DECchip 21130
PCI Integrated Graphics and Video Accelerator

Technical Reference Manual

Order Number: EC–QD2QC–TE

Revision/Update Information: This manual supersedes the *DECchip 21130 PCI Integrated Graphics and Video Accelerator Technical Reference Manual* (EC–QD2QB–TE). It describes chip revision DC7538C.

Digital Equipment Corporation
Maynard, Massachusetts

November 1995

Contents

Preface xix

1 Introduction

1.1 Features 1–1
1.2 Minimum System 1–5
1.3 Basic Programming Model 1–5
1.3.1 Extensions to the Basic Programming Model 1–6
1.4 Chip Revisions 1–7

2 Internal Architecture

2.1 PCI Interface 2–1
2.1.1 PCI Registers 2–1
2.2 Command FIFO 2–1
2.3 Command Parser 2–3
2.3.1 Pixel-Processing Pipeline Coherence 2–3
2.3.2 Frame Buffer Writes 2–4
2.3.2.1 Fill Mode 2–4
2.3.2.2 Bresenham Setup Hardware 2–4
2.4 Pixel Engine 2–4
2.5 Memory Controller 2–5
2.5.1 Pixel Merge 2–5
2.5.2 Write Buffer 2–6
2.5.3 Frame Buffer and Device Access Requests 2–6
2.6 Core Registers 2–7
2.7 DMA Read FIFO 2–7
2.8 Copy Buffer 2–8
2.9 Frame Buffer and Device Access 2–8
2.10 Generic Peripheral Port 2–8
2.11 VGA Subsystem 2–9
2.12 Frame Buffer Memory 2–9
2.12.1 Frame Buffer Configurations 2–10

2.12.2 Hardware Mode Restrictions 2–10
2.13 Video Back End 2–11
2.13.1 Monitor Timing 2–11
2.13.2 Video Refresh 2–11
2.13.3 Pixel Occlusion Bitmap 2–11
2.13.4 Cursor Generation 2–12
2.13.5 VAFC Port 2–12
2.13.6 Palette and DAC 2–13
2.14 Clocks and Clock Control 2–13

3 Pinout

3.1 Signal List 3–1
3.2 Signal Descriptions 3–4
3.3 Signals by Function and Direction 3–11

4 Electrical Specifications

4.1 PCI Electrical Specification Conformance 4–1
4.2 Absolute Maximum Ratings 4–1
4.3 Normal Operating Conditions 4–2
4.4 Supply Current and Power Dissipation 4–2
4.4.1 Test Conditions 4–2
4.5 dc Specifications 4–3
4.5.1 Operating Specifications 4–5
4.6 ac Specifications 4–6
4.6.1 Parameters for PCI Clock Domain Signals 4–7
4.6.2 PCI Cycle Timing 4–10
4.6.3 Memory Cycle Timing 4–16
4.6.4 ROM and GPP Data Cycle Timing 4–21
4.6.5 Parameters for Pixel Clock and VAFC Clock Domain Signals 4–23
4.6.6 VAFC Cycle Timing 4–26

5 Mechanical Specifications

6 Thermal Specifications

7 Address Space

7.1 Overview 7–1
7.2 Configuration Space 7–2
7.3 ROM Space 7–2
7.4 VGA Memory Space 7–2
7.5 2DA Memory Space 7–2
7.5.1 2DA Base Address 0 Memory Space 7–2
7.5.1.1 Base Address 0 Core Space Organization 7–3
7.5.1.2 Base Address 0 Register Space Organization 7–5
7.5.1.3 Base Address 0 Alternate Control Space Writes 7–9
7.5.2 Base Address 1 Memory Space 7–10
7.5.2.1 Base Address 1 VGA Alternate Register Space 7–10
7.5.2.2 Base Address 1 Generic Peripheral Port Space 7–11
7.5.2.3 Base Address 1 Interrupt Status Register Space 7–12
7.5.2.4 Base Address 1 Palette and DAC Register Space 7–12
7.5.2.5 Base Address 1 ROM Sparse Space Access 7–13

8 Register Descriptions

8.1 Overview 8–1
8.2 PCI Configuration Registers 8–10
8.2.1 PCI Identification Register 8–11
8.2.2 PCI Command and Status Register 8–12
8.2.3 PCI Class and Revision Register 8–16
8.2.4 PCI Latency Timer and Header Type Register 8–17
8.2.5 PCI Device Base Address Registers 8–18
8.2.5.1 PDBR0 Functions 8–19
8.2.5.2 PDBR1 Functions 8–19
8.2.6 PCI Expansion ROM Base Address Register 8–21
8.2.7 PCI Interrupt Line Register 8–22
8.2.8 PCI Clock Control Register 8–23
8.3 Miscellaneous Registers 8–25
8.3.1 Command Status Register 8–25
8.3.1.1 Write Memory Barrier 8–26
8.3.2 Interrupt Status Register 8–27
8.4 Graphics Command Registers 8–29
8.4.1 Slope Registers 8–30
8.4.2 Span Width Register 8–33
8.4.2.1 GSWR Read 8–33
8.4.2.2 GSWR Write 8–34

8.4.3 Continue Register 8–35
8.4.3.1 GCTR Write 8–35
8.4.3.2 GCTR Read 8–37
8.4.4 Copy-64 Source and Destination Registers 8–38
8.4.5 Copy-64A Source and Destination Registers 8–40
8.4.6 Repeat Begin and End Registers 8–41
8.5 Graphics Control Registers 8–42
8.5.1 Mode Register 8–43
8.5.2 Deep Register 8–46
8.5.2.1 Gib-Endian Support 8–47
8.5.3 Slope-No-Go Registers 8–50
8.5.3.1 GSNR<7:0> Write 8–50
8.5.3.2 GSNR<7:0> Read 8–51
8.5.4 Copy Buffer Registers 8–53
8.5.5 Pixel Shift Register 8–55
8.5.6 Address Register 8–56
8.5.7 Data Register 8–58
8.5.7.1 GDAR Opaque-Fill and Transparent-Fill Modes 8–58
8.5.7.2 GDAR Line Mode 8–59
8.5.8 Foreground and Background Registers 8–60
8.5.9 Raster Operation Register 8–63
8.5.10 Pixel Mask Register 8–66
8.5.10.1 GPXR Stipple Modes 8–66
8.5.10.2 GPXR Simple Mode 8–67
8.5.10.3 GPXR Any Mode 8–67
8.5.11 Bresenham 1 Register 8–68
8.5.11.1 GB1R Line Mode 8–68
8.5.11.2 GB1R Scaled-Copy Mode 8–69
8.5.12 Bresenham 2 Register 8–70
8.5.13 Bresenham 3 Register 8–71
8.5.13.1 GB3R Line Mode 8–71
8.5.13.2 GB3R Scaled-Copy Mode 8–72
8.5.14 Bresenham Width Register 8–73
8.5.15 DMA Base Address Register 8–74
8.5.15.1 GDBR DMA-Read Copy Mode 8–74
8.5.15.2 GDBR Scaled-Copy Mode 8–75
8.5.16 Scaled-Copy Control Register 8–76
8.5.17 Dither Row and Column Registers 8–81
8.6 Hardware Cursor Registers 8–82
8.6.1 Cursor Mode Register 8–82
8.6.2 Cursor Base Address Register 8–84
8.6.3 Cursor XY Register 8–85
8.7 Video Control Registers 8–85

