P9/P10®

Reference Guide Volume II

I/O Registers

DRAFT

3Dlabs®

P9/10®

Reference Guide Volume II -

I/O Registers

PROPRIETARY AND CONFIDENTIAL INFORMATION

Issue 2

Proprietary Notice

Email: info@3dlabs.com
Web: http://www.3dlabs.com

3D*labs*Ltd.
Meadlake Place
Thorpe Lea Road, Egham
Surrey, TW20 8HE
United Kingdom
Tel: +44 (0) 1784 470555
Fax: +44 (0) 1784 470699

3D*labs*K.K.
Shiroyama JT Mori Bldg 16F
40301 Toranomon
Minato-ku, Tokyo, 105, Japan
Tel: +81-3-5403-4653
Fax: +91-3-5403-4646

3D*labs*Inc.
480 Potrero Avenue
Sunnyvale, CA 94086,
United States
Tel: +1 (408) 530-4700
Fax: +1 (408) 530-4701

Change History

Document	Issue	Date	Change
174.2.2 01	1	07 June 2001	Creation
174.2.2 02	2	28 Feb 2002	Joint P9+P10 edition = P10 base issue plus: • Added **AGPDriveStrength** register (new) to PCICSR • LUT bookmarks • Changes to **BlockControl** reg • Added Texture field to MemoryTranslationEnable • Removed **AltTimingControlReg**, • **MemControl** register updates • Miscl. Video register corrections; addresses changed from rel to absolute • Corrected Latency bit in **VideoControl0** • Reserved 16-bit digital video mode bits in video port Mode • Added byRegionWidth • Clarify **ClkOutControl** of *GenLockClkOut* pin • PCIPll Drop bit • P9 configured registers • Corrected **VideoScale** conversion formula

User Note

This manual uses hyperlinks in MSWord file distributions to improve ease of access to relevant information for online users. To enable hyperlinks, the complete *Reference Guide* and *Programmer's Guide* file set need to be in a single Windows directory or folder.

This manual uses hyperlinks in MSWord file distributions to improve ease of access to relevant information for online users. To enable hyperlinks, the complete *Reference Guide* and *Programmer's Guide* file set should be in a single Windows directory or folder.

Where both P10 and P9 are in use, they should be placed in a common directory. All crosslinks use relative addressing. Configured variants are color-coded, e.g.:

Register Title [P9] - Indicates an entre configured register exclusive to a device type(s)
Configured content [P9] – Indicates a configured field or other content

Minor changes are released progressively but only rolled up into a major release level when the volume of changes warrants it. Readers are therefore advised to check the file dates and download the current version of their document set from the 3Dlabs website to ensure accuracy.

Table of Contents

5

Hardware Registers

This manual uses hyperlinks in **MSWord** file distributions to improve ease of access to relevant information for online users. To enable hyperlinks, the complete *Reference Guide* and *Programmer's Guide* file set should be in a single Windows directory or folder.

Where both P10 and P9 are in use, they should be placed in a common directory. All crosslinks use relative addressing. Configured variants are color-coded:

- *Register Title [p9]* - Indicates a configured register
- Configured content [P9] – Indicates a field or other content which is differently configured for P9, P10 etc.
- References to "P10" should be considered applicable to P9 unless there is a specific configuration variant.

Minor changes are released progressively but only rolled up into a major release level when the volume of changes warrants it. Readers are therefore advised to check the file dates and download the current version of their document set from the 3Dlabs website to ensure accuracy.

Chapter 5 lists P9/P10 hardware registers by region and functional offset group. Within each group, the registers are listed alphanumerically. Exceptionally, graphics core "software" registers (offset 8000-9FFF) are shown in chapter 6. Volumes II and III are P9/P10 common data with configuration notes to clarify differences between the two. See for example **PrimSetupMode** in volume III, Core Graphics.

Register details have the following format information:

Name	The register's name. Where register layout is configured for P9 the register name is shown in **red** in online versions or dark gray in printed versions.
Type	The region in which the register functions.
Offset	The offset of this register from the base address of the region.
Format	Can be bitfield or integer.
Bit	Bit Name
Read	Indicates whether the register bit can be read from. A ✓ mark indicates the register can be read from, a ✕ indicates the register bit is not readable.
Write	Indicates whether the register bit can be written to. A ✓ mark indicates the register can be written to, a ✕ indicates the register bit is not writable.
Reset	The value of the register following hardware reset.
Description	In the register descriptions:
Reserved Bits	Indicates bits that may be used in future members of the Permedia family. To ensure upwards compatibility software should not assume a value for these bits when read.
Not Used/ Unused Bits	Indicates bits that are adjacent to numeric fields. These may be used in future members of the Permedia family, but only to extend the dynamic range of these fields. The data returned from a read of these bits is undefined. When a Not Used field resides in the most significant position, a good convention to follow is to sign extend the numeric value, rather than masking the field to zero before writing the register. This will ensure compatibility if the dynamic range is increased in future.
Reserved Registers	Write accesses to reserved registers are accepted by the bus interface but the data is discarded. Read accesses return 0. Data written to reserved registers is never forwarded.
Configured Registers	The register name is shown in ***red italics*** in online versions (or gray on printed manuals). The configured fields or data are shown with yellow background in online versions.

For enumeration fields that do not specify the full range of possible values use only the specified values. An example of an enumeration field is the comparison field in the **DepthMode** register. Future chips may define a meaning for the unused values.

5.1.1 PCI Address Regions

The PCI Slave interface implements six PCI Address Regions, shown in the table below. The standard VGA compatible Memory and I/O Space addresses are decoded when the device has been suitably configured. These addresses do not form a single contiguous region, but are mentioned in the table for completeness:

PCI Address Regions

Region	Address Space	Size (bytes)	Description	Comments
Config	Configuration	256	PCI Configuration	PCI Special
Zero	Memory	256 K	Control Registers	relocatable
One	Memory	configured	Memory Aperture One	relocatable
Two	Memory	configured	Memory Aperture Two	relocatable
ROM	Memory	64 K	Expansion ROM	relocatable
VGA	Memory & I/O	-	VGA Address	optional & fixed

5.1.2 PCI Configuration Space

The PCI Configuration Space is intended to provide an appropriate set of configuration 'hooks' which satisfy the needs of current and anticipated system configuration mechanisms. The registers in this 256-byte space are accessed and modified by the use of PCI Configuration Read and Write commands, and are normally initialised by BIOS or similar low-level code at system power-up and reset.

When configured for multi-function operation the bus interface provides a unique 256-byte configuration space for each PCI function, but will map accesses to other regions to the same underlying hardware regardless of the function being addressed.

5.1.3 Region Zero Control Registers

Region Zero is a 256 KByte region containing control registers, and ports to and from the graphics processor. The control space is mapped twice within the 256 KByte region. In the second 128K the registers are mapped to be byte swappable for big endian hosts. See Section 3 of this document for further details of Region Zero.

5.1.4 Memory Apertures

Two separate apertures are provided to allow access to local memory. Each has a programmable size and can be disabled if required.

As well as being used to access local memory, these two apertures can also be programmed to allow reading and writing of the Expansion ROM. This ensures that the "ROM" is visible beyond system boot time, allowing an EEPROM device to be reprogrammed in the field. Finally, either aperture can be programmed to forward memory accesses to the VGA memory controller.

5.2 PCI Bus Interface Registers

The bus interface contains a number of PCI Configuration Registers, and also various Control Status registers for the chip, with accesses to these registers being handled entirely by the bus interface unit.

The bus interface also accesses the read-back path from the Graphics Processor. This is used to access registers in the GP without passing through the pipeline. The read-back operates by sending a tag to the read-back port and waiting a set number of clocks. When the delay has expired, data from the register corresponding to the tag will be present on the read-back port.

A separate port is provided to forward accesses to the RAMDAC Interface. The VGA unit is accessed via the Bypass FIFO. A separate port is also provided to access the EEPROM and VMI interface. The related registers are described below.

5.2.1 Reset

During soft reset the PCI Bus Region 0 registers are reset with the GP input and output FIFOs. However the bus master and slave state machines continue to run, which can result in the PCI trying to load the GPInFIFO during a reset. For details on Configuration and the Reset process see *P9/P10 Reference Guide* Volume IV, *Reset*.

Driver software must write to a PCI configuration register to disable the bus master before asserting a software reset. This ensures that the master is not trying to load the GP Input FIFO during a reset.

Note: When bus retries are disabled, the current implementation accepts and then discards all write accesses to the GP Input FIFO — this is different from the manner in which Bypass accesses are handled. The situation only occurs if driver software performs a soft reset and does not check that it has completed before writing to the FIFO.

5.3 PCI Configuration Region (0x00-0xFF)

The registers in this region have the following functions:

- Read device configuration from ROM following a bus reset.
- Decode PCI register inputs and generate control signals for the device.
- Compare incoming slave addresses with PCI Base Addresses.

When configured for multi-function operation the bus interface provides a unique 256-byte configuration space for each PCI function, but will map accesses to other regions to the same underlying hardware regardless of the function being addressed.

For ease of reference the configuration registers are categorized as:

1. Predefined
2. Base Address
3. AGP
4. Power Management
5. Multi-function
6. Indirect

5.3.1 Predefined

This section describes the predefined registers in the standard Type 00h Configuration Space header

CFGVendorID

Name	Type	Offset	Format
CFGVendorID	Configuration	0x00	Integer
	Control register		

Bits	Name	Read	Write	Reset	Description
0..15	Vendor ID	✓	×	0x3D3D	3Dlabs Company Code
16..31					See CFGDeviceID

Notes: Vendor Identification Number, 3D3D = **3D**labs Company code

CFGDeviceID

Name	Type	Offset	Format
CFGDeviceID	Configuration	0x02	Integer
	Control register		

Bits	Name	Read	Write	Reset	Description
0..15	DeviceID	✓	×	0x0024	[P9] = 0024h when **AltDeviceId** is 0 = 0026h when **AltDeviceId** is 1
		✓	×	0x0020	[P10] = 0020h when **AltDeviceId** is 0 = 0022h when **AltDeviceId** is 1
16..31	Reserved	×	×		

Notes: **AltDeviceId** 0 = 3Dlabs P9/P10 device, **AltDeviceId** 1 = alternative device

CFGCommand

Name	Type	Offset	Format
CFGCommand	Configuration	0x04	Bitfield

Control register

Bits	Name	Read	Write	Reset	Description
0	I/O Space Enable	✓	×	0	0 = Disable Accesses 1 = Enable Accesses If *VgaEnable* and *VgaFixed* are not both set, then fixed VGA addressing is disabled and this bit will be zero (read only).
1	Memory Space Enable	✓	✓	0	0 = Disable Accesses 1 = Enable Accesses
2	Bus Master Enable	✓	✓	0	0 = Disable access 1 = Enable access
3	Special Cycle Enable	✓	×	0	0 - P10 never responds to special cycle accesses
4	Memory Write and Invalidate Enable	✓	×	0	0 = "Memory Write and Invalidate" is never generated.
5	SVGA Palette Snoop Enable	✓	✓	0	0 = Treat palette accesses like other SVGA accesses 1 = Enable SVGA Palette snooping If VgaEnable and VgaFixed are not both set, then fixed VGA addressing will be disabled and this bit will be zero (read only).
6	Parity Error Response enable	✓	×	0	0: P9/P10 does not support parity error reporting
7	Address/Data stepping enable	✓	×	0	0: P9/P10 does not perform stepping
8	SERR driver enable	✓	×	0	0: P9/P10 does not support parity error reporting
9	Master Fast Back-to-Back Enable	✓	×	0	0: P9/P10 master does not do fast back-to-back accesses
10..15	Reserved	✓	×	0	

Notes: The command register provides control over a device's ability to generate and respond to PCI cycles. Writing 0 to a field in this register disconnects the specified device from the PCI for all except configuration accesses.

CFGStatus

Name	Type	Offset	Format
CFGStatus	Configuration	0x06	Bitfield

Control register

Bits	Name	Read	Write	Reset	Description
0...3	Reserved	✓	×	from PADPCIClk 66 pin	
4	Cap_List	✓	✓	1	1 = Capabilities Pointer is implemented from PCI 2.2 Spec.
5	66MHz Capable	✓	×	From pin	Bit 5 is set to value of PADPCIClk66 pin[1] where 0 = device is 33 MHz capable only 1 = device is 66 MHz capable
6	Reserved	✓	×	0000b	
7	Fast back-to-back capable	✓	×	1	1 = device can accept fast back-to-back PCI transactions
8	Master Data Parity Error	✓	×	0	device does not implement parity checking
9...10	DEVSEL Timing	✓	×	01b	0xib = device asserts DEVSEL# at medium speed
11	Signaled Target Abort	✓	×	0	device never signals Target-Abort
12	Received Target Abort	✓	✓	1	This bit is set by the bus master whenever its transaction is terminated with Target-Abort
13	Received Master Abort	✓	✓	1	This bit is set by the R5 bus master whenever its transaction is terminated with Master-Abort
14	Signalled System Error	✓	×	0	device never asserts a system error
15	Detected Parity Error	✓	×	0	device does not implement parity checking

Notes: The Status register is used to record status information for PCI bus related events. Reads to this register behave normally. Writes function differently in that bits can be reset, but not set ("Write-to-clear"). A bit is reset whenever the register is writen and the data in the corresponding bit location is a one.

[1] This uses a dual-purpose configuration pin. On P10 this is the *VidAHSync* pin. On P9 it is the *VidInData(0)* pin. For more information see the Reset chapter and Pinlist in *Reference Guide* volume IV.

CFGRevisionID

Name	Type	Offset	Format
CFGRevisionID	Configuration	0x08	Integer

Control register

Bits	Name	Read	Write	Reset	Description
0..7	RevisionID	✓	×	0x1	Revision Identification Number - 0x01 = Revision R01

Notes:

CFGClassCode[InterfaceClass]

Name	Type	Offset	Format
CFGClassCode[InterfaceClassCode]	Configuration	0x09	Bitfield

Control register

Bits	Name	Read	Write	Reset	Description
0..7	Interface	✔	×	Configured	Lower byte of ClassCode register. 00 = VGA or Other Display Controller

Notes: The lower byte of the **ClassCode** register identifies a specific register-level programming interface, so that device-independent software can interact with the device. The reset value of this register is determined by **CFGBusConfig**.

CFGClassCode

The Class Code register is read-only, and is used to identify the generic function of the PCI device. The register is best viewed as three byte-sized sub-registers, detailed below. The reset value of this register is determined by the contents of the CFGBusConfig register as follows:

CFGBusConfig				CFGClassCode			Meaning (see PCI 2.2 Specification Appendix D)
Base Class Zero	Vga Enable	Vga Fixed	SubClass3D (x = 0 or 1)	Base Class	Sub Class	Interface	
0	0	x	0	03h	80h	00h	"other" display controller
0	0	x	1	03h	02h	00h	3D controller
0	X	0	0	03h	80h	00h	"other" display controller
0	X	0	1	03h	02h	00h	3D controller
0	1	1	x	03h	00h	00h	VGA-compatible controller
1	0	x	x	00h	00h	00h	non VGA-compatible device
1	X	0	x	00h	00h	00h	non VGA-compatible device
1	1	1	x	00h	01h	00h	VGA-compatible device

- If the *BaseClassZero* bit in the **CFGBusConfig** register is zero, the Base Class is reported as 03h, since this device is a PCI display controller.
- If this bit is 1 (one) then the Base Class is reported as 00h, which allows Windows 95 to boot even though it may not interpret display controller class codes correctly.
- If the *VgaEnable* and *VgaFixed* bits are both one, the device is a VGA controller and fixed VGA address decoding will be enabled.

CFGClassCode[SubClass]

Name	Type	Offset	Format
CFGClassCode[SubClass]	Configuration	0x0A	Bitfield

Control register

Bits	Name	Read	Write	Reset	Description
0..7	SubClass	✔	✕	Configured	Middle byte of ClassCode register

Notes: The middle byte of the **ClassCode** register identifies the function of the device in more detail. The reset value of this register is determined by **CFGBusConfig**.

CFGClassCode[BaseClass]

Name	Type	Offset	Format
CFGClassCode[BaseClass]	Configuration	0x0B	Bitfield
	Control register		

Bits	Name	Read	Write	Reset	Description
0..7	BaseClass	✔	✕	configured	Upper byte of ClassCode register, classifies function type

Notes: The ClassCode register is read-only, and is used to identify the generic function of the device (see CFGClassCode table for definition) The register is best viewed as three byte-sized sub-registers including *Subclass* and *Interfaceclass*. The reset value is determined by **CFGBusConfig**

CFGCacheLine

Name	Type	Offset	Format
CFGCacheLineSize	Configuration	0x0C	Integer
	Control register		

Bits	Name	Read	Write	Reset	Description
0...7	Cache Line Size	✓	✕	0x00	00= Cache line size (not supported)

Notes: This register specifies the cache line size in units of 32 bit words. It is only implemented for PCI bus masters that use the "memory write and invalidate" command. The PCI bus master does not use this command so this register is zero and read-only

CFGLatencyTimer

Name	Type	Offset	Format
CFGLatencyTimer	Configuration	0x0D	Integer
	Control register		

Bits	Name	Read	Write	Reset	Description
0..7	Latency Timer Count	✓	✓	0x00	Sets the max number of PCI Clock cycles for master burst accesses

Notes: This register specifies, in PCI bus clocks, the value of the latency timer for this PCI bus master

CFGHeaderType

Name	Type	Offset	Format
CFGHeaderType	Configuration	0x0E	Integer
	Control register		

Bits	Name	Read	Write	Reset	Description
0..6	Header Type.	✓	×	from **CFGFunConfig**	0x00 = type 00 header layout
7	Multifunction	✓	×	0	PCI Definition: 0 = Single Function Device 1 = Multifunction device

Notes: The register identifies the layout of the second part of the predefined header (beginning at byte 10h in Configuration Space) and whether or not the device contains multiple functions. The reset value is set in **CFGFunConfig**.

CFGBuilt In Self-Test

Name	Type	Offset	Format
CFGBIST	Configuration	0x0F	Integer
	Control register		

Bits	Name	Read	Write	Reset	Description
0...7	BIST	✓	×	0x00	00 = BIST is unsupported over the PCI interface

Notes: Optional register used for control and status of Built-In Self Test (BIST).

5.3.2 Base Address Registers (0x10 – 0x28)

The Base Address registers allow boot software to relocate PCI devices in memory address space. At system power-up, device-independent software must be able to determine what devices are preset, build a consistent address map, and determine if a device has an Expansion ROM. These Base Address registers allow power-up software to determine the size of each Region, and set its base address within the memory map.

The predefined type 00h configuration header has six DWORD locations allocated for Base Address registers, starting from offset 10h in Configuration Space. This PCI device implements three Base Address registers, and the width of these registers is determined by the *PciAddress64* field in the **CFGBusConfig** register. The first Base Address register **CFGBaseAddr0** is always located at offset 10h. The offsets of the subsequent registers **CFGBaseAddr1** and **CFGBaseAddr2** are determined by the size of previous Base Address registers.

CFGCardBusCISPointer

Name	Type	Offset	Format
CFGCardBusCISPointer	Configuration	0x28	Integer
	Control register		

Bits	Name	Read	Write	Reset	Description
0..31	CardBus CIS Pointer	✓	×	0x000 0.0000	Not implemented

Notes: This register is optional and is not implemented.

CFGSubsystemVendorId

Name	Type	Offset	Format
CFGSubsystemVendorId	Configuration	0x2C	Integer
	Control register		

Bits	Name	Read	Write	Reset	Description
0..15	Subsystem VendorID	✓	×	0x3D3D	…or loaded from ROM

Notes: This register is used to identify the vendor of the add-in board or subsystem where the PCI device resides, and is normally loaded from the ROM during device initialisation – see Chapter 10, *Reset*, in *Reference Guide* Volume IV.

CFGSubsystemId

Name	Type	Offset	Format
CFGSubsystemId	Configuration	0x02E	Integer
	Control register		

Bits	Name	Read	Write	Reset	Description
0..15	SubsystemId	×	✓ once	0x0020	…or loaded from ROM

Notes: This register is used to identify the add-in board on which the PCI device resides. The reset value is normally loaded from ROM during device initialisation – see Chapter 10, *Reset*, in *Reference Guide* Volume IV.

CFGCapabilitiesPtr

Name	Type	Offset	Format
CFGCapabilitiesPtr	Configuration	0x34	Integer
	Control register		

Bits	Name	Read	Write	Reset	Description
0..7	Capability Ptr	✓	×	0x4C	Pointer to Power Management capability, address 0x4C.
8..31	Reserved	×	×	0	

Notes: This register holds an eight bit pointer used to provide an offset into the configuration space for the first item in a capabilities list of one of more configuration register sets, each of which supports a new feature or capability.

CFGInterruptLine

Name	Type	Offset	Format
CFGInterruptLine	Configuration	0x3C	Integer
	Control register		

Bits	Name	Read	Write	Reset	Description
0..7	Interrupt Line	✓	✓	0	Not read or written by this device itself.

Notes: The Interrupt Line register in an 8-bit register used to communicate interrupt line routing information. It is available for use by device drivers and operatiing systems but is not used by the PCI device itself..

CFGInterruptPin

Name	Type	Offset	Format
CFGIntPin	Configuration	0x3D	Integer
	Control register		

Bits	Name	Read	Write	Reset	Description
0...7	Interrupt Pin	✓	×	0x01	0x01 = uses Interrupt pin INTA#

Notes: The Interrupt Pin register tells the BIOS which interrupt line this device uses.

CFGMinGnt

Name	Type	Offset	Format
CFGMinGrant	Configuration	0x3E	Integer
	Control register		

Bits	Name	Read	Write	Reset	Description
0-7	Minimum grant	✓	×	0xC0	0xC0 = 48 microseconds

Notes: This register specifies how long a burst period the PCI device needs.

CFGMaxLat

Name	Type	Offset	Format
CFGMaxLat	Configuration	0x3F	Integer
	Control register		

Bits	Name	Read	Write	Reset	Description
0...7	Maximum latency	✓	×	0xC0	0xC0 = 48 microseconds

Notes: This register specifies how often the PCI device needs to gain access to the PCI bus.

5.3.3 Expansion ROM Registers (0x30 – 0x34)

The registers allow boot software to determine if a device has an Expansion ROM. These Base Address registers allow power-up software to determine the size of each Region, and set its base address within the memory map.

The predefined type 00h configuration header has six DWORD locations allocated for Base Address registers, starting from offset 10h in Configuration Space. This PCI device implements three Base Address registers, and the width of these registers is determined by the *PciAddress64* field in the **CFGBusConfig** register. The first Base Address register **CFGBaseAddr0** is always located at offset 10h. The offsets of the subsequent registers **CFGBaseAddr1** and **CFGBaseAddr2** are determined by the size of previous Base Address registers.

5.3.3.1 32- and 64-bit Base Address Registers

When **PciAddress64** is zero, the three Base Address registers are all 32 bits wide. When **PciAddress64** is one, the three Base Address registers are all 64 bits wide.

Definitions for both the 32-bit and the 64-bit versions of each Base Address register are given below, although obviously only one width will be visible at any given time depending on the value of **PciAddress64**. When the Base Address registers are only 32 bits wide, the three DWORD locations starting at offset 1Ch are always zero.

CFGBaseAddress0 [32-bit]

Name	Type	Offset	Format
CFGBaseAddr0	Configuration	0x10	Bitfield
	Control register		

Bits	Name	Read	Write	Reset	Description
0	Memory Space Indicator	✓	×	See Notes	0 = Region Zero is in PCI memory space.
1..2	Address Type	✓	×	00b	00b = Locate anywhere in 32 bit address space
3	Prefetchable	✓	×	See Notes	0 = Region is not prefetchable.
4..17	Size Indication	✓	×	See Notes	0 = Control registers must be mapped into 256 Kbyte region.
18...31	Base Address	✓	✓	See Notes	Loaded by software at boot time to set base address of PCI Region 0

Notes: Bse address registers allow boot software to relocate PCI devices in memory. The predefined type 00h configuration header has six DWORD locations for Base Address registers, starting from offset 10h in Configuration Space. This PCI device implements three Base Address registers, and the width of these registers is determined by the **PciAddress64** field in the CFGBusConfig register. The first Base Address register **CFGBaseAddr0** is always located at offThe Base Address 0 register contains the base address of the Control Region and defines the size and type of this region. This register has a 32-bit format when **PciAddress64** = zero. Reset value is configured by **CFGBusConfig**. – see Chapter 10, *Reset*, in *Reference Guide* Volume IV.

CFGBaseAddress0 [64-bit]

Name	Type	Offset	Format
CFGBaseAddr0	Configuration	0x10	Bitfield
	Control register		

Bits	Name	Read	Write	Reset	Description
0	Memory Space Indicator	✓	×	See Notes	0 = Region Zero is in PCI memory space.
1..2	Address Type	✓	×	10b	10b = Locate anywhere in 64-bit address space
3	Prefetchable	✓	×	See Notes	0 = Region is not prefetchable.
4..17	Size Indication	✓	×	See Notes	0 = Control registers must be mapped into 256 Kbyte region.
18…63	Base Address	✓	✓	See Notes	These bits reset to zero and are loaded by software at boot time to set the base address of Region Zero.

Notes: This register has a 64-bit format when **PciAddress64** is one. The Base Address 0 register contains the base address of the Control Region and defines the size and type of this region. Reset value is configured by **CFGBusConfig**. – see Chapter 10, *Reset*, in *Reference Guide* Volume IV.

CFGBaseAddress1 [32-bit]

Name	Type	Offset	Format
CFGBaseAddr1	Configuration	0x14	Bitfield

Control register

Bits	Name	Read	Write	Reset	Description
0	Memory Space Indicator	✓	✕	See notes	0 = Region Zero is in PCI memory space.
1..2	Address Type	✓	✕	00b	00b = Locate anywhere in 32 bit address space
3	Prefetchable	✓	✕	See notes	= 0 when PciPrefetchable is 0 = 1 when PciPrefetchable is 1
4…M	Size Indication	✓	✕	See notes	These bits are read-only zero to indicate the region size.
N…31	BaseAddress	✓	✓	See notes	These bits reset to zero, and are loaded by software at boot time to set the base address of Region One where N = (Base1AddrSize + 16) and M = (N – 1).

Notes:

- The Base Address 1 register contains the base address of Memory Aperture One and defines the size and type of this region.
- The predefined type 00h configuration header has six DWORD locations allocated for Base Address registers, starting from offset 10h in Configuration Space. This PCI device implements three Base Address registers, and the width of these registers is determined by the *PciAddress64* field in the **CFGBusConfig** register.
- The first Base Address register **CFGBaseAddr0** is always located at offset 10h. The offsets of the subsequent registers **CFGBaseAddr1** and **CFGBaseAddr2** are determined by the size of previous Base Address registers.
- When **Base1AddrSize** is zero, this register is Zero and Read-Only.
- This register has a 32-bit format when **PciAddress64** = zero. Reset value is configured by **CFGBusConfig**. – see Chapter 10, *Reset*, in *Reference Guide* Volume IV.

CFGBaseAddress1 [64-bit]

Name	Type	Offset	Format
CFGBaseAddr1	Configuration	0x18	Bitfield

Control register

Bits	Name	Read	Write	Reset	Description
0	Memory Space Indicator	✓	✕	See notes	0 = Region One is in PCI memory space.
1..2	Address Type	✓	✕	10b	10b = Locate anywhere in 64 bit address space
3	Prefetchable	✓	✕	See notes	= 0 when PciPrefetchable is 0 = 1 when PciPrefetchable is 1
4…M	Size Indication	✓	✕	See notes	These bits are read-only zero to indicate the region size.
N…63	BaseAddress	✓	✓	See notes	These bits reset to zero, and are loaded by software at boot time to set the base address of Region One *where* N = (*Base1AddrSize* + 16) *and* M = (N – 1).

Notes: The **BaseAddress1** register contains the base address of Memory Aperture One and defines the size and type of this region. This register has a 64-bit format when **PciAddress64** = one. When **Base1AddrSize** is zero this register is **Zero** and **Read-Only**. The reset value is configured by **CFGBusConfig**. – see Chapter 10, *Reset*, in *Reference Guide* Volume IV.

CFGBaseAddress2 [32-bit]

Name	Type	Offset	Format
CFGBaseAddr0	Configuration	0x18	Bitfield

Control register

0	Memory Space Indicator	✓	✕	See notes	0 = Region Two is in PCI memory space.
1..2	Address Type	✓	✕	00b	00b = Locate anywhere in 32 bit address space
3	Prefetchable	✓	✕	See notes	= 0 when PciPrefetchable is 0 = 1 when PciPrefetchable is 1
4…M	Size Indication	✓	✕	See notes	These bits are read-only zero to indicate the region size.
N…31	BaseAddress	✓	✓	See notes	These bits reset to zero, and are loaded by software at boot time to set the base address of Region Two *where* N = (*Base2AddrSize* + 16) *and* M = (N – 1).

Notes: This register has a 32-bit format when **PciAddress64** is 0. The Base Address 2 register contains the base address of Memory Aperture Two and defines the size and type of this region. When *Base2AddrSize* is zero, this register is Zero and Read-Only. Reset value is configured by **CFGBusConfig**. – see Chapter 10, *Reset*, in *Reference Guide* Volume IV.

CFGBase Address2 [64-bit]

Name	Type	Offset	Format
CFGBaseAddr2	Configuration	0x20	Bitfield

Control register

Bits	Name	Read	Write	Reset	Description
0	Memory Space Indicator	✓	✕	See notes	0 = Region Two is in PCI memory space.
1..2	Address Type	✓	✕	10b	10b = Locate anywhere in 64 bit address space
3	Prefetchable	✓	✕	See notes	= 0 when PciPrefetchable is 0 = 1 when PciPrefetchable is 1
4…M	Size Indication	✓	✕	See notes	These bits are read-only zero to indicate the region size.
N…63	BaseAddress	✓	✓	See notes	These bits reset to zero, and are loaded by software at boot time to set the base address of Region One *where* N = (*Base1AddrSize* + 16) *and* M = (N – 1).

Notes: The **BaseAddress2** register contains the base address of Memory Aperture One and defines the size and type of this region. This register has a 64-bit format when **PciAddress64** = one. When **Base2AddrSize** is zero this register is **Zero** and **Read-Only**. The reset value is configured by **CFGBusConfig**. – see Chapter 10, *Reset*, in *Reference Guide* Volume IV.

CFGRomAddr

Name	Type	Offset	Format
CFGRomAddr	Configuration	0x30	Bitfield

Control register

Bits	Name	Read	Write	Reset	Description
0	Address Decode Enable	✓	✕	0x000 00000	0 = disable Expansion ROM accesses 1 = enable Expansion ROM accesses The device only responds to accesses to the Expansion ROM when both this bit and the "Memory Space" bit in the CFGCommand register are set.
1…10	reserved	✓	✕		0 = reserved
11…M	Size Indication	✓	✕		These bits are read-only zero to indicate the region size.
N…31	Expansion ROM Base Address	✓	✓		These bits reset to zero and are loaded by software at boot time to set the base address of the Expansion ROM where N = (RomAddrSize + 16) and M = (N – 1).

Notes: This register contains the Base Address of the Expansion ROM, in PCI memory space.

5.3.4 AGP (0x40 – 0x4B)

The AGP Capability, Status, and Control registers occupy three DWORDs starting at offset 40h.

The abbreviation *AgpCapable* is used to indicate the logical OR of the *Rate1XCapable*, *Rate2XCapable*, and *Rate4XCapable* fields in **CGIBusConfig**, and controls those PCI Fast Write and AGP capabilities which are independent of the data transfer rate.

CFGAGPCapID

Name	Type	Offset	Format
CFGAGPCapID	Configuration	0x40	Integer

Control register

Bits	Name	Read	Write	Reset	Description
0..7	Capability ID	✓	×	see desc.	Configured by AGP Capable 0x02 when AGP Capable = 1 (AGP Capability ID)

Notes: This register specifies whether the device has AGP capability. The reset value is loaded by the **CFGBusConfig** AGP Rate Capability fields where *Capable* is the logical OR of the three fields.

CFGAGPNextPtr

Name	Type	Offset	Format
CFGAGPNextPtr	Configuration	0x41	Integer

Control register

Bits	Name	Read	Write	Reset	Description
0..7	Capability ID	✓	×	0x00	Pointer to Next Capability 00h = no further capabilities in list

Notes: . The AGP Pointer to Next Capability register points to the next capability in the list.

CFGAGPRevision

Name	Type	Offset	Format
CFGAGPRevision	Configuration	0x42	Integer

Control register

Bits	Name	Read	Write	Reset	Description
0…3	Rate	✓	×	See Notes	= 00h when AgpCapable is 1 (Minor Rev 0)
4…7	Major Rev	✓	×	See Notes	= 00h when AgpCapable is 0 = 02h when AgpCapable is 1

Notes: The AGP Revision register specifies the revision of the the AGP spec the device is built to. The reset value is configured in **CFGBusConfig**

CFGAGPStatus

Name	Type	Offset	Format
CFGAGPStatus	Configuration	0x44	Integer

Control register

Bits	Name	Read	Write	Reset	Description
0…2	Rate	✓	×	See Notes	Bit map indicating data transfer rates supported by this device. Bit 0 = value of *Rate1XCapable* (1X transfers supported) Bit 1 = value of *Rate2XCapable* (2X transfers supported) Bit 2 = value of *Rate4XCapable* (4X transfers supported)
3	Reserved	✓	×		0 = reserved
4	FastWrite	✓	×	See Notes	If set, this device supports Fast Write transactions. = 0 when *AgpCapable* is 0 = value of *PciFWCapable* when *AgpCapable* is 1
5	4G	✓	×	See Notes	If set, this device supports addresses greater than 4GB.
6…8	Reserved	✓	×		0 = reserved
9	SBA	✓	×	See Notes	If set, this device supports sideband addressing. = 0 when *AgpCapable* is 0 = value of *SbaCapable* when *AgpCapable* is 1
10…23	Reserved	✓	×		0 = reserved
24…31	RQ	✓	×		Maximum number of AGP requests this device can manage. = 00h when *AgpCapable* is 0 = 1Fh when *AgpCapable* is 1 (32 outstanding requests)

Notes: The **AGPStatus** register describes which AGP features are supported by the device. It is a read-only register, and will always read back as zero when *AgpCapable* is not set in **CFGBusConfig**.

