	0x84
CR16	0xA2
CR30	0x03
CR31	0x03
CR32	0x03
CR33	0x03
CR35	0x01
CR39	0x00
Watermark for 8 bpp at 100-MHz local memory speed	0x2220E000
Watermark for 16 bpp at 100-MHz local memory speed	0x22416000
Watermark for 24 bpp at 100-MHz local memory speed	0x44419000
Watermark for 8 bpp at 133-MHz local memory speed	0x22110000
Watermark for 16 bpp at 133-MHz local memory speed	0x22214000
Watermark for 24 bpp at 133-MHz local memory speed	0x44416000
MSR Mask	0x00

Parameters for Screen Resolution/Refresh Rate:	**1600x900_75Hz =**
Dot Clock Value	152 //Dot Clock
M Value	0x0011
N Value	0x0004
P Value	0x10
CR00	0x09
CR01	0xC7
CR02	0xC8
CR03	0x8C
CR04	0xE0
CR05	0x16
CR06	0xAA
CR07	0xFF
CR09	0x60
CR10	0x84
CR11	0x07
CR12	0x83
CR13	0xC8
CR15	0x84
CR16	0xAA
CR30	0x03
CR31	0x03
CR32	0x03
CR33	0x03
CR35	0x01
CR39	0x00
Watermark for 8 bpp at 100-MHz local memory speed	0x2220E000
Watermark for 16 bpp at 100-MHz local memory speed	0x22416000
Watermark for 24 bpp at 100-MHz local memory speed	0x44419000
Watermark for 8 bpp at 133-MHz local memory speed	0x22110000
Watermark for 16 bpp at 133-MHz local memory speed	0x22314000
Watermark for 24 bpp at 133-MHz local memory speed	0x44516000
MSR Mask	0x00

Parameters for Screen Resolution/Refresh Rate:	**1600x900_85Hz =**
Dot Clock Value	175 //Dot Clock
M Value	0x0031
N Value	0x000C
P Value	0x10
CR00	0x0B
CR01	0xC7
CR02	0xC8
CR03	0x8E
CR04	0xE0
CR05	0x16
CR06	0xAF
CR07	0xFF
CR09	0x60
CR10	0x84
CR11	0x07
CR12	0x83
CR13	0xC8
CR15	0x84
CR16	0xAF
CR30	0x03
CR31	0x03
CR32	0x03
CR33	0x03
CR35	0x01
CR39	0x00
Watermark for 8 bpp at 100-MHz local memory speed	0x2220E000
Watermark for 16 bpp at 100-MHz local memory speed	0x22416000
Watermark for 24 bpp at 100-MHz local memory speed	0x44419000
Watermark for 8 bpp at 133-MHz local memory speed	0x22110000
Watermark for 16 bpp at 133-MHz local memory speed	0x22314000
Watermark for 24 bpp at 133-MHz local memory speed	0x00000000
MSR Mask	0x00

Parameters for Screen Resolution/Refresh Rate:	1600x1200_60Hz =
Dot Clock Value	162 //162-MHz Dot Clock
M Value	0x0019
N Value	0x0006
P Value	0x10
CR00	0x09
CR01	0xC7
CR02	0xC7
CR03	0x8D
CR04	0xCF
CR05	0x07
CR06	0xE0
CR07	0x10
CR09	0x40
CR10	0xB0
CR11	0x03
CR12	0xAF
CR13	0xC8
CR15	0xAF
CR16	0xE1
CR30	0x04
CR31	0x04
CR32	0x04
CR33	0x04
CR35	0x01
CR39	0x00
Watermark for 8 bpp at 100-MHz local memory speed	0x2210B000
Watermark for 16 bpp at 100-MHz local memory speed	0x22416000
Watermark for 24 bpp at 100-MHz local memory speed	0x44419000
Watermark for 8 bpp at 133-MHz local memory speed	0x2210C000
Watermark for 16 bpp at 133-MHz local memory speed	0x22318000
Watermark for 24 bpp at 133-MHz local memory speed	0x44517000
MSR Mask	0x00

Parameters for Screen Resolution/Refresh Rate:	**1600x1200_65Hz =**
Dot Clock Value	175 //175.5-MHz Dot Clock
M Value	0x005D
N Value	0x0018
P Value	0x10
CR00	0x09
CR01	0xC7
CR02	0xC7
CR03	0x8D
CR04	0xCF
CR05	0x07
CR06	0xE0
CR07	0x10
CR09	0x40
CR10	0xB0
CR11	0x03
CR12	0xAF
CR13	0xC8
CR15	0xAF
CR16	0xE1
CR30	0x04
CR31	0x04
CR32	0x04
CR33	0x04
CR35	0x01
CR39	0x00
Watermark for 8 bpp at 100-MHz local memory speed	0x2210C000
Watermark for 16 bpp at 100-MHz local memory speed	0x22416000
Watermark for 24 bpp at 100-MHz local memory speed	0x44419000
Watermark for 8 bpp at 133-MHz local memory speed	0x00000000
Watermark for 16 bpp at 133-MHz local memory speed	0x00000000
Watermark for 24 bpp at 133-MHz local memory speed	0x00000000
MSR Mask	0x00