8.7.1 Video Base Address, Line Increment, and Line Width Registers 8–86
8.7.2 Video Valid Register 8–88
8.8 Video Format Registers 8–90
8.8.1 Video Pixel Format Register 8–91
8.8.1.1 Pixel Formatting 8–95
8.8.1.2 Addressing the RAM LUT in 15-bpp and 16-bpp True-Color Modes 8–96
8.8.1.3 Pixel Occlusion Bitmap 8–98
8.8.2 Video Pixel Occlusion Bitmap Base Address Register 8–100
8.8.3 Video Pixel Occlusion Bitmap Current Address Register 8–101
8.8.4 Video Current Refresh Address Register 8–102
8.8.5 Alternate Video Control Register 8–103
8.9 Palette and DAC Registers 8–104
8.9.1 Palette and DAC RAM Write and Read Address Registers . . . 8–105
8.9.2 Palette and DAC RAM Color Register 8–106
8.9.3 Palette and DAC Cursor Write and Read Address Registers 8–107
8.9.4 Palette and DAC Cursor Color Register 8–108
8.9.5 Palette and DAC Pixel Mask Register 8–109
8.9.6 Palette and DAC Status Register 8–110
8.9.7 Palette and DAC Command Register 0 8–111
8.9.8 Palette and DAC Command Register 1 8–112
8.9.9 Palette and DAC Signature Analysis Registers 8–114
8.10 VGA Register Overview 8–115
8.11 VGA External and General Registers 8–116
8.11.1 VGA Miscellaneous Output Register 8–117
8.11.2 VGA Feature Control Register 8–119
8.11.3 VGA Input Status 0 Register 8–120
8.11.4 VGA Input Status 1 Register 8–121
8.12 VGA Sequencer Registers 8–122
8.12.1 VGA Sequencer Index Register 8–122
8.12.2 VGA Sequencer Data Register 8–123
8.12.3 VGA Sequencer Reset Register 8–124
8.12.4 VGA Sequencer Clocking Mode Register 8–125
8.12.5 VGA Sequencer Plane Mask Register 8–127
8.12.6 VGA Sequencer Character Map Select Register 8–128
8.12.7 VGA Sequencer Memory Mode Register 8–129
8.13 VGA CRT Controller Registers 8–130
8.13.1 VGA CRTC Index Register 8–132
8.13.2 VGA CRTC Data Register 8–134
8.13.3 VGA CRTC Horizontal Total Register 8–135
8.13.4 VGA CRTC Horizontal Display End Register 8–136

8.13.5 VGA CRTC Start and End Horizontal Blank Registers 8–137
8.13.6 VGA CRTC Start and End Horizontal Sync Registers 8–139
8.13.7 VGA CRTC Vertical Total Register..................... 8–141
8.13.8 VGA CRTC Overflow Register 8–142
8.13.9 VGA CRTC Preset Row Register 8–143
8.13.10 VGA CRTC Maximum Scanline Register 8–144
8.13.11 VGA CRTC Cursor Start and End Registers 8–145
8.13.12 VGA CRTC Start Address High and Low Registers 8–147
8.13.13 VGA CRTC Cursor Location High and Low Registers 8–148
8.13.14 VGA CRTC Start and End Vertical Sync Register 8–149
8.13.15 VGA CRTC End Vertical Display Register 8–151
8.13.16 VGA CRTC Offset Register 8–152
8.13.17 VGA CRTC Underline Row Scan Register 8–153
8.13.18 VGA CRTC Start and End Vertical Blanking Registers 8–154
8.13.19 VGA CRTC Mode Control Register 8–155
8.13.20 VGA CRTC Line Compare Register 8–157
8.14 VGA Extended Registers 8–157
8.14.1 VGA Extended Paging Control Register................. 8–158
8.14.2 VGA Extended Host Page Offset A and B Registers 8–160
8.14.3 VGA Extended Split-Screen Start Address High and Low Byte Register 8–161
8.14.4 VGA Extended Interlace Control Register 8–162
8.14.5 VGA Extended Equalization Start and End Registers 8–163
8.14.6 VGA Extended Half-Line Register 8–164
8.14.7 VGA Extended Timing Control A Register 8–165
8.14.8 VGA Extended Timing Control B Register 8–166
8.14.9 VGA Extended Video FIFO Control Register 8–167
8.14.10 VGA Extended Clock Control A and B Registers 8–168
8.14.11 VGA Extended Interface Control Register 8–171
8.15 VGA Graphics Controller Registers 8–171
8.15.1 VGA Graphics Controller Index Register 8–172
8.15.2 VGA Graphics Controller Data Register 8–173
8.15.3 VGA Graphics Controller Set/Reset Register 8–174
8.15.4 VGA Graphics Controller Enable Set/Reset Register 8–175
8.15.5 VGA Graphics Controller Color Compare Register......... 8–176
8.15.6 VGA Graphics Controller Data Rotate Register 8–177
8.15.7 VGA Graphics Controller Read Map Select Register 8–178
8.15.8 VGA Graphics Controller Mode Register 8–179
8.15.9 VGA Graphics Controller Miscellaneous Register 8–181
8.15.10 VGA Graphics Controller Color Don’t Care Register 8–182
8.15.11 VGA Graphics Controller Bit Mask Register 8–183
8.16 VGA Attribute Controller Registers 8–183
8.16.1 VGA Attribute Controller Index/Data Register 8–184

8.16.2 VGA Attribute Controller Palette Registers 8–186
8.16.3 VGA Attribute Controller Mode Register 8–187
8.16.4 VGA Attribute Controller Overscan Register 8–188
8.16.5 VGA Attribute Controller Color Plane Enable Register 8–189
8.16.6 VGA Attribute Controller Pixel Panning Register 8–190
8.16.7 VGA Attribute Controller Color Select Register 8–191
8.17 VGA Color Registers 8–191
8.17.1 VGA Color Pixel Address Write Mode and Read Mode Registers 8–192
8.17.2 VGA Color DAC State Register 8–193
8.17.3 VGA Color Pixel Data Register 8–194
8.17.4 VGA Color Pixel Mask Register 8–195

9 PCI Operations

9.1 Configuration Operations 9–1
9.2 Memory Reads and Writes 9–2
9.2.1 Memory Write to Core Space 9–2
9.2.2 Memory Read of Core Space 9–3
9.2.2.1 Read Interlock 9–3
9.3 Target Operations 9–4
9.3.1 Access Granularity 9–5
9.3.2 Transaction Termination 9–5
9.4 Master Operation 9–9
9.4.1 DMA Read Transfer 9–10
9.4.2 Transaction Termination 9–10
9.4.3 Aborted DMA Transaction Termination 9–11
9.5 Parity 9–11
9.6 Bus Parking 9–11
9.7 Functions Not Supported 9–12

10 Graphics Operations

10.1 Overview 10–1
10.1.1 Frame Buffer Writes 10–1
10.1.2 Graphics Command Register Writes 10–2
10.1.3 Invoking Graphics Operations 10–3
10.1.4 Register Load Synchronization 10–4
10.1.5 Source and Destination Operands 10–5
10.2 Graphics Modes 10–6
10.2.1 Simple Mode 10–7
10.2.2 Opaque-Stipple Mode 10–9
10.2.2.1 Opaque Bit-Reversed Stipple Mode 10–11

10.2.3	Transparent-Stipple Mode	10–12
10.2.3.1	Transparent-Stipple with Pixel Mask Modes	10–13
10.2.4	Opaque-Fill Mode	10–14
10.2.4.1	Opaque Extended-Pattern Fill Mode	10–16
10.2.5	Transparent-Fill Mode	10–17
10.2.5.1	Transparent Extended-Pattern Fill Mode	10–18
10.2.6	Copy Mode	10–19
10.2.6.1	Source and Destination Alignment	10–22
10.2.6.2	Backward Copies	10–25
10.2.6.3	Priming and Flushing the Residue Register	10–26
10.2.6.4	Copy Direction Flag	10–28
10.2.6.5	64-Byte Unmasked Span Copies	10–29
10.2.6.6	Copy Buffer Operation	10–29
10.2.6.7	Fast Frame Buffer Access Using the Copy Buffer Registers	10–32
10.2.7	DMA-Read Copy Mode	10–33
10.2.7.1	Priming and Flushing the Residue Register	10–36
10.2.8	Scaled-Copy Mode	10–39
10.2.8.1	Video Rendering Pixel Flow	10–42
10.2.8.2	YUV Pixel Formats	10–44
10.2.8.3	16-bpp and 32-bpp RGB Formats	10–46
10.2.8.4	Rendering Full Frames	10–46
10.2.8.5	Unoccluded or Trivially Occluded Target Windows	10–46
10.2.8.6	Nontrivially Occluded Windows	10–48
10.2.8.7	Determining the Command FIFO Entry Availability	10–48
10.2.8.8	Scaling	10–48
10.2.8.9	Programming the Bresenham Scaler for Unoccluded Spans	10–49
10.2.8.10	Scaling of Occluded Spans	10–50
10.2.8.11	Specifying Span Starting and Trailing Edges	10–51
10.2.8.12	Required Software Interlock	10–51
10.2.9	Opaque-Line Mode	10–52
10.2.9.1	Drawing Lines with Frame Buffer Writes	10–52
10.2.9.2	Drawing Lines with the Slope Registers	10–54
10.2.9.3	Extending and Linking 2D Lines	10–57
10.2.10	Transparent-Line Mode	10–62