CFGAGPCommand

Name	Type	Offset	Format
CFGAGPCommand	Configuration	0x48	Integer

Command register

Bits	Name	Read	Write	Reset	Description
0…2	DATA_RATE	✓	✓		One (and only one) bit in this field must be set to indicate the desired data transfer rate. The same bit must be set on both master and target. Setting no bits or more than one but should disable AGP mastering. Setting this field to a value not supported in the CFGAGPStatus register should also disable AGP bus master operation. 1 = 1X transfer rate 2 = 2X transfer rate 4 = 4X transfer rate
3	Reserved	✓	×		0=reserved
4	FW_ENABLE	✓	✓		0 = use standard PCI protocol to receive memory space writes 1 = use PCI Fast Write protocol to receive memory space writes
5	4G_ENABLE	✓	✓		4G_ENABLE 0 = the AGP master must only generate 32-bit addresses 1 = enable AGP master addressing above 4G boundary
6, 7	Reserved	✓	×		0=reserved
8	AGP_ ENABLE	✓	✓		0 = disable AGP master operation 1 = enable AGP master operation
9	SBA_ENABLE	✓	✓		0 = disable Sideband Address mechanism 1 = enable Sideband Address mechanism If SBA is not set in the CFGAGPStatus register but SBA_ENABLE is set in this register, then the AGP bus master should be disabled.
10…23	Reserved	✓	×		0 = reserved
24…31	RQ_DEPTH	✓	✓		The value in this field should never exceed the value of RQ from the **CFGAGPStatus** register.The maximum queue depth used internally is the lower of *RQ* and *RQ_DEPTH* fields in case this field has been programmed incorrectly.

Notes: The **AGPCommand** register is programmed by operating system software to enable AGP operation and select which data rate and features to use. If *AgpCapable* is not set all writes to this register are discarded and the entire register should read back as zero.

5.3.5 Power Management

The power management registers support power states D0, D1, and D3. When a PCI function within the device is in any power state other than D0, decoding of slave I/O and memory accesses and the initiation of bus master transactions should be disabled for that function only. This is the equivalent of the I/O Space, Memory Space, and Bus Master bits in the **CFGCommand** register for that function being unset.

Configuration Space accesses must be decoded at all times, regardless of the power state. These requirements conform to Version 1.1 of the *PCI Power Management Specification*.

CFGPMCapID

Name	Type	Offset	Format
CFGPMCapID	Configuration *Control register*	0x4C	Integer

Bits	Name	Read	Write	Reset	Description
0..7	Capability ID	✓	×	see desc.	01h = Power Management Capability

Notes: The PM Capability ID register specifies that the device has Power Management Capability.

CFGPMNextPtr

Name	Type	Offset	Format
CFGPMNextPtr	Configuration *Control register*	0x4D	Integer

Bits	Name	Read	Write	Reset	Description
0..7		✓	×	See Notes	Pointer to Next Capability = 00h when *AgpCapable* is 0 (no more capabilities in list) = 40h when *AgpCapable* is 1 (pointer to AGP Capability)

Notes: The PM Pointer to Next Capability register points to the next capability in the list. Reset is configured in the **CFGBusConfig** AGP Capability fields. (The abbreviation *AgpCapable* is used to indicate the logical OR of the *Rate1XCapable*, *Rate2XCapable*, and *Rate4XCapable* fields in **CGIBusConfig**, and controls those PCI Fast Write and AGP capabilities which are independent of the data transfer rate.)

CFGPMC

Name	Type	Offset	Format
CFGPMC	Configuration	0x4E	Integer
	Control register		

Bits	Name	Read	Write	Reset	Description
0..2	Version	✓	×	0x222	010b = complies with Rev 1.1 of the PCI Power Management Interface spec
3	PME Clock	✓	×		0 = PME# is not supported in any state
4	Reserved	✓	×		0 = reserved
5	DSI	✓	×		1 = this device requires special initialization following transition to D0 uninitialized state
6…8	Reserved	✓	×		0 = reserved
9	D1_Support	✓	×		1 = D1 power state is supported
10	D2_Support	✓	×		0 = D2 power state is not supported
11…15	PME_Support	✓	×		0 = PME# signal is not asserted in any power state

Notes: The Power Management Capabilities (PMC) register.

CFGPMCSR

Name	Type	Offset	Format
CFGPMCSR	Configuration	0x50	Integer
	Control register		

Bits	Name	Read	Write	Reset	Description
0,1	PowerState	✓	✓		0 = D0 1 = D1 3 = D3(hot) Valid states are 0, 1 and 3
2…7	Reserved	✓	×		
8	PME_En	✓	×		0 = PME# signal is not asserted in D3(cold)
12…9	Data_Select	✓	×		0 = Data register not supported
14,13	Data_Scale	✓	×		0 = Data register not supported
15	PME_Status	✓	×		0 = PME# signal is not asserted in D3 (cold)

Notes: The Power Management Control/Status (PMCSR) register. If the value 2 is written to *PowerState* the write is discarded (power state D2 is not supported). When no PCI functions within the device are in the D0 power state the internal "ConfLowPower" control signal is asserted, and this is used to disable the generation of interupts and reduce the power consumption of the device.

CFGPMCSR_BSE

Name	Type	Offset	Format
CFGPMCSR_BSE	Configuration	0x52	Integer

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Bridgesupport	✓	×		00h = this device is not a PCI-to-PCI bridge

Notes: This register specifies the Power Management PCI-to-PCI bridge support

CFGPMData

Name	Type	Offset	Format
CFGPMData	Configuration	0x53	Integer

Control register

Bits	Name	Read	Write	Reset	Description
0…7	PowerMgmt Data	✓	×	0xx00	00h = this register is reserved but not implemented

Notes: This register implements the optional Power Management Data register

5.4 Multi-function registers

P9/P10 can be configured to operate as a multi-function device to support multi-head displays. The device includes the required PCI Configuration Space registers for **two** functions to match the number of video heads provided.

The user-defined **CFGFunConfig** register controls how the PCI is configured for multi-function operation. This register cannot be changed dynamically from software, but instead is loaded from external ROM after a hard reset. The value of the *MaxFunction* field is one bit wide in this implementation.

Previous sections of this document have described the Configuration Space registers as they appear in **function zero**. This section lists the differences between other functions and function zero, and how their registers interact to control the operation of each individual function and the device as a whole.

Note: Registers which are used to report status and capabilities have the same value in all functions and are not duplicated here.

5.4.1 Predefined Registers (Multi-function, function > zero)

See the function zero definitions (above) for predefined registers not listed here.

CFGDeviceID [function > zero]

Name	Type	Offset	Format
CFGDeviceID	Configuration	0x02	Integer
	Control register		

Bits	Name	Read	Write	Reset	Description
0..15	DeviceID	✓	✕	0x0024 See Notes	**[P9]** = 0024h when **MultiUniqDevId** is 0 and **AltDeviceId** is 0 (P9/P10 device) = 0025h when **MultiUniqDevId** is 1 and **AltDeviceId** is 0 (alternative) = 0026h when **MultiUniqDevId** is 0 and **AltDeviceId** is 1 (alternative) = 0027h when **MultiUniqDevId** is 1 and **AltDeviceId** is 1 (alternative)
0..15	DeviceID	✓	✕	0x0020 See Notes	**[P10]** = 0020h when **MultiUniqDevId** is 0 and **AltDeviceId** is 0 (P9/P10 device) = 0021h when **MultiUniqDevId** is 1 and **AltDeviceId** is 0 (alternative) = 0022h when **MultiUniqDevId** is 0 and **AltDeviceId** is 1 (alternative) = 0023h when **MultiUniqDevId** is 1 and **AltDeviceId** is 1 (alternative)
16..31	Reserved	✕	✕		

Notes: The DeviceID register contains the device identification number. The same Device ID is normally used for all functions, although a bit is provided in the **CFGFunConfig** register to give every function a unique Device ID (which is formed by adding the function number to the "standard" Device ID).
The reset value may also be loaded from ROM]

CFGClassCode[BaseClass] [function > zero]

Name	Type	Offset	Format
CFGClassCode[BaseClass]	Configuration	0x0B	Bitfield
	Control register		

Bits	Name	Read	Write	Reset	Description
0..7	BaseClass	✔	✕	configured	Upper byte of ClassCode register, classifies function type

Notes: The ClassCode register is read-only, and is used to identify the generic function of the device (see CFGClassCode table for definition) The register is best viewed as three byte-sized sub-registers including *Subclass* and *Interfaceclass*. The reset value is determined by **CFGBusConfig.**
Only function zero supports VGA operation so **VgaEnable** and *VgaFixed* are always assumed to be zero when generating the Class Code for any other function. This limits the available Class Codes to 3D controller or "other" display controller for functions other than zero.

CFGCommand [function > zero]

Each function has its own **CFGCommand** register, which controls the ability of that function to generate and respond to PCI bus cycles. When a zero is written to this register, the relevant function is logically disconnected from the bus for all accesses except configuration accesses.

Slave address decoding can therefore be disabled on a per-function basis by writing to the individual **CFGCommand** register for each function. However, all the functions share a common PCI Master unit and in some circumstances it is possible for bus mastering to be disabled for only one function. In this situation the bus master must continue to operate, and only be disabled when the Bus Master field has been cleared in the CFGCommand register for *every* function.

Only function zero is permitted to support VGA operation if enabled by the **CFGBusConfig** register. When VGA operation is enabled then the standard fixed VGA I/O and Memory Space addresses are decoded through function zero, and the slave response to these addresses is enabled and disabled by the **CFGCommand** register in that function. The **CFGCommand** registers in functions other than zero have no effect on VGA operation, and their I/O Space and VGA Palette Snoop fields are zero.

CFGCommand

Name	Type	Offset	Format
CFGCommand	Configuration	0x04	Bitfield

Control register

Bits	Name	Read	Write	Reset	Description
0	I/O Space	✓	✕	0x0000	0 = disable I/O space accesses 1 = enable I/O space accesses If *VgaEnable* and VgaFixed in the CFGBusConfig register are not both set, then fixed VGA addressing will be disabled and this bit will be zero (read only).
1	Memory Space	✓	✕		0 = disable memory space accesses for this function 1 = enable memory space accesses for this function
2	Bus Master	✓	✕		0 = disable master accesses for this function 1 = enable master accesses for this function
3	Special Cycles	✓	✕		0 = this device never responds to special cycle accesses
4	Memory Write and Invalidate Enable	✓	✕		0 = "Memory Write and Invalidate" is never generated
5	VGA Palette Snoop	✓	✕		0 = this function does not support VGA operation
6	Parity Error Response	✓	✕		0 = this device does not support parity error reporting
7	Address/Data Stepping	✓	✕		0 = this device does not perform address/data stepping
8	SERR# Enable	✓	✕		0 = this device does not support parity error reporting
9	Fast Back-to-Back Enable	✓	✕		0 = this device does not perform fast back-to-back accesses
10…15	Reserved	✓	✕		0 = reserved

Notes:

CFGStatus [function > zero]

See the function zero definition.

In a *single-function device* the *Received Target Abort* and *Received Master Abort* bits are set in the **CFGStatus** register following the abort condition regardless of whether the bus master is enabled in **CFGCommand**.

In a *multi-function device* all the functions in the device share a single PCI Master so the relevant abort bits are set in the **CFGStatus** register of *every* function where bus master operation is enabled in its **CFGComand**.

CFGLatencyTimer [function > zero]

See the function zero definition.

Each function has its own copy of the **CFGLatencyTimer** register. During multi-function operation the shared PCI Master uses the highest latency timer count from all functions with an enabled bus master.

CFGInterruptLine [function > zero]

See the function zero definition.

Each function has its own copy of the **CFGInterruptLine** register. When *MultiShareIntLine* = one then writing to **CFGInterruptLine** for any function will update this register for all functions, otherwise only the register for the function actually written is affected.

The registers in the Interrupt Controller are shared between all functions, and can be accessed through Region Zero which is always visible for each function. Any head-specific interrupts such as vertical blank must have a bit per head provided in the appropriate registers in the Interrupt Controller.

5.4.2 Base Address Registers (Multi-function, function > zero)

Each function has its own set of Base Address Registers. The 256 KByte Region Zero is always mapped in for every function. Configuration bits are provided in **CFGFunConfig** to disable the BARs for Regions One and Two in all functions other than zero, which may be useful to reduce the total bus address space consumed by multi-function configurations where both memory apertures do not need to be visible for every function.

CFGBaseAddr0 [function > zero]

See function zero definition.

This register always has the same format in all functions as **CFGBaseAddr0** in function zero. All slave accesses to Region Zero of any function are mapped through the bus interface to the same underlying "Region Zero" hardware in the device, regardless of which function is actually addressed.

CFGBaseAddr1 [function > zero]

See function zero definition.

When **MultiBar1Enable** = 0 this register is *Zero and Read Only* for functions other than zero.

When **MultiBar1Enable** = 1 this register has the same format in all functions as **CFGBaseAddr1** in function zero. All slave accesses to Region One of any function are mapped through the bus interface to the same underlying "Region One" hardware in the device, regardless of the function addressed.

CFGBaseAddr2 [function > zero]

See function zero definition.

When **MultiBar2Enable**=0 this register is *Zero and Read Only* for functions other than zero.

When **MultiBar2Enable**=1 this register has the same format in all functions as CFGBaseAddr2 in function zero. All slave accesses to Region Two of any function are mapped through the bus interface to the same underlying "Region Two" hardware in the device, regardless of the function addressed.

CFGRomAddr [function > zero]

The Expansion ROM Base Address Register can only be accessed through function zero, and is always *Zero and Read Only* for all other functions.

5.4.3 AGP Registers (Multi-function, function > zero)

Function zero supports AGP operation when enabled by the *AgpCapable* registers in **CFGBusConfig**.

Although not required by the *AGP Interface Specification* it is not impossible that some system software will look at the configuration space of a device and decide whether or not to allocate system resources based on its AGP capabilities. (For example, DirectDraw might decide that GART-based DMA services are not available to function One if it indicates that it is not AGP capable.) For this reason there are configuration bits in **CFGFunConfig** to make AGP registers visible in functions other than zero, and to configure whether the **CFGAGPCommand** register should be shared between all functions.

When **MultiAgpCapable** is zero all the AGP registers (**CFGAgpCapId**, **CFGAGPNextPtr**, **CFGAGPRevision**, **CFGAGPStatus**, and **CFGAGPCommand**) are *Zero and Read Only* for functions other than zero. When both **AgpCapable** and **MultiAgpCapable** = 1 then these registers are visible in all functions. Details of AGP registers not listed here can be found in the earlier function zero definitions.

CFGAGPCommand [function > zero]

See function zero definition.

Each function has its own copy of the CFGAGPCommand register. When **MultiShareAgpCmd** and **MultiAgpCapable** are both one then writing to the CFGAGPCommand register for any given function will update this register for all functions, otherwise only the register for the function actually written will be affected.

Note: *Regardless of how the other functions in a multi-function device are configured, the operation and mode of the AGP Master is only affected by the **CFGAGPCommand** register in function zero.*

5.4.4 Power Management Registers (Multi-function, function > zero)

All functions share the common PCI and AGP Bus Master units. These will only be disabled as a result of power management when there are no functions remaining in the D0 power state. The power state of the entire device reflects the power state of all functions. For example, if function zero is in power state D3 and function one is in power state D1 then the device is in power state D1.

See the earlier function zero definitions for Power Management registers not listed here.

CFGPMNextPtr [function > zero]

See function zero definition.

When **MultiAgpCapable** is zero this register *Zero and Read Only* for functions other than zero.

CFGPMCSR [function > zero]

See function zero definition.

Each function has its own copy of the **CFGPMCSR** register, and power states D0, D1, and D3 are *supported separately for each function*.

Putting a function into a power state other than D0 disables slave address decoding, bus mastering, and interrupt generation for that function only. Other functions still in power state D0 may continue to respond to slave accesses and generate interrupts and bus master transactions. Configuration accesses must be decoded at all times, regardless of power state.

5.5 Indirect PCI Space Access to Regions 0 – 3 and ROM

The **IndirectData, IndirectAddress**, and **IndirectTrigger** registers are used to access Regions Zero, One, Two, and the ROM region indirectly through PCI Configuration Space.

The region to be accessed and the offset into the region are programmed into the **IndirectAddress** register. Write data is loaded into the **IndirectData** register, and is written to the location pointed to by the **IndirectAddress** register when the **IndirectTrigger** register is written.

Reading the **IndirectTrigger** register returns the value at the location pointed to by the **IndirectAddress** register. The byte enables used for the internal read or write operation are taken from the actual bus transaction to the **IndirectTrigger** register.

Each of these three user-defined registers is shared between all functions in a multi-function device, and accesses through any function are mapped to the same underlying register hardware by the bus interface.

A similar approach is used for VGA Indirect Addressing, through VGA I/O Space. Examples are shown in Appendix 2, below.

CFGIndirectData

Name	Type	Offset	Format
CFGIndirectData	Configuration	0xF4	Integer
	Control register		

Bits	Name	Read	Write	Reset	Description
0…31	IndirectData	✓	✓	0x000 00000	

Notes: The **IndirectData** register is used to hold write data for indirect transfers using PCI Configuration Space. See the description above for details of how this register should be used.

CFGIndirectAddress

Name	Type	Offset	Format
CFGIndirectAddress	Configuration	0xF8	Integer
	Control register		

Bits	Name	Read	Write	Reset	Description
0…28	AddressOffset	✓	✓	0x000 0,0000	
29…31	RegionSelect				0 = select Region 0 1 = select Region 1 2 = select Region 2 3 = reserved 4 = reserved 5 = reserved 6 = reserved 7 = select ROM region Reserved values can be written to and read from this register, but will result in indirect writes being discarded and indirect reads returning zero.

Notes: The **IndirectAddress** register is used to hold the region to be accessed for indirect transfers using PCI Configuration Space, and the address offset within that region. See the text above for details of how this register should be used.

CFGIndirectTrigger

Name	Type	Offset	Format
CFGIndirectTrigger	Configuration	0xFC	Integer

Control register

Bits	Name	Read	Write	Reset	Description
0…31	Indirect Trigger	✓	✓	0x000 0,0000	

Notes: The **IndirectTrigger** register is used to trigger indirect accesses to the device using PCI Configuration Space. See the text above for details of how this register should be used.

5.6 Region 0 Reserved Registers (0x09000 – 0x0EFFF and 0x29000 – 0x2EFFF)

All accesses to reserved sub-regions in the table above are intercepted and handled by the bus interface: writes are discarded, and reads return zero. Accesses to non-reserved sections of the address map are forwarded to the appropriate target unit.

The bus interface has no information about the internal register map of individual target units, so where target units have a sparse register map they themselves are responsible for handling accesses to reserved registers. By convention, they too should absorb writes and read back zero from reserved addresses.

5.7 Region 0 Control Registers (0x0000-0x01FF)

Region Zero is a 256 KByte region containing control registers and ports to and from the graphics processor. The control space is mapped in two 128K ranges: in the second 128K the registers are mapped to be byte swappable for Big Endian hosts. See the *P9/P10 Reference Guide* volume I for further details of Region Zero.

The bus interface has its own internal set of CSR registers, which are described in detail in the PCI CSR Unit Specification. They include Reset, Power Management, and Bus Master control registers, but in a departure from previous 3Dlabs designs the interrupt and error registers now reside in a separate Interrupt Control unit which allows a much more generic and re-usable implementation of the bus interface CSR registers.

5.7.1 Bus Interface CSR (0x00000 – 0x00FFF)

ResetStatus

Name	Type	Offset	Format
ResetStatus	Bus Interface	0x00	Integer

Control register

Bits	Name	Read	Write	Reset	Description
0…30	Reserved	✓	×		0=reserved
31	Indirect Trigger	✓	✓	0x000 00000	Software Reset Flag 0 = graphics processor is ready for use 1 = graphics process is being reset and must not be used

Notes: Writing to this register resets the graphics processor software. It does ***not*** reset the bus interface. The reset takes a number of cycles to complete during which the graphics processor should not be used. A flag in the register shows that the software reset is still in progress.
Note: All PCI CSR registers are reset by this software reset unless explicitly stated otherwise.

PowerManagement

Name	Type	Offset	Format
PowerManagement	Bus Interface	0x08	Integer
	Control register		

Bits	Name	Read	Write	Reset	Description
0	SlavePM Enable	✓	✓		0 = respond to slave accesses in all power states 1 = respond to slave accesses only in State D0
1	MasterPM Enable	✓	✓		0 = allow bus masters to access the bus in all power states 1 = allow bus masters to access the bus only in State D0
2	InterruptPM Enable	✓	✓		0 = allow **INTA#** to be asserted in all power states 1 = allow **INTA#** to be asserted only in State D0
3	InterruptGate Mode	✓	✓		0 = de-assert the **INTA#** output on entering low-power mode 1 = allow **INTA#** to remain asserted during low-power mode if it was already asserted before entering low-power mode
4…31	Reserved	✓	×		0=reserved

Notes: This register controls the behaviour of the bus interface in power states other than the D0 state.

ApertureOne
ApertureTwo

Name	Type	Offset	Format
ApertureOne	Region Zero	0x10	Bitfield
ApertureTwo	Region Zero	0x18	Bitfield
	Control register		

Bits	Name	Read	Write	Reset	Description
0, 1	ApertureMode	✓	✓	0	0 = access the local memory directly 1 = access the memory through the VGA subsystem 2 = use this aperture to access the Expansion ROM 3 = reserved (access the local memory directly)
2..31	Reserved	✓	×	0	

Notes: Two memory apertures are provided, each being a PCI region with a configured size (see CFGBusConfig). . The **ApertureOne** and **ApertureTwo** registers allow the Apertures to be used to access the VGA or ROM instead of the memory controller. When the *VGAAccess* bit in either of the **ApertureOne** or **ApertureTwo** registers is set, then all accesses to the relevant aperture are forwarded to the VGA Unit rather than directly to the memory controller. Writing a "reserved" value to this register configures the aperture to access the memory directly, but this may change in future implementations.

BusErrorFlags

Name	Type	Offset	Format
BusErrorFlags	Region Zero	0x20	Bitfield

Control register

Bits	Name	Read	Write	Reset	Description
0	Soft Reset Access Error	✓	✓	0	0 = no error 1 = Graphics Core / Memory / VGA access discarded during Soft Reset
1	Completion Discard Error	✓	✓	0	0 = no error 1 = Delayed Completion discarded after master failed to repeat request
2	VGA Snoop Failure Error	✓	✓	0	0 = no error 1 = VGA Snoop failed as no buffer space available to receive the data
3	Target Abort Error	✓	✓	0	0 = no error 1 = PCI Master transaction terminated by Target Abort
4	Master Abort Error	✓	✓	0	0 = no error 1 = PCI Master transaction terminated by Master Abort
5..31	Reserved	✓	×	0	

Notes: The BusErrorFlags register shows which errors are outstanding in the bus interface. Flag bits are reset by writing to this register with the corresponding bit set to a one. Flags at positions where the bits are set to zero will be unaffected by the write.

BusErrorEnable

Name	Type	Offset	Format
ResetStatus	Region Zero	0x028	Bitfield

Control register

Bits	Name	Read	Write	Reset	Description
0	Soft Reset Access Enable	✓	✓		0 = disable 1 = enable bus error generation
1	Completion Discard Enable	✓	✓		0 = disable 1 = enable bus error generation
2	VGA Snoop Failure Enable	✓	✓		0 = disable 1 = enable bus error generation
3	Target Abort Enable	✓	✓		0 = disable 1 = enable bus error generation
4	Master Abort Enable	✓	✓		0 = disable 1 = enable bus error generation
5…31	Reserved	✓	×		0=reserved

Notes: The BusErrorEnable register selects which error conditions are allowed to generate a Bus Error interrupt signal to the Interrupt Controller Unit.

PciMasterControl

Name	Type	Offset	Format
PciMasterControl	Region Zero	0x030	Integer

Control register

Bits	Name	Read	Write	Reset	Description
0	RdConcat Enable	✓	✓	0x000 00003	0 = do not attempt to concatenate read requests 1 = concatenate adjacent read bursts on the bus
1	WrConcat Enable	✓	✓		0 = do not attempt to concatenate write requests 1 = concatenate adjacent write bursts on the bus
2…31	Reserved (Read Only)	✓	×		0=reserved

Notes: The PciMasterControl register is used to control the behaviour of the PCI Master. This register is not affected by the software reset caused by writing to the **ResetStatus** register

PciAbortStatus

Name	Type	Offset	Format
PciAbortStatus	Region Zero	0x038	Bitfield

Control register

Bits	Name	Read	Write	Reset	Description
0,1	Ident	✓	✓	0x000 00000	The identity of the DMA source which caused the Abort.This field is only valid when the Status field is not zero. 0 = Interrupt Controller 1 = GPIO Upload Unit 2 = Memory Controller
2…29	Reserved	✓	×		0=reserved
30…31	Status	✓	✓		0 = no transactions terminated by Abort 1 = Write transaction terminated by Abort 2 = Read transaction terminated by Abort

Notes: **PciAbortStatus** reports whether an operation initiated by the PCI Master in this device has been terminated with an abort on the bus. Only details of the first such abort are recorded, and are not overwritten by subsequent aborts until **PciAbortStatus** has been cleared. Writing any value to the **PciAbortStatus** register will clear it together with the **PciAbortAddrLo** and **PciAbortAddrHi** registers.

PciAbortAddrLo

Name	Type	Offset	Format
PciAbortAddrLo	Region Zero	0x040	Integer
	Control register		

Bits	Name	Read	Write	Reset	Description
0…31	AbortAddrLo	✓	✓	0x000 00000	Lower 32 address bits of aborted transaction. The contents of this register are only valid when the **PciAbortStatus** *Status* bit reports a read or write abort

Notes: This register records the lower 32 bits of the bus address which caused the abort recorded in the **PciAbortStatus** register's *Status* bit. Writing any value to **PciAbortStatus** clears the **PciAbortAddrLo** register

PciAbortAddrHi

Name	Type	Offset	Format
PciAbortAddrHi	Region Zero	0x048	Bitfield
	Control register		

Bits	Name	Read	Write	Reset	Description
0…31	AbortAddrHi	✓	✓	0x000 00000	Higher 32 address bits of aborted transaction. The contents of this register are only valid when "PciAbortStatus.Status" reports a read or write abort

Notes: This register records the higher 32 bits of the bus address which caused the abort recorded in **PciAbortStatus** register's *Status* bit. Writing any value to **PciAbortStatus** clears this register

AgpMasterControl

Name	Type	Offset	Format
AgpMasterControl	Region Zero	0x050	Bitfield

Control register

Bits	Name	Read	Write	Reset	Description
0	GntDisable	✓	✓	0x000 00003	0 = allow AGP Master to generate REQ# and receive GNT#s normally 1 = connect PCI Master directly to REQ# and GNT# (for debug only)
1	ReadThrottle	✓	✓		0 = use the RBF# pin to throttle start of low-priority read data transfers 1 = only request read data when space to receive it (RBF# never asserted)
2,3	RdBurstSize	✓	✓		Length of requested AGP Read transactions. All longer requests from DMA sources are broken up into a series of transactions of this length. 0 = 16 bytes (2 QuadWords) 1 = 32 bytes (4 QuadWords) 2 = 48 bytes (6 QuadWords) 3 = 64 bytes (8 QuadWords)
4,5	WrBurstSize	✓	✓		Length of requested AGP Write transactions. All longer requests from DMA sources are broken up into a series of transactions of this length. 0 = 16 bytes (2 QuadWords) 1 = 32 bytes (4 QuadWords) 2 = 48 bytes (6 QuadWords) 3 = 64 bytes (8 QuadWords)
6…31	Reserved	✓	×		0=reserved

Notes: This register is used to set up the behaviour of the AGP Master. It will normally be programmed during device initialisation and should not be modified during runtime operation unless the AGP Master is idle and there are no outstanding AGP Read or AGP Write requests to the target

PciPLLControl

Name	Type	Offset	Format
PciPLLControl	Region Zero	0x058	Bitfield

Control register

Bits	Name	Read	Write	Reset	Description
0…2	PciPLLSetup	✓	✓	0x800 0000F	7h = standard setup [reset value]
3	PciPLLEnable	✓	✓		1 = enabled [reset value]
4…30	Reserved [P10]	✓	×		0=reserved
4…29	Reserved [P9]	✓	×		0=reserved
30	PciPLLDrop	✓	✓	x	0 = lock maintained 1 = lock dropped
31	PciPLLLock	✓	×		0 = not locked 1 = locked

Notes: The **PciPLLControl** register is used to control the PLL which multiplies the incoming 15ns **CLK** signal from the bus to generate an internal 266MHz clock (this is required to transmit data and sideband addresses in AGP 2X and 4X transfer modes). The top register bit reports the PLL status. This register is not affected by the software reset caused by writing to the **ResetStatus** register

AgpAutoCalCount

Name	Type	Offset	Format
AgpAutoCalCount	Region Zero	0x060	Bitfield

Control register

Bits	Name	Read	Write	Reset	Description
0…11	AutoCal Count	✓	×	0x000 00FFF	FFFh = minimum count value
12…31	AutoCalCount	✓	✓		

Notes: The AgpAutoCalCount register controls the number of bus clocks between automatic calibrations of the output drivers for the **SBA[7::0]**, **AD[31::00]** and **C/BE[3::0]** bus interface pins while operating in AGP 4X transfer mode. The bottom 12 bits of this register are always set, ensuring a sensible minimum interval between calibration operations.
This register is not affected by the software reset caused by writing to the ResetStatus register

AgpDriveStrength

Name	Type	Offset	Format
AgpAutoCalCount	Region Zero	0x068	Bitfield

Control register

Bits	Name	Read	Write	Reset	Description
0…2	AutoCal Count	✓	✓	0x000 00000	SbDriveStrength: Programmed drive strength for **SBA[7::0]** outputs.
3	SbDriveSelect	✓	✓		Select SBA[7::0] output buffer drive strength. 0 = use automatically measured drive strength 1 = use programmed value of SbDriveStrength Bits 2-0
4…6	AdDrive Strength	✓	✓		Programmed drive strength for AD[31::00] and C/BE[3::0] outputs.
7	AdDriveSelect	✓	✓		Select AD[31::00] and C/BE[3::0] output buffer drive strength. 0 = use automatically measured drive strength 1 = use programmed value of AdDriveStrength
8…10	ZsDriveStrength (Read Only)	✓	×		Automatically measured drive strength for AGPZSET pin. The value of this field may change over time, as a result of changes in the device operating environment.
11	ZsDriveValid (Read Only)	✓	×		0 = drive strength currently being updated 1 = **ZsDriveStrength** register field is valid
12…31	Reserved	✓	×		

Notes: The AgpDriveStrength register is used to monitor the required bus interface output buffer drive strength, which is normally measured and updated automatically while operating in AGP 4X transfer mode. To assist with electrical debugging, the AgpDriveStrength register can also directly control the output drive strength of the bus interface pins regardless of the AGP transfer mode selected.
Note: This register is not affected by the software reset caused by writing to the ResetStatus register.

AgpCalibration

Name	Type	Offset	Format
AgpCalibration	Region Zero	0xF00-F7F	Integer

Control register

Bits	Name	Read	Write	Reset	Description
0…31	Reserved	✓	×	0x000 00000	0=reserved

Notes: This range of register offsets is reserved to receive dummy writes during automatic calibration of the output drivers for the **AD[31::00]** and **C/BE[3::0]** pins while operating in AGP 4X transfer mode. All data written to this address range is discarded, and all read operations return zero. This is the default behaviour for "reserved" registers and therefore does not require any special implementation

CoreControl

Name	Type	Offset	Format
CoreControl	Region Zero	0x070	Integer

Control register

Bits	Name	Read	Write	Reset	Description
0…5	GpMemory Disable	✓	✓	0x000 00000	Disable accesses from graphics core memory ports. See the P9/P10 Memory Pipe Specification for details
6	Context Disable	✓	✓	0x000 00000	0 = enable tag snooping in the Context Unit 1 = disable tag snooping and hence context saving
7…31	Reserved	✓	×		0=reserved

Notes: The CoreControl register controls the operation of the Context and Memory Pipe units

5.7.1.1 PCI Router Status, Profiling and Manufacturing Registers

See Appendix 2 where provided

5.7.2 Interrupt Control (0x01000 – 0x01FFF) 4 K

The following registers are all 32 bits wide and aligned to 64 bits. Writes to undefined addresses are discarded, reads from undefined addresses return zero. Any register bits which are not explicitly defined should be treated as reserved and should not be modified by writes and return zero for reads.

Interrupts work on arrays of interrupt signals. The implementation should match the individual interrupt signals to bit positions in the arrays through the following tables:

Interrupts			
Bit	**Bit Name**	**Signal Name**	**Description**
0	Command	GPIOCommIntrApi	Command interrupt
1	IsocCommand	GPIOCommIntrIso	Command interrupt from isochronous channel
2	Sync	GPIOSyncIntrApi	Sync interrupt
3	IsocSync	GPIOSyncIntrIso	Sync interrupt from isochronous channel
4	ContextTimeout	GPIOTimerIntr	Context scheduler timeout
5	PageFault	MemoryAddressFaultInterrupt	Memory page fault
6	PageDMA	MemoryPageDMACompleteInterrupt	Page controller DMA complete
7	Error	ErrorInterrupt	Error, check flags for source
8	External	ExternalInterrupt	From external pin
9	VideoPort0	VPortFrameIntr0	Start of frame from video input port
10	VideoPort1	VPortFrameIntr1	Start of frame from video input port
11	VBlank0	VideoVBlank0	Vertical blank
12	VBlank1	VideoVBlank1	Vertical blank

The rest of P9/P10 does not distinguish between errors and interrupts, so in the *InterruptValid* array the errors follow on from the interrupts.