Parameters for Screen Resolution/Refresh Rate:	**1600x1200_70Hz =**
Dot Clock Value	189 //189-MHz Dot Clock
M Value	0x003D
N Value	0x000E
P Value	0x10
CR00	0x09
CR01	0xC7
CR02	0xC7
CR03	0x8D
CR04	0xCF
CR05	0x07
CR06	0xE0
CR07	0x10
CR09	0x40
CR10	0xB0
CR11	0x03
CR12	0xAF
CR13	0xC8
CR15	0xAF
CR16	0xE1
CR30	0x04
CR31	0x04
CR32	0x04
CR33	0x04
CR35	0x01
CR39	0x00
Watermark for 8 bpp at 100-MHz local memory speed	0x2220E000
Watermark for 16 bpp at 100-MHz local memory speed	0x22416000
Watermark for 24 bpp at 100-MHz local memory speed	0x44419000
Watermark for 8 bpp at 133-MHz local memory speed	0x22110000
Watermark for 16 bpp at 133-MHz local memory speed	0x22416000
Watermark for 24 bpp at 133-MHz local memory speed	0x00000000
MSR Mask	0x00

Parameters for Screen Resolution/Refresh Rate:	1600x1200_72Hz =
Dot Clock Value	195 //195-MHz Dot Clock
M Value	0x003F
N Value	0x000E
P Value	0x10
CR00	0x0B
CR01	0xC7
CR02	0xC7
CR03	0x8F
CR04	0xD5
CR05	0x0B
CR06	0xE1
CR07	0x10
CR09	0x40
CR10	0xB0
CR11	0x03
CR12	0xAF
CR13	0xC8
CR15	0xAF
CR16	0xE2
CR30	0x04
CR31	0x04
CR32	0x04
CR33	0x04
CR35	0x01
CR39	0x00
Watermark for 8 bpp at 100-MHz local memory speed	0x2220E000
Watermark for 16 bpp at 100-MHz local memory speed	0x22416000
Watermark for 24 bpp at 100-MHz local memory speed	0x44419000
Watermark for 8 bpp at 133-MHz local memory speed	0x22210000
Watermark for 16 bpp at 133-MHz local memory speed	0x22416000
Watermark for 24 bpp at 133-MHz local memory speed	0x00000000
MSR Mask	0x00

Parameters for Screen Resolution/Refresh Rate:	**1600x1200_75Hz =**
Dot Clock Value	202 //202.5-MHz Dot Clock
M Value	0x0024
N Value	0x0007
P Value	0x10
CR00	0x09
CR01	0xC7
CR02	0xC7
CR03	0x8D
CR04	0xCF
CR05	0x07
CR06	0xE0
CR07	0x10
CR09	0x40
CR10	0xB0
CR11	0x03
CR12	0xAF
CR13	0xC8
CR15	0xAF
CR16	0xE1
CR30	0x04
CR31	0x04
CR32	0x04
CR33	0x04
CR35	0x01
CR39	0x00
Watermark for 8 bpp at 100-MHz local memory speed	0x2220E000
Watermark for 16 bpp at 100-MHz local memory speed	0x22416000
Watermark for 24 bpp at 100-MHz local memory speed	0x44419000
Watermark for 8 bpp at 133-MHz local memory speed	0x22210000
Watermark for 16 bpp at 133-MHz local memory speed	0x22418000
Watermark for 24 bpp at 133-MHz local memory speed	0x00000000
MSR Mask	0x00

Parameters for Screen Resolution/Refresh Rate:	**1600x1200_85Hz =**
Dot Clock Value	229 //229.5-MHz Dot Clock
M Value	0x0029
N Value	0x0007
P Value	0x10
CR00	0x09
CR01	0xC7
CR02	0xC7
CR03	0x8D
CR04	0xCF
CR05	0x07
CR06	0xE0
CR07	0x10
CR09	0x40
CR10	0xB0
CR11	0x03
CR12	0xAF
CR13	0xC8
CR15	0xAF
CR16	0xE1
CR30	0x04
CR31	0x04
CR32	0x04
CR33	0x04
CR35	0x01
CR39	0x00
Watermark for 8 bpp at 100-MHz local memory speed	0x22210000
Watermark for 16 bpp at 100-MHz local memory speed	0x22416000
Watermark for 24 bpp at 100-MHz local memory speed	0x00000000
Watermark for 8 bpp at 133-MHz local memory speed	0x00000000
Watermark for 16 bpp at 133-MHz local memory speed	0x00000000
Watermark for 24 bpp at 133-MHz local memory speed	0x00000000
MSR Mask	0x00