11 Programming

11.1	PCI Configuration Firmware	11–1
11.1.1	Device Address Mapping	11–1
11.1.2	Bus Mastering	11–2
11.1.3	Interrupt Routing	11–3
11.1.4	Expansion ROM	11–3
11.2	Mode Switching	11–3
11.2.1	VGA-to-2DA Mode Switching	11–3
11.2.2	2DA-to-VGA Mode Switching	11–4
11.2.2.1	Expected 2DA Operation During VGA Mode	11–5
11.3	Bit-Block Transfers	11–5
11.3.1	Screen-to-Screen Copy	11–5
11.3.2	Host-to-Screen Copy	11–8
11.3.3	Scaled-Copy	11–8
11.4	Dither Mathematics	11–8
11.5	Scaling Filters	11–9
11.6	Overlays	11–10
11.6.1	Flicker-Free Monochrome Overlay Support	11–10
11.6.2	True Monochrome Overlay with Pixel Occlusion Bitmap Hardware	11–11
11.6.3	True 8-bpp Overlay in 16-bpp or 32-bpp Frame Buffers	11–12
11.7	Fills	11–13
11.7.1	Solid	11–13
11.7.2	Stippling or Filling with a Monochrome Brush	11–14
11.7.3	Tiling or Filling with a Non-Monochrome Brush	11–14
11.8	Lines	11–15
11.8.1	Line Drawing Under X	11–15
11.8.2	Line Drawing Under Win32	11–17
11.9	Text	11–19
11.10	Repeat Loop Examples	11–19
11.11	Video Registers	11–22
11.11.1	Modifying the Contents of the Video Registers	11–22
11.11.2	Video Registers in 64-Bit and 32-Bit Frame Buffer Modes	11–23
11.11.3	Video Refresh Calculations	11–25
11.12	Programming for Alpha CPUs	11–26
11.12.1	Programmed I/O Through the CPU Write Buffer	11–27
11.12.2	Address and Continue Register Access	11–28

12 Hardware Interface

12.1 Frame Buffer Interface 12–1
12.1.1 Hardware Mode Restrictions 12–1
12.1.2 Frame Buffer Configuration Sensing 12–5
12.2 VGA Subsystem 12–6
12.2.1 PCI–to–VGA Interface 12–6
12.2.2 VGA-to-Frame Buffer Memory Interface 12–6
12.2.3 VGA-to-Video Back End Interface 12–6
12.3 ROM and Generic Peripheral Port Interface 12–8
12.3.1 GPP Read and Write Access 12–8
12.3.2 GPP Interrupts 12–9
12.4 Video Port and Display Monitor Interface 12–9
12.4.1 VESA Advanced Feature Connector 12–9
12.4.1.1 VAFC Operation 12–10
12.4.1.2 Relationship Between vafc_vclk and vafc_dclk 12–10
12.4.1.3 VAFC Pixel Output Modes 12–10
12.4.1.4 VAFC Pixel Input Modes 12–11
12.4.1.5 VAFC Input Windows 12–12
12.4.1.6 VAFC Blank Enable 12–12
12.4.1.7 VAFC Output Screen Resolutions 12–12
12.4.1.8 VAFC Input Screen Resolutions 12–13
12.4.2 Video Port Transceivers 12–13
12.4.3 Monitor Connection 12–13
12.4.4 Display Power Management Signaling 12–14
12.4.5 Display Data Channel 12–14
12.5 Clocks and Clock Control 12–15
12.5.1 Memory Clock 12–15
12.5.2 Core Clock 12–15
12.5.3 Pixel Clock 12–15
12.5.4 VGA Dot Clock 12–16
12.5.5 Test Clock 12–16

A Pin Summary

B Register Summary

C Technical Support, Ordering Information, and Associated Literature

Index

Figures

1–1	Supported VESA Display Modes	1–4
1–2	DECchip 21130 in Minimum System Configuration	1–5
1–3	Explanation of 21130 Chip Face Labels	1–8
2–1	DECchip 21130 Block Diagram	2–2
4–1	Clock Domains	4–3
4–2	PCI Clock Domain Signal ac Parameter Measurements	4–7
4–3	PCI Write — Cycle Start Timing	4–10
4–4	PCI Read — Cycle Start Timing	4–12
4–5	PCI Read or Write — Cycle End Timing	4–13
4–6	PCI Target Disconnect or Abort — pci_stop# Timing	4–14
4–7	PCI Configuration Cycle — pci_idsel Timing	4–15
4–8	PCI Parity — pci_par Timing	4–16
4–9	Hyperpage Mode Memory Write Cycle Timing	4–17
4–10	Hyperpage Mode Memory Read Cycle Timing	4–18
4–11	Read-Modify-Write Memory Cycle Timing	4–19
4–12	CAS-Before-RAS Memory Refresh Cycle Timing	4–20
4–13	ROM Data Cycle Timing	4–21
4–14	GPP Data Cycle Timing	4–22
4–15	Pixel Clock Domain Signal ac Parameter Measurements	4–23
4–16	VAFC Clock Domain Signal ac Parameter Measurements	4–24
4–17	VAFC Request Cycle Timing	4–26
4–18	VAFC Video Data Transfer Cycle Timing	4–27
5–1	DECchip 21130 208-Pin PQFP Package	5–1
7–1	Memory Space Organization	7–4
7–2	Core Space Maps	7–4
7–3	Base Address Register 0 Register Space Organization	7–5
7–4	ROM Sparse Space PCI Read Data Format	7–15
8–1	Slope Registers and Drawing Octants	8–32

8–2	Gib-Endian Transfers	8–48
8–3	Copy Buffer Layout	8–52
8–4	Foreground and Background as a Function of Bitmap Depth	8–62
8–5	Cursor Value Bits to Pixels Mapping	8–83
8–6	Variable Pixel Formats	8–94
8–7	RAM LUT Addressing in 15-bpp and 16-bpp True-Color Modes	8–97
8–8	Pixel Occlusion Bitmap Format	8–98
8–9	Screen Parameters	8–131
10–1	Simple Mode PCI Write-Data Format	10–7
10–2	Opaque-Stipple Mode PCI Write-Data Format	10–9
10–3	Opaque-Stipple Mode Operation	10–10
10–4	Transparent-Stipple Mode PCI Write-Data Format	10–12
10–5	Transparent-Stipple Mode Operation	10–13
10–6	Opaque-Fill Mode PCI Write-Data Format	10–14
10–7	Opaque-Fill Mode Operation	10–15
10–8	Copy Mode PCI Write Data Formats	10–20
10–9	Forward Span Copy	10–24
10–10	Primed Forward Span Copy	10–27
10–11	Copy Buffer Layout	10–31
10–12	DMA-Read Copy-Mode PCI Write-Data Format	10–34
10–13	DMA-Read Copy	10–38
10–14	Scaled-Copy Mode PCI Write Data Format	10–40
10–15	Scaled-Copy PCI DMA Start Address	10–41
10–16	Host-to-Screen Scaled-Copy and Video Rendering Pixel Flow	10–43
10–17	MCSR Format	10–48
10–18	Opaque-Line Mode PCI Write-Data Format	10–53
10–19	Opaque Line Drawing	10–58
10–20	Opaque-Line Drawing Sequence	10–61
11–1	BitBlt Using Copy Mode Example	11–7
11–2	Overlay Data in 16-bpp and 32-bpp Frame Buffers	11–13
11–3	Drawing Clipped Lines	11–16
11–4	Frame Buffer Address Space in 64-Bit and 32-Bit Modes	11–24
11–5	Video Address in 64-Bit and 32-Bit Modes	11–25
11–6	Video Scanline Addresses	11–26