Errors			
Bit	**Bit Name**	**Signal Name**	**Description**
0	BusError	BusError	Error from PCI, check bus register for cause
1	PixAddrTimeOut	PixAddrWatchdogInt	Watchdog timeout from pixel address unit
2	TexTimeOut	TexWatchdogInt	Watchdog timeout from texture units
3	IndexError0	GPIOIdxErr0	Index references invalid address
4	IndexError0	GPIOIdxErr1	Index references invalid address
5	IndexError0	GPIOIdxErr2	Index references invalid address

6	IndexError0	GPIOIdxErr3	Index references invalid address
7	IndexError0	GPIOIdxErr4	Index references invalid address
8	IndexError0	GPIOIdxErr5	Index references invalid address
9	IndexError0	GPIOIdxErr6	Index references invalid address
10	IndexError0	GPIOIdxErr7	Index references invalid address
11	IndexError0	GPIOIdxErr8	Index references invalid address
12	IndexError0	GPIOIdxErr9	Index references invalid address
13	IndexError0	GPIOIdxErr10	Index references invalid address
14	IndexError0	GPIOIdxErr11	Index references invalid address
15	IndexError0	GPIOIdxErr12	Index references invalid address
16	IndexError0	GPIOIdxErr13	Index references invalid address
17	IndexError0	GPIOIdxErr14	Index references invalid address
18	IndexError0	GPIOIdxErr15	Index references invalid address
19	VUnderflow0	VideoUnderflowError0	Video underflow from head 0
20	VUnderflow1	VideoUnderflowError1	Video underflow from head 1
21	ShdTimeOut	ShadUnitWDogIntr	Watchdog timeout for shading unit
22	PixVtgTimeOut	PixAddrVtgSyncIntr	Watchdog timeout for syncing on vtg
23	PixTimeOut	PixUnitWDogIntr	Watchdog timeout for pixel unit

ItcInterruptEnable

Name	Type	Offset	Format
ItcInterruptEnable	Region Zero	0x01000	Mask

Control register

Bits	Name	Read	Write	Reset	Description
0…31	Mask	✓	×	0000.0000	Mask of internal signals that should cause a bus interrupt. See above for bit assignments

Notes:

ItcInterruptPending

Name	Type	Offset	Format
ItcInterruptPending	Region Zero	0x01008	Integer

Control register

Bits	Name	Read	Write	Reset	Description
0…29	Mask	✓	✓		Mask of internal signals that are asserted. Write 1 to each bit to be cleared. See above for bit assignments
30	Host	✓	✓		Set to 1 to raise interrupt under software control
31	VGA	✓	×		0=reserved

Notes:

ItcErrorEnable

Name	Type	Offset	Format
ItcErrorEnable	Region Zero	0x01010	Mask

Control register

Bits	Name	Read	Write	Reset	Description
0…31	Mask	✓	✓	0x000 0.0000	Mask of internal signals that should cause a bus interrupt. See above for bit assignments

Notes:

ItcErrorPending

Name	Type	Offset	Format
ItcErrorPending	Region Zero	0x01018	Bitfield

Control register

Bits	Name	Read	Write	Reset	Description
0…30	Mask	✓	✓	0x000 0.0000	Mask of internal signals that are asserted. Write 1 to each bit to be cleared. See above for bit assignments
31	Reserved	✓	×		0=reserved

Notes:

ItcTrigger

Name	Type	Offset	Format
ItcTrigger	Region Zero	0x01020	Mask

Control register

Bits	Name	Read	Write	Reset	Description
0	Valid	✓	×		Set to 1 when register is written, cleared when program has executed
1	Program	✓	✓	0x000 0.0000	
2…31	Reserved				

Notes: A write to this register triggers the operation of a program (the value in the register specifies the program number to run

ItcProgramControl0

Name	Type	Offset	Format
	Region Zero	0x01040	Bitfield

Control register

Bits	Name	Read	Write	Reset	Description
0	SrcA	✓	✓		0=0 1 = Memory
1	SrcB	✓	✓		0=Mask 1=Variable
2,3	Operation	✓	✓		0 = Add 1 = Subtract 2 = AND 3 = OR
4	Dst	✓	✓		0 = Discard 1 = Memory
5…8	Enables	✓	✓		0=reserved
9…15	Reserved	✓	×		0=reserved
16…31	Mask	✓	✓		0 = Mask 1 = Variable

Notes: The value for *SrcA* can be set to zero or read from memory; the value for *SrcB* can be set to the mask of pending interrupts or a variable from a register. The operation is one of:
Dst = SrcA + SrcB
Dst = SrcA – SrcB
Dst = SrcA AND SrcB
Dst = SrcA OR SrcB
The destination value can be written to memory or discarded. All data values are 32 bits, but writes to memory use a specified byte enable mask.

ItcProgramAddrLow0

Name	Type	Offset	Format
ItcProgramAddrLow0	Region Zero	0x01048	Bitfield

Control register

Bits	Name	Read	Write	Reset	Description
0…1	Reserved	✓	×	0x000 0.0000	Reserved
2…31	Address	✓	×		Address

Notes:

ItcProgramAddrHigh0

Name	Type	Offset	Format
ItcProgramAddrHigh0	Region Zero	0x01050	Bitfield
	Control register		

Bits	Name	Read	Write	Reset	Description
0…31	Address	✓	✓	0x000 0.0000	

Notes:

ItcProgramVariable0

Name	Type	Offset	Format
	Region Zero	0x01058	Bitfield
	Control register		

Bits	Name	Read	Write	Reset	Description
0…31	Data	✓	✓	0x000 0.0000	Data

Notes:

ItcProgramControl1

Name	Type	Offset	Format
	Region Zero	0x01060	Bitfield

Control register

Bits	Name	Read	Write	Reset	Description
0	SrcA	✓	✓	0x000 0.0000	0=0 1 = Memory
1	SrcB	✓	✓		0=Mask 1=Variable
2,3	Operation	✓	✓		0 = Add 1 = Subtract 2 = AND 3 = OR
4	Dst	✓	✓		0 = Discard 1 = Memory
5…8	Enables	✓	✓		Enables
9…15	Reserved	✓	×		0=reserved
16…31	Mask	✓	✓		0 = Mask 1 = Variable

Notes: The value for *SrcA* can be set to zero or read from memory; the value for *SrcB* can be set to the mask of pending interrupts or a variable from a register. The operation is one of:
Dst = SrcA + SrcB Dst = SrcA – SrcB
Dst = SrcA AND SrcB Dst = SrcA OR SrcB
The destination value can be written to memory or discarded. All data values are 32 bits, but writes to memory use a specified byte enable mask.

ItcProgramAddrLow1

Name	Type	Offset	Format
ItcProgramAddrHigh0	Region Zero	0x01068	Bitfield

Control register

Bits	Name	Read	Write	Reset	Description
0…1	Reserved	✓	×		
1…31	Address	✓	✓	0x000 0.0000	

Notes:

ItcProgramAddrHigh1

Name	Type	Offset	Format
ItcProgramAddrHigh0	Region Zero	0x01070	Bitfield
	Control register		

Bits	Name	Read	Write	Reset	Description
0…31	Address	✓	✓	0x000 0.0000	

Notes:

ItcProgramVariable1

Name	Type	Offset	Format
ItcProgramVariable1	Region Zero	0x01078	Bitfield
	Control register		

Bits	Name	Read	Write	Reset	Description
0…31	Data	✓	✓	0x000 0.0000	Data

Notes:

5.7.3 Video Head 0 Control (0x02000 – 0x02FFF) (4Kb)

The Video Control implementation is described in the *P10 Reference Guide* volume 1. Each of the two video heads in the current P9/P10 implementation has its own 4K Byte control register space within Region Zero.

Unless explicitly noted, each register in this list is repeated for each head in the system. Any reserved fields in a register should read back as zero.

The implementation uses one DClk process for each head in the system, but one PClk process for all heads. The PClk process holds the register read/write controls and must route the accesses to the appropriate head; the head number is used explicitly in the code. Each DClk process handles the pixel processing, and as there is an indentical set of processes for each head the head number is not used explcitly. An access to a register just references that register, and does not specify which head it belongs to. If the head number needs to be referenced directly the symbol # is used which should be replaced the number of that particular head; this is mainly needed in code controlling shared resources such as pins.

5.7.3.1 Direct Access Registers

The following registers are accessed directly by reading or writing the defined address.

Note: Unlike other registers, Video Control registers are all 8 bytes wide and set on 8 byte boundaries in the PCI address range. When accessed from the VGA they are packed on byte boundaries.

VideoPaletteWriteAddress

Name	Type	Offset	Format
ItcErrorEnable	Region Zero	0x02000	Address

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Address	✓	✓	0x000 0.0000	

Notes:

VideoPaletteData

Name	Type	Offset	Format
VideoPaletteData	Region Zero	0x02008	Int

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Data	✓	✓	0xXX XX.X XXX	Data

Notes: If the color resolution is 6 bits, bits 7 and 6 are returned as zero for reads and ignored for writes. In this mode, bits 5 to 0 are read from, or written to, bits 7 to 2 of the palette. Autoincrements VideoPaletteReadAddress and VideoPaletteWriteAddress

VideoPixelMask

Name	Type	Offset	Format
VideoPixelMask	Region Zero	0x02010	Mask

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Mask	✓	✓	0xXX XX.X XXX	Mask

Notes: The contents of this register is ANDed with the index into the color palette. The same mask is applied seperately to red, green, and blue components. It is only applied to LUT0

VideoPaletteReadAddress

Name	Type	Offset	Format
VideoPaletteReadAddress	Region Zero	0x02018	Address
	Control register		

Bits	Name	Read	Write	Reset	Description
0…7	Address	✓	✓	0xXX XXX XXX	

Notes: The index of the palette entry to be read is written to this register. If this register is read, its operation is determined by the state of the *LastReadAddress* bit in the **VideoControl0** register.
LastReadAddress = 0 register returns mode of last access to palette, 0 = write, 3 = read.
LastReadAddress = 1, register returns palette read address

VideoIndexLow

Name	Type	Offset	Format
VideoIndexLow	Region Zero	0x02020	Int
	Control register		

Bits	Name	Read	Write	Reset	Description
0…7	Index	✓	✓	0xXX XXX XXX	Index

Notes: This register, with **VideoIndexHigh**, selects the register that will be accessed when the **VideoIndexData** register is written or read

VideoIndexHigh

Name	Type	Offset	Format
VideoIndexHigh	Region Zero	0x02028	Int
	Control register		

Bits	Name	Read	Write	Reset	Description
0…3	Index	✓	✓	0xXX XXX XXX	Index

Notes: This register, with **VideoIndexLow**, selects the register that will be accessed when the **VideoIndexData** register is written or read

VideoIndexedData

Name	Type	Offset	Format
VideoIndexedData	Region Zero	0x02030	Int
	Control register		

Bits	Name	Read	Write	Reset	Description
0…7	Data	✓	✓	0xXX XXXX XXXX	

Notes: A read or write to this register accesses the register pointed to by the **VideoIndex** register. Following a read or write to this register, the index is incremented if *AutoIncrement* is enabled in **VideoIndexControl**

VideoIndexControl

Name	Type	Offset	Format
VideoIndexControl	Region Zero	0x02038	Bitfield
	Control register		

Bits	Name	Read	Write	Reset	Description
0	Auto Increment	✓	✓	0x000 0.0000	0 = Disabled 1 = Enabled
1…7	Reserved	✓	×		

Notes:

5.7.3.2 Indirect Access Registers

The following registers may be accessed indirectly by first loading the index into the **IndexLow** and **IndexHigh** registers, and then reading or writing the **VideoIndexedData** register. They may be accessed directly by forming a byte address from the index and accessing on 32 bit alignments (so 4 registers are read or written at a time, although byte enables are honoured for writes). They are packed together, any CPUs that cannot support byte writes should use the indirection register. The indices are not all consecutive.

VideoUpdate

Name	Type	Index	Format
VideoUpdate	Region Zero	0x40	Bitfield

Control register

Bits	Name	Read	Write	Reset	Description
0	UnderlayReg [P10]	✓	✓	0x000 0.0000	0 = Update complete 1 = Update pending
	Reserved [P9]	✓	✗		Reserved [P9]
1	Underlay Buffer [P10]	✓	✓	0x000 0.0000	0 = Update complete 1 = Update pending
	Reserved [P9]	✓	✗		Reserved [P9]
2	MainReg	✓	✓	0x000 0.0000	0 = Update complete 1 = Update pending
3	MainBuffer	✓	✓	0x000 0.0000	0 = Update complete 1 = Update pending
4	OverlayReg	✓	✓	0x000 0.0000	0 = Update complete 1 = Update pending
5	OverlayBuffer	✓	✓	0x000 0.0000	0 = Update complete 1 = Update pending
6	CursorReg [P10]	✓	✓	0x000 0.0000	0 = Update complete 1 = Update pending
	Reserved [P9]	✓	✗		Reserved [P9]
7	CursorBuffer [P10]	✓	✓		0 = Update complete 1 = Update pending
	Reserved [P9]	✓	✗		Reserved [P9]

Notes: Set flag when registers have been modified; flag cleared when new values have been registered. 'Update complete' means that new data can be written into the registers, so for example when triple-buffered there is at least one space in the buffer 'queue'. See **VideoBufferControl** for information on buffer queueing.

VideoControl0

Name	Type	Index	Format
VideoControl0	Region Zero	0x41	Bitfield
	Control register		

Bits	Name	Read	Write	Reset	Description
0	LastRead Address	✓	✓	0x000 0.0000	Controls data returned by read from **VideoPaletteReadAddress** register 0 = Disabled (return palette access state). 1 = Enabled (return last palette read address).
1	PixelScale	✓	✓		0 = Disabled. 1 = Enabled
2	LineScale	✓	✓		0 = Disabled. 1 = Enabled
3	RGB	✓	✓		0 = Color order = BGR. 1 = Color order = RGB
4	Latency	✓	✔		1 = Set.. Improves underrun control, should always be set
5	Priority	✓	✓		Sets memory requests to high priority
6,7	Reserved	✓	×		

Notes:

VideoControl1

Name	Type	Index	Format
VideoControl1	Region Zero	0x42	Bitfield
	Control register		

Bits	Name	Read	Write	Reset	Description
0	AccessLUT	✓	✓	0x000 0.1000	Controls which LUT is read/written [P10] Must = 0 [P9]
1	ExtendLUT	✓	✓		Linearly extend LUT data on load
2	Interlace	✓	✓		
3	StereoFrame	✓	×	.	Flag indicates which field is being displayed
4	MainStereo	✓	✓	.	0 = Disabled 1 = Enabled
5	OverlayStereo	✓	✓		0 = Disabled 1 = Enabled
6	InvertStereo	✓	✓		0 = Disabled 1 = Enabled
7	Filter	✓	✓		0 = Disabled 1 = Enabled

Notes:

VideoBufferControl

Name	Type	Index	Format
VideoBufferControl	Region Zero	0x43	Bitfield

Control register

Bits	Name	Read	Write	Reset	Description
0,1	Underlay	✓	×		0 = SingleBuffer 1 = DoubleBuffer 2 = TripleBuffer
	Reserved [P9]	✓	×		Reserved [P9]
2,3	Main	✓	×		0 = SingleBuffer 1 = DoubleBuffer 2 = TripleBuffer
4,5	Overlay	✓	×		0 = SingleBuffer 1 = DoubleBuffer 2 = TripleBuffer
6,7	Cursor	✓	×		0 = SingleBuffer 1 = DoubleBuffer 2 = TripleBuffer
	Reserved [P9]	✓	×		Reserved [P9]

Notes: When running single-bufffered the 'queue' of buffers is zero deep so screen updates happen at once. When double-buffered there is a queue of one item waiting for the next vblank. When triple-buffered there are 2 updates queued for the next and next-plus-one vblank.
Buffer availability is reported by the **VideoUpdate** register.

VideoUnderlayPan [P10]

Name	Type	Index	Format
VideoUnderlayPan	Region Zero	0x44	Bitfield

Control register

Bits	Name	Read	Write	Reset	Description
0…2	X	✓	✓	0x0X XX.0 XXX	X offset within first tile of first valid byte
3	Reserved	✓	×		
4…6	Y	✓	✓		Y offset within first tile of first valid byte
7	Reserved	✓	×		

Notes: **Reserved register in P9, enabled in P10**

VideoMainPan

Name	Type	Index	Format
VideoMainPan	Region Zero	0x45	Bitfield
	Control register		

Bits	Name	Read	Write	Reset	Description
0…2	X	✓	✓	0x0X XX.0 XXX	X offset within first tile of first valid byte
3	Reserved	✓	×		
4…6	Y	✓	✓		Y offset within first tile of first valid byte
7	Reserved	✓	×		

Notes:

VideoOverlayPan

Name	Type	Index	Format
VideoOverlayPan	Region Zero	0x46	Bitfield
	Control register		

Bits	Name	Read	Write	Reset	Description
0…2	X	✓	✓	0x0X XX.0 XXX	X offset within first tile of first valid byte
3	Reserved	✓	×		
4..6	Y	✓	✓		Y offset within first tile of first valid byte
7	Reserved	✓	×		

Notes:

VideoCursorPan [P10]

Name	Type	Index	Format
VideoCursorPan	Region Zero	0x47	bitfield
	Control register		

Bits	Name	Read	Write	Reset	Description
0…2	X	✓	✓	0x0X XX.0 XXX	X offset within first tile of first valid byte
3	Reserved	✓	×		
4..6	Y	✓	✓		Y offset within first tile of first valid byte
7	Reserved	✓	×		

Notes: **Reserved register in P9, enabled in P10**

VideoUnderlayAddress0 [P10]

Name	Type	Index	Format
VideoUnderlayAddress0	Region Zero	0x48	Address
	Control register		

Bits	Name	Read	Write	Reset	Description
0…7	Address	✓	✓	0xXX XX.X XXX	Holds LSB of tile address of main image

Notes: **Reserved register in P9, enabled in P10**

VideoUnderlayAddress1 [P10]

Name	Type	Index	Format
VideoUnderlayAddress1	Region Zero	0x49	Address
	Control register		

Bits	Name	Read	Write	Reset	Description
0…7	Address	✓	✓	0xXX XX.X XXX	

Notes: **Reserved register in P9, enabled in P10**

VideoUnderlayAddress2 [P10]

Name	Type	Index	Format
VideoUnderlayAddress2	Region Zero	0x4A	Address

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Address	✓	✓	0xXX XXX XXX	

Notes: **Reserved register in P9, enabled in P10**

VideoUnderlayAddress3 [P10]

Name	Type	Index	Format
VideoUnderlayAddress3	Region Zero	0x4B	Address

Control register

Bits	Name	Read	Write	Reset	Description
0…3	Address	✓	✓	0x000 0XX XX	Holds MSB of tile address of main image
4…7	Reserved	✓	✕		

Notes: **Reserved register in P9, enabled in P10**

VideoUnderlayStride0 [P10]

Name	Type	Index	Format
VideoUnderlayStride0	Region Zero	0x4C	Address

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Address	✓	✓	0xXX XXX XXX	

Notes: **Reserved register in P9, enabled in P10**

VideoUnderlayStride1 [P10]

Name	Type	Index	Format
VideoUnderlayStride1	Region Zero	0x4D	Address

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Address	✓	✓	0xXX XX.X XXX	

Notes: **Reserved register in P9, enabled in P10**

VideoUnderlayStride2 [P10]

Name	Type	Index	Format
VideoUnderlayStride2	Region Zero	0x4E	Address

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Address	✓	✓	0xXX XX.X XXX	

Notes: **Reserved register in P9, enabled in P10**

VideoUnderlayStride3 [P10]

Name	Type	Index	Format
VideoUnderlayStride3	Region Zero	0x4F	Address

Control register

Bits	Name	Read	Write	Reset	Description
0…3	Address	✓	✓	0x000 0.XX XX	
4…7	Reserved	✓	✕		

Notes: **Reserved register in P9, enabled in P10**

VideoMainAddress0

Name	Type	Index	Format
VideoMainAddress0	Region Zero	0x50	Address

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Address	✓	✓	0xXX XXX XXX	

Notes:

VideoMainAddress1

Name	Type	Index	Format
VideoMainAddress1	Region Zero	0x51	Address

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Address	✓	✓	0xXX XXX XXX	

Notes:

VideoMainAddress2

Name	Type	Index	Format
VideoMainAddress2	Region Zero	0x52	Address

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Address	✓	✓	0xXX XXX XXX	

Notes:

VideoMainAddress3

Name	Type	Index	Format
VideoMainAddress3	Region Zero	0x53	Address
	Control register		

Bits	Name	Read	Write	Reset	Description
0…7	Address	✓	✓	0x000 0.XX XX	Holds MSB of tile address of main image

Notes:

VideoMainStride0

Name	Type	Index	Format
VideoMainStride0	Region Zero	0x54	Address
	Control register		

Bits	Name	Read	Write	Reset	Description
0…7	Address	✓	✓	0xXX XX.X XXX	

Notes:

VideoMainStride1

Name	Type	Index	Format
VideoMainStride1	Region Zero	0x55	Address
	Control register		

Bits	Name	Read	Write	Reset	Description
0…7	Address	✓	✓	0xXX XX.X XXX	
8…31	Reserved	✓	×		

Notes:

VideoMainStride2

Name	Type	Index	Format
VideoMainStride2	Region Zero	0x56	Address

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Address	✓	✓	0xXX XX.X XXX	

Notes:

VideoMainStride3

Name	Type	Index	Format
VideoMainStride3	Region Zero	0x57	Address

Control register

Bits	Name	Read	Write	Reset	Description
0…3	Address	✓	✓	0x000 0.XX XX	
4…7	Reserved	✓	×		

Notes:

VideoOverlayAddress0

Name	Type	Index	Format
VideoOverlayAddress0	Region Zero	0x58	Address

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Address	✓	✓	0xXX XX.X XXX	Holds LSB of tile address of overlay

Notes:

VideoOverlayAddress1

Name	Type	Index	Format
VideoOverlayAddress1	Region Zero	0x59	Address

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Address	✓	✓	0xXX XX.X XXX	

Notes:

VideoOverlayAddress2

Name	Type	Index	Format
VideoOverlayAddress2	Region Zero	0x5A	Address

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Address	✓	✓	0xXX XX.X XXX	

Notes:

VideoOverlayAddress3

Name	Type	Index	Format
VideoOverlayAddress3	Region Zero	0x58	Address

Control register

Bits	Name	Read	Write	Reset	Description
0…3	Address	✓	✓	0x000 0.XX XX	Holds MSB of tile address of overlay

Notes:

VideoOverlayStride0

Name	Type	Index	Format
VideoOverlayStride0	Region Zero	0x5C	Address

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Address	✓	✓	0xXX XXX XXX	Holds LSB of tile address of overlay

Notes:

VideoOverlayStride1

Name	Type	Index	Format
VideoOverlayStride1	Region Zero	0x5D	Address

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Address	✓	✓	0xXX XXX XXX	Holds LSB of tile address of overlay

Notes:

VideoOverlayStride2

Name	Type	Index	Format
VideoOverlayStride2	Region Zero	0x5E	Address

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Address	✓	✓	0xXX XXX XXX	

Notes:

VideoOverlayStride3

Name	Type	Index	Format
VideoOverlayStride3	Region Zero	0x5F	Address

Control register

Bits	Name	Read	Write	Reset	Description
0…3	Address	✓	✓	0x000 0.XX XX	
4…7	Reserved	✓	✕		

Notes:

VideoCursorAddress0 [P10]

Name	Type	Index	Format
VideoCursorAddress0	Region Zero	0x60	Address

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Address	✓	✓	0xXX XX.X XXX	Holds LSB of tile address of cursor

Notes: **Reserved register in P9, enabled in P10**

VideoCursorAddress1 [P10]

Name	Type	Index	Format
VideoCursorAddress1	Region Zero	0x61	Address

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Address	✓	✓	0xXX XX.X XXX	

Notes: **Reserved register in P9, enabled in P10**

VideoCursorAddress2 [P10]

Name	Type	Index	Format
VideoCursorAddress2	Region Zero	0x62	Address

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Address	✓	✓	0xXX XX.X XXX	

Notes: **Reserved register in P9, enabled in P10**

VideoCursorAddress3 [P10]

Name	Type	Index	Format
VideoCursorAddress3	Region Zero	0x63	Address

Control register

Bits	Name	Read	Write	Reset	Description
0…3	Address	✓	✓	0x00. XXX X	Holds MSB of tile address of cursor
4…7	Reserved	✓	×		

Notes: **Reserved register in P9, enabled in P10**

VideoCursorStride0 [P10]

Name	Type	Index	Format
VideoCursorStride0	Region Zero	0x64	Address

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Address	✓	✓	0xXX XX.X XXX	Holds LSB of tile address of overlay

Notes: **Reserved register in P9, enabled in P10**

VideoCursorStride1 [P10]

Name	Type	Index	Format
VideoCursorStride1	Region Zero	0x65	Address

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Address	✓	✓	0xXX XXX XXX	Holds LSB of tile address of overlay

Notes: **Reserved register in P9, enabled in P10**

VideoCursorStride2 [P10]

Name	Type	Index	Format
VideoCursorStride2	Region Zero	0x66	Address

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Address	✓	✓	0xXX XXX XXX	Holds LSB of tile address of overlay

Notes: **Reserved register in P9, enabled in P10**

VideoCursorStride3 [P10]

Name	Type	Index	Format
VideoCursorStride3	Region Zero	0x67	Address

Control register

Bits	Name	Read	Write	Reset	Description
0…3	Address	✓	✓	0x000 0XX XX	Holds LSB of tile address of overlay
4…7	Reserved	✓	×		

Notes: **Reserved register in P9, enabled in P10**

VideoMainStereoAddress0

Name	Type	Index	Format
VideoMainStereoAddress0	Region Zero	0x68	Address

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Address	✓	✓	0xXX XXX XXX	Holds LSB of tile address of stereo main image

Notes:

VideoMainStereoAddress1

Name	Type	Index	Format
VideoMainStereoAddress1	Region Zero	0x69	Address

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Address	✓	✓	0xXX XXX XXX	Holds LSB of tile address of stereo main image

Notes:

VideoMainStereoAddress2

Name	Type	Index	Format
VideoMainStereoAddress2	Region Zero	0x6A	Address

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Address	✓	✓	0xXX XXX XXX	Holds LSB of tile address of stereo main image

Notes:

VideoMainStereoAddress3

Name	Type	Index	Format
VideoOverlayStride1	Region Zero	0x6B	Address

Control register

Bits	Name	Read	Write	Reset	Description
0…3	Address	✓	✓	0x0000.XXXX	Holds MSB of tile address of stereo main image
4…7	Reserved	✓	×		

Notes:

VideoOverlayStereoAddress0

Name	Type	Index	Format
VideoOverlayStereoAddress0	Region Zero	0x6C	Address

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Address	✓	✓	0xXXXX.XXXX	Holds LSB of tile address of stereo overlay image

Notes:

VideoOverlayStereoAddress1

Name	Type	Index	Format
VideoOverlayStereoAddress1	Region Zero	0x6D	Address

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Address	✓	✓	0xXXXX.XXXX	

Notes:

VideoOverlayStereoAddress2

Name	Type	Index	Format
VideoOverlayStereoAddress2	Region Zero	0x6E	Address

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Address	✓	✓	0x000.000	

Notes:

VideoOverlayStereoAddress3

Name	Type	Index	Format
VideoOverlayStereoAddress3	Region Zero	0x6F	Address

Control register

Bits	Name	Read	Write	Reset	Description
0…3	Address	✓	✓	0x0000.XXXX	Holds MSB of tile address of stereo overlay image
4…7	Reserved	✓	×		

Notes:

VideoTiming

Name	Type	Index	Format
VideoTiming	Region Zero	0x70	Bitfield

Control register

Bits	Name	Read	Write	Reset	Description
0	TimingEnable	✓	✓	0x000.000	0 = Disabled. 1 = Enabled
1	Underlay Enable	✓	✓		0 = Disabled. 1 = Enabled
	Reserved [P9]	✓	✕		Reserved
2	MainEnable	✓	✓		0 = Disabled. 1 = Enabled
3	OverlayEnable	✓	✓		0 = Disabled. 1 = Enabled
4	CursorEnable	✓	✓		0 = Disabled. 1 = Enabled
	Reserved [P9]	✓	✕		Reserved
5…7	Reserved	✓	✕		Reserved

Notes:

VideoGenlock

Name	Type	Index	Format
VideoGenlock	Region Zero	0x71	Bitfield

Control register

Bits	Name	Read	Write	Reset	Description
0,1	Mode	✓	✓	0x0XXX.XXXX	0 = Off 1 = External 2 = Internal 3 = Reserved
2,3	Head [P10]	✓	✓		Head to lock to for internal mode [P10]
	Reserved [P9]	✓	✕		Reserved [P9]
4	LockStereo	✓	✕		
5	InvertHSync	✓	✕		
6	InvertVSync	✓	✕		
7	VOnly	✓	✕		Ignore horizontal sync, lock to vertical only

Notes:

VideoLineCountLow

Name	Type	Index	Format
VideoLineCountLow	Region Zero	0x72	Integer

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Count	✓	×	0xXX XXX XXX	Low byte of line number being processed

Notes:

VideoLineCountHigh

Name	Type	Index	Format
VideoLineCountHigh	Region Zero	0x73	Integer

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Count	✓	×	0xXX XXX XXX	High byte of line number being processed

Notes:

VideoHSyncStartLow

Name	Type	Index	Format
VideoHSyncStartLow	Region Zero	0x74	Address

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Address	✓	✓	0xXX XXX XXX	

Notes:

VideoHSyncStartHigh

Name	Type	Index	Format
VideoHSyncStartHigh	Region Zero	0x75	Integer

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Count	✓	✓	0xXX XXX XXX	
8…31	Reserved	✓	×		

Notes:

VideoHSyncEndLow

Name	Type	Index	Format
VideoHSyncEndLow	Region Zero	0x76	Integer

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Count	✓	✓	0xXX XXX XXX	
8…31	Reserved	✓	×		

Notes:

VideoHSyncEndHigh

Name	Type	Index	Format
VideoOverlayStride1	Region Zero	0x77	Integer

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Count	✓	✓	0xXX XXX XXX	
8…31	Reserved	✓	×		

Notes:

VideoHBlankEndLow

Name	Type	Index	Format
VideoHBlankEndLow	Region Zero	0x78	Integer

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Count	✓	✓	0xXX XX.X XXX	
8…31	Reserved	✓	×		

Notes:

VideoHBlankEndHigh

Name	Type	Index	Format
VideoHBlankEndHigh	Region Zero	0x79	Integer

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Count	✓	✓	0xXX XX.X XXX	
8…31	Reserved	✓	×		

Notes:

VideoHTotalLow

Name	Type	Index	Format
VideoHTotalLow	Region Zero	0x7A	Integer

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Count	✓	✓	0xXX XX.X XXX	
8…31	Reserved	✓	×		

Notes:

VideoHTotalHigh

Name	Type	Index	Format
VideoHTotalHigh	Region Zero	0x7B	Integer

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Count	✓	✓	0xXX XX.X XXX	

Notes:

VideoVSyncStartLow

Name	Type	Index	Format
VideoVSyncStartLow	Region Zero	0x7C	Integer

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Count	✓	✓	0xXX XX.X XXX	

Notes:

VideoVSyncStartHigh

Name	Type	Index	Format
VideoVSyncStartHigh	Region Zero	0x7D	Integer

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Count	✓	✓	0xXX XX.X XXX	

Notes:

VideoVSyncEndLow

Name	Type	Index	Format
VideoVSyncEndLow	Region Zero	0x7E	Integer

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Count	✓	✓	0xXX XXX XXX	

Notes:

VideoVSyncEndHigh

Name	Type	Index	Format
VideoVSyncEndHigh	Region Zero	0x7F	Integer

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Count	✓	✓	0xXX XXX XXX	

Notes:

VideoVBlankEndLow

Name	Type	Index	Format
VideoOverlayStride1	Region Zero	0x80	Integer

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Count	✓	✓	0xXX XXX XXX	

Notes:

VideoVBlankEndHigh

Name	Type	Index	Format
VideoVBlankEndHigh	Region Zero	0x81	Integer

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Count	✓	✓	0xXX XX.X XXX	

Notes:

VideoVTotalLow

Name	Type	Index	Format
VideoVTotalLow	Region Zero	0x82	Integer

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Count	✓	✓	0xXX XX.X XXX	

Notes:

VideoVTotalHigh

Name	Type	Index	Format
VideoVTotalHigh	Region Zero	0x83	Integer

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Count	✓	✓	0xXX XX.X XXX	

Notes:

VideoGenlockHLow

Name	Type	Index	Format
VideoGenlockHLow	Region Zero	0x84	Integer

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Count	✓	✓	0xXX XX.X XXX	

Notes:

VideoGenlockHHigh

Name	Type	Index	Format
VideoGenlockHHigh	Region Zero	0x85	Integer

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Count	✓	✓	0xXX XX.X XXX	

Notes:

VideoGenlockVLow

Name	Type	Index	Format
VideoGenlockVLow	Region Zero	0x86	Integer

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Count	✓	✓	0xXX XX.X XXX	

Notes:

VideoGenlockVHigh

Name	Type	Index	Format
VideoGenlockVHig	Region Zero	0x87	Integer

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Count	✓	✓	0xXX XXX XXX	

Notes:

VideoUnderlayFormat [P10]

Name	Type	Index	Format
VideoUnderlayFormat	Region Zero	0x88	Bitfield

Control register

Bits	Name	Read	Write	Reset	Description
0,1	PixelSize	✓	✓	0xXX XXX XXX	0 = 8 bits. 1 = 16 bits. 2 = 32 bits
2…5	Format	✓	✓		
6	AlphaLUT Select	✓	✓		0 = Take LUT select from LUTSelect register 1 = Take LUT select from alpha channel
7	Linear	✓	✓		0 = Undefined color bits set to zero 1 = Undefined color bits linearly extended from upper bits

Notes: **Reserved register in P9, enabled in P10**

VideoMainFormat

Name	Type	Index	Format
VideoMainFormat	Region Zero	0x89	Bitfield
	Control register		

Bits	Name	Read	Write	Reset	Description
0,1	PixelSize	✓	✓	0xXX XX.X XXX	0 = 8 bits. 1 = 16 bits. 2 = 32 bits
2…5	Format	✓	✓		
6	AlphaLUT Select	✓	✓		0 = Take LUT select from LUTSelect register 1 = Take LUT select from alpha channel
7	Linear	✓	✓		0 = Undefined color bits set to zero 1 = Undefined color bits linearly extended from upper bits

Notes: The Format field and pixelsize considerations are described in the Video Format table below.