12–1	VGA Subsystem Interfaces	12–7
12–2	Clock Generation	12–17

Tables

1–1	21130 Chip Revision Levels	1–7
2–1	Mode Restrictions	2–11
3–1	Signal List	3–1
3–2	Signal Description	3–4
3–3	Signals by Function	3–11
3–4	Signals and Active Levels by Direction	3–15
4–1	Absolute Maximum Ratings	4–2
4–2	Normal Operating Conditions	4–2
4–3	Pin Characteristics	4–4
4–4	dc Parameters	4–6
4–5	PCI Clock Domain Signal ac Parameters	4–8
4–6	PCI Write — Cycle Start Timing Parameters	4–10
4–7	PCI Read — Cycle Start Timing Parameters	4–12
4–8	PCI Read or Write — Cycle End Timing Parameters	4–13
4–9	PCI Target Disconnect or Abort — pci_stop# Timing Parameters	4–14
4–10	PCI Configuration Cycle — pci_idsel Timing Parameters	4–15
4–11	PCI Parity — pci_par Timing Parameters	4–16
4–12	Hyperpage Mode Memory Write Cycle Timing Parameters	4–17
4–13	Hyperpage Mode Memory Read Cycle Timing Parameters	4–18
4–14	Read-Modify-Write Memory Cycle Timing Parameters	4–19
4–15	CAS-Before-RAS Memory Refresh Cycle Timing Parameters	4–20
4–16	ROM and GPP Data Cycle Timing Parameters	4–21
4–17	Pixel Clock and VAFC Clock Domain Signal ac Parameters	4–25
4–18	VAFC Request Cycle Timing Parameters	4–26
4–19	VAFC Video Data Transfer Cycle Timing Parameters	4–27
6–1	Airflow Versus Temperature	6–1
7–1	Core Space per Frame Buffer Option	7–3
7–2	Registers Supported by Write Alias Spaces 1, 4, 5, 6, and 7	7–6

7–3	Base Address Register 0 Register Map	7–7
7–4	Targets for Writes to Alternate Control Space	7–9
7–5	Base Address Register 1 Memory Space Map	7–10
7–6	Base Address Register 1 VGA Register Map	7–11
7–7	Base Address Register 1 Palette and DAC Register Map	7–13
7–8	ROM Sparse Space PCI Read Data Field Description	7–15
8–1	21130 Registers	8–2
8–2	Palette Snoop Response	8–14
8–3	Memory Clock Frequency Select	8–24
8–4	Graphics Modes	8–45
8–5	Boolean Raster Operations	8–64
8–6	GSCR Mode Field Description	8–77
8–7	Typical Scaled-Copy Mode Operations	8–79
8–8	Cursor Pixel Value Bit Description	8–83
8–9	Video Pixel Formats	8–92
8–10	Variable Pixel Formats	8–95
8–11	Pixel Occlusion Bitmap Field Description	8–98
8–12	VGA Register Port Map	8–115
8–13	VGA External and General Register Port Map	8–116
8–14	Displayed Vertical Size as Function of HSP and VSP	8–118
8–15	VGA CRTC and Extended Register Indices	8–132
8–16	Typical Pixel Clock Frequencies	8–170
8–17	VGA Graphics Controller Write Modes	8–180
9–1	PCI Transactions to 2DA Memory Space	9–6
9–2	PCI Transactions to Configuration Space and Expansion ROM Space	9–7
9–3	PCI Transactions to VGA Memory and I/O Space	9–8
9–4	Snooped DAC Write PCI Transactions to VGA Space	9–9
10–1	Mode-Dependent Frame Buffer Write Operations	10–2
10–2	Graphics Command Register Write Operations	10–3
10–3	Source and Destination Operands According to Mode	10–5
10–4	Simple Mode Parameters	10–7
10–5	Opaque-Stipple Mode Parameters	10–9
10–6	Transparent-Stipple Mode Parameters	10–12
10–7	Opaque-Fill Mode Parameters	10–14
10–8	Transparent-Fill Mode Parameters	10–17
10–9	Copy Mode Parameters	10–19

10–10	Copy Mode Span Limits	10–21
10–11	Assigning the Pixel Shift Value	10–25
10–12	DMA-Read Copy-Mode Parameters	10–33
10–13	Edge Mask Settings in DMA-Read Copy Mode	10–37
10–14	Edge Mask for Short Spans in DMA-Read Copy Mode	10–37
10–15	Scaled-Copy Mode Parameters	10–39
10–16	Scaled-Copy Mode PCI Write Data Field Description	10–40
10–17	Opaque-Line Mode Parameters	10–52
10–18	Opaque-Line Mode Parameters Using Slope Registers	10–55
10–19	Transparent-Line Mode Parameters	10–62
11–1	21130 Base Address and Memory Space Enable Fields	11–1
11–2	PCI Latency Timer and Bus Master Enable Fields	11–3
11–3	Cursor Color Displayed with Monochrome Overlay	11–12
11–4	Fully Shadowed, Pseudo-Shadowed, and Video-Disabled Registers	11–22
11–5	Video Address Configuration Registers	11–25
12–1	Pin Usage in VGA Mode	12–2
12–2	Pin Usage in 32-bit GPP and ROM Modes	12–2
12–3	Pin Usage in 64-bit GPP and ROM Modes	12–3
12–4	Pin Usage in 32-bit GPP and VAFC Modes	12–4
12–5	DPMS States	12–14
A–1	Signals by Function	A–1
B–1	21130 Register Alphabetical List	B–1
B–2	21130 Register Summary	B–5

Preface

This manual describes the architecture, internal design, external interface, and specifications of the DECchip 21130 PCI Integrated Graphics and Video Accelerator.

Audience

This manual is for system designers, software developers, and hardware engineers who use the DECchip 21130.

Manual Organization

This manual includes the following chapters and appendices and an index.

Chapter 1 Introduction
Chapter 2 Internal Architecture
Chapter 3 Pinout
Chapter 4 Electrical Specifications
Chapter 5 Mechanical Specifications
Chapter 6 Thermal Specifications
Chapter 7 Address Space
Chapter 8 Register Descriptions
Chapter 9 PCI Operations
Chapter 10 Graphics Operations
Chapter 11 Programming
Chapter 12 Hardware Interface
Appendix A Pin Summary
Appendix B Register Summary
Appendix C Technical Support, Ordering, and Associated Literature
Index

Conventions

The following conventions are used throughout this manual.

Abbreviations

- **bpp**

  The terms ‘‘bits per pixel’’ and ‘‘bits/pixel’’ are abbreviated as bpp.

- **Binary Multiples**

  When representing binary multiples, the abbreviations K, M, and G (kilo, mega, and giga) have the following values.

K =	2^{10} (1024)
M =	2^{20} (1,048,576)
G =	2^{30} (1,073,741,824)

  For example:

2KB =	2 kilobytes =	2×2^{10} bytes
4MB =	4 megabytes =	4×2^{20} bytes
8GB =	8 gigabytes =	8×2^{30} bytes
2K pixels =	2 kilopixels =	2×2^{10} pixels
4M pixels =	4 megapixels =	4×2^{20} pixels

- **Register Access**

  The abbreviations used to indicate the type of access to register fields and bits have the following definitions:

  IGN — Ignore

  Bits and fields specified as IGN are ignored when written.

  MBZ — Must Be Zero

  Software must never place a nonzero value in bits and fields specified as MBZ. Reads return unpredictable values. Such fields are reserved for future use.

  RAZ — Read As Zero

  Bits and fields specified as RAZ are ignored on writes and return a zero when read.

  RC — Read Clears

  Bits and fields specified as RC are cleared when read. Unless otherwise specified, such fields cannot be written.

RES — Reserved

Bits and fields specified as RES are reserved by Digital and should not be used; however, zeros can be written to reserved fields that cannot be masked.

RO — Read Only

Bits and fields specified as RO can be read and are ignored (not written) on writes.

RW — Read/Write

Bits and fields specified as RW can be read and written.

R/W1C — Read/Write One to Clear

Bits and fields specified as R/W1C can be read. Writing a one clears these bits for the duration of the write; writing a zero has no effect.

WO — Write Only

Bits and fields specified as WO can be written but not read.

Addresses

Unless otherwise noted, all addresses and offsets are hexadecimal.

Aligned and Unaligned

The terms *aligned* and *naturally aligned* are interchangeable and refer to data objects that are powers of two in size. An aligned datum of size 2^n is stored in memory at a byte address that is a multiple of 2^n; that is, one that has n low-order zeros. For example, an aligned 64-byte stack frame has a memory address that is a multiple of 64.

A datum of size 2^n is *unaligned* if it is stored in a byte address that is not a multiple of 2^n.

Bit Notation

Multiple-bit fields can include contiguous and noncontiguous bits contained in angle brackets (<>). Multiple contiguous bits are indicated by a pair of numbers separated by a colon (:). For example, <9:7,5,2:0> specifies bits 9,8,7,5,2,1, and 0. Similarly, single bits are frequently indicated with angle brackets. For example, <27> specifies bit 27.

Caution

Cautions indicate potential damage to equipment or loss of data.

Data Units

The following data unit terminology is used throughout this manual.

Term	Words	Bytes	Bits	Other
Byte	½	1	8	—
Word	1	2	16	—
Dword	2	4	32	Longword
Quadword	4	8	64	2 Dwords
Hexaword	16	32	256	8 Dwords

External

Unless otherwise stated, throughout this manual the term external means not contained in the DECchip 21130.

Note

Notes emphasize particularly important information.

Numbering

All numbers are decimal or hexadecimal unless otherwise indicated. In cases of ambiguity, a subscript indicates the radix of nondecimal numbers. For example, 19 is decimal, but 19_{16} and 19A are hexadecimal. (Also see Addresses.)

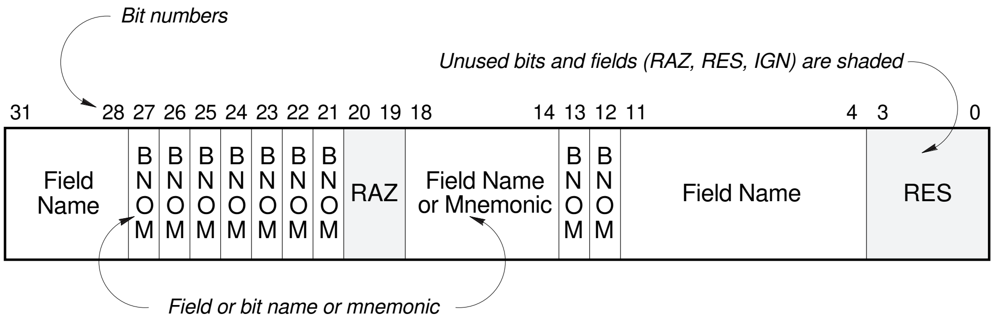
Ranges and Extents

Ranges are specified by a pair of numbers separated by two periods (..) and are inclusive. For example, a range of integers 0..4 includes the integers 0, 1, 2, 3, and 4.

Extents are specified by a pair of numbers in angle brackets (<>) separated by a colon (:) and are inclusive. Bit fields and register sets are often specified as extents. For example, bits <7:3> specifies bits 7, 6, 5, 4, and 3, and GSLR<7:0> specifies a set of eight graphics slope registers.

Register Figures

The following figure defines the conventions used in register format figures.



Signal Names

Signal names are printed in lowercase, boldface type. Low-asserted signals are indicated by the number sign (**#**) suffix. For example, **pll_clk_in** is a high-asserted signal, and **pll_clk_in#** is a low-asserted signal.

1 Introduction

The DECchip 21130 is a DRAM-based, 2D graphics accelerator for desktop systems running Microsoft Windows 3.1, Windows 95, and Windows NT. The 21130 integrates a peripheral component interconnect (PCI) interface, graphics accelerator, RAMDAC, phase-locked loop (PLL) timing generators, VGA controller, and video windowing hardware to merge graphics and video data.

The 21130 continues to refine Digital's proven graphics architecture. It incorporates most of the 2D graphics features introduced with the DECchip 21030 and adds many new features in a smaller package at lower cost.

1.1 Features

The following is a summary of the 21130 hardware features.

PCI Interface

The 21130 provides a glueless PCI interface with separate access to the VGA controller, 2D accelerator, and external BIOS ROM. It includes the PCI registers and supports PCI master and target transactions and direct memory access (DMA) read capability. The interface is fully electrically compliant with the *PCI Local Bus Specification, Revision 2.0.* The 21130 and connected external devices present only one PCI load.

Integrated Multimedia Real-Time Video Display Acceleration

Includes:

- YUV-to-RGB index color space conversion
- Image scaling for arbitrary source and destination bitmap sizes
- Support for the Microsoft display control interface (DCI) for video acceleration (Video for Windows)
- Support for industry-standard codecs (Indeo, Cinepak, Video1, MPEG1, JPEG, Px64)

1.1 Features

- VESA advanced feature connector (VAFC)

Faster and Simpler Line Drawing

In many systems, software is responsible for all of the cumbersome line setup calculations, including generating the Bresenham error and address increments for the major and minor axis steps as well as the initial error term.

In the 21130's streamlined interface, software needs to write only the absolute *dx* and absolute *dy* values of the line segment to one of eight slope registers, implicitly specifying the drawing octant. The 21130 then automatically generates all of the Bresenham terms, initializes the Bresenham registers, and draws up to 16 pixels. Lines can be extended beyond 16 pixels simply by using the continue register.

Graphics and Multimedia Video Pipeline

The pipeline includes the command FIFO and parser, pixel engine, DMA read FIFO, and graphics registers. It provides real-time scale, dither, and YUV-to-RGB index conversion for video display in a window or full-screen video display.

DMA Engine for Image Data

The 21130 has the DMA-read copy mode for fast host-to-screen bit-block transfers (BitBlts) and the scaled-copy mode for host-to-screen stretchBlt transfers. These modes allow large, contiguous regions to be directly transferred from main memory to the frame buffer. The onchip PCI interface allows main memory, other PCI graphics devices, or PCI video devices to be the external source.

Proprietary Dithering

The 21130 implements Digital's AccuLook dithering algorithm (patent-pending) to support rendering to 8-bpp and 16-bpp bitmaps. The quality of dithered 8-bpp pseudo-color images surpasses standard 16-bpp, direct-color image quality. The quality of the dithered 16-bpp, direct-color images is comparable to 24-bpp, true-color image quality.

64-Byte Copy Buffer

The copy buffer supports high-bandwidth local frame buffer BitBlts.

Bresenham Line Drawing Engine and Setup Hardware

The onchip Bresenham line drawing engine and setup hardware performs Bresenham per-pixel line stepping and most of the Bresenham-term setup.

Color Expansion

The 21130 expands monochrome bitmaps to various pixel depths for drawing text or filling regions with solid or bitonal brushes.

1.1 Features

32-Bit or 64-Bit DRAM Display Memory Controller

The memory controller provides a 64-bit data path to the frame buffer. When the VAFC port is active, the memory controller is in 32-bit frame buffer mode, and the upper half of the data path is available for full-time video I/O. The memory controller supports hyperpage mode (extended data out—EDO) DRAMs, and linear frame-buffer addressing in 1, 2, or 4MB frame buffer configurations.

VGA Controller

The VGA controller supports VGA modes through 13_{16}. It includes the video timing function and VGA registers.

Palette and DAC

The 21130's palette and DAC includes a 24-bit, true-color DAC; a 256-color RAM look-up table (LUT) for graphics; and a 256-color ROM LUT for video.

VAFC Port

The VAFC port passes bidirectional RGB or indexed 8- or 16-bit video on the upper 32 bits of the DRAM data path.

Generic Peripheral Port

The generic peripheral port (GPP) provides access to 8-bit devices such as video processing circuits, audio chips, or I^2C controllers. The 8-bit GPP is multiplexed on the DRAM data path.

64 × 64 × 2 Onchip Cursor

The 21130 incorporates onchip cursor control. It retains a cursor image in its off-screen frame buffer memory and passes its control to the palette and DAC.

High-Performance CRT Controller

The 21130 CRTC supports the following VESA-standard, 75-Hz, noninterlaced resolutions (Figure 1–1):

1280 × 1024	8-bpp
1024 × 768	8- and 16-bpp
800 × 600	8-, 16-, and 24-bpp
640 × 480	8-, 16-, and 24-bpp

Note

For resolutions with 1024 or more vertical scanlines, the vertical front porch must be a minimum of two scanlines.

1.1 Features

Figure 1–1 shows the supported VESA modes. The figure does not show lower-resolution, VGA-compatible display formats for VGA text and graphics (VGA modes 0 through 13_{16}).