Pixel Format Table

Format	Name	RGB	Bits/pixel	R	G	B	A	Index
0	CI8	-	8	-	-	-	-	0-7
1	3:3:2	0	8	0-2	3-5	6-7	-	-
1	3:3:2	1	8	5-7	2-4	0-1	-	-
2	5:5:5:1	0	16	0-4	5-9	10-14	15	-
2	5:5:5:1	1	16	10-14	5-9	0-4	15	-
3	5:6:5	0	16	0-4	5-10	11-15	-	-
3	5:6:5	1	16	11-15	5-10	0-4	-	-
4	8:8:8:8	0	32	0-7	8-15	16-23	24-31	-
4	8:8:8:8	1	32	16-23	8-15	0-7	24-31	-
5	10:10:10:2	0	32	0-9	10-19	20-29	30-31	-
5	10:10:10:2	1	32	20-29	10-19	0-9	30-31	-
6	CI4	-	4	-	-	-	-	0-3, 4-7

The pixel size is independent of the color format, so it is possible to have an 8 bit pixel with a 32 bit stride. The bitmask format is different because it uses 4 bits per pixel regardless of pixel size; this format must be used with a one byte pixel size. The pipeline maintains 16 bits per component, but various operations use different numbers of bits. Color key uses 8 bits, blends use 8 bits, LUTs use 8 bits for input but output 10 bits.

VideoOverlayFormat

Name	Type	Index	Format
VideoOverlayFormat	Region Zero	0x8A	Bitfield
	Control register		

Bits	Name	Read	Write	Reset	Description
0,1	PixelSize	✓	✓	0xXX XX.X XXX	0 = 8 bits. 1 = 16 bits. 2 = 32 bits
2…5	Format	✓	✓		
6	AlphaLUT Select	✓	✓		0 = Take LUT select from LUTSelect register 1 = Take LUT select from alpha channel
7	Linear	✓	✓		0 = Undefined color bits set to zero 1 = Undefined color bits linearly extended from upper bits

Notes:

VideoCursorFormat [P10]

Name	Type	Index	Format
VideoCursorFormat	Region Zero	0x8B	Bitfield
	Control register		

Bits	Name	Read	Write	Reset	Description
0,1	PixelSize	✓	✓	0xXX XX.X XXX	0 = 8 bits. 1 = 16 bits. 2 = 32 bits
2…5	Format	✓	✓		
6	AlphaLUT Select	✓	✓		0 = Take LUT select from LUTSelect register 1 = Take LUT select from alpha channel
7	Linear	✓	✓		0 = Undefined color bits set to zero 1 = Undefined color bits linearly extended from upper bits
8…31	Reserved	✓	✕		

Notes: **Reserved register in P9, enabled in P10**

VideoUnderlayXStartLow [P10]

Name	Type	Index	Format
VideoUnderlayXStartLow	Region Zero	0x8C	Integer

Control register

Bits	Name	Read	Write	Reset	Description
0…7	count	✓	✓	0xXX XX.X XXX	
8…31	Reserved	✓	×		

Notes: **Reserved register in P9, enabled in P10**

VideoUnderlayXStartHigh [P10]

Name	Type	Index	Format
VideoUnderlayXStart High	Region Zero	0x8D	Integer

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Count	✓	✓	0xXX XX.X XXX	
8…31	Reserved	✓	×		

Notes: **Reserved register in P9, enabled in P10**

VideoUnderlayYStartLow [P10]

Name	Type	Index	Format
VideoUnderlayYStartLow	Region Zero	0x8E	Integer

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Count	✓	✓	0xXX XX.X XXX	
8…31	Reserved	✓	×		

Notes: **Reserved register in P9, enabled in P10**

VideoUnderlayYStartHigh [P10]

Name	Type	Index	Format
VideoUnderlayYStartHigh	Region Zero	0x8F	Integer

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Count	✓	✓	0xXX XXX XXX	
8…31	Reserved	✓	×		

Notes: **Reserved register in P9, enabled in P10**

VideoUnderlayXEndLow [P10]

Name	Type	Index	Format
VideoUnderlayXEndLow	Region Zero	0x90	Integer

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Count	✓	✓	0xXX XXX XXX	
8…31	Reserved	✓	×		

Notes: **Reserved register in P9, enabled in P10**

VideoUnderlayXEndHigh [P10]

Name	Type	Index	Format
VideoUnderlayXEndHigh	Region Zero	0x91	Integer

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Count	✓	✓	0xXX XXX XXX	
8…31	Reserved	✓	×		

Notes: **Reserved register in P9, enabled in P10**

VideoUnderlayYEndLow [P10]

Name	Type	Index	Format
VideoUnderlayYEndLow	Region Zero	0x92	Integer

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Count	✓	✓	0xXX XXX XXX	
8…31	Reserved	✓	×		

Notes: **Reserved register in P9, enabled in P10**

VideoUnderlayYEndHigh [P10]

Name	Type	Index	Format
VideoUnderlayYEndHigh	Region Zero	0x93	Integer

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Count	✓	✓	0xXX XXX XXX	
8…31	Reserved	✓	×		

Notes: **Reserved register in P9, enabled in P10**

VideoMainXStartLow

Name	Type	Index	Format
VideoMainXStartLow	Region Zero	0x94	Integer

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Count	✓	✓	0xXX XXX XXX	
8…31	Reserved	✓	×		

Notes:

VideoMainXStartHigh

Name	Type	Index	Format
VideoMainXStartHigh	Region Zero	0x95	Integer

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Count	✓	✓	0xXX XX.X XXX	
8…31	Reserved	✓	×		

Notes:

VideoMainYStartLow

Name	Type	Index	Format
VideoMainYStartLow	Region Zero	0x96	Integer

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Count	✓	✓	0xXX XX.X XXX	
8…31	Reserved	✓	×		

Notes:

VideoMainYStartHigh

Name	Type	Index	Format
VideoMainYStartHigh	Region Zero	0x97	Integer

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Count	✓	✓	0xXX XX.X XXX	
8…31	Reserved	✓	×		

Notes:

VideoMainXEndLow

Name	Type	Index	Format
VideoMainXEndLow	Region Zero	0x98	Integer

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Count	✓	✓	0xXX XX.X XXX	
8…31	Reserved	✓	×		

Notes:

VideoMainXEndHigh

Name	Type	Index	Format
VideoMainXEndHigh	Region Zero	0x99	Integer

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Count	✓	✓	0xXX XX.X XXX	
8…31	Reserved	✓	×		

Notes:

VideoMainYEndLow

Name	Type	Index	Format
VideoMainYEndLow	Region Zero	0x9A	Integer

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Count	✓	✓	0xXX XX.X XXX	
8…31	Reserved	✓	×		

Notes:

VideoMainYEndHigh

Name	Type	Index	Format
VideoMainYEndHigh	Region Zero	0x09B	Integer

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Count	✓	✓	0xXX XX.X XXX	
8…31	Reserved	✓	×		

Notes:

VideoOverlayXStartLow

Name	Type	Index	Format
VideoOverlayXStartLow	Region Zero	0x9C	Integer

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Count	✓	✓	0xXX XX.X XXX	
8…31	Reserved	✓	×		

Notes:

VideoOverlayXStartHigh

Name	Type	Index	Format
VideoOverlayXStartHigh	Region Zero	0x9D	Integer

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Count	✓	✓	0xXX XX.X XXX	
8…31	Reserved	✓	×		

Notes:

VideoOverlayYStartLow

Name	Type	Index	Format
VideoOverlayYStartLow	Region Zero	0x9E	Integer

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Count	✓	✓	0xXX XXX XXX	
8…31	Reserved	✓	×		

Notes:

VideoOverlayYStartHigh

Name	Type	Index	Format
VideoOverlayYStartHigh	Region Zero	0x9F	Integer

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Count	✓	✓	0xXX XXX XXX	
8…31	Reserved	✓	×		

Notes:

VideoOverlayXEndLow

Name	Type	Index	Format
VideoOverlayXEndLow	Region Zero	0xA0	Integer

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Count	✓	✓	0xXX XXX XXX	
8…31	Reserved	✓	×		

Notes:

VideoOverlayXEndHigh

Name	Type	Index	Format
VideoOverlayXEndHigh	Region Zero	0xA1	Integer

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Count	✓	✓	0xXX XX.X XXX	
8…31	Reserved	✓	×		

Notes:

VideoOverlayYEndLow

Name	Type	Index	Format
VideoOverlayYEndLow	Region Zero	0xA2	Integer

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Count	✓	✓	0xXX XX.X XXX	
8…31	Reserved	✓	×		

Notes:

VideoOverlayYEndHigh

Name	Type	Index	Format
VideoOverlayYEndHigh	Region Zero	0xA3	Integer

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Count	✓	✓	0xXX XX.X XXX	
8…31	Reserved	✓	×		

Notes:

VideoCursorXStartLow [P10]

Name	Type	Index	Format
VideoCursorXStartLow	Region Zero	0xA4	Integer

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Count	✓	✓	0xXX XX.X XXX	
8…31	Reserved	✓	×		

Notes: **Reserved register in P9, enabled in P10**

VideoCursorXStartHigh [P10]

Name	Type	Index	Format
VideoCursorXStartHigh	Region Zero	0xA5	Integer

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Count	✓	✓	0xXX XX.X XXX	
8…31	Reserved	✓	×		

Notes: **Reserved register in P9, enabled in P10**

VideoCursorYStartLow [P10]

Name	Type	Index	Format
VideoCursorYStartLow	Region Zero	0xA6	Integer

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Count	✓	✓	0xXX XX.X XXX	
8…31	Reserved	✓	×		

Notes: **Reserved register in P9, enabled in P10**

VideoCursorYStartHigh [P10]

Name	Type	Index	Format
VideoCursorYStartHigh	Region Zero	0xA7	Integer

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Count	✓	✓	0xXX XX.X XXX	
8…31	Reserved	✓	×		

Notes: **Reserved register in P9, enabled in P10**

VideoCursorXEndLow [P10]

Name	Type	Index	Format
VideoCursorXEndLow	Region Zero	0xA8	Integer

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Count	✓	✓	0xXX XX.X XXX	
8…31	Reserved	✓	×		

Notes: **Reserved register in P9, enabled in P10**

VideoCursorXEndHigh [P10]

Name	Type	Index	Format
VideoCursorXEndHigh	Region Zero	0xA9	Integer

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Count	✓	✓	0xXX XX.X XXX	
8…31	Reserved	✓	×		

Notes: **Reserved register in P9, enabled in P10**

VideoCursorYEndLow [P10]

Name	Type	Index	Format
VideoCursorYEndLow	Region Zero	0xAA	VideoCursorYEndLow
	Control register		

Bits	Name	Read	Write	Reset	Description
0…7	Count	✓	✓	0xXX XX.X XXX	
8…31	Reserved	✓	×		

Notes: **Reserved register in P9, enabled in P10**

VideoCursorYEndHigh [P10]

Name	Type	Index	Format
VideoCursorYEndHigh	Region Zero	0xAB	Integer
	Control register		

Bits	Name	Read	Write	Reset	Description
0…7	Count	✓	✓	0xXX XX.X XXX	
8…31	Reserved	✓	×		

Notes: **Reserved register in P9, enabled in P10**

VideoBackgroundR

Name	Type	Index	Format
VideoBackgroundR	Region Zero	0xB0	Integer
	Control register		

Bits	Name	Read	Write	Reset	Description
0…7	Red	✓	✓	0xXX XX.X XXX	
8…31	Reserved	✓	×		

Notes:

VideoBackgroundG

Name	Type	Index	Format
VideoBackgroundG	Region Zero	0xB1	Integer
	Control register		

Bits	Name	Read	Write	Reset	Description
0…7	Green	✓	✓	0xXX XX.X XXX	

Notes:

VideoBackgroundB

Name	Type	Index	Format
VideoBackgroundB	Region Zero	0xB2	Integer
	Control register		

Bits	Name	Read	Write	Reset	Description
0…7	Blue	✓	✓	0xXX XX.X XXX	

Notes:

VideoScale

Name	Type	Index	Format
VideoScale	Region Zero	0xB3	Bitfield
	Control register		

Bits	Name	Read	Write	Reset	Description
0…3	HScale	✓	✓	0xXX XX.X XXX	Horizontal scale
4…7	VScale	✓	✓		Vertical scale

Notes: The equation used to work out the scale factor is:
*horizontal factor =(source width / display width) – 1/4) * 32*
*vertical factor =(source height / display height) – 1/4) * 32*

VideoUnderlayKeyTest [P10]

Name	Type	Index	Format
VideoUnderlayKeyTest	Region Zero	0xB4	Bitfield

Control register

Bits	Name	Read	Write	Reset	Description
0,1	TestMode	✓	✓	0xXX XX.X XXX	0 = Off 1 = Always 2 = Equal3 = NotEqual
2	Test	✓	✓		0 = Src 1 = Dst
3	TestColor	✓	✓		
4	TestAlpha	✓	✓		
5	ReplaceOnFail	✓	✓		
6	ReplaceAlpha	✓	✓		
7	ReplaceLUT	✓	✓		

Notes: **Reserved register in P9, enabled in P10**

Each pixel to be displayed may have contributions from any of four channels. The pixel color is determined by working through the channels in order from underlay (Key0), main, overlay, to cursor (Key 3). The first destination field in the sequence (underlay), is set to the background color so that any undefined regions of the screen have a color. This is mainly used when the framebuffer does not match a fixed resolution display - the image can be centered on the display and the background color used to fill around the edges. The sequence of operations in the key process is:

- Test.
- Apply logic op between source and destination to form new destination.
- Blend.

For more information see the *Programmer's Guide*:discussion of Blending and Video Channel Initialization.

VideoUnderlayKeyOp [P10] (configured)

Name	Type	Index	Format
VideoUnderlayKeyOp	Region Zero	0xB5	Bitfield

Control register

Bits	Name	Read	Write	Reset	Description
0,1	LogicOp	✓	✓	0xXX XX.X XXX	0 = Off 1 = Dst XOR Src 2 = Dst OR Src 3 = Dst AND Src
2…4	BlendMode	✓	✓		0 = Off 1 = Register 2 = SrcAlpha 3 = DstAlpha 4 = SrcColor
5	Conditional Blend	✓	✓		If enabled, do not apply blend if current test fails
6	BlendLUT	✓	✓		Change LUT select based on blend factor
7	Overlay	✓	✓		Special alpha-as-overlay mode
8…31	Reserved	✓	×		

Notes: **Reserved register in P9, enabled in P10**

VideoMainKeyTest

Name	Type	Index	Format
VideoMainKeyTest	Region Zero	0xB6	Bitfield
	Control register		

Bits	Name	Read	Write	Reset	Description
0,1	TestMode	✓	✓	0xXX XXX XXX	0 = Off 1 = Always 2 = Equal 3 = NotEqual
2	Test	✓	✓		0 = Src 1 = Dst
3	TestColor	✓	✓		
4	TestAlpha	✓	✓		
5	ReplaceOnFail	✓	✓		
6	ReplaceAlpha	✓	✓		
7	ReplaceLUT	✓	✓		
8…31	Reserved	✓	×		

Notes: Each pixel to be displayed may have contributions from any of four channels. The pixel color is determined by working through the channels in order from underlay (Key0), main, overlay, to cursor (Key 3).

Note: in P9 only the Main and Overlay channels are supported.

The first destination field in the sequence, is set to the background color so that any undefined regions of the screen have a color. This is mainly used when the framebuffer does not match a fixed resolution display - the image can be centered on the display and the background color used to fill around the edges. The sequence of operations in the key process is:

- Test.
- Apply logic op between source and destination to form new destination.
- Blend.

For more information see the *Programmer's Guide*:discussion of Blending and Video Channel Initialization.

VideoMainKeyOp

Name	Type	Index	Format
VideoMainKeyOp	Region Zero	0xB7	Bitfield

Control register

Bits	Name	Read	Write	Reset	Description
0,1	LogicOp	✓	✓	0xXX XX.X XXX	0 = Off 1 = Dst XOR Src 2 = Dst OR Src 3 = Dst AND Src
2…4	BlendMode	✓	✓		0 = Off 1 = Register 2 = SrcAlpha 3 = DstAlpha 4 = SrcColor
5	Conditional Blend	✓	✓		If enabled, do not apply blend if current or previous tests fail
6	BlendLUT	✓	✓		Change LUT select based on blend factor
7	Overlay	✓	✓		Special alpha-as- overlay mode
8…31	Reserved	✓	×		

Notes:

VideoOverlayKeyTest

Name	Type	Index	Format
VideoOverlayKeyTest	Region Zero	0xB8	Bitfield

Control register

Bits	Name	Read	Write	Reset	Description
0,1	TestMode	✓	✓	0xXX XX.X XXX	0 = Off 1 = Always 2 = Equal3 = NotEqual
2	Test	✓	✓		0 = Src 1 = Dst
3	TestColor	✓	✓		
4	TestAlpha	✓	✓		
5	ReplaceOnFail	✓	✓		
6	ReplaceAlpha	✓	✓		
7	ReplaceLUT	✓	✓		
8…31	Reserved	✓	✕		

Notes: Each pixel to be displayed may have contributions from any of four channels. The pixel color is determined by working through the channels in order from underlay (Key0), main, overlay, to cursor (Key 3).

Note: in P9 only the Main and Overlay channels are supported.

The first destination field in the sequence (underlay), is set to the background color so that any undefined regions of the screen have a color. This is mainly used when the framebuffer does not match a fixed resolution display - the image can be centered on the display and the background color used to fill around the edges. The sequence of operations in the key process is:

- Test.
- Apply logic op between source and destination to form new destination.
- Blend.

For more information see the *Programmer's Guide*:discussion of Blending and Video Channel Initialization.

VideoOverlayKeyOp

Name	Type	Index	Format
VideoOverlayKeyOp	Region Zero	0xB9	Bitfield

Control register

Bits	Name	Read	Write	Reset	Description
0,1	LogicOp	✓	✓	0xXX XX.X XXX	0 = Off 1 = Dst XOR Src 2 = Dst OR Src 3 = Dst AND Src
2…4	BlendMode	✓	✓		0 = Off 1 = Register 2 = SrcAlpha 3 = DstAlpha 4 = SrcColor
5	ConditionalBlend	✓	✓		If enabled, do not apply blend if current or previous tests fail
6	BlendLUT	✓	✓		Change LUT select based on blend factor
7	Overlay	✓	✓		Special alpha-as-overlay mode
8…31	Reserved	✓	×		

Notes:

VideoCursorKeyTest [P10]

Name	Type	Index	Format
VideoCursorKeyTest	Region Zero	0xBA	Bitfield

Control register

Bits	Name	Read	Write	Reset	Description
0,1	TestMode	✓	✓	0xXX XXX XXX	0 = Off 1 = Always 2 = Equal 3 = NotEqual
2	Test	✓	✓		0 = Src 1 = Dst
3	TestColor	✓	✓		
4	TestAlpha	✓	✓		
5	ReplaceOnFail	✓	✓		
6	ReplaceAlpha	✓	✓		
7	ReplaceLUT	✓	✓		
8…31	Reserved	✓	×		

Notes: **Reserved register in P9, enabled in P10**

Each pixel to be displayed may have contributions from any of four channels. The pixel color is determined by working through the channels in order from underlay (Key0), main, overlay, to cursor (Key 3). The first destination field in the sequence (underlay), is set to the background color so that any undefined regions of the screen have a color. This is mainly used when the framebuffer does not match a fixed resolution display - the image can be centered on the display and the background color used to fill around the edges. The sequence of operations in the key process is:

- Test.
- Apply logic op between source and destination to form new destination.
- Blend.

For more information see the *Programmer's Guide* discussion of Blending and Video Channel Initialization.

VideoCursorKeyOp [P10]

Name	Type	Index	Format
VideoCursorKeyOp	Region Zero	0xBB	Bitfield

Control register

Bits	Name	Read	Write	Reset	Description
0…1	LogicOp	✓	✓	0xXX XXX XXX	0 = Off 1 = Dst XOR Src 2 = Dst OR Src 3 = Dst AND Src
2…4	BlendMode	✓	✓		0 = Off 1 = Register 2 = SrcAlpha 3 = DstAlpha 4 = SrcColor
5	Conditional Blend	✓	✓		If enabled, do not apply blend if current or previous tests fail
6	BlendLUT	✓	✓		Change LUT select based on blend factor
7	Overlay	✓	✓		Special alpha-as-overlay mode
8…31	Reserved	✓	×		

Notes: **Reserved register in P9, enabled in P10**

VideoUnderlayKeyR [P10]

Name	Type	Index	Format
VideoUnderlayKeyR	Region Zero	0xBC	Integer

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Red	✓	✓	0xXX XXX XXX	
8…31	Reserved	✓	×		

Notes: **Reserved register in P9, enabled in P10**

VideoUnderlayKeyG [P10]

Name	Type	Index	Format
VideoUnderlayKeyG	Region Zero	0xBD	Integer

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Green	✓	✓	0xXX XXX XXX	
8…31	Reserved	✓	×		

Notes: **Reserved register in P9, enabled in P10**

VideoUnderlayKeyB [P10]

Name	Type	Index	Format
VideoUnderlayKeyB	Region Zero	0xBE	Integer

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Blue	✓	✓	0xXX XX.X XXX	
8…31	Reserved	✓	×		

Notes: **Reserved register in P9, enabled in P10**

VideoUnderlayKeyA [P10]

Name	Type	Index	Format
VideoUnderlayKeyA	Region Zero	0xBF	Integer

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Alpha	✓	✓	0xXX XX.X XXX	
8…31	Reserved	✓	×		

Notes: **Reserved register in P9, enabled in P10**

VideoMainKeyR

Name	Type	Index	Format
VideoMainKeyR	Region Zero	0xC0	Integer

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Red	✓	✓	0xXX XX.X XXX	
8…31	Reserved	✓	×		

Notes:

VideoMainKeyG

Name	Type	Index	Format
VideoMainKeyG	Region Zero	0xC1	Integer

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Green	✓	✓	0xXX XXX XXX	
8…31	Reserved	✓	×		

Notes:

VideoMainKeyB

Name	Type	Index	Format
VideoMainKeyB	Region Zero	0xC2	Integer

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Blue	✓	✓	0xXX XXX XXX	
8…31	Reserved	✓	×		

Notes:

VideoMainKeyA

Name	Type	Index	Format
VideoMainKeyA	Region Zero	0xC3	Integer

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Alpha	✓	✓	0xXX XXX XXX	
8…31	Reserved	✓	×		

Notes:

VideoOverlayKeyR

Name	Type	Index	Format
VideoOverlayKeyR	Region Zero	0xC4	Integer

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Red	✓	✓	0xXX XX.X XXX	
8…31	Reserved	✓	×		

Notes:

VideoOverlayKeyG

Name	Type	Index	Format
VideoOverlayKeyG	Region Zero	0xC5	Integer

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Green	✓	✓	0xXX XX.X XXX	
8…31	Reserved	✓	×		

Notes:

VideoOverlayKeyB

Name	Type	Index	Format
VideoOverlayKeyB	Region Zero	0xC6	Integer

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Blue	✓	✓	0xXX XX.X XXX	
8…31	Reserved	✓	×		

Notes:

VideoOverlayKeyA

Name	Type	Index	Format
VideoOverlayKeyA	Region Zero	0xC7	Integer
	Control register		

Bits	Name	Read	Write	Reset	Description
0…7	Alpha	✓	✓	0xXX XXX XXX	
8…31	Reserved	✓	×		

Notes:

VideoCursorKeyR [P10]

Name	Type	Index	Format
VideoCursorKeyR	Region Zero	0xC8	Integer
	Control register		

Bits	Name	Read	Write	Reset	Description
0…7	Red	✓	✓	0xXX XXX XXX	
8…31	Reserved	✓	×		

Notes: **Reserved register in P9, enabled in P10**

VideoCursorKeyG [P10]

Name	Type	Index	Format
VideoCursorKeyG	Region Zero	0xC9	Integer
	Control register		

Bits	Name	Read	Write	Reset	Description
0…7	Green	✓	✓	0xXX XXX XXX	
8…31	Reserved	✓	×		

Notes: **Reserved register in P9, enabled in P10**

VideoCursorKeyB [P10]

Name	Type	Index	Format
VideoCursorKeyB	Region Zero	0xCA	Integer

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Blue	✓	✓	0xXX XX.X XXX	
8…31	Reserved	✓	×		

Notes: **Reserved register in P9, enabled in P10**

VideoCursorKeyA [P10]

Name	Type	Index	Format
VideoCursorKeyA	Region Zero	0xCB	Integer

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Alpha	✓	✓	0xXX XX.X XXX	
8…31	Reserved	✓	×		

Notes: **Reserved register in P9, enabled in P10**

VideoUnderlayBlend [P10]

Name	Type	Index	Format
VideoUnderlayBlend	Region Zero	0xCC	Integer

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Factor	✓	✓	0xXX XX.X XXX	
8…31	Reserved	✓	×		

Notes: **Reserved register in P9, enabled in P10**

VideoMainBlend

Name	Type	Index	Format
VideoMainBlend	Region Zero	0xCD	Integer
	Control register		

Bits	Name	Read	Write	Reset	Description
0…7	Factor	✓	✓	0xXX XX.X XXX	
8…31	Reserved	✓	×		

Notes:

VideoOverlayBlend

Name	Type	Index	Format
VideoOverlayBlend	Region Zero	0xCE	Integer
	Control register		

Bits	Name	Read	Write	Reset	Description
0…7	Factor	✓	✓	0xXX XX.X XXX	
8…31	Reserved	✓	×		

Notes:

VideoCursorBlend [P10]

Name	Type	Index	Format
VideoCursorBlend	Region Zero	0xCF	Integer
	Control register		

Bits	Name	Read	Write	Reset	Description
0…7	Factor	✓	✓	0xXX XX.X XXX	
8…31	Reserved	✓	×		

Notes: **Reserved register in P9, enabled in P10**

VideoLUT0

Name	Type	Index	Format
VideoLUT0	Region Zero	0xD0	Bitfield
	Control register		

Bits	Name	Read	Write	Reset	Description
0,1	Mode	✓	✓	0x000 0.0000	0 = RGB 1 = ColorIndex 2 = AlphaIndex
2,3	Width	✓	✓		0 = 6 bit LUT 1 = 8 bit LUT 2 = 10 bit LUT
4	Underlay Enable	✓	✓		Enable = 1 [P10]
	Reserved [P9]	✓	×		Reserved [P9]
5	MainEnable	✓	✓		
6	OverlayEnable	✓	✓		
7	CursorEnable [P10]	✓	✓		Enable = 1 [P10]
	Reserved [P9]	✓	×		Reserved [P9]
8…31	Reserved	✓	×		

Notes:

VideoLUT(1) [P10]

Name	Type	Index	Format
VideoLUT1	Region Zero	0xD1	Bitfield
	Control register		

Bits	Name	Read	Write	Reset	Description
0,1	Mode	✓	✓	0x000 0.000	0 = RGB 1 = ColorIndex 2 = AlphaIndex
2,3	Width	✓	✓		0 = 6 bit LUT 1 = 8 bit LUT 2 = 10 bit LUT
4	Underlay Enable	✓	✓		
5	MainEnable	✓	✓		
6	OverlayEnable	✓	✓		
7	CursorEnable	✓	✓		
8…31	Reserved	✓	×		

Notes: **Reserved register in P9, enabled in P10**

VideoLUTMode

Name	Type	Index	Format
VideoLUTMode	Region Zero	0xD2	Bitfield
	Control register		

Bits	Name	Read	Write	Reset	Description
0	Key	✓	✓	0x000 0.0000	
1,2	Chanel	✓	✓		0 = Underlay [P10 only, reserved in P9] 1 = Main 2 = Overlay 3 = Cursor [P10 only, reserved in P9]
8…31	Reserved	✓	×		

Notes:

VideoInterleaveControl

Name	Type	Index	Format
VideoInterleaveControl	Region Zero	0xD3	Bitfield
	Control register		

Bits	Name	Read	Write	Reset	Description
0	Underlay [P10]	✓	✓	0x000 0.0000	0 = Disabled 1 = Enabled
	Reserved [P9]	✓	×		Reserved [P9]
1	Main	✓	✓		0 = Disabled. 1 = Enabled
2	Overlay	✓	✓		0 = Disabled. 1 = Enabled
3	Cursor	✓	✓		0 = Disabled. 1 = Enabled
	Reserved [P9]	✓	×		Reserved [P9]
4,5	Channel	✓	✓		1 & 2 valid in P9, others reserved
6	ID	✓	✓		
7…31	Reserved	✓	×		

Notes:

VideoInterleaveData

Name	Type	Index	Format
VideoInterleaveData	Region Zero	0xD4	Bitfield
	Control register		

Bits	Name	Read	Write	Reset	Description
0	Underlay	✓	✓	0x000 0.0000	0 = Disabled. 1 = Enabled [P10]
	Reserved [P9]	✓	✕		Reserved [P9]
1	Main	✓	✓		0 = Disabled. 1 = Enabled
2	Overlay	✓	✓		0 = Disabled. 1 = Enabled
3	Cursor	✓	✓		0 = Disabled. 1 = Enabled
	Reserved [P9]	✓	✕		Reserved [P9]
4,5	LUT	✓	✓		
7…31	Reserved	✓	✕		

Notes:

VideoInterleaveOffsetX

Name	Type	Index	Format
VideoInterleaveOffsetX	Region Zero	0xD5	Integer
	Control register		

Bits	Name	Read	Write	Reset	Description
0…6	XOffset	✓	✓	0xXX XX.X XXX	Pixel offset
7…31	Reserved	✓	✕		

Notes:

VideoInterleaveOffsetY

Name	Type	Index	Format
VideoInterleaveOffsetY	Region Zero	0xD6	Integer
	Control register		

Bits	Name	Read	Write	Reset	Description
0…6	YOffset	✓	✓	0xXX XX.X XXX	Line offset
7…31	Reserved	✓	✕		

Notes:

VideoWindowID

Name	Type	Index	Format
VideoWindowID	Region Zero	0xD8	Bitfield
	Control register		

Bits	Name	Read	Write	Reset	Description
0	Enable	✓	✓	0x000 0.0000	
1,2	Channel [P10]	✓	✓		0 = Underlay 1 = Main 2 = Overlay 3 = Cursor
	Channel [P9]	✓	✓		0 = Reserved 1 = Main 2 = Overlay 3 = Reserved
3…31	Reserved	✓	×		

Notes:

VideoWindowTable0

Name	Type	Index	Format
VideoWindowTable0	Region Zero	0xD9	Integer
	Control register		

Bits	Name	Read	Write	Reset	Description
0…7	Data	✓	✓	0xXX XX.X XXX	

Notes:

VideoWindowTable1

Name	Type	Index	Format
VideoWindowTable1	Region Zero	0xDA	Integer
	Control register		

Bits	Name	Read	Write	Reset	Description
0…7	Data	✓	✓	0xXX XX.X XXX	

Notes:

VideoClipRectEnable0

Name	Type	Index	Format
VideoClipRectEnable0	Region Zero	0xDC	Bitfield

Control register

Bits	Name	Read	Write	Reset	Description
0…3	Reserved [P9]	✓	✕		Reserved [P9]
0	UnderlayA [P10]	✓	✓	0xXXXX.XXXX	
1	UnderlayB [P10]	✓	✓		
2	UnderlayC [P10]	✓	✓		
3	UnderlayD [P10]	✓	✓		
4	MainA	✓	✓		
5	MainB	✓	✓		
6	MainC	✓	✓		
7	MainD	✓	✓		

Notes:

VideoClipRectEnable1

Name	Type	Index	Format
VideoClipRectEnable1	Region Zero	0xDD	Bitfield

Control register

Bits	Name	Read	Write	Reset	Description
0	OverlayA	✓	✓	0xXXXX.XXXX	
1	OverlayB	✓	✓		
2	OverlayC	✓	✓		
3	OverlayD	✓	✓		
4…7	Reserved [P9]	✓	✕		Reserved in P9, enabled in P10-
4	CursorA [P10]	✓	✓		
5	CursorB [P10]	✓	✓		
6	CursorC [P10]	✓	✓		
7	CursorD [P10]	✓	✓		
8…31	Reserved	✓	✕		

Notes:

VideoClipRectXStartLowA

Name	Type	Index	Format
VideoClipRectXStart LowA	Region Zero	0xE0	Integer

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Count	✓	✓	0xXX XX.X XXX	

Notes:

VideoClipRectXStartHighA

Name	Type	Index	Format
VideoClipRectXStart HighA	Region Zero	0xE1	Integer

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Count	✓	✓	0xXX XX.X XXX	

Notes:

VideoClipRectYStartLowA

Name	Type	Index	Format
VideoClipRectYStart LowA	Region Zero	0xE2	Integer

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Count	✓	✓	0xXX XX.X XXX	

Notes:

VideoClipRectYStartHighA

Name	Type	Index	Format
VideoClipRectYStartHighA	Region Zero	0xE3	Integer

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Count	✓	✓	0xXX XXX XXX	

Notes:

VideoClipRectXEndLowA

Name	Type	Index	Format
VideoClipRectXEnd LowA	Region Zero	0xE4	Integer

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Count	✓	✓	0xXX XXX XXX	

Notes:

VideoClipRectXEndHighA

Name	Type	Index	Format
VideoClipRectXEnd HighA	Region Zero	0xE5	Integer

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Count	✓	✓	0xXX XXX XXX	

Notes:

VideoClipRectYEndLowA

Name	Type	Index	Format
VideoClipRectYEnd LowA	Region Zero	0xE6	Integer

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Count	✓	✓	0xXX XXX XXX	

Notes:

VideoClipRectYEndHighA

Name	Type	Index	Format
VideoClipRectYEnd HighA	Region Zero	0xE7	Integer

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Count	✓	✓	0xXX XXX XXX	

Notes:

VideoClipRectXStartLowB

Name	Type	Index	Format
VideoClipRectXStart LowB	Region Zero	0xE8	Integer

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Count	✓	✓	0xXX XXX XXX	

Notes:

VideoClipRectXStartHighB

Name	Type	Index	Format
VideoClipRectXStartHighB	Region Zero	0xE9	Integer

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Count	✓	✓	0xXX XXX XXX	

Notes:

VideoClipRectYStartLowB

Name	Type	Index	Format
VideoClipRectYStartLowB	Region Zero	0xEA	Integer

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Count	✓	✓	0xXX XXX XXX	

Notes:

VideoClipRectYStartHighB

Name	Type	Index	Format
VideoClipRectYStartHighB	Region Zero	0xEB	Integer

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Count	✓	✓	0xXX XXX XXX	

Notes:

VideoClipRectXEndLowB

Name	Type	Index	Format
VideoClipRectXEndLowB	Region Zero	0xEC	Integer

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Count	✓	✓	0xXX XX.X XXX	

Notes:

VideoClipRectXEndHighB

Name	Type	Index	Format
VideoClipRectXEnd HighB	Region Zero	0xED	Integer

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Count	✓	✓	0xXX XX.X XXX	

Notes:

VideoClipRectYEndLowB

Name	Type	Index	Format
VideoClipRectYEndLowB	Region Zero	0xEE	Integer

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Count	✓	✓	0xXX XX.X XXX	

Notes:

VideoClipRectYEndHighB

Name	Type	Index	Format
VideoClipRectYEnd HighB	Region Zero	0xEF	Integer

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Count	✓	✓	0xXX XX.X XXX	

Notes:

VideoClipRectXStartLowC

Name	Type	Index	Format
VideoClipRectXStart LowC	Region Zero	0xF0	Integer

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Count	✓	✓	0xXX XX.X XXX	

Notes:

VideoClipRectXStartHighC

Name	Type	Index	Format
VideoClipRectXStart HighC	Region Zero	0xF1	Integer

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Count	✓	✓	0xXX XX.X XXX	

Notes:

VideoClipRectYStartLowC

Name	Type	Index	Format
VideoClipRectYStartLowC	Region Zero	0xF2	Integer

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Count	✓	✓	0xXX XX.X XXX	

Notes:

VideoClipRectYStartHighC

Name	Type	Index	Format
VideoClipRectYStartHighC	Region Zero	0xF3	Integer

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Count	✓	✓	0xXX XX.X XXX	

Notes:

VideoClipRectXEndLowC

Name	Type	Index	Format
VideoClipRectXEnd LowC	Region Zero	0xF4	Integer

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Count	✓	✓	0xXX XX.X XXX	

Notes:

VideoClipRectXEndHighC

Name	Type	Index	Format
VideoClipRectXEnd HighC	Region Zero	0xF5	Integer

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Count	✓	✓	0xXX XX.X XXX	

Notes:

VideoClipRectYEndLowC

Name	Type	Index	Format
VideoClipRectYEnd LowC	Region Zero	0xF6	Integer

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Count	✓	✓	0xXX XX.X XXX	

Notes:

VideoClipRectYEndHighC

Name	Type	Index	Format
VideoClipRectYEnd HighC	Region Zero	0xF7	Integer

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Count	✓	✓	0xXX XX.X XXX	

Notes:

VideoClipRectXStartLowD

Name	Type	Index	Format
VideoClipRectXStart LowD	Region Zero	0xF8	Integer

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Count	✓	✓	0xXX XX.X XXX	

Notes:

VideoClipRectXStartHighD

Name	Type	Index	Format
VideoClipRectXStartHighD	Region Zero	0xF9	Integer

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Count	✓	✓	0xXX XX.X XXX	

Notes:

VideoClipRectYStartLowD

Name	Type	Index	Format
VideoClipRectYStart LowD	Region Zero	0xFA	Integer

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Count	✓	✓	0xXX XX.X XXX	

Notes:

VideoClipRectYStartHighD

Name	Type	Index	Format
VideoClipRectYStart HighD	Region Zero	0xFB	Integer

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Count	✓	✓	0xXX XX.X XXX	

Notes:

VideoClipRectXEndLowD

Name	Type	Index	Format
VideoClipRectXEnd LowD	Region Zero	0xFC	Integer

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Count	✓	✓	0xXX XX.X XXX	

Notes:

VideoClipRectXEndHighD

Name	Type	Index	Format
VideoClipRectXEnd HighD	Region Zero	0xFD	Integer

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Count	✓	✓	0xXX XX.X XXX	

Notes:

VideoClipRectYEndLowD

Name	Type	Index	Format
VideoClipRectYEnd LowD	Region Zero	0xFE	Integer

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Count	✓	✓	0xXX XX.X XXX	

Notes:

VideoClipRectYEndHighD

Name	Type	Index	Format
VideoClipRectYEnd HighD	Region Zero	0xFF	Integer

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Count	✓	✓	0xXX XX.X XXX	

Notes:

VideoDACControl

Name	Type	Index	Format
VideoDACControl	Region Zero	0x100	Bitfield

Control register

Bits	Name	Read	Write	Reset	Description
0	DACPower Ctl	✓	✓	0x000 0.0000	0 = Normal operation 1 = LowPower
1	Reserved	✓	✓		Reserved for future use
2	BlankRed	✓	✓		0 = Disabled 1 = Enabled
3	BlankGreen	✓	✓		0 = Disabled 1 = Enabled
4	BlankBlue	✓	✓		0 = Disabled 1 = Enabled
5	Pedestal	✓	✓		0 = Disabled 1 = Enabled
6,7	Reserved	✓	×		

Notes: Sync on Green is not supported on P10.

VideoDACSyncControl

Name	Type	Index	Format
VideoDACSyncControl	Region Zero	0x101	Bitfield

Control register

Bits	Name	Read	Write	Reset	Description
0	HSyncCtl	✓	✓	0x000 0.0000	0 = Active low at pin 1 = Active high at pin
1	VSyncCtl				
2,3	Hsync Override				0 = Run 1 = Force tristate 2 = Force low 3 = Force high
4,5	Vsync Override				0 = Run 1 = Force tristate 2 = Force low 3 = Force high
6	Composite				0 = Enable. 1 = Disable
7	Reserved	✓	×		

Notes:

VideoDACSense

Name	Type	Index	Format
VideoDACSense	Region Zero	0x102	Bitfield

Control register

Bits	Name	Read	Write	Reset	Description
0	Red	✓	×	0x000 0.0XX X	
1	Green	✓	×		
2	Blue	✓	×		
3…7	Reserved	✓	×		

Notes:

VideoDACDDC

Name	Type	Index	Format
VideoDACDDC	Region Zero	0x103	Bitfield

Control register

Bits	Name	Read	Write	Reset	Description
0	ClkOut	✓	✓	0x000 0.XX XX	
1	DataOut	✓	✓		
2	ClkIn	✓	×		
3	DataIn	✓	×		
4…7	Reserved	✓	×		

Notes:

VideoDPMode

Name	Type	Index	Format
VideoDPMode	Region Zero	0x104	Bitfield

Control register

Bits	Name	Read	Write	Reset	Description
0,1	Mode	✓	✓	0x000 0.0000	0 = Off 1 = SinglePixel 2 = DoublePixel 3 = AlphaPixel
2…5	StrobeDelay	✓	✓		Delay applied to output strobe in 250pS taps
6	StrobeInvert	✓	✓		
7	Reserved	✓	×		

Notes:

VideoDPSyncControl

Name	Type	Index	Format
VideoDPSyncControl	Region Zero	0x105	Bitfield

Control register

Bits	Name	Read	Write	Reset	Description
0	HSyncCtl	✓	✓	0x000 0.0000	0 = Active low at pin. 1 = Active high at pin
1	VSyncCtl	✓	✓		0 = Active low at pin. 1 = Active high at pin.
2,3	Hsync Override	✓	✓		0 = Run 1 = Reserved 2 = Force low 3 = Force high
4,5	Vsync Override	✓	✓		0 = Run 1 = Reserved 2 = Force low 3 = Force high
6	Composite	✓	✓		0 = Enable. 1 = Disable
7	BlankCtl	✓	✓		0 = Active low at pin. 1 = Active high at pin

Notes:

VideoDPDDC [P10]

Name	Type	Index	Format
VideoDPDDC	Region Zero	0x106	Address

Control register

Bits	Name	Read	Write	Reset	Description
0	ClkOut	✓	✓	0x000 0.XXII	
1	DataOut	✓	✓		
2	ClkIn	✓	×		
3	DataIn	✓	×		
4…7	Reserved	✓	×		

Notes: **Reserved register in P9, enabled in P10**

VideoTest

Name	Type	Index	Format
VideoTest	Region Zero	0x180	Address

Control register

Bits	Name	Read	Write	Reset	Description
0,1	CRC	✓	✓	0x000 0.0000	0 = Off 1 = Line 2 = Frame 3 = Complete
2…7	Reserved	✓	×		

Notes:

VideoCRC[0-3]

Name	Type	Index	Format
VideoCRC0	Region Zero	0x184	Integer
VideoCRC1		0x185	
VideoCRC2		0x186	
VideoCRC3		0x187	

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Data	✓	×	0xXX XX.X XXX	

Notes:

VideoGPEvent

Name	Type	Index	Format
VideoGPEvent	Region Zero	0x188	Bitfield

Control register

Bits	Name	Read	Write	Reset	Description
0	Enable	✓	✓	0x000 0.XX X0	Signal GP on update
1…3	Source	✓	✓		0 = UnderlayReg – **Reserved in P9** 1 = UnderlayBuffer – **Reserved in P9** 2 = MainReg 3 = MainBuffer 4 = OverlayReg 5 = OverlayBuffer 6 = CursorReg – **reserved in P9** 7 = CursorBuffer- **reserved n P9**
4…7	Reserved	✓	×		

Notes:

VideoLock[0-1]

Name	Type	Index	Format
VideoLock0	Region Zero	0x190	Bitfield
VideoLock1		0x191	

Control register

Bits	Name	Read	Write	Reset	Description
0…3	Reg	✓	✓	0x000 0.0000	Mask of channels that lock register loads 0 = Underlay – **Reserved in P9** 1 = Main 2 = Overlay 3 = Cursor – **Reserved in P9**
4…7	Buffer				Mask of channels that lock buffer swaps: 0 = Underlay – **Reserved in P9** 1 = Main 2 = Overlay 3 = Cursor – **Reserved in P9**

Notes: Used to ensure synchronization between channels or heads. Two independent locks can be configured per head using **VideoLock0** and **VideoLock1**

VideoDigitalPortControl

Name	Type	Index	Format
VideoDigitalPortControl	Region Zero	0x1A0	Bitfield

Control register

Bits	Name	Read	Write	Reset	Description	
0…2	Mode	✓	✓	0xXX XXX 000	0 = Off 2 = In0 4 = In1	1 = Shared 3 = Out0 5 = Out1
3	Channel0				0 = In	1 = Out
4	Channel1				0 = In	1 = Out
5	DoubleEdge				0 = Off	1 = On
6	Stereo				Head to output stereo signal	
7	StereoOutput				Output stereo signal	

Notes: *This register is common across all heads*

BlockControl[0-3]

Name	Type	Index	Format
BlockControl0	Region Zero	0x1A4	Integer
BlockControl1		0x1A5	
BlockControl2		0x1A6	
BlockControl3		0x1A7	

Control register

Bits	Name	Read	Write	Reset	Description
0	TexturePipe0	✓	✓	0x000 0.0000	
1	TexturePipe1	✓	✓		
2	TexturePipe2	✓	✓		
3	TexturePipe3	✓	✓		
4	Memory Controller1	✓	✓		
5	VideoOut1	✓	✓		

Notes: *Common across all heads.*

FunctionalScanMode

Name	Type	Offset	Format
FunctionalScanMode	Region Zero	0x1A8	integer

Control register

Bits	Name	Read	Write	Reset	Description
0	Clk	✓	✓	0x000 0.0000	0 = Low 1 = High
1	Stop	✓	✓		0 = Off 1 = On (stop all clocks except PClk)
2	Mode	✓	✓		0 = Off 1 = On (enable PClk domain ring fence)
3	ScanEnable	✓	✓		0 = Off 1 = On (clock data along scan chain)
4	TestRAM	✓	✓		0 = Off 1 = On
5	TestShadow	✓	✓		0 = Off 1 = On
6,7	Reserved	✓	×		

Notes: *Common across all heads*

FunctionalScanMux[0-7]

Name	Type	Index	Format
FunctionalScanMux0	Region Zero	0x1B0	Integer
FunctionalScanMux1		0x1B1	
FunctionalScanMux2		0x1B2	
FunctionalScanMux3		0x1B3	
FunctionalScanMux4		0x1B4	
FunctionalScanMux5		0x1B5	
FunctionalScanMux6		0x1B6	
FunctionalScanMux7		0x1B7	

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Data	✓	✓	0x000 0.0000	

Notes: *Common across all heads*

FunctionalScanIn[0-7]

Name	Type	Index	Format
FunctionalScanIn0	Region Zero	0x1B8	Integer
FunctionalScanIn1		0x1B9	
FunctionalScanIn2		0x1BA	
FunctionalScanIn3		0x1BB	
FunctionalScanIn4		0x1BC	
FunctionalScanIn5		0x1BD	
FunctionalScanIn6		0x1BE	
FunctionalScanIn7		0x1BF	

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Data	✓	✓	0x000 0.0000	Input to scan chain

Notes: *Common across all heads*

FunctionalScanOut[0-7]

Name	Type	Index	Format
FunctionalScanOut0	Region Zero	0x1C0	Integer
FunctionalScanOut1		0x1C1	
FunctionalScanOut2		0x1C2	
FunctionalScanOut3		0x1C3	
FunctionalScanOut4		0x1C4	
FunctionalScanOut5		0x1C5	
FunctionalScanOut6		0x1C6	
FunctionalScanOut7		0x1C7	

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Data	✓	✓	0x000 0.0000	Output of scan chain

Notes: *Common across all heads*

FunctionalScanClock

Name	Type	Index	Format
FunctionalScanClock	Region Zero	0x1C8	Bitfield
	Control register		

Bits	Name	Read	Write	Reset	Description
0	VGAClk	✓	✓	0x01	Unsigned long
1	D0Clk	✓	✓		Unsigned long
2	D1Clk	✓	✓		Unsigned long
3	KClk	✓	✓		Unsigned long
4	MClk	✓	✓		Unsigned long

Notes: *Common across all heads*

PLL0Select

Name	Type	Index	Format
PLL0Select	Region Zero	0x1FF	Bitfield
	Control register		

Bits	Name	Read	Write	Reset	Description
0,1	RegSet	✓	✓	0x000 0.X00 1	0 = Disable 1 = Enable
2…7					
8…31	Unused	✓	×		

Notes: *Common across all heads*

PLL0Control

Name	Type	Index	Format
PLL0Control	Region Zero	0x200	Bitfield
	Control register		

Bits	Name	Read	Write	Reset	Description
0	Enable	✓	✓	0x000 0.X00 1	0 = Disable 1 = Enable
1,2	Ref				0 = Internal 1 = External 2 = PClk
3	Lock	✓	×		0 = Not locked 1 = Locked
4…7	Reserved	✓	×		

Notes: *Common across all heads*

PLL0PreScaleA

Name	Type	Index	Format
PLL0PreScaleA	Region Zero	0x201	Integer
	Control register		

Bits	Name	Read	Write	Reset	Description
0…7	Value	✓	✓	0x000 0.0011	

Notes: *Common across all heads*

PLL0FeedbackScaleA

Name	Type	Index	Format
PLL0FeedbackScaleA	Region Zero	0x202	Integer
	Control register		

Bits	Name	Read	Write	Reset	Description
0…7	Value	✓	✓	0x001 1.1000	Holds LSB of tile address of overlay

Notes: *Common across all heads*

PLL0PostScaleA

Name	Type	Index	Format
PLL0PostScaleA	Region Zero	0x203	Bitfield
	Control register		

Bits	Name	Read	Write	Reset	Description
0…2	Scale	✓	✓	0x000 0.0011	0 = Divide by 1. 1 = Divide by 2. 2 = Divide by 4. 3 = Divide by 8. 4 = Divide by 16.
3…7	Reserved	✓	×		

Notes: *Common across all heads*

PLL0PreScaleB

Name	Type	Index	Format
PLL0PreScaleB	Region Zero	0x204	Integer
	Control register		

Bits	Name	Read	Write	Reset	Description
0…7	Value	✓	✓	0x000 0.0100	

Notes: *Common across all heads*

PLL0FeedbackScaleB

Name	Type	Index	Format
PLL0FeedbackScaleB	Region Zero	0x205	Integer
	Control register		

Bits	Name	Read	Write	Reset	Description
0…7	Value	✓	✓	0x010 0.1111	

Notes: *Common across all heads*

PLL0PostScaleB

Name	Type	Index	Format
PLL0PostScaleB	Region Zero	0x206	Bitfield
	Control register		

Bits	Name	Read	Write	Reset	Description
0…2	Scale	✓	✓	0x000 0.0011	0 = Divide by 1. 1 = Divide by 2. 2 = Divide by 4. 3 = Divide by 8. 4 = Divide by 16.
3…7	Reserved	✓	×		

Notes: *Common across all heads*

PLL0PreScaleC

Name	Type	Index	Format
PLL0PreScaleC	Region Zero	0x207	Integer
	Control register		

Bits	Name	Read	Write	Reset	Description
0…7	Value	✓	✓	0xXX XX.X XXX	

Notes: *Common across all heads*

PLL0FeedbackScaleC

Name	Type	Index	Format
PLL0FeedbackScaleC	Region Zero	0x208	Integer
	Control register		

Bits	Name	Read	Write	Reset	Description
0…7	Value	✓	✓	0xXX XX.X XXX	

Notes: *Common across all heads*

PLL0PostScaleC

Name	Type	Index	Format
PLL0PostScaleC	Region Zero	0x209	Bitfield
	Control register		

Bits	Name	Read	Write	Reset	Description
0…2	Scale	✓	✓	0xXX XX.X XXX	0 = Divide by 1. 1 = Divide by 2. 2 = Divide by 4. 3 = Divide by 8. 4 = Divide by 16.
3…7	Reserved	✓	×		

Notes: *Common across all heads*

PLL0PreScaleD

Name	Type	Index	Format
PLL0PreScaleD	Region Zero	0x20A	Integer
	Control register		

Bits	Name	Read	Write	Reset	Description
0…7	Value	✓	✓	0xXX XX.X XXX	

Notes: *Common across all heads*

PLL0FeedbackScaleD

Name	Type	Index	Format
PLL0FeedbackScaleD	Region Zero	0x20B	Integer
	Control register		

Bits	Name	Read	Write	Reset	Description
0…7	Value	✓	✓	0xXX XX.X XXX	

Notes: *Common across all heads*

PLL0PostScaleD

Name	Type	Index	Format
PLL0PostScaleD	Region Zero	0x20C	Bitfield
	Control register		

Bits	Name	Read	Write	Reset	Description
0…2	Scale	✓	✓	0xXX XX.X XXX	0 = Divide by 1. 1 = Divide by 2. 2 = Divide by 4. 3 = Divide by 8. 4 = Divide by 16.
3…7	Reserved	✓	✕		

Notes: *Common across all heads*

PLL1Control

Name	Type	Index	Format
PLL1Control	Region Zero	0x20D	Bitfield

Control register

Bits	Name	Read	Write	Reset	Description
0	Enable	✓	✓	0x0000.X000	
1,2	Ref	✓	✓		0 = Internal 1 = External 2 = PClk
3	Lock	✓	×		0 = Not locked. 1 = Locked
4…7	Reserved	✓	×		

Notes: *Common across all heads*

PLL1PreScale

Name	Type	Index	Format
PLL1PreScale	Region Zero	0x20E	Integer

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Value	✓	✓	0xXX.XXXX	

Notes: *Common across all heads*

PLL1FeedbackScale

Name	Type	Index	Format
PLL1FeedbackScale	Region Zero	0x20F	Integer

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Value	✓	✓	0xXX.XXXX	

Notes: *Common across all heads*

PLL1PostScale

Name	Type	Index	Format
PLL1PostScale	Region Zero	0x210	Bitfield

Control register

Bits	Name	Read	Write	Reset	Description
0…2	Scale	✓	✓	0xXX XX.X XXX	0 = Divide by 1. 1 = Divide by 2. 2 = Divide by 4. 3 = Divide by 8. 4 = Divide by 16.
3…7	Reserved	✓	×		

Notes: *Common across all heads*

PLL2Control

Name	Type	Index	Format
PLL2Control	Region Zero	0x214	Bitfield

Control register

Bits	Name	Read	Write	Reset	Description
0	Enable	✓	✓	0x000 0.X00 0	0 = Disable 1 = Enable
1,2	Ref	✓	✓		0 = Internal 1 = External 2 = PClk
3	Lock	✓	×		0 = Not locked. 1 = Locked
4…7	Reserved	✓	×		

Notes: *Common across all heads*

PLL2PreScale

Name	Type	Index	Format
PLL2PreScale	Region Zero	0x215	Integer

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Value	✓	✓	0xXX XX.X XXX	

Notes: *Common across all heads*

PLL2FeedbackScale

Name	Type	Index	Format
PLL2FeedbackScale	Region Zero	0x216	Integer
	Control register		

Bits	Name	Read	Write	Reset	Description
0…7	Value	✓	✓	0xXX XX.X XXX	

Notes: *Common across all heads*

PLL2PostScale

Name	Type	Index	Format
PLL2PostScale	Region Zero	0x217	Bitfield
	Control register		

Bits	Name	Read	Write	Reset	Description
0…2	Scale	✓	✓	0xXX XX.X XXX	0 = Divide by 1. 1 = Divide by 2. 2 = Divide by 4. 3 = Divide by 8. 4 = Divide by 16.
3…7	Reserved	✓	×		

Notes: *Common across all heads*

PLL3Control

Name	Type	Index	Format
PLL3Control	Region Zero	0x218	Bitfield
	Control register		

Bits	Name	Read	Write	Reset	Description
0	Enable	✓	✓	0x000 0.X00 0	0 = Disable 1 = Enable
1,2	Ref	✓	✓		0 = Internal 1 = External 2 = PClk
3	Lock	✓	×		0 = Not locked. 1 = Locked
4…7	Reserved	✓	×		

Notes: *Common across all heads*

PLL3PreScale

Name	Type	Index	Format
PLL3PreScale	Region Zero	0x219	Integer
	Control register		

Bits	Name	Read	Write	Reset	Description
0…7	Value	✓	✓	0xXX XX.X XXX	

Notes: *Common across all heads*

PLL3FeedbackScale

Name	Type	Index	Format
PLL3FeedbackScale	Region Zero	0x21A	Integer
	Control register		

Bits	Name	Read	Write	Reset	Description
0…7	Value	✓	✓	0xXX XX.X XXX	

Notes: *Common across all heads*

PLL3PostScale

Name	Type	Index	Format
PLL3PostScale	Region Zero	0x21B	Bitfield
	Control register		

Bits	Name	Read	Write	Reset	Description
0…2	Scale	✓	✓	0xXX XX.X XXX	0 = Divide by 1. 1 = Divide by 2. 2 = Divide by 4. 3 = Divide by 8. 4 = Divide by 16.
3…7	Reserved	✓	✕		

Notes: *Common across all heads*

ClkOutControl [P10]

Name	Type	Index	Format
ClkOutControl	Region Zero	0x220	Bitfield
	Control register		

Bits	Name	Read	Write	Reset	Description	
0,1	Mode	✓	✓	0x100 0.0001	0 = Disable 2 = Divide by 2	1 = Divide by 1 3 = Divide by 4
2…5	Source	✓	✓		0 = PClk 2 = PLL0 4 = PLL2 6 = VGAClk 8 = MClk 10 = D1Clk	1 = Reserved 3 = PLL1 5 = PLL3 7 = KClk 9 = D0Clk
6	Reserved	✓	×			
7	Invert	✓	✓			

Notes: Common across all heads. Controls the clock output at the *GenLockClkOut* pin. **Reserved register in P9, enabled in P10**

VGAClkControl

Name	Type	Index	Format
VGAClkControl	Region Zero	0x221	Bitfield
	Control register		

Bits	Name	Read	Write	Reset	Description	
0,1	State	✓	✓	0x000 0.0000	0 = Drive Low 2 = Run	1 = Drive High 3 = Reserved
2,3	Source	✓	✓		0 = DClk0 2 = Reserved	1 = DClk1 3 = Reserved
4	Div2	✓	✓		0 = Off	1 = On
5…7	Reserved	✓	×			

Notes: *Common across all heads*

DClk0Control

Name	Type	Index	Format
DClk0Control	Region Zero	0x222	Bitfield

Control register

Bits	Name	Read	Write	Reset	Description
0,1	State	✓	✓	0x000 0.1110	0 = Drive Low 1 = Drive High 2 = Run 3 = Reserved
2…4	Source	✓	✓		0 = PClk 1 = External 2 = Reserved 3 = PLL0 4 = PLL1 5 = PLL2 6 = PLL3
5	Div2				0 = Off 1 = On
6,7	Reserved	✓	×		

Notes: *Common across all heads*

DClk1Control

Name	Type	Index	Format
DClk1Control	Region Zero	0x223	Bitfield

Control register

Bits	Name	Read	Write	Reset	Description
0,1	State	✓	✓	0x000 0.1110	0 = Drive Low 1 = Drive High 2 = Run 3 = Reserved
2…4	Source	✓	✓		0 = PClk 1 = External 2 = Reserved 3 = PLL0 4 = PLL1 5 = PLL2 6 = PLL3
5	Div2				0 = Off 1 = On
6,7	Reserved	✓	×		

Notes: *Common across all heads*

KClkControl

Name	Type	Index	Format
KClkControl	Region Zero	0x224	Bitfield

Control register

Bits	Name	Read	Write	Reset	Description
0,1	State	✓	✓	0x000 0.0010	0 = Drive Low 1 = Drive High 2 = Run 3 = Reserved
2…4	Source	✓	✓		0 = PClk 1 = External 2 = Reserved 3 = PLL0 4 = PLL1 5 = PLL2 6 = PLL3
5	Div2				0 = Off 1 = On
6,7	Reserved	✓	×		

Notes: *Common across all heads*

MClkControl

Name	Type	Index	Format
MClkControl	Region Zero	0x225	Bitfield

Control register

Bits	Name	Read	Write	Reset	Description
0,1	State	✓	✓	0x000 0.0010	0 = Drive Low 1 = Drive High 2 = Run 3 = Reserved
2…4	Source	✓	✓		0 = PClk 1 = External 2 = Reserved 3 = PLL0 4 = PLL1 5 = PLL2 6 = PLL3
5	Div2				0 = Off 1 = On
6,7	Reserved	✓	×		

Notes: *Common across all heads*

5.7.4 Memory Control (0x03000 – 0x03FFF) 4 K

The following regitsters control the operation of the memory controller. The memory should be idle before any changes are made. This can be tested by checking the busy flag in the **MemoryControl** register. All registers are on 64 bit boundaries except the fifo registers which are packed to allow bursts. The register definitions show addresses in multiples of 32 bits.

5.7.4.1 DMA Controller

When a page fault is detected data will normally have to be transaferred from system memory to video memory. This may done directly by the CPU reading system data and writing it directly to the chip, or by programming the DMA controller. The sequence of operations to fix a fault will usually be:

- Fault detected.
- Cause of fault retrieved from PageControl fifo.
- Remedy determined, list of pages to be paged out and paged in constructed.

- DMA and table update commands sent to PageControl fifo.

If the system memory used as source or destination of paging is not locked down then the CPU must handle the copy directly through the bypass. Table update commands should still be sent through the PageControl fifo to ensure correct ordering. All commands sent to the PageControl fifo take 4 dwords (1 AGP fast write). The format is:

Word	Bits	Description
1	0..1	Command 0 = Table update 1 = System to video DMA 2 = Video to system DMA 3 = Invalidate
	2	Interrupt on completion
	3..9	Reserved
	10..31	Page
Command = Table update		
2	0..31	Table entry
3	0..31	Table entry
Command = DMA (either type)		
2	0..5	Reserved
	6..7	Type 0 = Reserved 1 = PCI 2 = AGP
	8..11	Reserved
	12..31	System page
3	0..31	System page
Command = Invalidate		
2	0..31	Reserved
3	0..31	Reserved
All commands		
4	0..7	Restart
	8..15	Reserved
	16..23	Suspend
	24..31	Reserved

The first word holds the type of command in the lower 2 bits. The options are to modify a page table entry, to transfer data from system to video memory, to transfer data from video to system memory, or to invalidate the entries in the TLB (translation look-aside buffer, a cache of page table entries). An additional bit indicates that an interrupt should be raised when the command has completed. The upper bits of the first word hold a page number that this command will use.

MemoryPageControlFifo[0-3]

Name	Type	Offset	Format
MemoryPageControlFifo0	Region Zero	0x03000	Integer
MemoryPageControlFifo1		0x03004	
MemoryPageControlFifo2		0x03008	
MemoryPageControlFifo3		0x0300c	

Control register

Bits	Name	Read	Write	Reset	Description
0…31	Data	✓	✓	0x XXX X.XX XX	

Notes: Writing to this register puts data into fifo, read gets data out. Used to control page downloads and modifications to table, and to report faults

MemoryPageControlFifoSpace

Name	Type	Offset	Format
MemoryPageControl FifoSpace	Region Zero	0x03010	Bitfield

Control register

Bits	Name	Read	Write	Reset	Description
0…4	Space	✓	✓	0x000. 0008	
5…31	Reserved	✓	×		

Notes:

MemoryPageControlFifoCount

Name	Type	Offset	Format
MemoryPageControl FifoCount	Region Zero	0x03018	Integer

Control register

Bits	Name	Read	Write	Reset	Description
0…4	Count	✓	✓	0x000. 0000	
5…31	Reserved	✓	×		

Notes:

MemoryControl

Name	Type	Offset	Format
MemoryControl	Region Zero	0x03020	Bitfield

Control register

Bits	Name	Read	Write	Reset	Description
0	Busy	✓	✓	0x XXX X.XX XX	Set if the memory controller is still processing requests. This bit should tested and found clear before modifying any control registers
1…6	Reserved	✓	×		
7	VideoMemory Width	✓	✓		Selects 128 and 256 bus width. P9/P10 is usually configured with 256bit but can use 128bit. 0 = 128 1 = 256
8	VideoDevice Type	✓	✓		VGA type selection: 0 = 16 1 = 32
9	Internal Strobes	✓	✓		0 = Use external strobes 1 = Use internal clock as strobe
10…31	Reserved	✓	×		

Notes:

MemoryTranslationEnable (

Name	Type	Offset	Format
MemoryTranslation Enable	Region Zero	0x03028	Bitfield

Control register

Bits	Name	Read	Write	Reset	Description
0	Bypass	✓	✓	0x000. 0000	Enables bypass read/write accesses
1	VGA/VIP'/ Texture	✓	✓		Controls Texture read accesses
2	Graphics Processor	✓	✓		Controls graphics core read/write accesses, apart from texture accesses
3	Command Processor	✓	✓		Controls the GPIO read accesses
4	Video Processor0	✓	✓		Control the read accesses made for video refresh
5	Video Processor1	✓	✓		Control the read accesses made for video refresh
6…31	Reserved [p10]	✓	×		
6…29	Reserved [P9]	✓	×		
30,31	Segment [p9]	✓	✓		Upper 2 bits of system address if translation disabled.

Notes: The P9 virtual address range is 4GB of contiguous space. If virtual addressing is turned off the upper 2 bits of the address indicate the type of memory access (video memory, PCI, AGP) which reduces the range to 1GB. As system addresses can be above 1GB the segment bits suplement the address. They are only used when virtual addressing is OFF and are intended for debugging.