Figure 1–1 Supported VESA Display Modes

	8–bpp	16–bpp	24–bpp*	32–bpp
640 X 480	31.50 *101*	63.00 *111*	94.50 *112*	126.00 *112*
800 X 600	49.50 *103*	99.00 *114*	148.50 *115*	198.00 *115*
1024 X 768	78.75 *104*	157.50 *117*		
1280 X 1024	135.00 *107*			

*Packed pixels, not accelerated

Legend:

31.50	Peak memory bandwidth (MB/s)
101	VESA Int10 mode (hexadecimal)

(white box)	1MB frame buffer
(shaded box)	2MB frame buffer

The peak memory bandwidth is the product of the pixel rate and pixel depth.

Display Power-Management Signaling

The 21130 also supports the VESA display power-management signaling (DPMS) for EPA Energy Star (Green PC) requirements. (See the *VESA Monitor Timing Proposed Standard for 640X480, 800X600, and 1280X1024 at 75 Hz, VDMT 75HZ Rev 1.2P* and the VESA *Display Power Management Signaling (DPMS) Proposal, Version 1.0p, Revision 0.7p* for more information.)

Functions Not Supported

The 21130 does not support the complete Windows set of 256 Boolean raster operations. The Windows manager and most applications typically use only three or four of the 256 Boolean raster operations (raster ops or ROPs). The most commonly used ROPs are included in the 16 functions supported by the 21130 hardware. For the infrequent cases when either the Windows NT display driver or Windows 95 display driver encounters an unsupported

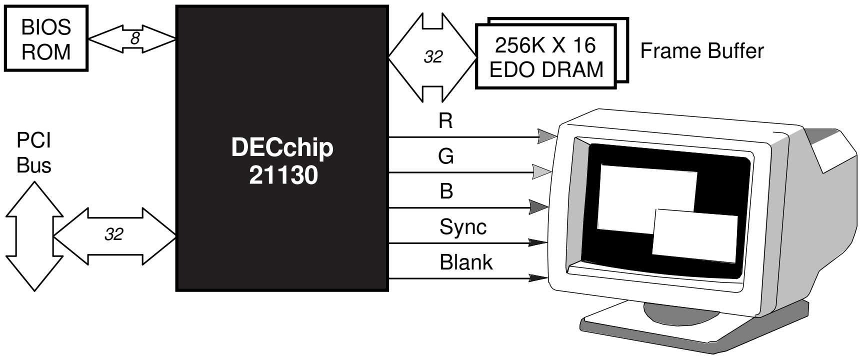
ROP, it defaults to the graphics device interface (GDI). This does not affect performance for Windows or the majority of applications (including Windows benchmarks) and has only a negligible effect on the performance of the remaining minority of applications.

1.2 Minimum System

The DECchip 21130 plays with a multiplicity of processors and operating systems and graphical user interfaces (GUIs). Its high level of integration facilitates the lowest cost graphics and multimedia subsystem implementation with the minimum real-estate requirements on the motherboard or as a plug-and-play option.

The 21130 allows simple, glueless, multimedia subsystem implementations. Figure 1–2 shows the simplest configuration, requiring only four chips.

Figure 1–2 DECchip 21130 in Minimum System Configuration



1.3 Basic Programming Model

In the basic 21130 programming model, the processor writes directly to addresses within the 21130's frame buffer address space. The data is interpreted according to the current graphics mode to perform the desired operation. Exceptions to this paradigm are described next.

There are four primary 21130 operating modes: simple, stipple, line, and copy. Each primary mode of operation has an associated byte mask, Boolean raster operation, and bitmap depth. The byte mask determines which bits in a pixel can be modified during a write. The raster operation provides one of sixteen 2-operand Boolean functions of source (or pattern) and destination, and automatically performs a read-modify-write cycle when necessary. The bitmap depth specifies how pixel data maps to frame buffer Dwords.

Simple Mode

In simple mode, writes to the frame buffer are similar to writes to main memory, except for the optional effects of the byte mask, pixel mask, raster operation, and bitmap depth. In this mode, the byte mask and the pixel mask determine which pixels are written.

Stipple Mode

In stipple mode (color expansion mode), data written to the frame buffer is interpreted as a monochrome pattern, in which the following occurs:

- Ones are expanded into foreground pixels.
- Zeros are either expanded into background pixels (opaque stipple mode) or not expanded (transparent stipple mode).

In the opaque and transparent stipple modes, the pixel mask can be programmed to write fewer than 32 pixels.

Line Mode

In line mode, the processor sets up registers for the Bresenham engine and then writes into the frame buffer at the starting address of the line. The data written by the processor is interpreted as a monochrome pattern, in which the following occurs:

- Ones are expanded into foreground pixels.
- Zeros are either expanded into background pixels (opaque line mode) or have no effect (transparent line mode).

Copy Mode

In copy mode, the processor writes alternately to the source and destination address within the frame buffer. The data written by the processor is interpreted as a bit mask that specifies which pixels are to be read (source) or written (destination).

1.3.1 Extensions to the Basic Programming Model

Several extensions to the basic programming model are available.

Stipple-Fill Mode

In stipple-fill mode, each write causes the 21130 to fill as many as 2K pixels on a scanline, using the 32-bit data as a 32-bit monochrome pattern.

Line Mode

In line mode, the eight slope registers (one per octant) allow the processor to offload some of the traditional line setup computations. The processor writes the absolute values of the line rise and run to one of the slope registers, implicitly specifying a drawing octant, and causes the 21130 to generate the Bresenham address and error terms and draw up to 16 pixels at one time. Consequently, the processor can specify a short, connected line with one 32-bit write. Lines can be extended beyond 16 pixels simply by writing the continue register.

Copy Mode

In copy mode, the copy-64 source and copy-64 destination registers allow the processor to read 64 unmasked bytes from the source and write 64 unmasked bytes to the destination with one write to each register. This makes full use of the 64-byte copy buffer for large area copies of 8-bit pixels.

In the DMA copy modes, the processor can specify the addresses of the source in PCI memory space. One write to the frame buffer then causes the 21130 to begin reading from the PCI and writing to the frame buffer.

1.4 Chip Revisions

There are 21130 chips with different revision levels. As a result, you need to check for the chip revision number printed on the face of the chip. Table 1–1 describes the revision levels and Figure 1–3 shows where to locate the printed revision level on the chip. This information is also available in the PCI class and revision register (PCRR, see Section 8.2.3).

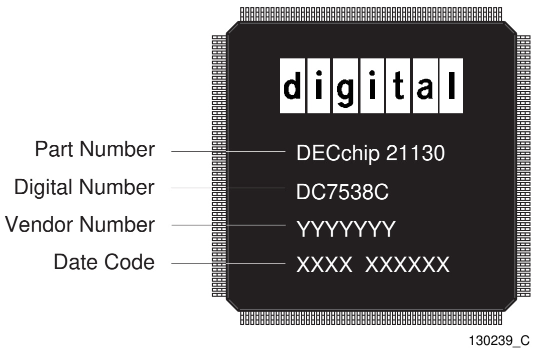
This manual describes revision C (DC7538C).

Table 1–1 21130 Chip Revision Levels

Revision Level	Chip Revision Number
A	DC7538A
B	DC7538B
C	DC7538C

1.4 Chip Revisions

Figure 1–3 Explanation of 21130 Chip Face Labels



2

Internal Architecture

This chapter describes the DECchip 21130 microarchitecture. Figure 2–1 is a block diagram of the chip showing its major functional areas.

2.1 PCI Interface

The PCI interface connects the 21130 core to the PCI bus. The primary function of the PCI interface is to keep the command FIFO filled with writes and commands issued over the PCI to the 21130 registers and frame buffer.

The PCI interface supports most of the PCI bus commands as a target. It also allows the 21130 to be a PCI master for direct memory access (DMA) operations, transferring pixel data between memory that can be accessed from the PCI and the 21130 frame buffer. DMA read data is taken from the PCI and passed to the DMA read FIFO. As a target or master, the PCI interface initiates and responds to different types of termination sequences. The PCI interface controls all access to the PCI configuration registers.

2.1.1 PCI Registers

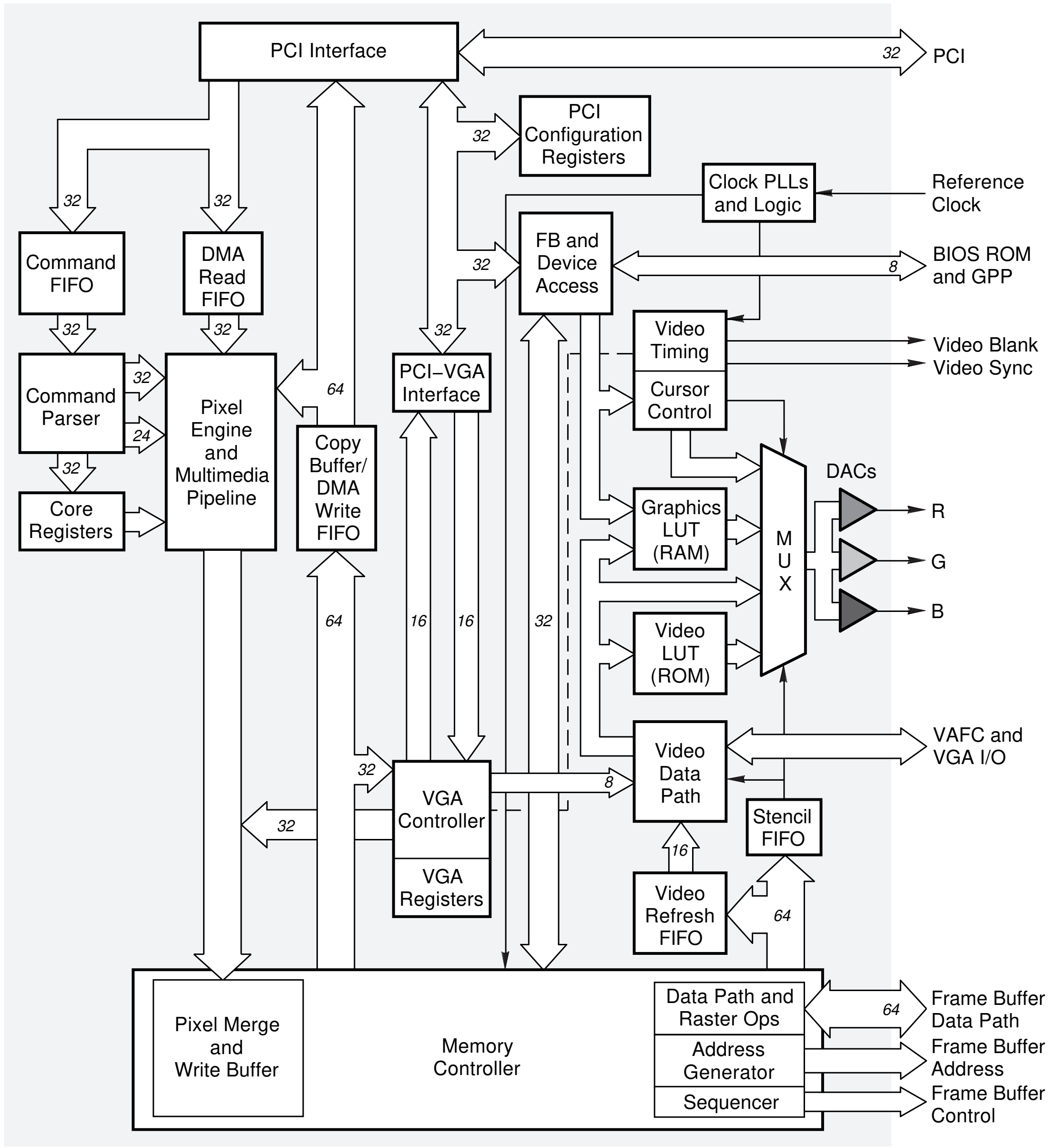
The PCI registers reside in the 21130 PCI configuration space and include the device-independent registers required for all PCI devices as well as the PCI device-specific registers in the 21130.

2.2 Command FIFO

The command FIFO contains 64 Dword entries. It buffers writes to the frame buffer and core registers for processing by the 21130 core. The PCI interface loads the command FIFO with an address followed by an arbitrary number of data entries. The command parser unloads the entries and initiates processing.

2.2 Command FIFO

Figure 2–1 DECchip 21130 Block Diagram



The command FIFO contains only core-space write data, such as writes to the 21130 core registers, alternate control space, and frame buffer space. Because the PCI interface accepts burst memory writes to base address 0 as a PCI target, the command FIFO can independently store an address or data in each of its 64 entries. In other words, the command FIFO can hold any combination of addresses and data, from one address and 63 entries of burst data to 32 pairs of address and data entries. If the command parser detects a sequence of one address and multiple data entries, it generates and matches the correct address to each data entry when it unloads the command FIFO.

The command FIFO is a boundary between chip clocking domains (see Figure 4–1). The input runs at the PCI clock rate and the output runs at the 21130 core clock rate.

2.3 Command Parser

The command parser processes graphics commands and register write accesses. It unloads graphics commands (in the form of address and data) from the command FIFO and performs initial processing before passing commands to the pixel engine. If the command parser detects a sequence of one address and multiple data entries, it generates an address for each data entry.

The command parser runs at the 21130 core clock rate.

2.3.1 Pixel-Processing Pipeline Coherence

The pixel-processing pipeline consists of the pixel engine, pixel-merge function, write buffer, and memory controller. The command parser imposes hardware register interlocks to ensure coherent processing through the pipeline. The interlocks allow the pipeline to operate concurrently with register updates; that is, updates to graphics operation parameters.

Most of the parameter registers are double-buffered. The command parser schedules buffered-register loading and swapping, and, in certain cases, delays command processing to maintain parameter coherence through the pipeline. In the case of writes to the command status register, raster operation register, mode register, and scaled-copy control register, the interlock mechanism waits until the pipeline has been flushed before it resumes processing.

Note

The deep register is not managed by hardware interlock and requires software scheduling.

2.3.2 Frame Buffer Writes

The command parser detects all writes to the frame buffer and begins processing the graphics command specified by the current graphics mode. The command parser does not perform any pixel address or data calculations, but forwards *predigested* commands to the pixel engine for processing.

2.3.2.1 Fill Mode

For all fill mode drawing, the command parser breaks large-span fill commands into 32-pixel span commands which the pixel engine can accept and process. The pixel engine can process individual pixels, 16-pixel lines, and 32-pixel spans.

2.3.2.2 Bresenham Setup Hardware

The command parser incorporates the Bresenham setup hardware (the Bresenham engine is in the pixel engine). When the command parser receives a write to the slope registers, span width register, or slope-no-go registers, it calculates the Bresenham terms: length, initial error, error increments 1 and 2, and address increments 1 and 2. When the write is to a slope register or the span width register, the command parser also forwards the line command to the pixel engine. The command parser forwards all other line, span, and pixel mode drawing commands directly to the pixel engine.

2.4 Pixel Engine

The pixel engine does all of the pixel address and value calculations. It receives single-pixel, 16-pixel line, and 32-pixel span commands from the command parser and reduces them into individual 16-bit-aligned or 64-bit-aligned frame buffer address and data pairs destined for the memory controller. The pixel engine contains the following pixel processing hardware to generate pixel addresses and data.

- Stipple logic

  The stipple logic expands a monochrome bitmap (and optional bitmap mask) into foreground or background color (or neither), on a per-pixel basis over a 16-pixel line or 32-pixel span.

- Bresenham engine

  The Bresenham engine steps through the pixels of a line (up to 16 pixels at a time), generating a pixel address for each step.

- Dither logic

  The dither logic implements Digital's AccuLook dithering algorithm. The algorithm maps 8 bits per channel (24-bpp) YUV or RGB color to various combinations of 8-, 16-, or 32-bpp YUV or RGB. The source of the 24-bpp RGB is the scaled-copy, 24-bit pixel stream.

After the pixel engine reduces spans into pixels and calculates the mode-dependent pixel data, it translates pixel addresses into frame buffer addresses as a function of the frame buffer depth and target bitmap. The pixel engine forwards each memory access to the pixel-merge function.

The pixel engine receives data directly from the memory controller for copy operations. In copy mode, the pixel engine first forwards a series of read requests, tagged with the source address, to the pixel-merge function. Eventually, the memory controller returns source pixel data (64 bits at a time) to the copy buffer. Then, when instructed by the command parser, the pixel engine unloads the copy buffer and forwards that data back to the pixel-merge function as a write tagged with the destination address.