MemoryPageTableLower

Name	Type	Offset	Format
MemoryPageTableLower	Region Zero	0x03030	Bitfield

Control register

Bits	Name	Read	Write	Reset	Description
0,1	Type	✓	✓	0x XXX X.XX XX	0 = Video 1 = System PCI 2 = System AGP 3 = Reserved
2…11	Reserved	✓	×		
12…31	Address				

Notes:

MemoryPageTableUpper

Name	Type	Offset	Format
MemoryPageTableUpper	Region Zero	0x03038	Bitfield

Control register

Bits	Name	Read	Write	Reset	Description
0…31	Address	✓	✓	0x XXX X.XX XX	

Notes:

MemoryPageTableLimit

Name	Type	Offset	Format
MemoryPageTableLimit	Region Zero	0x03040	Bitfield

Control register

Bits	Name	Read	Write	Reset	Description
0…21	Limit	✓	✓	0x XXX X.XX XX	The number of page table entries.
22…31	Reserved	✓	×		

Notes: Each entry is a 64-bit word detailing page address, access rights and so on. Max.virtual address is thus: (number of Entries) * (Page Size (4K))

MemoryCounter

Name	Type	Offset	Format
MemoryCounter	Region Zero	0x03048	Bitfield

Control register

Bits	Name	Read	Write	Reset	Description
0…31	Timer	✓	✓	0x XXX X.XX XX	Free running counter at MClk frequency

Notes:

MemoryProfileControl

Name	Type	Offset	Format
MemoryProfileControl	Region Zero	0x03050	Bitfield

Control register

Bits	Name	Read	Write	Reset	Description
0…3	Mode	✓	✓	0x000 0.0000	0 = Reset 1 = Memory low power 2 = Memory idle 3 = Memory active 4 = Memory reads 5 = Memory writes 6 = Memory refreshes
4	Controller	✓	✓		Memory controller to read results from. Counters in both controllers always run but must be read from one at a time
5…31	Reserved	✓	×		

Notes:

MemoryProfileCount

Name	Type	Offset	Format
MemoryProfileCount	Region Zero	0x03058	Bitfield

Control register

Bits	Name	Read	Write	Reset	Description
0…31	Count	✓	✓	0x000 0.0000	

Notes:

GPProfileControl

Name	Type	Offset	Format
GPProfileControl	Region Zero	0x03060	Bitfield

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Mode0	✓	✓	0x000 0.0010	For counter 0 (see below for modes)
8…15	Mode1	✓	✓		For counter 1 (see below for modes)
16…23	Mode2	✓	✓		For counter 3 (see below for modes)
24…31	Reserved	✓	×		

Notes: Modes:
0 = Reset 1 = PageMiss (count of TLB misses)
2 = PageFault (count of page faults)3 = PageDMA (time spent loading pages by DMA)
4 = PageStall (time spent stalled due to TLB miss, fault reporting, page loading)
5 = VertexShadingIdle 6 = VertexShadingInputVertex
7 = GeomBlocked 8 = PrimSetUpPoints
9 = PrimSetUpLines 10 = PrimSetUpTriangles
11 = RasteriserStalled 12 = RasteriserIdle
13 = RasteriserTiles 14 = ContextCacheMiss
15 = ContextIsocChanges 16 = ContextGeomChanges
17 = GSDEarlyExit 18 = GSDSameTile
19 = LBCacheMiss 20 = PixelCacheMiss
21 = TextureAddressPrimaryCacheMiss
22 = SecondaryTextureCacheMiss 23 = HostOutTile
24 = MemLBRead 25 = MemPixelRead
26 = MemTextureRead 27 = MemContextRead
28 = RectRasterizeTiles 29 = GPIO Context
30 = GPIO Message

GPProfileCount[0-3]

Name	Type	Offset	Format
GPProfileCount0	Region Zero	0x03068	integer
GPProfileCount1		0x03070	
GPProfileCount2		0x03078	
GPProfileCount3		0x03080	

Control register

Bits	Name	Read	Write	Reset	Description
0…31	count	✓	✓	0x000 0.0010	

Notes:

MVTimingA

Name	Type	Offset	Format
MVTimingA	Region Zero	0x03088	Bitfield

Control register

Bits	Name	Read	Write	Reset	Description
0…4	RowCycle	✓	✓	0x XXX X.XX XX	Minimum time between active RAS cycles to the same bank
5…9	RAS2CAS Write	✓	✓		Minimum time between RAS and CAS cycles to same bank
10…14	RAS2CAS Read	✓	✓		Minimum time between RAS and CAS cycles to same bank
15…19	CAS2RAS Write	✓	✓		Row precharge time + burst length
20…24	CAS2RAS Read	✓	✓		Row precharge time + burst length
25…29	RefreshCycle	✓	✓		Minimum time taken to complete refresh command
30…31	Reserved	✓	×		

Notes:

MVTimingB (

Name	Type	Offset	Format
MVTimingB	Region Zero	0x03090	Bitfield

Control register

Bits	Name	Read	Write	Reset	Description
0	RefreshEnable	✓	✓	0x XXX X.XX XX	0 = refresh disabled 1 = refresh enabled
1…14	RefreshPeriod	✓	✓		Number of memory clocks between refresh cycles. Minimum value = 63
15	Reserved	✓	×		
16…20	Activate2 Activate	✓	✓		Minimum time between RAS cycles to different banks
21…23	Write2Read	✓	✓		Delay from write cycle to read cycle. Equal to bus turnaround time + 1 + burst length/2
24…27	Read2Write	✓	✓		Delay from read cycle to write cycle. Equal to CAS latency + 1 + bus turn-around time
28…31	PowerDown Exit	✓	×		Minimum time to repower memory array [P9] Reserved – set to 0 [P10]

Notes:

MVCaps

Name	Type	Offset	Format
MClkControl	Region Zero	0x03098	Bitfield

Control register

Bits	Name	Read	Write	Reset	Description
0	LowPower	✓	×	0x XXX X.XX XX	Reserved – set to 0 [P10] Low Power [P9]
1,2	Column Address	✓	✓		Number of bits of column address. 0 = 8 1 = 9 2 = 10 3 = 11
3…5	CAS latency	✓	✓		Determines the CAS latency expected by the memory controller. The CasLatency parameter can be loaded directly with the appropriate value from the memory device data sheet plus 1.
6…31	Reserved	✓	×		

Notes:

MVMode

Name	Type	Offset	Format
MVMode	Region Zero	0x030A0	Bitfield

Control register

Bits	Name	Read	Write	Reset	Description
0…15	Mode	✓	✓	0x XXX X.XX XX	Bit pattern to load into mode register during initialization
16…31	ExtendedMode	✓	✓		Bit pattern to load into extended mode register during initialization

Notes:

MV0Clock

Name	Type	Offset	Format
MV0Clock	Region Zero	0x030A8	Bitfield

Control register

Bits	Name	Read	Write	Reset	Description
0…3	Delay	✓	✓	0x XXX X.XX XX	4 bit field giving values 0 to 15 in 220ps taps. 0 = 0 delay 15 = 15 x 220ps delay
4	Invert	✓	✓		clock invert
6…31	Reserved	✓	×		

Notes: To set a delay of 0.9ns, for example, would require more than 4 taps: of 220ps:
4 taps = 880ps , not enough
5 taps = 1000ps (1.0ns)

MV0StrobeInvert

Name	Type	Offset	Format
MV0StrobeInvert	Region Zero	0x030B0	Bitfield

Control register

Bits	Name	Read	Write	Reset	Description
0…7	In0..In7 [P9]	✓	✓	0x XXX X.XX XX	StrobeIn delays in 220ps taps
0…15	In0..In15 [P10]				
8…15	Reserved [P9]	✓	×		Reserved [P9]
16…23	Out0..Out7 [P9]	✓	✓		StrobeOut delays in 220ps taps
16…31	Out0..Out15 [P10]	✓	✓		
24…31	Reserved [P9]	✓	✓		Reserved

Notes: Delay chain per strobe out or strobe in to configure for varying memory types and layouts.

MV0StrobeOutDelay0

Name	Type	Offset	Format
MV0StrobeOutDelay0	Region Zero	0x030B8	Bitfield
	Control register		

Bits	Name	Read	Write	Reset	Description
0…3	Delay0	✓	✓	0x XXX X.XX XX	
4…7	Delay1	✓	✓		
8…11	Delay2	✓	✓		
12…15	Delay3	✓	✓		
16…19	Delay4	✓	✓		
20…23	Delay5	✓	✓		
24…27	Delay6	✓	✓		
28…31	Delay7	✓	✓		

Notes: Delay chain per strobe out or strobe in to configure for varying memory types and layouts.

MV0StrobeOutDelay1

Name	Type	Offset	Format
MV0StrobeOutDelay0	Region Zero	0x030C0	Bitfield
	Control register		

Bits	Name	Read	Write	Reset	Description
0…31	Reserved [P9]	0	✕	0x XXX X.XX XX	Reserved address in P9. Write has no effect, read as zero
0…3	Delay8	✓	✓	0x XXX X.XX XX	
4…7	Delay9	✓	✓		
8…11	Delay10	✓	✓		
12…15	Delay11	✓	✓		
16…19	Delay12	✓	✓		
20…23	Delay13	✓	✓		
24…27	Delay14	✓	✓		
28…31	Delay15	✓	✓		

Notes: Delay chain per strobe out or strobe in to configure for varying memory types and layouts.

MV0StrobeInDelay0

Name	Type	Offset	Format
MV0StrobeInDelay0	Region Zero	0x030C8	Bitfield
	Control register		

Bits	Name	Read	Write	Reset	Description
0…3	Delay0	✓	✓	0x XXX X.XX XX	
4…7	Delay1	✓	✓		
8…11	Delay2	✓	✓		
12…15	Delay3	✓	✓		
16…19	Delay4	✓	✓		
20…23	Delay5	✓	✓		
24…27	Delay6	✓	✓		
28…31	Delay7	✓	✓		

Notes: Delay chain per strobe out or strobe in to configure for varying memory types and layouts.

MV0StrobeInDelay1

Name	Type	Offset	Format
MV0StrobeInDelay0	Region Zero	0x030D0	Bitfield
	Control register		

Bits	Name	Read	Write	Reset	Description
0…31	Reserved [P9]	0	✗	0x XXX X.XX XX	Reserved address in P9. Write has no effect, read as zero
0…3	Delay8	✓	✓	0x XXX X.XX XX	
4…7	Delay9	✓	✓		
8…11	Delay10	✓	✓		
12…15	Delay11	✓	✓		
16…19	Delay12	✓	✓		
20…23	Delay13	✓	✓		
24…27	Delay14	✓	✓		
28…31	Delay15	✓	✓		

Notes: Delay chain per strobe out or strobe in to configure for varying memory types and layouts.

MV1Clock

Name	Type	Offset	Format
MV1Clock	Region Zero	0x030D8	Bitfield

Control register

Bits	Name	Read	Write	Reset	Description
0…3	Delay	✓	✓	0x XXX X.XX XX	
4	Invert	✓	✓		
6…31	Reserved	✓	×		

Notes:

MV1StrobeInvert

Name	Type	Offset	Format
MV1StrobeInvert	Region Zero	0x030E0	Bitfield

Control register

Bits	Name	Read	Write	Reset	Description
0…7	In0..In7 [P9]	✓	✓	0x XXX X.XX XX	StrobeIn delays in 220ps taps
0…15	In0..In15 [P10]				
8…15	Reserved [P9]	✓	×		Reserved [P9]
16…23	Out0..Out7 [P9]	✓	✓		StrobeOut delays in 220ps taps
16…31	Out0..Out15 [P10]	✓	✓		
24…31	Reserved [P9]	✓	✓		Reserved

Notes:

MV1StrobeOutDelay0

Name	Type	Offset	Format
MV1StrobeOutDelay0	Region Zero	0x030E8	Bitfield

Control register

Bits	Name	Read	Write	Reset	Description
0…3	Delay0	✓	✓	0x XXX X.XX XX	
4…7	Delay1	✓	✓		
8…11	Delay2	✓	✓		
12…15	Delay3	✓	✓		
16…19	Delay4	✓	✓		
20…23	Delay5	✓	✓		
24…27	Delay6	✓	✓		
28…31	Delay7	✓	✓		

Notes: Delay chain per strobe out or strobe in to configure for varying memory types and layouts.

MV1StrobeOutDelay1

Name	Type	Offset	Format
MV1StrobeOutDelay0	Region Zero	0x030F0	Bitfield

Control register

Bits	Name	Read	Write	Reset	Description
0…31	Reserved [P9]	0	✗	0x XXX X.XX XX	Reserved address in P9. Write has no effect, read as zero
0…3	Delay8	✓	✓	0x XXX X.XX XX	
4…7	Delay9	✓	✓		
8…11	Delay10	✓	✓		
12…15	Delay11	✓	✓		
16…19	Delay12	✓	✓		
20…23	Delay13	✓	✓		
24…27	Delay14	✓	✓		
28…31	Delay15	✓	✓		

Notes: Delay chain per strobe out or strobe in to configure for varying memory types and layouts.

MV1StrobeInDelay0

Name	Type	Offset	Format
MV1StrobeInDelay0	Region Zero	0x030F8	Bitfield

Control register

Bits	Name	Read	Write	Reset	Description
0…3	Delay0	✓	✓	0x XXX X.XX XX	
4…7	Delay1	✓	✓		
8…11	Delay2	✓	✓		
12…15	Delay3	✓	✓		
16…19	Delay4	✓	✓		
20…23	Delay5	✓	✓		
24…27	Delay6	✓	✓		
28…31	Delay7	✓	✓		

Notes: Delay chain per strobe out or strobe in to configure for varying memory types and layouts.

MV1StrobeInDelay1

Name	Type	Offset	Format
MV1StrobeInDelay1	Region Zero	0x03100	Bitfield

Control register

Bits	Name	Read	Write	Reset	Description
0…31	Reserved [P9]	0	✗	0x XXX X.XX XX	Reserved address in P9. Write has no effect, read as zero
0…3	Delay8	✓	✓	0x XXX X.XX XX	
4…7	Delay9	✓	✓		
8…11	Delay10	✓	✓		
12…15	Delay11	✓	✓		
16…19	Delay12	✓	✓		
20…23	Delay13	✓	✓		
24…27	Delay14	✓	✓		
28…31	Delay15	✓	✓		

Notes: Delay chain per strobe out or strobe in to configure for varying memory types and layouts.

MVSystemControl

Name	Type	Offset	Format
MVSystemControl	Region Zero	0x03108	Bitfield

Control register

Bits	Name	Read	Write	Reset	Description
0…7	TimeSlice	✓	✓	0x XXX X.XX XX	Sets the time in clocks between forced pre-emptions. Access to the PCI/AGP will normally be granted to the most efficient type of access from the perspective of the bus bandwidth, but this may be at odds with the the needs of the rest of the system so setting this to a low value will share access to the bus more fairly
8..15	Throttle	✓	✓		Sets the time between system accesses in clocks. Some systems do not operate efficiently if flooded with requests. Setting this value to greater than zero will insert wait states between requests to the PCI/AGP bus; the value should be set to approximately the balance the rate of generating requests with the rate at which they are serviced, and should account for both clock speed differences and the size of requests
16	FlipBypass	✓	✓		
17	FlipVGA/ VIP/Texture	✓	✓		
18	FlipGraphics Processor	✓	✓		
19	FlipCommand Processor	✓	✓		
20	FlipVideoProcessor0	✓	✓		
21	FlipVideo Processor1	✓	✓		
22	FlipTLB Update	✓	✓		
23	FlipPage Handler	✓	✓		
24…31	Reserved	✓	×		

Notes: **MVSystemControl**.*Flip** reverses the priority of arbitration for the source.

MVVideoControl

Name	Type	Offset	Format
MVVideoControl	Region Zero	0x03110	Bitfield

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Low	✓	✓	0x XXX X.XX XX	Sets the low threshold for display video requests. Increasing this value increases the amount of time the video will wait before making requests so increasing the potential bandwidth of units accessing memory.
8..15	High	✓	✓		Sets the high threshold for display video requests. Increasing this value increases the amount of time the video will wait before making high priority requests. It should be set to the highest value that does not result in video under-runs.
16	FlipBypass	✓	✓		
17	FlipVGA/ VIP/Texture	✓	✓		
18	FlipGraphics Processor	✓	✓		
19	FlipCommand Processor	✓	✓		
20	FlipVideo Processor0	✓	✓		
21	FlipVideo Processor1	✓	✓		
22	FlipTLB Update	✓	✓		
23	FlipPage Handler	✓	✓		
24	VideoPipe	✓	✓		Causes video requests to be taken into account during arbitration of requests from the graphics process
25…27	Q	✓	✓		Controls the number of requests that can be queued in the memory controller. The value should be set according to the relative clock frequencies of K and M clks to queue the minimum number that does not cause bubbles. The value must be non-zero (>0)
28…31	Reserved	✓	✕		

Notes: **MVVideoControl.***Flip** reverses the priority of arbitration for the source.

MPipeControl

Name	Type	Offset	Format
MPipeControl	Region Zero	0x03118	Bitfield

Control register

Bits	Name	Read	Write	Reset	Description
0…3	GPQ	✓	✓	0x XXX X.XX XX	
4…7	GPTurn Around)	✓	✓		
8…31	Reserved	✓	×		

Notes:

5.7.5 VGA Control 0x04000 – 0x04FFF4 K

The VGA registers generally follow industry VGA conventions. The registers described below are not comprehensive but include all chip-specific variants accessible via VGA I/O and addressable memory (described here) together with the Index registers which support them (**GraphicsIndexReg, SequencerIndexReg**) and others.

As well as the standard fixed I/O addresses at 0xA0000 through 0xBFFFF, the VGA Control registers are mapped into a 4K Byte space within Region Zero.[2] This allows driver software to initialise the VGA Controller without having to make I/O Space accesses (which can be difficult under some operating systems).

5.7.5.1 General Registers

These non-proprietary VGA registers are provided here for convenience only. For further information on VGA registers see, for example, IBM document SA14-2413-00 titled *Video Graphics Adaptor (VGA) Core*.

[2] VGA Control accesses through Region Zero are not affected by the **PciVgaIOColorDecode** enable signal so both monochrome and color accesses are always forwarded to the VGA Unit provided the *VgaEnable* bit is set in the **CFGBusConfig** register..

MiscOutputReg

Name	Type	Offset	Format
MiscOutputReg	VGA	0x3c2 0x3cc	Bitfield

General VGA register

Bits	Name	Read	Write	Reset	Description
0	IOAddress	✓	✓	0x XXX X.XX XX	This bit selects the I/O address for either Monochrome (0) or Colour (1) modes. The registers which change their port address are: 0 CRTC IndexReg 0x3b4 CRTC DataReg 0x3b5 FeatureControlReg 0x3ba InputStatus1Reg 0x3ba 1 CRTC IndexReg 0x3d4 CRTC DataReg 0x3d5 FeatureControlReg 0x3da InputStatus1Reg 0x3da
1	EnableRam	✓	✓		This bit controlles access to the display memory by the host. 0=No access to the display memory is made1. 1=Access to the display memory are made.
2,3	ClockSelect	✓	✓		These two bits control the frequency of the dot clock for video generation and are passed to an external clock synthesiser which generates the VClock. It is the programmers responsibility to ensure a clean transition by using the ResetRegister to force a reset state before changing the clock2.
4	Reserved	✓	×		.
5	PageSelect	✓	✓		This bit affects the meaning of the LSB of the Display Memory address when in Even/Odd mode (i.e. MemoryModeReg.EvenOdd = 1 and GraphicsMiscReg.ChainEvenOdd = 0 and MemoryModeReg.Chain4 = 0). In this case: 0=Odd memory locations are selected. 1=Even memory locations are selected.
6,7	SyncPolarity	✓	✓		These are dual purpose bits used to select screen size and the polarity of the sync signals. Screen size: 0 400 lines3. 1 400 lines. 2 350 lines. 3 480 lines. Active Sync Polarity: *HSync* *VSync* 0 High High 1 Low High 2 High Low 3 Low Low

Notes: 1. This signal is used by the external decode logic to enable/disable memory read and write decodes. When disabled no response is made to memory accesses in the VGA range.
2. Changing the clock (even using the ResetReg) is still likely to be a potentially dangerous thing to implement, giving rise to spikes and glitches so this needs to be actively taken into account in the implementation.
3. This is marked as reserved by other VGA chips, but it needs to be defined as doing something. These bits don't get used to set up the internal video timing and are only used to help control a multisync monitor.

FeatureControlReg

Name	Type	Offset	Format
FeatureControlReg	VGA	0x3ca 0x3?a	Bitfield

General VGA register

Bits	Name	Read	Write	Reset	Description
0…2	Reserved	✓	✕	0x XXX X.XX XX	
3	VSyncControl	✓	✓		0 = Normal vertical sync 1 = The VSync signal is logically ored with DisplayEnable (an internal signal) before going to the RAMDAC.
4…7	Reserved	✓	✕		

Notes: "?" in offset address (port) == b when in Monochrome Mode (MiscOutputRegIOAddress=0) or == d when in Color Mode (MiscOutputRegIOAddress=1)

InputStatus0Reg

Name	Type	Offset	Format
InputStatus0Reg	VGA	0x3c2	Bitfield

General VGA register

Bits	Name	Read	Write	Reset	Description
0…3	Reserved	✓	✕	0x XXX X.XX XX	
4	SwitchSense	✓	✕		Always returns 0 as there are no switches to read..
5,6	Reserved	✓	✕		
7	Vsync Interrupt	✓	✕		0=Vertical interrupt is clear 1=Vertical interrupt is pending

Notes:

InputStatus1Reg

Name	Type	Offset	Format
InputStatus1Reg	VGA	0x3?a	Bitfield

General VGA register

Bits	Name	Read	Write	Reset	Description
0	DisplayEnable	✓	✕	0x XXX X.XX XX	This bit follows the state of the DisplayEnable signal (HDisplayEnable & VDisplayEnable) from the video output stage. Note this signal spans clock domains. 0=Video is being displayed 1=Blanking is active
1,2	Reserved	✓	✕		
3	VSync	✓	✕		This bit follows the state of the VerticalSync signal from the video output stage. Note this signal spans clock domains. 0=Vertical retrace is not in progress 1=Vertical retrace is in progress
4,5	Diagnostic	✓	✕		These bits follow two of the eight outputs of the Pixel FIFO. The selection is made according to ColourPlaneEnableReg.VideoStatusMux. The value of this field selects the output (P7:P0) of the VData lines from the Video Timing Generator as follows1: VideoMuxSelect *Bit 5* *Bit 4* 0 P2 P0 1 P5 P4 2 P3 P1 3 P7 P6
6,7	Reserved	✓	✕		

Notes:
- "?" in offset address (port) == b when in Monochrome Mode (MiscOutputRegIOAddress=0) or == d when in Color Mode (MiscOutputRegIOAddress=1)
- There is some disagreement between the VGA chips as to which bits are selected by this field. We have followed the majority (the Cirrus chip GD542 is the odd one out).
- A side effect of reading this register is to clear the attribute toggle flip flop.

DACReadIndexReg

Name	Type	Offset	Format
DACReadIndexReg	VGA	0x3c7	Bitfield

General VGA register

Bits	Name	Read	Write	Reset	Description
0…7		✕	✓	0x XXX X.XX XX	This field holds the address of the entry in the RAMDAC LUT to be read when a read is done to the DACDataReg port. This index is automatically incremented on the conclusion of every third read to the DACDataReg.

Notes:
- This register is in the RAMDAC so writes will just be passed through with the necessary protocol changes.
- This port address is used to return status when read so cannot be used to return the current index value. This functionality is imposed by the RAMDAC and accepted as part of the normal VGA behaviour.

DACWriteIndexReg

Name	Type	Offset	Format
DACWriteIndexReg	VGA	0x3c8	Bitfield

General VGA register

Bits	Name	Read	Write	Reset	Description
0…7		✓	✓	0x XXX X.XX XX	This field holds the address of the entry in the RAMDAC LUT to be written when a write is done to the DACDataReg port. This index is automatically incremented on the conclusion of every third write to the DACDataReg

Notes:
- This register is in the RAMDAC so writes will just be passed through with the necessary protocol changes..

DACDataReg

Name	Type	Offset	Format
DACDataReg	VGA	0x3c9	Bitfield

General VGA register

Bits	Name	Read	Write	Reset	Description
0…7		✓	✓	0x XXX X.XX XX	This field holds the LUT data for the RAMDAC. Before writing to this register, write the first (or only) LUT index to **DACWriteIndexReg.** The next three writes to this register, corresponding to Red, Green and Blue values are written to the LUT at the current write index value. The write index value is then incremented[1]. Before reading this register write the first (or only) LUT index to **DACReadIndexReg**. The next three reads to this register will return the Red, Green and Blue values from the LUT at the current read index value. The read index value is then incremented

Notes:
- This register is in the RAMDAC so writes will just be passed through with the necessary protocol changes
- At which point the new LUT data is written to the LUT in the RAMDAC (all three words at once, or individually) depends on the RAMDAC.
- The effect of mixing writes and reads part way through a colour triplet is not defined and again will be a function of the RAMDAC.
- Some RAMDACs impose a maximum rate at which their internal registers can be written to (by a host). There is nothing in any of the VGA chips with internal RAMDACs which indicates this is something for the programmer to take into account when updating the LUT (for example). Any time required by the RAMDAC is handled outside of the VGA core by the RAMDAC interface..

DACStateReg

Name	Type	Offset	Format
DACStateReg	VGA	0x3c7	Bitfield

General VGA register

Bits	Name	Read	Write	Reset	Description
0,1	LastAccess	✓	✓	0x XXX X.XX XX	These two bits return the current state of the RAMDAC and will always be the same. The two values are 0=A write operation is in progress or occurred last. 3=A read operation is in progress or occurred last
2…7	Reserved	✓	✕		

Notes:
- This register is in the RAMDAC so writes will just be passed through with the necessary protocol changes..

DACMaskReg

Name	Type	Offset	Format
DACMaskReg	VGA	0x3c6	Bitfield

General VGA register

Bits	Name	Read	Write	Reset	Description
0…7		✓	✓	0x XXX X.XX XX	This field holds a mask value which is applied to the 8 bit pixel data passed to the RAMDAC prior to it being used to index the look up table

Notes: This register is in the RAMDAC so writes will just be passed through with the necessary protocol changes..

5.7.5.2 Sequencer Registers

The sequencer registers mainly control the global clock and memory modes. They also contain all but one of the non-standard (i.e. extended) VGA registers:

- **VGAControlReg** – assorted 3Dlabs-specific controls.
- Locking registers – lock the extended registers.
- Bank switching registers – enable bank switching of the 0xa0000/0xb0000 regions through the bypass.
- Genlocking registers – allow the VTG to be synchronized to an external video source.
- Scratch registers – available for use as an information store.
- Indirect base registers – follow the state of the HIndirectBase signals from the PCI interface.
- Alternative timing registers – intended for flat panels: they provide greater screen resolution and are protected from host interference.

Note: The following Sequencer registers are non-standard 3Dlabs additions: Sequencer Index 0x05 to Index 0x37

SequencerIndexReg

Name	Type	Offset	Format
SequencerIndexReg	VGA	0x3c4	Bitfield

VGA Sequencer register

Bits	Name	Read	Write	Reset	Description
0…5	Index	✓	✓	0x XXX X.XX XX	This index points to one of the sequencer registers which will get read or written on the next I/O access to the SequencerPort (0x3c5). The registers and their corresponding indices are: 0x00 ResetReg 0x01 ClockModeReg 0x02 MapMaskReg 0x03 CharacterMapSelectReg 0x04 MemoryModeReg 0x05 VGAControlReg 0x06 LockExtended1Reg 0x07 LockExtended2Reg 0x08 BankALowReg 0x09 BankAHighReg 0x0a BankBLowReg 0x0b BankBHighReg 0x0c PCIControlReg 0x0d HLockShiftReg 0x0e VLockShiftReg 0x0f GenLockControlReg 0x10 .. 0x1f ScratchRegs 0x20 .. 0x23 IndirectBaseRegs 0x24 AltHTotalLowReg 0x25 AltHDisplayEndLowReg 0x26 AltHBlankingStartLowReg 0x27 AltHBlankingEndLowReg 0x28 AltHSyncStartLowReg 0x29 AltHSyncEndReg 0x2a AltVTotalLowReg 0x2b AltVDisplayEndLowReg 0x2c AltVBlankingStartLowReg 0x2d AltVBlankingEndLowReg 0x2e AltVSyncStartLowReg 0x2f AltVSyncEndReg 0x30 AltHOverflowReg 0x31 AltVOverflow1Reg 0x32 AltVOverflow2Reg

					0x33 AltVOverflow3Reg 0x34 AltLineCompareLowReg 0x35 AltLineCompareHighReg 0x36 reserved 0x37 HeadSelectReg 0x38 .. 0x3f None[1]
6,7	Reserved	✓	×		

Notes:
- This register is in the RAMDAC so writes will just be passed through with the necessary protocol changes..

ResetReg

Name	Type	Offset	Format
ResetReg	VGA	0x3c5 index 00	Bitfield

VGA Sequencer register

Bits	Name	Read	Write	Reset	Description
0	AsyncReset	✓	✓	0x XXX X.XX XX	0=Places the logic in a safe state so a different dot clock can be switch in to control the video timing. The display is blanked and HSync and VSync placed in their quiescent state (depending on their respective polarity)[1, 2]. 1=Normal VGA operation.
1	SyncReset	✓	✓		0=Places the logic in a safe state so a different dot clock can be switch in to control the video timing. The display is blanked and HSync and VSync placed in their quiescent state (depending on their respective polarity)[2]. 1=Normal VGA operation.
2…7	Reserved	✓	×		

Notes:
- The exact use of these resets is a bit vague (in the VGA documentation), however the real use is to prevent corruption of the display memory when the clock frequency is changed by the MiscOutputReg.ClockSelect. In this design this is never a possibility as the clock domain for memory accesses is independent of the dot clock. How these signals interact with each other is not consistent between the VGA chips. The AsyncReset is treated as the SyncReset and either can be used (the IBM VGA does this).
- Not all VGA chips define this effect on the monitor signals but it is probably desirable otherwise a monitor can be driven in an undesirable way.

ClockingModeReg

Name	Type	Offset	Format
ClockingModeReg	VGA	0x3c5 index 01	Bitfield

VGA Sequencer register

Bits	Name	Read	Write	Reset	Description
0	Character Dot8_9	✓	✓	0x XXX X.XX XX	Controls how the dot clock is divided to produce the character clock[1]. 0 Selects that each character has 9 dots horizontally. 1 Selects that each character has 8 dots horizontally.
1	Reserved	✓	×		Reserved.
2..4	VideoLoad Control	✓	✓		This field determines how frequently the video shift registers are loaded: 0 Every character clock 1 Every other character clock 2 Every fourth character clock 3 Every fourth character clock Note this two bit field is not contiguous.
3	DotClock DivTwo	✓	✓		This bit controls the relationship between the dot clock and the master clock (whose frequency is selected by **MiscOutput**.*ClockSelect*). 0 Use the master clock 1 Divide the master clock by two
5	ScreenOff	✓	✓		This bit blanks the screen to prevent video access to the display memory so all the available bandwidth is devoted to the host. The bandwidth saving feature is not necessary in this design so is ignored, however the side effect is still used. 0=Normal operation. 1=The pixel colour is taken from the **OverscanColourReg**.
6,7	Reserved	✓	×		

Notes:
- Most of the bits in this register are used in the video clock domain.
- This field is ignored when in Graphics mode (**AttributeModeReg**.*GraphicsMode*) and there is always 8 dots per character..

MapMaskReg

Name	Type	Offset	Format
MapMaskReg	VGA	0x3c5 index 02	Bitfield

VGA Sequencer register

Bits	Name	Read	Write	Reset	Description
0…3	Mask	✓	✓	0x XXX X.XX XX	These bits effectively act as a byte mask for memory writes: Bit 0 controls the least significant byte bit 3 controls the most significant byte.
4…7	Reserved	✓	×		

Notes: Each byte is sometimes called a Map or Plane and each map has assigned functionality in different modes (i.e. in Text mode map 0 holds the character code, map 1 the attribute byte). A 0 in a bit position means that no write occurs to the corresponding byte

CharMapSelectReg

Name	Type	Offset	Format
CharMapSelectReg	VGA	0x3c5 index 03	Bitfield

VGA Sequencer register

Bits	Name	Read	Write	Reset	Description
0,1,4	CharMap SelectB	✓	✓	0x XXX X.XX XX	Selects the "B" character set to use. See table in the notes section. These bits are renamed F0 (4), F1 (0) and F2 (1) to reflect the binary weighting they represent. *Note: this three bit field is not contiguous.*
5,3,2	CharMap SelectA	✓	×		Selects the "A" character set to use. See table below. These bits are renamed F0 (5), F1 (2) and F2 (3) to reflect the binary weighting they represent. *Note: this three bit field is not contiguous.*
6,7	Reserved	✓	×		

Notes:

- The select value determines the base offset of the map as follows:

Value	Offset
0	0K
1	16K
2	32K
3	48K
4	8K
5	24K
6	40K
7	56K

- Bit 3 of the attribute byte normally controls the intensity of the foreground colour. This bit may be redefined to be a switch between character sets allowing 512 displayable characters. This switch is enabled whenever CharMapSelectA is different to CharMapSelectB and MemoryModeReg.ExtendedMemory is 1.
- The address of the character in plane 2 of the display memory is given by the concatenation of the three bit fields: [F2:0][C7:0][R4:0], where F is the active CharMapSelect value, C is the ASCII character code and R is the character row...

MemoryModeReg

Name	Type	Offset	Format
MemoryModeReg	VGA	0x3c5 index 04	Bitfield

VGA Sequencer register

Bits	Name	Read	Write	Reset	Description
0	Reserved	✓	×		
1	ExtendedMemory	✓	✓	0x XXX X.XX XX	Controls access to the display memory by the host. 0=The effective memory size is 64KBytes regardless of the actual installed memory. 1=Allows access to all the installed memory. Also enables character map selection in the CharMapSelectReg.
2	EvenOdd	✓	✓		0=Even host addresses will access planes 0 and 2 in the display memory (corresponds to bytes 0 and 2). Odd host addresses will access planes 1 and 3 in the display memory (i.e. the LSB of the host address selects between pairs of bytes). 1=The host address selects 32 bit words and the MapMaskReg determines which bytes get written. This bit is set to 0 for text modes and should have the be opposite state to GraphicsModeReg.EvenOdd field for consistent operation (programmer's responsibility). This bit only effect writes to the display memory by the host and the next bit (ChainFour) must be 0 for this bit to have any effect.
3	ChainFour	✓	✓		0=No effect. 1=The LS two bits of the host address are used to select to byte to read or write from the display memory. The word address is taken from the remaining bits. This bit has priority over the previous EvenOdd bit. The ReadMapReg is ignored. This bit is only set when in mode 13 - 256 colour mode.
4..7	Reserved	✓	×		

Notes: •

VGAControlReg

Name	Type	Offset	Format
VGAControlReg	VGA	0x3c5 index 05	Bitfield

VGA Sequencer register

Bits	Name	Read	Write	Reset	Description
0	EnableHost Memory Access	✓	✓	0x XXX X.XX XX	Controls access to the display memory by the host. 0=No access to the display memory is made in response to host VGA memory accesses. Writes are ignored and reads always return zero. All the host bus cycles are completed as normal. 1=Normal access to the display memory occurs. This bit is further qualified by the VGAEnable signal which acts as a global disable.
1	EnableHost DacAccess	✓	✓		Controls access to the RAMDAC by the host. 0=No access to the RAMDAC is made in response to host Dac accesses. Writes are ignored and reads always return zero. All the host bus cycles are completed as normal. 1=Normal access to the RAMDAC occurs. This bit is further qualified by the VGAEnable signal which acts as a global disable.
2	Enable Interrupts	✓	✓		0=Prevents any interrupts from being generated by the VGA core. 1=Enables interrupt generation from the VGA core providing the VerticalSyncEndReg.DisableVerticalInterrupt field is set to zero. This bit is further qualified by the VGAEnable signal which acts as a global disable. This additional enable bit is provided so the VGA core can be disabled from one place.
3	EnableVGA Display	✓	✓		Controls access to the display memory by the Memory Reader for the purpose of keeping the display refreshed. It also tells (on the VGAVidSelect signal) the video select logic external to the VGA core that the display should be driven from the VGA core. 0=No accesses to display memory are to be made and the video source should not be the VGA core. The Memory Reader, Attribute Controller and Video Timing Generator are held in their reset state. 1=Accesses to the display memory are made and the video to be displayed comes from the VGA core. This bit is further qualified by the VGAEnable signal which acts as a global disable.
4	DacAddr2	✓	✓		This bit extends the RAMDAC address range.
5	DacAddr3	✓	✓		This bit extends the RAMDAC address range.

6	EnableVTG	✓	✓		0=Stops the VTG running and producing sync pulses. 1=Enables the VTG to run and produce sync pulses. This bit only has an effect when the VGA display has been disabled by EnableVGADisplay. When the display has been disabled by VGAEnable this bit is ignored. When the VGA dispaly is active then this bit is ignored.
7	InvertVBlank	✓	×		0=No Invert VBlank. 1=Invert VBlank

Notes:
- On reset *EnableHostMemoryAccess*, *EnableHostDacAccess* and *EnableVGADisplay* are enabled, *EnableInterrupts* is disabled and *DacAddr2* and *DacAddr3* bits are set to 0, *InvertVBlank* is set to 0.
- This is a non-standard (i.e. extended) VGA register

LockExtended1Reg
LockExtended2Reg

Name	Type	Offset	Format
LockExtended1Reg	VGA	0x3c5	Bitfield
LockExtended2Reg		index 06	
		index 07	

VGA Sequencer register

Bits	Name	Read	Write	Reset	Description
0…7		×	✓	0x XXX X.XX XX	These 2 registers act as a lock for the extended registers.

Notes: On reset extended registers are locked – they cannot be written and read back as 0, and the sequencer index behaves as a 3-bit index. Writing the value 0x3d to **LockExtended1Reg** followed by 0xdb to **LockExtended2Reg** unlocks the extended registers. Writing any other values locks them

BankALowReg
BankBLowReg

Name	Type	Offset	Format
BankALowReg BankBLowReg	VGA	0x3c5 index 08 index 0A	Bitfield

VGA Sequencer register

Bits	Name	Read	Write	Reset	Description
0…7	Bank[A or B]7_0	✓	✓	0x XXX X.XX XX	Holds the 8 low order bits of the 10-bit BankA base address.

Notes:
- The 2 high order bits can be found in **BankAHighReg** or **BankBHighReg**.
- The BankA and Bank B base addresses are used for bank switching the 0xa0000 and 0xb0000 regions through the bypass (if enabled).
- The BankA bits provide the HBankA signals to the PCI interface, the BankB bits provide the HBankB signals to the PCI interface.
- These fields should not be confused with the **Mode640Reg** Bank A and Bank B fields which are deprecated.

BankAHighReg
BankBHighReg

Name	Type	Offset	Format
BankAHighReg BankBHighReg	VGA	0x3c5 index 09 index 0B	Bitfield

VGA Sequencer register

Bits	Name	Read	Write	Reset	Description
0,1		✓	✓	0x XXX X.XX XX	Holds the 2 high order bits of the 10-bit BankB base address.
2…7	Reserved	✓	✕		

Notes:
- The 8 low order bits can be found in **BankALowReg** and **BankBLowReg** .
- The BankB base address is used for bank switching the 0xa0000 and 0xb0000 region through the bypass (if enabled).
- The BankA bits provide the HBankA signals to the PCI interface. The BankB bits provide the HBankB signals to the PCI interface.
- These fields should not be confused with the **Mode640Reg** Bank A and Bank B fields which are deprecated

PCIControlReg

Name	Type	Offset	Format
DACWriteIndexReg	VGA	0x3c5 index 0x0C	Bitfield

VGA Sequencer register

Bits	Name	Read	Write	Reset	Description
0	BankEnable	✓	✓	0x XXX X.XX XX	If set, enables bank switching of the 0xa0000/0xb0000 regions through the bypass, using the 10-bit BankA/BankB base addresses. This bit provides the HBankEnable signal to the PCI interface.
1	IndirectEnable	✓	✓		If set, enables access to chip registers via I/O ports 0x3b0/0x3b1/0x3d0/0x3d1. This bit provides the HIndirectEnable signal to the PCI interface.
2…7	Reserved	✓	✕		

Notes:

HLockShiftReg

Name	Type	Offset	Format
HLockShiftReg	VGA	0x3c5 index 0E	Bitfield

VGA Sequencer register

Bits	Name	Read	Write	Reset	Description
0…7		✓	✓	0x XXX X.XX XX	If genlocking is enabled, this field specifies the number of characters by which the horizontal blank end is delayed

Notes:

VLockShiftReg

Name	Type	Offset	Format
VLockShiftReg	VGA	0x3c5 index 0E	Bitfield

VGA Sequencer register

Bits	Name	Read	Write	Reset	Description
0…7		✓	✓	0x XXX X.XX XX	If genlocking is enabled, this field specifies the number of scanlines by which the vertical blank end is delayed

Notes:

GenLockControlReg

Name	Type	Offset	Format
GenLockControlReg	VGA	0x3c5 index 0F	Bitfield

VGA Sequencer register

Bits	Name	Read	Write	Reset	Description
0	Enable	✓	✓	0x XXX X.XX XX	Allows the VTG to be synchronized to an external video source via the *GenLockHSync* and *GenLockVSync* pins
1…7	Reserved	✓	✕		

Notes: Enabling GenLock causes the horizontal & vertical sync starts & blank ends to be delayed. Sync starts are delayed until the arrival of the ExtHSync & ExtVSync signals. Blank ends are delayed by the numbers specified in the **HLockShiftReg** & **VLockShiftReg** registers..

ScratchReg[0x0-0xF]

Name	Type	Offset	Format
ScratchReg[[0x0-0xF]	VGA	0x3c5 index 10 to index 1F	Bitfield

VGA Sequencer register

Bits	Name	Read	Write	Reset	Description
0…7		✓	✓	0x XXX X.XX XX	These registers are available for use as an information store and do not affect the VGA operation

Notes:

IndirectBaseReg[0x0-0x3]

Name	Type	Offset	Format
IndirectBaseReg[0x0-0x3]	VGA	0x3c5 index 20 to index 23	Bitfield

VGA Sequencer register

Bits	Name	Read	Write	Reset	Description
0…7		✓	✓	0x XXX X.XX XX	These 4 registers follow the state of the HIndirectBase signals from the PCI interface. **IndirectBaseReg[0]** returns bits 7..0, **IndirectBaseReg[1]** returns bits 15..8, **IndirectBaseReg[2]** returns bits 23..16, and **IndirectBaseReg[3]** returns bits 31..24

Notes:

HeadSelectReg

Name	Type	Offset	Format
HeadSelectReg	VGA	0x3c5 index 0x37	Bitfield

VGA Sequencer register

Bits	Name	Read	Write	Reset	Description
0,1	DisplayHeadEnable	✓	✓	0x XXX X.XX XX	Condition the 2-bit VGAVidSelect signal. Bit 0 enables output to Head 0. Bit 1 enables output to Head 1.
2	RamdacHead Select	✓	✓		Selects the display head for Ramdac accesses.
3…7	Reserved	✓	✗		

Notes: Allows specific head enable/disable e.g. when changing dual head video mode. See the *Programmer's Guide* section on Dual Head Video Output.

5.7.5.3 CRTC Registers

The CRTC registers provide timing and control of video display characteristics such as .Display Start/End, Syncing and Blanking. These are non-proprietary registers and can be found in VGA reference manuals. See for example IBM document SA14-2413-00 titled *Video Graphics Adaptor (VGA) Core*

5.7.5.4 Graphics Registers

These are primarily non-proprietary VGA registers which are provided here for convenience only. For further information on VGA registers see, for example, IBM document SA14-2413-00 titled *Video Graphics Adaptor (VGA) Core*.

Note: The following Graphics register is a non-standard 3Dlabs addition: Graphics Index 0x09 – ***Mode640Reg.***

GraphicsIndexReg

Name	Type	Offset	Format
GraphicsIndexReg	VGA	0x3ce	Bitfield

VGA Graphics register

Bits	Name	Read	Write	Reset	Description
0…3	Index	✓	✓	0x XXX X.XX XX	This index points to one of the Graphics registers which will get read or written on the next I/O access to the GraphicsPort (0x3cf). The registers and their corresponding indices are: 0x0 SetResetReg 0x1 SetResetEnableReg 0x2 ColourCompareReg 0x3 DataRotateReg 0x4 ReadMapSelectReg 0x5 GraphicsModeReg 0x6 GraphicsMiscReg 0x7 ColourDontCareReg 0x8 BitMaskReg 0x9 Mode640Reg 0xa None1 : : : : 0xf None1
4…7	Reserved	✓	×		

Notes: Writes to a field denoted 'None' have no effect as the write is simply discarded. Reading from a field denoted 'None' returns zero

SetResetReg

Name	Type	Offset	Format
SetResetReg	VGA	0x3CF index 0x0	Bitfield

VGA Graphics register

Bits	Name	Read	Write	Reset	Description
0…3	Mask	✓	✓	0x XXX X.XX XX	This mask is the value written to the respective memory planes for Write Modes 0 and 3. Bit zero controls byte 0, the least significant byte. All eight bits in a plane or byte are written with the same value. This is described more fully in the **WriteMode** description
4…7	Reserved	✓	×		

Notes:

SetResetEnableReg

Name	Type	Offset	Format
SetResetEnableReg	VGA	0x3CF index 0x1	Bitfield

VGA Graphics register

Bits	Name	Read	Write	Reset	Description
0..3	Mask	✓	✓	0x XXX X.XX XX	These bits, together with **SetResetReg**.*Mask* determine the values written into the display memory when in Write Mode 0.
3…7	Reserved	✓	×		

Notes: If a bit in this field is set to 1 then the corresponding value in SetResetReg.Mask will be written into the corresponding display memory byte (all bits take the same value). If this bit is set to 0 then the corresponding value from the host data bus will be written into the corresponding display memory plane (all bits take the same value).
This is described more fully in the **WriteMode** description.

ColorCompareReg

Name	Type	Offset	Format
ColorCompareReg	VGA	0x3CF index 0x2	Bitfield

VGA Graphics register

Bits	Name	Read	Write	Reset	Description
0..3	Mask	✓	✓	0x XXX X.XX XX	These 4 bits provide the colour to compare each of the 8 four bit pixels read from display memory when in ReadMode 1
4…7	Reserved	✓	×		

Notes:

DataRotateReg

Name	Type	Offset	Format
DataRotateReg	VGA	0x3CF index 0x2	Bitfield

VGA Graphics register

Bits	Name	Read	Write	Reset	Description
0…2	RotateCount	✓	✓	0x XXX X.XX XX	This field specifies the number of bit positions the host data is to be rotated right before being submitted for further processing. This rotation only occurs in WriteModes 0 and 3
3,4	FunctionSelect	✓	✓		This field specifies the logical operation between the host data (after rotation) and the data in the DataLatch. 0=XOR
5…7	Reserved	✓	×		

Notes:

ReadMapSelectReg

Name	Type	Offset	Format
ReadMapSelectReg	VGA	0x3CF index 0x4	Bitfield

VGA Graphics register

Bits	Name	Read	Write	Reset	Description
0,1	Select	✓	✓	0x XXX X.XX XX	Specifies the byte in display memory which is to be read by the host when in ReadMode 0. If **MemoryModeReg**.*EvenOdd* = 0 then bit 0 is ignored and the least significant bit of the address used instead. This field is ignored when the **MemoryModeReg** .*ChainFour* field is 1
2…7	Reserved	✓	✕		

Notes:

GraphicsMiscReg

Name	Type	Offset	Format
GraphicsMiscReg	VGA	0x3CF index 0x6	Bitfield

VGA Graphics register

Bits	Name	Read	Write	Reset	Description
0	GraphicsMode	✓	✓	0x XXX X.XX XX	0=Test 1=Graphics
1	ChainEven Odd	✓	✓		0=No chaining
2,3	MemoryMap	✓	✓		This field specifies the size and position of the visible display memory in the host address space. Value Start Address Length 0 0xa0000 128K 1 0xa0000 64K 2 0xb0000 32K 3 0xb8000 32K Accesses to the start address are translated to accesses in the display memory at address 0. This field is passed to the external address decoder so only the appropriate address range is decoded and passed to the VGA core.

Notes:

ColorDontCareReg

Name	Type	Offset	Format
ColorDontCareReg	VGA	0x3CF index 0x7	Bitfield

VGA Graphics register

Bits	Name	Read	Write	Reset	Description
0…3	Mask	✓	✓	0x XXX X.XX XX	These four bits control whether the corresponding bit in a 4 bit pixel will take part in the colour compare operation. If a bit is 0 then the corresponding bit will not take part in the colour compare test and will consequently return a true result for this bit.
4…7	Reserved	✓	×		

Notes:

BitMaskReg

Name	Type	Offset	Format
BitMaskReg	VGA	0x3CF index 0x8	Bitfield

VGA Graphics register

Bits	Name	Read	Write	Reset	Description
0…7	Mask	✓	✓	0x XXX X.XX XX	Each bit in this field controls whether the corresponding bit in the display memory is written in WriteModes 0, 2 and 3. If a bit is 0 then the corresponding bit in display memory will not be written. It is the programmers responsibility to have read from the address already so its contents are in the DataLatch.

Notes:

Mode640Reg

Name	Type	Offset	Format
Mode640Reg	VGA	0x3CF index 0x9	Bitfield

VGA Graphics register

Bits	Name	Read	Write	Reset	Description
0…2	BankA[2:0]	✓	✓	0x XXX X.XX XX	This field provides the additonal address bits needed when the horizontal screen resolution is 640 pixels and a host address is beign made to the 64K region starting at address 0xa0000.
3..5	BankB[2:0]	✓	✓		This field provides the additonal address bits needed when the horizontal screen resolution is 640 pixels and a host address is beign made to the 64K region starting at address 0xb0000.
6	StartAddress16	✓	✓		The most significant bit of the StartAddress when mode 640 is enabled.
7	Enable	✓	✓		0=No action. 1=The VGA core operates in 640 resolution mode.

Notes: This is a non-standard (i.e. extended) VGA register which supports the 640 horizontal resolution modes used in SVGA

5.7.5.5 Attribute Registers

These are non-proprietary VGA registers. For further information on VGA registers see, for example, IBM document SA14-2413-00 titled *Video Graphics Adaptor (VGA) Core*.

5.7.6 ROM Control (0x05000 – 0x05FFF) (4KB)

This unit controls accesses to a serial ROM and any other devices sharing the same bus. The serial bus uses a standard 2-wire protocol that is compatible with ROMs such as Xicor 24512 and Atmel AT24C512; it is also used to control other devices such as TV encoders.

As many as 4 ROM devices can be fitted to the bus. Each is expected to be 64Kbytes and are consecutive in addressing. This allows more storage for situations where more than a simple VGA BIOS is needed (e.g. the UGA pCode). The chip configuration data is held at the end of the first 64K ROM.

If the ROM controller is used to access something other than a ROM, the CPU should use the **ROMControl** register to issue appropriate commands and read or write bytes of data according to the bus protocol. This mechanism uses the ROM pulse width and setup parameters as specified in the **ROMTiming** register, and the command field is returned to NOP when the operation is complete. Alternatively, setting the *Override* bit in the **ROMcontrol** register allows direct control of the SBClk* and SBData* pins using the Data in/out and Clock in/out fields to bypass the timing parameters..

At reset the PCI configuration unit reads setup data from the ROM assuming the dual-purpose UseROMConfig pin[3]. During reset the controller must be able to access the ROM without software assistance. Reads from the PCI unit are transferred through a FIFO - each read is for 32 bits converted into four byte reads. The ROM auto-incrementing address is used to improve performance.

Performance can also be improved by setting the timing parameters (pulse width and setup) in **ROMTIming** to values appropriate to the ROM. The reset timings are conservative and can be modified by the PCI configuration unit as part of the bootstrap (i.e. accurate timings are read from the ROM and then loaded into this unit).

The registers are aligned to 64 bit boundaries (i.e. the addressing units are 32 bits) with byte address offsets from the region base address.

[3] On P10 the UseROMConfig pin is *VidAVSync*. On P9 the UseROMConfig pin is VidInData(1). See the Reset and Pinlist chapters in *Reference Guide* Volume IV for more information.

ROMTiming

Name	Type	Offset	Format
ROMTiming	Region Zero	0x05000	Bitfield

Control register

Bits	Name	Read	Write	Reset	Description
0…15	Command	✓	✓	0x002 0.0200	Count of PCI clocks for command operations (pulse width)
16…23	Data	✓	✓		Count of PCI clocks for data operations (setup and hold for data transfers)
24…31	ReadBurst	✓	✓		Count of 32 bit reads that will be done to consecutive addresses

Notes:
- There are 3 fields: *Command*, *Data*, and *ReadBurst*. I2C defines different timing parameters for command and data transfers.
- Command should be loaded with the required pulse width; this value will also be used for command setup and hold (units PClks).
- Data should be loaded with required setup and hold for data transfers (units PClks).
- Burst is a special optimization. Load it with the number of 32 bit reads you intend to do from the ROM and it will burst that number at a higher speed. If burst is set to anything other than 0 you must always do a multiple of burst reads (because readburst is burst size minus 1).

ROMControl

Name	Type	Offset	Format
ROMControl	Region Zero	0x05008	Bitfield

Control register

Bits	Name	Read	Write	Reset	Description
0	Override	✓	✓	0x000 0.0000	0 = Normal operation 1 = Drive SBClk and SBData pins direct from this register
1…3	Command	✓	✓		0 = NOP 1 = Start 2 = Stop 3 = Read 4 = Write 5 = ReadAck
4	Busy	✓	×		0 = Controller is idle 1 = Access taking place
5	Error	✓	✓		Cleared by writing 1 0 = Correct operation 1 = Slave reported error during transaction
6	ClockIn	✓	×		SbClk
7	DataIn	✓	×		SBData
8	ClockOut	✓	✓		SBClk
9	DataOut	✓	✓		SBData
10	AckPolling	✓	✓		0 = Disabled 1 = Poll until ack received
11…15	Reserved	✓	×		
16…23	ByteIn	✓	✓		
24…31	ByteOut	✓			

Notes: The *Override* field allows direct control of the *SBClock* and *SBData* pins.
Common across all heads

ROMSpinlock

Name	Type	Offset	Format
ROMSpinlock	Region Zero	0x05010	Bitfield

Control register

Bits	Name	Read	Write	Reset	Description
0	Lock	✓	✓	0x000 0.0000	0 = Free 1 = Locked
1…31	ID	✓	×		Read/write field holding ID of current user.

Notes:

5.7.7 Bypass Control (0x06000 – 0x06FFF) (4 KB)

The PCI Bypass control registers occupy a 4K section of Region Zero, and all the register offsets listed below are defined from the base of this 4K Bypass section. Writes to undefined registers in this 4K area will be discarded, and reads will return the value zero.

The write data bus to this unit is 128 bits wide. The Bypass control and status registers are placed on 256-bit boundaries to allow for future increases in bus width.

The PCI Bypass Unit connects the PCI bus with the P10 memory controller. It provides a “bypass” path around the graphics core, through which software can read and write local memory directly. The memory controller reads and writes 64bytes at a time from local memory. The PCI Bypass unit has the following functions:

- Combine writes from the bus interface to reduce memory bandwidth requirements.
- Track outstanding memory writes to determine when all data has reached memory.
- Cache read data from the memory to reduce subsequent bus interface read latency.
- Optionally convert between linear and planar byte tile memory accesses based on the bypass address.

Typically the Bypass path is used to download graphics commands into local memory, after which a write is made to a Graphics Core register to program it to fetch the downloaded commands.

5.7.7.1 Cache Control Registers

ByCacheFlush

Name	Type	Offset	Format
ResetStatus	Region Zero	0x06000	Integer
	Control register		

Bits	Name	Read	Write	Reset	Description
0…31	ByCacheFlush	✓	✓	0x000 0,000	Writes flush the cache Reads return zero

Notes: The PCI Router checks the *BypassWriteComplete* signal before accessing the Graphics Core or VGA Units, and when *BypassWriteComplete* is not asserted the PCI Router automatically writes to the **ByCacheFlush** register to flush buffered write data out to the memory and invalidate the read cache. The PCI Router then waits for BypassWriteComplete to be asserted before starting the required access to the Graphics Core or VGA Unit. Reads from this register always return the value zero.

ByCacheStatus

Name	Type	Offset	Format
ByCacheStatus	Region Zero	0x06020	Bitfield
	Control register		

Bits	Name	Read	Write	Reset	Description
0	WriteBuffer Status	✓	✓	0x000 0,0000	0 = bypass write buffer empty 1 = buffer holding write data
1	ReadCache Status	✓	✓		0 = bypass read cache empty 1 = cache holding valid data
2…7	Reserved	✓	×		0=reserved
8…15	MemWrite Count	✓	✓		Count of writes which have been issued to the memory controller but have not yet completed (zero = all writes complete).
16…31	Reserved	✓	×		0=reserved

Notes: The **ByCacheStatus** register reports the status of the bypass read cache and write buffer and the number of memory writes which have been issued not yet completed. Write data is not guaranteed to have reached the memory until the bypass *writebufferstatus* is reported empty and *MemWriteCount* is zero

ByCacheControl

Name	Type	Offset	Format
ByCacheControl	Region Zero	0x06040	Bitfield

Control register

Bits	Name	Read	Write	Reset	Description
0	WriteBuffer Enable	✓	✓	0x000 00000	0 = bypass write buffer disabled 1 = bypass write buffer enabled
1	ReadCache Enable	✓	✓		0 = bypass read cache disabled 1 = bypass read cache enabled
2…31	Reserved	✓	×		0=reserved

Notes: The **ByCacheControl** register controls the behaviour of the bypass cache. Writing any value to this register will flush all currently buffered write data out to memory and mark all cached data as invalid

5.7.7.2 Region Control Registers

ByRegionFormat0
ByRegionFormat1
ByRegionFormat2
ByRegionFormat3
ByRegionFormat4
ByRegionFormat5
ByRegionFormat6
ByRegionFormat7

Name	Type	Offset	Format
ByRegionFormat0	Region Zero	0x06100	Bitfield
ByRegionFormat1		0x06180	
ByRegionFormat2		0x06200	
ByRegionFormat3		0x06280	
ByRegionFormat4		0x06300	
ByRegionFormat5		0x06380	
ByRegionFormat6		0x06400	
ByRegionFormat7		0x06480	

Control register

Bits	Name	Read	Write	Reset	Description
0	Enable	✓	✓	0	0 = Disable region 1 = Enable region
1,2	PixelSize	✓	✓	X	0 = 8 bits 1 = 16 bits 2 = 32 bits
3,4	ByteSwap	✓	✓	X	0 = ABCD 1 = BADC 2 = CDAB 3 = DCBA
5…7	Reserved	✓	×		
8…15	MultiWrite Mask	✓	✓	X	Mask of regions to repeat writes to
16…31	Reserved	✓	×		Reserved, read as 0

Notes:

ByRegionStart0
ByRegionStart1
ByRegionStart2
ByRegionStart3
ByRegionStart4
ByRegionStart5
ByRegionStart6
ByRegionStart7

Name	Type	Offset	Format
ByRegionStart0	Region Zero	0x06120	Bitfield
ByREgionStart1		0x061A0	
ByREgionStart2		0x06220	
ByRegionStart3		0x062A0	
ByRegionStart4		0x06320	
ByRegionStart5		0x063A0	
ByRegionStart6		0x06420	
ByRegionStart7		0x064A0	

Control register

Bits	Name	Read	Write	Reset	Description
0…8	Reserved	✓	×		
9…30	Address	✓	✓	0x XXX X.XX XX	Start address of region
31	Reserved	✓	×		

Notes:

ByRegionEnd0
ByRegionEnd1
ByRegionEnd2
ByRegionEnd3
ByRegionEnd4
ByRegionEnd5
ByRegionEnd6
ByRegionEnd7

Name	Type	Offset	Format
ByRegionEnd0	Region Zero	0x06140	Bitfield
ByRegionEnd1		0x061C0	
ByRegionEnd2		0x06240	
ByRegionEnd3		0x062C0	
ByRegionEnd4		0x06340	
ByRegionEnd5		0x063C0	
ByRegionEnd6		0x06440	
ByRegionEnd7		0x064C0	

Control register

Bits	Name	Read	Write	Reset	Description
0…8	Reserved	✓	×		
9…30	Address	✓	✓	0x XXX X.XX XX	End address of region
31	Reserved	✓	×		

Notes:

ByRegionWidth0
ByRegionWidth1
ByRegionWidth2
ByRegionWidth3
ByRegionWidth4
ByRegionWidth5
ByRegionWidth6
ByRegionWidth7

Name	Type	Offset	Format
ByRegionWidth0	Region Zero	0x06160	Bitfield
ByRegionWidth1		0x061E0	
ByRegionWidth2		0x06260	
ByRegionWidth3		0x062E0	
ByRegionWidth4		0x06360	
ByRegionWidth5		0x063E0	
ByRegionWidth6		0x06460	
ByRegionWidth7		0x064E0	

Control register

Bits	Name	Read	Write	Reset	Description
0…8	Reserved	✓	×		
9…30	Address	✓	✓	0x XXX X.XX XX	End address of region
31	Reserved	✓	×		

Notes:

5.7.8 Video Port Control 0x07000 – 0x07FFF (4K)

The Video Port implements a VESA Video Interface Port (VIP) master. The Video Port supports:

- ITU-R BT.656 video stream – 8-bit @ 27MHz
- VIP1.1 port – 8-bit @ 27Mhz
- VIP2 Level I port – 8-bit @ 75MHz

The Video Port does not support:

- VIP1.1 or VIP2 host port
- VIP2 Level II or Level III video port

For further information see the Video chapter in the *Reference Guide,* Volume I.

5.7.8.1 Register Interface

The 4-Kbyte region defines 32-bit registers on 64-bit boundaries as follows.

Enable

Name	Type	Offset	Format
Enable	Region Zero	0x07000	Bitfield

Control register

Bits	Name	Read	Write	Reset	Description
0	Enable	✓	✓	0	When 0, VPUIClk is forced into reset. When 1, VPUIClk is taken out of reset
1…31	Reserved	✓	×		0=reserved

Notes: Drives the corresponding PClk → IClk signal. Resynchronized from PClk → Iclk.

Mode

Name	Type	Offset	Format
Mode	Region Zero	0x07008	Bitfield

Control register

Bits	Name	Read	Write	Reset	Description
0…2	Reserved	✓	×	0xXX XX.X XXX	Set to 0
3	SkipData	✓	✓		0 = Empty cycles during active video are processed. 1 = Empty cycles during active video are discarded.
4	HBStore	✓	✓		0 = Horizontal blanking data is discarded. 1 = Horizontal blanking data is stored.
5	VBStore	✓	✓		0 = Vertical blanking data is discarded. 1 = Vertical blanking data is stored.
6	Interlaced	✓	✓		0 = Store video source as non-interlaced frames. The video source can be non-interlaced or interlaced. 1 = Store video source as interlaced frames. The video source must be interlaced.
7	StartField	✓	✓		In interlaced video, this is matched against the EAV Field (F) bit to determine the 1st field in the frame.
8	MaxIdx	✓	✓		Maximum index 0 = 1 buffer 1 = 2 buffers 2 = 3 buffers 3 = Reserved (3 buffers)
10…31	Reserved	✓	×		

Notes: Not resynchronised from PClk → Iclk.

SAVPos

Name	Type	Offset	Format
SAVPos	Region Zero	0x07010	Bitfield
	Control register		

Bits	Name	Read	Write	Reset	Description
0…11	Pos	✓	✓	0xXX XX.X XXX	SAV position, counted from 0 at the start of the horizontal blanking interval
12…31	Reserved	✓	×		

Notes: Not resynchronized from PClk → Iclk.

EAVPos

Name	Type	Offset	Format
SAVPos	Region Zero	0x07018	Bitfield
	Control register		

Bits	Name	Read	Write	Reset	Description
0…11	Pos	✓	✓	0xXX XX.X XXX	EAV position, counted from 0 at the start of the horizontal blanking interval
12…31	Reserved	✓	×		

Notes: Not resynchronized from PClk → Iclk.

BufAddr[0..1][0..2]

Name	Type	Offset	Format
BufAddr[0..1][0..2]	Region Zero	0x07020, 0x0728 0x07030, 0x0738 0x07040, 0x0748	Bitfield

Control register

Bits	Name	Read	Write	Reset	Description
0…24	BufAddr	✓	✓	0xXX XX.X XXX	Buffer address aligned to a 4-tile boundary
25…31	Reserved	✓	×		

Notes: Not resynchronized from PClk → Iclk.

RdIdx

Name	Type	Offset	Format
RdIdx	Region Zero	0x07050	Bitfield

Control register

Bits	Name	Read	Write	Reset	Description
0,1	RdIdx0	✓	✓	0xXX XX.X XXX	Read index (task 0).
2,3	RdIdx1	✓	✓		Read index (task 1).
4…31	Reserved	✓	×		0=reserved

Notes: Resynchronized from PClk → Iclk.

WrIdx

Name	Type	Offset	Format
WrIdx	Region Zero	0x07058	Bitfield

Status register

Bits	Name	Read	Write	Reset	Description
0,1	RdIdx0	✓	×	0xXX XX.X XXX	Read index (task 0).
2,3	RdIdx1	✓	×		Read index (task 1).
4…31	Reserved	✓	×		0=reserved

Notes: Resynchronized from PClk → Iclk. Reset value = undefined at chip reset, 0 at VPUIClk reset

5.7.9 Video Head 1 Control (0x08000 – 0x08FFF) (4KB)

The Video Control registers are described above in *Video Head 0 Control* and in the *P10 Programmer's Guide*. Each of the two video heads in the current P10 implementation has its own 4K Byte control register space within Region Zero. Unless explicitly noted, each register in this list is repeated for each head in the system. Any reserved fields in a register should read back as zero.

5.7.10 Reserved (0x09000 – 0x0EFFF) (24KB)

5.7.11 GPIO Driver (0x0F000 – 0x0FFFF)

The 128-Kbyte address region is sub-decoded into a 4-Kbyte driver region and a 64-Kbyte user region. The 4-Kbyte driver region defines 32-bit registers on 64-bit boundaries.

ImsgReady[Drv,Iso]

Name	Type	Offset	Format
ImsgReady[Drv,Iso]	Region Zero	0x0F000 0x0F030	Bitfield

Control register

Bits	Name	Read	Write	Reset	Description
0	Ready	✓	×	0x1	0 = the message assembly is not ready to be written. 1 = the message assembly is ready to be written. Reset value = 1. Read-write access.
1…31	Reserved	✓	×		Reserved.

Notes:

ImsgTag[Drv,Iso]

Name	Type	Offset	Format
ImsgTag[Drv,Iso]	Region Zero	0x0F008 0x0F038	Bitfield

Control register

Bits	Name	Read	Write	Reset	Description
0…9	Tag	✓	✓		Message tag. Reset value = undefined. Read-write access.
10,11	-	✓	×		Reserved for future tag expansion.
12,13	Size	✓	✓		Message size, in 32-bit data words – 1. Reset value = undefined. Read-write access.
14…31	-	✓	×		Reserved.

Notes:

ImsgData[0…3][Drv]

Name	Type	Offset	Format
ImsgData[0…3][Drv]	Region Zero	0x0F010 0x0F018 0x0F020 0x0F028	Data

Control register

Bits	Name	Read	Write	Reset	Description
0…31	Data	✓	✓		Message data. Reset value = undefined. Read-write access.

Notes:

ImsgData[0…3][Iso]

Name	Type	Offset	Format
ImsgData[0…3][Iso]	Region Zero	0x0F040 0x0F048 0x0F050 0x0F058	Data

Control register

Bits	Name	Read	Write	Reset	Description
0…31	Data	✓	✓		Message data. Reset value = undefined. Read-write access.

Notes:

ScheduleUsr

Name	Type	Offset	Format
ScheduleUsr	Region Zero	0x0F060	Bitfield

Control register

Bits	Name	Read	Write	Reset	Description
0	Enable	✓	✓	0	0 = the user context scheduler is disabled and the driver input message port and circular buffer enabled. 1 = the user context scheduler is enabled and the driver input message port and circular buffer disabled.
1	Reserved	✓	×		Reserved.
2…31	Timeout	✓	✓	***	Time slice available to user contexts before pre-emption, in clocks. At 200 MHz, 230 clocks equal about 5 seconds.

Notes:

MagicWrPtrUsr

Name	Type	Offset	Format
MagicWrPtrUsr	Region Zero	0x0F068	Bitfield

Control register

Bits	Name	Read	Write	Reset	Description
0…22	Reserved	✓	×		Reserved
22…30	Magic	✓	✓	***	Magic number required in the corresponding bits of the **CBufWrPtrUsr**
31	Reserved	✓	×		Reserved

Notes:

SuspendUsr

Name	Type	Offset	Format
SuspendUsr	Region Zero	0x0F070	Bitfield

Control register

Bits	Name	Read	Write	Reset	Description
0…22	Reserved	✓	×		Reserved
22…30	Magic	✓	✓	***	Magic number required in the corresponding bits of the **CBufWrPtrUsr**
31	Reserved	✓	×		Reserved

Notes:

CBufEnableBusyUsr[0..15]

Name	Type	Offset	Format
CBufEnableBusyUsr[0..15]	Region Zero	0x0F080	Bitfield

Control register

Bits	Name	Read	Write	Reset	Description
0	Enable	✓	✓	0	When 0, the circular buffer is disabled. When 1, the circular buffer is enabled.
1	Busy	✓	×	***	When 0, the circular buffer is idle, i.e. (RdPtr == WrPtr). When 1, the circular buffer is busy, i.e. (RdPtr != WrPtr). Read-only access, but writing the register clears the read and write pointers, so clearing this bit.
2	RqBusy	✓	×	***	When 0, the DMA engine is idle, i.e. (RqPtr == WrPtr). When 1, the DMA engine is busy, i.e. (RqPtr != WrPtr). Read-only access, but writing the register clears the request and write pointers, so clearing this bit.
3	CtxtEnable	✓	✓	***	When 0, the context buffer is disabled. When 1, the context buffer is enabled. Reset value = undefined.
4…31	CtxtAddr	✓	✓	***	Context buffer address, in 64-byte tiles.

Notes: This accesses one register in an array of 16, indexed by address bits 11:8.
This is locked while the circular buffer is busy.

CBufAddrUsr[0..15]

Name	Type	Offset	Format
CBufAddrUsr[0..15]	Region Zero	0x0F088	Bitfield

Control register

Bits	Name	Read	Write	Reset	Description
0,1	ByteSwap	✓	×		Byte swap: 0 = ABCD (no swap) 1 = BADC 2 = CDAB 3 = DCBA
2…8	-	✓	×		Reserved.
9	VideoMemHint [p9]	✓	✓		After address translation, the circular buffer address will be in video memory, so break up the request in a more optimal way for the memory controller.
	Reserved [p10]	✓	×		
10…29	Addr	✓	✓	***	Circular buffer address, in 4-Kbyte pages, undefined reset value
30,31	Reserved [p9]	✓	×		Reserved
	Addr [p10]	✓	✓	***	Continues address in bits 10…29.

Notes: This accesses one register in an array of 16, indexed by address bits 11:8.
This is locked while the circular buffer is busy

CBufEnableBusyIso

Name	Type	Offset	Format
CBufEnableBusyIso	Region Zero	0x0F090	Bitfield

Control register

Bits	Name	Read	Write	Reset	Description
0	Enable	✓	✓	0	When 0, the circular buffer is disabled and the input message port enabled. When 1, the circular buffer is enabled and the input message port disabled. Read-write access.
1	Busy	✓	×	***	When 0, the circular buffer is idle, i.e. (RdPtr == WrPtr). When 1, the circular buffer is busy, i.e. (RdPtr != WrPtr). Read-only access, but writing the register clears the read and write pointers, so clearing this bit.
2	RqBusy	✓	×		When 0, the DMA engine is idle, i.e. (RqPtr == WrPtr). When 1, the DMA engine is busy, i.e. (RqPtr != WrPtr). Read-only access, but writing the register clears the request and write pointers, so clearing this bit.
3…31	Reserved	✓	×		Reserved.

Notes: This is locked while the circular buffer is busy.

CBufAddrIso

Name	Type	Offset	Format
CBufAddrIso	Region Zero	0x0F098	Bitfield

Control register

Bits	Name	Read	Write	Reset	Description
0,1	ByteSwap	✓	×		Byte swap: 0 = ABCD (no swap) 1 = BADC 2 = CDAB 3 = DCBA
2,9	Reserved	✓	×		Reserved for future expansion.
10…31	Addr	✓	×	***	

Notes: This is locked while the circular buffer is busy.

CBufWrPtrIso

Name	Type	Offset	Format
CBufWrPtrIso	Region Zero	0x0F0A0	Bitfield

Control register

Bits	Name	Read	Write	Reset	Description
0…10	Reserved	✓	✓	***	Circular buffer write-pointer, as an offset in words from the start.
20,21	Reserved	✓	×		Reserved for future expansion.
22…31	Reserved	✓	×		Reserved

Notes: The write pointer is initialised to 0 when the circular buffer is enabled

CBufRdPtrIso

Name	Type	Offset	Format
CBufRdPtrIso	Region Zero	0x0F0a8	Bitfield

Control register

Bits	Name	Read	Write	Reset	Description
0…22	Reserved	✓	×	***	Circular buffer read-pointer, as an offset in words from the start.
20,21	Reserved	✓	×		Reserved for future expansion.
22…31	Reserved	✓	×		Reserved

Notes: The read pointer is initialised to 0 when the circular buffer is enabled

CommandIdIso

Name	Type	Offset	Format
CommandIdIso	Region Zero	0x0F0B0	Bitfield

Control register

Bits	Name	Read	Write	Reset	Description
0…29	CommandID	✓	×	***	Value of the most recently processed CommandId message.
30,31	Reserved	✓	×		Reserved

Notes:

SyncIdIso

Name	Type	Offset	Format
SyncIdIso	Region Zero	0x0F0B8	Bitfield

Control register

Bits	Name	Read	Write	Reset	Description
0…29	Sync	✓	×	***	Value of the most recently processed Sync command
30,31	Reserved	✓	×		

Notes:

OMsgReady

Name	Type	Offset	Format
OMsgReady	Region Zero	0x0F0C0	Bitfield

Control register

Bits	Name	Read	Write	Reset	Description
0	Ready	✓	×	0	When 0, the output message is not ready to be read. When 1, the output message is ready to be read
1…31	Reserved	✓	×		

Notes:

OMsgTag

Name	Type	Offset	Format
OMsgTag	Region Zero	0x0F0C8	Bitfield

Control register

Bits	Name	Read	Write	Reset	Description
0…9	Tag	✓	×	***	Message Tag
10,11	Reserved	✓	×		Reserved for future tag expansion
12…31	Reserved	✓	×		Reserved

Notes:

OMsgData[0..3]

Name	Type	Offset	Format
OMsgData[0..3]	Region Zero	0x0F0D0 0x0F0D8 0x0F0E0 0x0F0E8	Data

Control register

Bits	Name	Read	Write	Reset	Description
0…31	Data	✓	×	***	Message data

Notes:

MagicWrPtrUsr

Name	Type	Offset	Format
MagicWrPtrUsr	Region Zero	0x0F068	Bitfield

Control register

Bits	Name	Read	Write	Reset	Description
0…22	Reserved	✓	×		Reserved
22…30	Magic	✓	✓	***	Magic number required in the corresponding bits of the **CBufWrPtrUsr**
31	Reserved	✓	×		Reserved

Notes:

CommIntrMask

Name	Type	Offset	Format
CommIntrMask	Region Zero	0x0F0F0	Bitfield

Control register

Bits	Name	Read	Write	Reset	Description
0…15	Mask	✓	✓	***	Pending interrupt mask. A per-context bit is set when a Command or Sync interrupt is asserted. Bits are cleared by writing a 1 to them.
16…31	Reserved	✓	×		

Notes:

SyncMask

Name	Type	Offset	Format
SyncIntrMask	Region Zero	0x0F0F8	Bitfield

Control register

Bits	Name	Read	Write	Reset	Description
0…15	Mask	✓	✓	***	Pending interrupt mask. A per-context bit is set when a Command or Sync interrupt is asserted. Bits are cleared by writing a 1 to them.
16…31	Reserved	✓	×		

Notes:

5.7.12 GPIO User (0x10000 – 0x1FFFF)

CbufWrPtrUsr[0-15]

Name	Type	Offset	Format
CbufWrPtrUsr[0]	Region Zero	0x10010	
CbufWrPtrUsr[1]		0x11010	
CbufWrPtrUsr[2]		0x12010	
CbufWrPtrUsr[3]		0x13010	
CbufWrPtrUsr[4]		0x14010	
CbufWrPtrUsr[5]		0x15010	
CbufWrPtrUsr[6]		0x16010	
CbufWrPtrUsr[7]		0x17010	
CbufWrPtrUsr[8]		0x18010	
CbufWrPtrUsr[9]		0x19010	
CbufWrPtrUsr[10]		0x1a010	
CbufWrPtrUsr[11]		0x1b010	
CbufWrPtrUsr[12]		0x1c010	
CbufWrPtrUsr[13]		0x1d010	
CbufWrPtrUsr[14]		0x1e010	
CbufWrPtrUsr[15]		0x1f010	

Status register

Bits	Name	Read	Write	Reset	Description
0…19	WrPtr	✓	×	0xXX XXX XXX	Circular buffer write-pointer, as an offset in words from the start. Reset value = undefined. Read-write access when the magic bits match, read-only otherwise.
20,21	-	✓	×		Reserved for future expansion of preceding field.
22…30	Magic	✓	×		When written, these bits must match the corresponding bits set via the **MagicWrPtrUsr**. When read, these bits return 0.
31	Yield	✓	×		When written with 0, the context finishes its timeslice (default behaviour). When written with 1, the context yields its timeslice (effectively forcing it to 0) at the next opportunity. When read, this bit returns 0.

Notes:	•	The write-pointer is initialised to 0 when the circular buffer is enabled.
	•	This accesses one register in an array of 16, indexed by address bits 15:12.

CbufRdPtrUsr[0-15]

Name	Type	Offset	Format
CbufRdPtrUsr[0]	Region Zero	0x10018	
CbufRdPtrUsr[1]		0x11018	
CbufRdPtrUsr[2]		0x12018	
CbufRdPtrUsr[3]		0x13018	
CbufRdPtrUsr[4]		0x14018	
CbufRdPtrUsr[5]		0x15018	
CbufRdPtrUsr[6]		0x16018	
CbufRdPtrUsr[7]		0x17018	
CbufRdPtrUsr[8]		0x18018	
CbufRdPtrUsr[9]		0x19018	
CbufRdPtrUsr[10]		0x1a018	
CbufRdPtrUsr[11]		0x1b018	
CbufRdPtrUsr[12]		0x1c018	
CbufRdPtrUsr[13]		0x1d018	
CbufRdPtrUsr[14]		0x1e018	
CbufRdPtrUsr[15]		0x1f018	

Status register

Bits	Name	Read	Write	Reset	Description
0…19	RdPtr	✓	×	0xXX XX.X XXX	Circular buffer read-pointer, as an offset in words from the start. Reset value = undefined. Read-only access.
20,21	Reserved	✓	×		Reserved for future expansion of preceding field.
22…31	Reserved	✓	×		Reserved.

Notes:

CommandIDUsr[0-15]

Name	Type	Offset	Format
CommandIDUsr[0]	Region Zero	0x10020	
CommandIDUsr[1]		0x11020	
CommandIDUsr[2]		0x12020	
CommandIDUsr[3]		0x13020	
CommandIDUsr[4]		0x14020	
CommandIDUsr[5]		0x15020	
CommandIDUsr[6]		0x16020	
CommandIDUsr[7]		0x17020	
CommandIDUsr[8]		0x18020	
CommandIDUsr[9]		0x19020	
CommandIDUsr[10]		0x1a020	
CommandIDUsr[11]		0x1b020	
CommandIDUsr[12]		0x1c020	
CommandIDUsr[13]		0x1d020	
CommandIDUsr[14]		0x1e020	
CommandIDUsr[15]		0x1f020	

Status register

Bits	Name	Read	Write	Reset	Description
0…29	CommandId	✓	×		Value of the most recently processed CommandId message. Reset value = undefined. Read-only access.
30,31	Reserved	×	×		Reserved.

Notes: This accesses one register in an array of 16, indexed by address bits 15:12

SyncIDUsr[0-15]

Name	Type	Offset	Format
SyncIDUsr[0]	Region Zero	0x10028	
SyncIDUsr[1]		0x11028	
SyncIDUsr[2]		0x12028	
SyncIDUsr[3]		0x13028	
SyncIDUsr[4]		0x14028	
SyncIDUsr[5]		0x15028	
SyncIDUsr[6]		0x16028	
SyncIDUsr[7]		0x17028	
SyncIDUsr[8]		0x18028	
SyncIDUsr[9]		0x19028	
SyncIDUsr[10]		0x1a028	
SyncIDUsr[11]		0x1b028	
SyncIDUsr[12]		0x1c028	
SyncIDUsr[13]		0x1d028	
SyncIDUsr[14]		0x1e028	
SyncIDUsr[15]		0x1f028	

Status register

Bits	Name	Read	Write	Reset	Description
0…29	SyncId	✓	×		Value of the most recently processed SyncId message. Reset value = undefined. Read-only access.
30,31	Reserved	×	×		Reserved.

Notes: This accesses one register in an array of 16, indexed by address bits 15:12

5.8 Memory Apertures 1 & 2

Access to memory apertures is controlled by the ApertureOne and ApertureTwo registers in Region Zero.

5.9 Expansion ROM

A region is provided for a standard 64 KByte PCI Expansion ROM. Code will never be executed directly from this ROM but will always be loaded into host memory before execution.

CFGBusConfig

Name	Type	Offset	Format
CFGBusConfig	ROM	0xF0	Bitfield
	Control register		

Bits	Name	Read	Write	Reset	Description
0	BaseClassZero	✓	✓	See note	0 = use the correct PCI BaseClass Code 1 = force the PCI BaseClase Code to zero (backward compatible)
1	VgaEnable	✓	✓	See note	0 = disable internal VGA subsystem 1 = enable internal VGA subsystem

2	VgaFixed	✓	✓	See note	0 = disable VGA fixed address decoding 1 = enable VGA fixed address decoding (Fixed VGA address decoding is only active when both the **VgaFixed** and **VgaEnable** fields are set.) Bit 1
3	VgaNoAlias	✓	✓	See note	0 = decode only 10 bits of VGA I/O addresses 1 = decode all 32-bits of VGA I/O addresses Bit 2
4	SubClass3D	✓	✓	See note	0 = set Display SubClass Code to "other display" when VGA disabled 1 = set Display SubClass Code to "3D controller" when VGA disabled
5	RetryDisable	✓	✓	See note	0 = enable PCI Retry using "Disconnect-Without-Data" 1 = disable PCI Retry using "Disconnect-Without-Data"
6	DelayRd Disable	✓	✓	See note	0 = enable Delayed PCI Read transactions 1 = disable Delayed PCI Read transactions
7	DelayWr Disable	✓	✓	See note	0 = enable Delayed PCI Write transactions 1 = disable Delayed PCI Write transactions
8	Rate1X Capable	✓	✓	See note	0 = device does not support 1X data transfer rate 1 = device supports 1X data transfers (AGP only)
9	Rate2X Capable	✓	✓	See note	0 = device does not support 2X data transfer rate 1 = device supports 2X data transfers (AGP or FW)
10	Rate4X Capable	✓	✓	See note	0 = device does not support 4X data transfer rate 1 = device supports 4X data transfers (AGP or FW)
11	PciFWCapable	✓	✓	See note	0 = device does not support PCI Fast Write transactions 1 = device can accept PCI Fast Write (FW) transactions
12	Agp4G Capable	✓	✓	See note	0 = the AGP bus master can only generate 32-bit addresses 1 = the AGP master supports addresses above 4GB boundary
13	SbaCapable	✓	✓	See note	0 = device not capable of AGP sideband addressing 1 = device can generate AGP sideband addressing
14	AgpRdEnable	✓	✓	See note	0 = disable AGP read master operation 1 = enable AGP read master operation
15	AgpWrEnable	✓	✓	See note	0 = disable AGP write master operation 1 = enable AGP write master operation
16	AutoCal Enable	✓	✓	See note	0 = disable AGP output driver auto-calibration 1 = enable AGP output driver auto-calibration
17	ShortReset	✓	✓	See note	0 = generate normal reset pulse to rest of chip (functional mode) 1 = generate short reset pulse to rest of chip (fast simulation only)
18	AgpAutoReset	✓	✓	See note	0 = rely on software not to trigger a SoftReset until AGP Master idle 1 = automatically terminate outstanding AGP requests on SoftReset
19	ByAutoFlush	✓	✓	See note	0 = rely on driver software to ensure Bypass/Core synchronisation 1 = automatically flush the Bypass when switching to Core accesses

20	FWDisc WithData	✓	✓	See note	0 = use "Disconnect-Without-Data" in PCI Fast Write mode 1 = use "Disconnect-With-Data" for PCI Fast Write transfers
21	PciPrefetch Enable	✓	✓	See note	0 = disable internal prefetch of slave read data for Regions 1 and 2 1 = enable internal prefetch of slave read data for Regions 1 and 2
22	Pci Prefetchable	✓	×	See note	This controls the Base Address Register "Prefetchable" bits and cannot be altered once it has been loaded from ROM. 0 = Base Address Regions 1 and 2 are not marked prefetchable 1 = Base Address Regions 1 and 2 are both marked prefetchable
23	PciAddress64	✓	×	See note	0 = the PCI slave is located anywhere in 32-bit address space 1 = the PCI slave is located anywhere in 64-bit address space [The PCI master can always use DAC to access 64-bit space]
24…27	Base1AddrSize	✓	×	See note	Controls the size of Region 1, cannot be altered once it has been loaded from ROM See *Base2AddrSize* for values
28…31	Base2AddrSize	✓	×	See note	Controls the size of Region 2, cannot be altered once it has been loaded from ROM. 0x0 = not enabled 0x1 = 128 KB 0x2 = 256 KB 0x3 = 512 KB 0x4 = 1 MB 0x5 = 2 MB 0x6 = 4 MB 0x7 = 8 MB 0x8 = 16 MB 0x9 = 32 MB 0xA = 64 MB 0xB = 128 MB 0xC = 256 MB 0xD = 512 MB 0xE = 1 GB 0xF = 2 GB

Notes:
- The **BusConfig** register is normally loaded from the external expansion ROM and controls the configuration of the bus interface. Most of the fields in this register can be updated by PCI Configuration Space writes. This allows boot-time software to change the default power-up values, where this makes sense. The fields controlling Base Address region sizes and Base Address register widths are read-only, since it is not sensible to update these from software after power-up.
- The field names of this register are used throughout this document to indicate how other registers are initialised and configured. For example, the *BaseClassZero* field is used to control the contents of the **CFGClassCode** register. The abbreviation ***AgpCapable*** is used to indicate the logical OR of ***the Rate1XCapable, Rate2XCapable***, and ***Rate4XCapable*** fields, and controls those PCI Fast Write and AGP capabilities which are independent of the data transfer rate.

Note: Before changing the AGPRate the AGP Master should be disabled (reqEnable = Off) or the result may be an undefined state.

The Reset value default is 0xBB180000 but can also be loaded from ROM

CFGFunConfig

Name	Type	Offset	Format
CFGFunConfig	Config *Control register*	0xE4	Bitfield

Bits	Name	Read	Write	Reset	Description
0	MaxFunction	✓	×	0	0 = only function zero is enabled (single function device) 1 = functions zero and one enabled (multi function device)
1, 2	Reserved	✓	×	0	0 = reserved (for extending the MaxFunction field)
3	MultiUniq DevId	✓	×	0	0 = all functions have the same Device ID 1 = each function has a unique Device ID
4	MultiShare IntLine	✓	×	0	0 = every function has a separate Interrupt Line register 1 = all functions share a common Interrupt Line register
5	MultiBar1 Enable	✓	×	0	0 = disable the Region One BAR for functions greater than zero 1 = all functions have the same size Region One as function zero
6	MultiBar2 Enable	✓	×	0	0 = disable the Region Two BAR for functions greater than zero 1 = all functions have the same size Region Two as function zero
7	MultiAgp Capable	✓	×	0	0 = functions greater than zero do not implement AGP registers 1 = all functions have AGP registers when **AGPCapable** is one Unimplemented AGP registers are always Zero and Read Only.

8	MultiShare AgpCmd	✓	×	0	0 = every function has a separate AGP Command register 1 = all functions share a common AGP Command register This field has no effect when **MultiAgpCapable** is zero, in which case the AGP registers for functions greater than zero will be Zero and Read Only.
9…28	Reserved	✓	×	0	
29, 30	RomAddrSize	✓	×	0	This controls the size of the Expansion ROM Region. 0x0 = 64 KB 0x1 = 128 KB 0x2 = 256 KB 0x3 = 512 KB
31	AltDeviceId	✓	×	0	0 = use "standard" Device IDs (starting from 0020h) 1 = use "alternate" Device IDs (starting from 0022h)

Notes: The field names of this register are used throughout this document to indicate how other registers are initialised and configured. For example, the *MaxFunction* field is used to control the contents of the **CFGHeaderType** configuration register for each function.
The **FunConfig** register is normally loaded from the external expansion ROM and controls how the bus interface is configured for multi-function operation. This entire register is read-only, since it is not sensible to update the number of functions or **BaseAddress** Registers from software after power-up.
The Reset value default is 0x00000000 but can be loaded from ROM

CFGDevConfig

Name	Type	Offset	Format
CFGDevConfig	Config *Control register*	0xEC	Bitfield

Bits	Name	Read	Write	Reset	Description
0…31	DevConfig	✓	✓	0	Generic Device configuration

Notes: The **DevConfig** register is normally loaded from the external ROM (see *Reference Guide* volume IV, Chapter 10, *Reset*) and contains configuration information for the rest of the device – it does not affect the behaviour of the bus interface. The output of this register is presented to the rest of the chip as a 32-bit configuration data bus, together with a "valid" flag indicating when the register been loaded. When **CFGDevConfig** is not loaded from the ROM, the "valid" flag is not asserted to the rest of the chip until the register has been loaded from software. This register can be read and written using PCI Configuration Space accesses, with the **CFGDevConfigMask** register providing a 32-bit write mask.

The Reset value default is 0x00000000 but can be loaded from ROM

CFGDevConfigMask

Name	Type	Offset	Format
CFGDevConfigMask	Config *Control register*	0xE8	Bitfield

Bits	Name	Read	Write	Reset	Description
0…31	DevConfig	✓	✓	0	Generic Device configuration

Notes: The **DevConfigMask** register is normally loaded from the external ROM (see *Reference Guide* volume IV, Chapter 10, *Reset*) and. contains the write mask for the **DevConfig** register. When a bit is set in the write mask, the corresponding bit in the **DevConfig** register is writeable. This configurable write mask reduces the need for device-specific masking to be included in the bus interface, while still providing write-protection for fields which should not be dynamically updated by software after power-up.

The Reset value default is 0xFFFFFFFF but can be loaded from ROM

5.10 VGA Registers (0xA0000 - 0xBFFFF)

The bus interface can be configured to respond to the standard VGA-compatible Memory and I/O Space addresses using the *VgaFixed* bit in the **CFGBusConfig** register[4], provided the internal VGA controller has been enabled using the *VgaEnable* bit in the same register. The bus interface will then respond to Memory Space addresses A0000h through BFFFFh, and also I/O Space addresses within the ranges 3B0h to 3BBh and 3C0h to 3DFh (and aliases of these I/O addresses when appropriately configured). These are all fixed addresses, unaffected by the base address registers in PCI Configuration Space.

It is also possible to access the VGA control registers through the relocatable Region Zero in PCI memory space, and the VGA memory through either of the relocatable local memory apertures (Regions One and Two).

5.10.1 Fixed address decoding

All 32 bits of a PCI Memory Space address are decoded to determine if a fixed memory address access is being made to the VGA Unit. The fixed memory VGA address range is divided into four 32 KByte sections, each of which has its own independent range decode enable from the VGA Unit. These enable signals are used to select which of the possible memory address sub-ranges are currently active – as far as the bus interface is concerned any combination of these sub-ranges can be active, depending on the mode of the VGA Core Unit:

VGA Unit Address Range Enable Signals	
VHDL Signal Name	**Memory Address Range Enabled**
PciVgaMemA0Decode	1 = decode addresses `0xA0000` – `0xA7FFF`
PciVgaMemA8Decode	1 = decode addresses `0xA8000` – `0xAFFFF`
PciVgaMemB0Decode	1 = decode addresses `0xB0000` – `0xB7FFF`
PciVgaMemB8Decode	1 = decode addresses `0xB8000` – `0xBFFFF`

5.10.2 Memory Aperture Accesses

When the *VgaAccess* bit is set in either of the **ApertureOne** or **ApertureTwo** registers[5], then all accesses to the relevant Region One or Region Two aperture will be forwarded to the VGA Unit rather than directly to the memory controller. Where the memory aperture is configured to be larger than the 128 KByte VGA memory size, then VGA memory space will be aliased within the total aperture address size.

[4] See the *PCI Config Unit Specification* for details of the CFGBusConfig register.

[5] See the *PCI CSR Unit Specification* for details of the ApertureOne and ApertureTwo registers.

VGA accesses using these relocatable memory apertures are not affected by the enables from the VGA Unit, but will always be forwarded to the VGA Unit provided the **VgaEnable** bit is set in the CFGBusConfig register.

5.10.3 Fixed I/O Addresses

The number of I/O address bits that are decoded by the bus interface depends on the *VgaNoAlias* configuration bit in the **CFGBusConfig** register. When *VgaNoAlias* is set, all 32 bits of I/O address are decoded. When *VgaNoAlias* is not set then only the bottom 10 bits of the address are decoded, and the bus interface responds to all I/O address aliases.

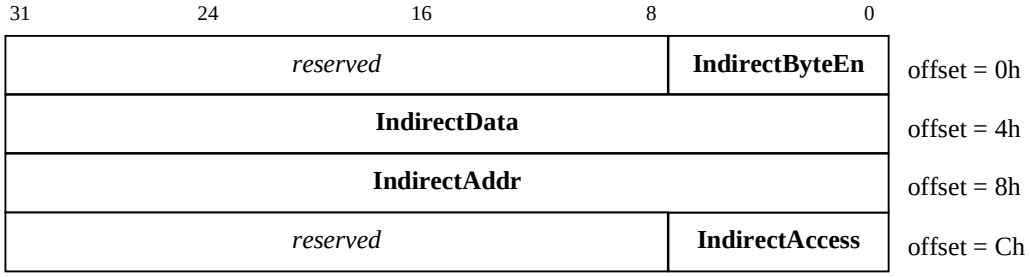
The exact I/O Space addresses which the bus interface should respond to is a function of the configuration and mode of the VGA Core Unit. Enable signals from the VGA Unit are used to select which of the possible I/O address sub-ranges are currently active. For example, in monochrome and color VGA modes a different subset of I/O ports must be enabled.

VGA Unit Address Range Enable Signals			
VHDL Signal Name	**Signal Value**	**I/O Addresses Decoded**	**Mode**
PciVgaIOColorDecode	0	0x3B4, 0x3B5, 0x3BA	Mono
		0x3C0 – 0x3C2, 0x3C4 – 0x3CA 0x3CC, 0x3CE, 0x3CF	
PciVgaIOColorDecode	1	0x3D4, 0x3D5, 0x3DA	Color
		0x3C0 – 0x3C2, 0x3C4 – 0x3CA 0x3CC, 0x3CE, 0x3CF	

5.10.4 Indirect VGA I/O Registers

The bus interface slave address decoder includes four registers which provide a mechanism to perform indirect accesses through VGA I/O Space. These registers occupy a special sixteen byte "VGA Indirect" region, which itself can only be accessed indirectly using a series of VGA byte (or word) I/O transactions. This is similar to the set of registers which are used to access any of Regions Zero, One, Two, and the ROM region indirectly through PCI Configuration Space.

The four VGA registers and their use are described first, followed by details of how to read and write these registers through VGA I/O Space together with examples.



The **IndirectAddr** register has the same format as the **CFGIndirectAddress** register in configuration space, specifying a region in its top three bits, and an offset within that region in the remaining bits. To make an indirect access, the **IndirectAddr** register is first loaded with the destination and the offset within that region.

For an indirect write, 32 bits of data are written into **IndirectData** and four byte enables into **IndirectByteEn**. A single byte write to the **IndirectAccess** register triggers the write. For an indirect read, a single byte read of the **IndirectAccess** register causes 32 bits of data to be read from inside the device, and loaded into the **IndirectData** register. Subsequent reads of the **IndirectData** register can be used to obtain the data without causing side effects to the rest of the device (allowing 32-bit indirect data to be read back using byte I/O transfers). Note that reads or writes to the IndirectAccess register must always be single-byte transfers.

5.10.5 Reading from a Region Zero register

The following examples show how to use the VGA Indirect registers to read and write internal registers. The I/O ports shown are the monochrome ports at 0x3B0 and 0x3B1. If the VGA Unit is in colour mode then the ports at 0x3D0 and 0x3D1 should be used instead. Byte accesses are shown in these examples, but word accesses to combine the offset and data would be valid (and potentially more efficient) except when accessing the IndirectAccess register.

The following code shows how to read the contents of the register at offset 0x1234 in Region Zero.

// write the indirect address 0x1234 to the internal **IndirectAddr** register

```
Addr = (0 << 29) | 0x1234              // (0 << 29) selects Region 0
port(0x3B0) ← 0x08                     // select byte 0 of IndirectAddr
port(0x3B1) ← ((Addr >>  0) & 0xFF)    // load byte 0 of address & region
port(0x3B0) ← 0x09                     // select byte 1 of IndirectAddr
port(0x3B1) ← ((Addr >>  8) & 0xFF)    // load byte 1 of address & region
port(0x3B0) ← 0x0A                     // select byte 2 of IndirectAddr
port(0x3B1) ← ((Addr >> 16) & 0xFF)    // load byte 2 of address & region
port(0x3B0) ← 0x0B                     // select byte 3 of IndirectAddr
```

```
port(0x3B1) ← ((Addr >> 24) & 0xFF)      // load byte 3 of address & region

// set the byte enables to read all four data bytes

port(0x3B0) ← 0x00                       // select IndirectByteEn
port(0x3B1) ← 0x0F                       // set all four byte enables

// trigger the read internally using byte read of IndirectAccess register

port(0x3B0) ← 0x0C                       // select IndirectAccess register
tempvar ← port(0x3B1)            // throw away the return value

// IndirectData now contains the data so read it back one byte at a time.

port(0x3B0) ← 0x04                       // select byte 0 of IndirectData
Data0 ← port(0x3B1)                      // read byte 0 of final data dword
port(0x3B0) ← 0x05                       // select byte 1 of IndirectData
Data1 ← port(0x3B1)                      / read byte 1 of final data dword
port(0x3B0) ← 0x06                       // select byte 2 of IndirectData
Data2 ← port(0x3B1)                      // read byte 2 of final data dword
port(0x3B0) ← 0x07                       // select byte 3 of IndirectData
Data3 ← port(0x3B1)              // read byte 4 of final data dword

FinalData = (Data3 << 24) | (Data2 << 16) | (Data1 << 8) | Data0
```

5.10.6 Writing to a Region Two Register

The following code shows how to write the value "`Data`" to offset `0x1234` in Region Two.

```
// write the indirect address 0x1234 to the internal IndirectAddr register

Addr = (2 << 29) | 0x1234        // (2 << 29) selects Region 2
port(0x3B0) ← 0x08                       // select byte 0 of IndirectAddr
port(0x3B1) ← ((Addr >>  0) & 0xFF)      // load byte 0 of address & region
port(0x3B0) ← 0x09                       // select byte 1 of IndirectAddr
port(0x3B1) ← ((Addr >>  8) & 0xFF)      // load byte 1 of address & region
port(0x3B0) ← 0x0A                       // select byte 2 of IndirectAddr
port(0x3B1) ← ((Addr >> 16) & 0xFF)      // load byte 2 of address & region
port(0x3B0) ← 0x0B                       // select byte 3 of IndirectAddr
port(0x3B1) ← ((Addr >> 24) & 0xFF)      // load byte 3 of address & region

// set the byte enables to write all four data bytes

port(0x3B0) ← 0x00                       // select IndirectByteEn
port(0x3B1) ← 0x0F                       // set all four byte enables

// load IndirectData with the data to be written, one byte at a time

port(0x3B0) ← 0x04                       // select byte 0 of IndirectData
port(0x3B1) ← ((Data >>  0) & 0xFF)      // load byte 0 of dword to write
port(0x3B0) ← 0x05                       // select byte 1 of IndirectData
port(0x3B1) ← ((Data >>  8) & 0xFF)      // load byte 1 of dword to write
port(0x3B0) ← 0x06                       // select byte 2 of IndirectData
port(0x3B1) ← ((Data >> 16) & 0xFF)      // load byte 2 of dword to write
port(0x3B0) ← 0x07                       // select byte 3 of IndirectData
port(0x3B1) ← ((Data >> 24) & 0xFF)      // load byte 3 of dword to write

// trigger the write internally using byte write of IndirectAccess register

port(0x3B0) ← 0x0C               // select IndirectAccess register
port(0x3B1) ← 0x00               // write any value to trigger the write
```