The pixel engine runs at the core clock rate.

2.5 Memory Controller

The memory controller provides the interface to the frame buffer and responds to requests from two sources: the pixel engine and the frame buffer and device access function. It responds to requests from the pixel engine (through the pixel-merge function and write buffer) for accelerated drawing operations in the frame buffer.

2.5.1 Pixel Merge

To improve drawing performance, the pixel merge function merges pixel writes to eliminate consecutive writes to different bytes at the same quadword address. (Such write sequences occur frequently during line drawing and multimedia operations.) The pixel merge function:

- Receives requests for frame buffer writes from the pixel engine
- Temporarily stores the requests
- Determines if the writes are to different bytes in the same quadword
- Merges consecutive writes to the same quadword

Consecutive writes are not merged if they do not use the same raster operation or if consecutive operations are writing to the same byte. The pixel merge buffer is flushed when the write buffer is empty.

The pixel merge function runs at the core clock rate.

2.5.2 Write Buffer

The DRAM sequencer often processes write commands faster than the graphics engine generates them, and the write buffer helps to optimize DRAM use between drawing operations and screen refresh. The write buffer is a 16-entry FIFO that buffers merged writes from the pixel engine. The memory controller unloads the buffer when the DRAM is available for drawing.

The write buffer is written at the core clock rate and read at the memory clock rate (twice the core clock rate).

As long as the write buffer contains valid entries, the memory controller continues to unload addresses and data. The memory controller performs a Boolean ROP function on each write, as specified by the raster operation register. If the ROP is a function of the destination, the memory controller automatically performs the necessary read-modify-write operation.

The memory controller returns requested read data to the pixel engine through the copy buffer.

2.5.3 Frame Buffer and Device Access Requests

The memory controller responds to occasional asynchronous requests from the frame buffer and device access function for the following:

- Direct host reads of the frame buffer
- External EEPROM reads and writes
- GPP reads and writes

Note

The 21130 supports one external (E)(E)PROM. It and its associated functions are referred to as the BIOS ROM, EEPROM, flash ROM, PCI expansion ROM (space), and ROM ((sparse) space).

To conserve 21130 pins, the GPP address, data, and control lines are tied to a subset of the memory controller address and data lines. Therefore, to read or write the external EEPROM or a generic peripheral, the memory controller must interrupt processing of write buffer address and data. (See Section 2.12.2 for more information about mode restrictions due to shared pins.)

When the memory controller receives a request from the frame buffer and device access function, it suspends write-buffer entry processing within a maximum latency and services the request. The frame buffer and device access function specifies the type of access and passes address and data as required. The memory controller performs the following cycles:

- GPP or EEPROM access—the memory controller drives an address and either drives (write) or latches (read) a byte of data.
- Cursor data fetch—the memory controller performs a frame buffer read at the specified address and returns two successive quadwords to the frame buffer and device access function.

After an asynchronous access has been serviced, the memory controller resumes processing addresses and data from the write buffer.

The memory controller also issues CAS-before-RAS refresh cycles frequently enough to keep the dynamic memory refreshed.

The memory controller runs at twice the core clock rate, and can perform CAS page-mode cycles at the core clock rate.

2.6 Core Registers

The core registers are all the registers physically implemented in the base address 0 register space core (Section 7.5.1). Many of the core registers are double-buffered to allow pipelined graphics processing to overlap register updates. The command parser controls the core register read access, write access, and double-buffering.

2.7 DMA Read FIFO

The DMA read FIFO contains 32 Dword entries. It is loaded by the PCI interface during a DMA-read copy operation and unloaded by the pixel engine. The DMA read FIFO contains only pixel data.

The DMA read FIFO is a boundary between chip clocking domains (see Figure 4–1). The input runs at the PCI clock rate and the output runs at the 21130 core clock rate.

2.8 Copy Buffer

The copy buffer contains 8 quadword (64-bit) entries. It is used when transferring data from a frame buffer source to a destination in the frame buffer.

The memory controller returns source data to the copy buffer. In copy mode, the pixel engine forwards the data, tagged with a destination address, down the pixel processing pipeline to the memory controller.

2.9 Frame Buffer and Device Access

The frame buffer and device access (FBDA) function collects requests for access to the frame buffer and external devices (GPP and EEPROM) from several sources. It prioritizes and forwards the requests to the memory controller. The memory controller processes the requests as interrupts to write-buffer processing. The following requests are routed to the FBDA function:

- Direct frame buffer read—from the host through the PCI interface.
- External EEPROM read and write—PCI expansion ROM space read requests detected and routed by the PCI interface.

The FBDA function provides the signals required to write one 8-bit EEPROM and a generic peripheral.

2.10 Generic Peripheral Port

The generic peripheral port (GPP) consists of an 8-bit data bus and several control signals. It can connect, with little or no glue logic, to many types of devices including audio chips, DSPs, and a variety of video processing components.

To conserve 21130 pins, the GPP address, data, and control lines are tied to a subset of the memory controller address and data lines. Therefore, to read or write a generic peripheral (or the external EEPROM), the memory controller must interrupt processing of write buffer address and data. (See Section 2.12.2 for more information about mode restrictions due to shared pins.)

Note

The GPP should be accessed only during vertical blank.

The GPP will also connect to an I^2C controller, which in turn can control multiple devices through the I^2C serial bus. The target devices that were the basis for GPP definition include the Philips PCD8584 (I^2C controller), SAA7110, SAA7191B, and SAA7199, and the Brooktree Bt812, Bt819, and Bt855. The GPP also connects to an ICS2595.

2.11 VGA Subsystem

The 21130 powers up with VGA active and the 2DA inactive (at reset, the VGA enable bit in the deep register (GDER <22>, Section 8.5.2) is set). When the 21130 is operating in VGA mode, the PCI address decoders are disabled, and addresses propagate through to the PCI-to-VGA interface, which contains its own address decoders.

Because the VGA controller has ISA characteristics on its system interface, the PCI-to-VGA interface translates PCI protocols, data formats, and addresses into their ISA-like equivalents. The PCI-to-VGA interface is a layer of logic and state machines between the back of the PCI interface and the ISA front end of the VGA controller.

In VGA mode, two RAS signals independently control a 16-bit-wide memory bank. In 2DA mode, the two RAS signals are tied internally and have identical timing to drive 32 or 64 bits of frame buffer DRAMs. (A third RAS signal is active in 2DA mode, if there is a second bank of frame buffer memory.) The VGA controller uses only 32 bits of frame buffer, regardless of the actual memory width.

The VGA memory control, address, and data signals are multiplexed with their equivalents from the 2DA memory controller immediately before the pins. In VGA mode, the VGA controller has complete control of the frame buffer, including display refresh and DRAM refresh functions.

See Section 2.12.2 for more information about mode restrictions due to shared pins.

The 21130 uses the VGA CRT controller (CRTC) for the VGA and 2DA modes of operation. It generates timing for graphics resolutions up to 1280×1024.

2.12 Frame Buffer Memory

The 21130 uses one size of memory device, the 256K $\times$ 16 DRAM with *extended data out* (EDO) capability. This high-bandwidth feature is also known as *hyperpage mode*.

2.12 Frame Buffer Memory

2.12.1 Frame Buffer Configurations

Frame buffer memory is organized as one or two banks of 32-bit or 64-bit memory. The *minimum configuration* uses two DRAMs comprising a frame buffer 32 bits wide and 256K (one bank) deep. The *minimum high-performance configuration* uses four DRAMs in parallel for a 64-bit data path, 256K deep. The *maximum configuration* adds a second bank of 4 DRAMs (8 DRAMs total) for a frame buffer configuration 64 bits wide and 512K deep.

Software can deliberately set frame buffer width to 32 bits when a 64-bit frame buffer is present, but should *not* set width to 64 bits if only a 32-bit frame buffer is present. In the VGA mode of operation, or when the VAFC port is in active use, the frame buffer operates in 32-bit mode, regardless of the actual amount of memory attached to the frame buffer interface. At power-up, the 21130 memory controller is in the 32-bit mode; the CPU must intervene to set it to the 64-bit mode of operation.

2.12.2 Hardware Mode Restrictions

In normal operation, the 86 memory data bus signals represent data and control signals for frame buffer memory cycles. However, the physical pins are shared with other subsystems on the 21130 chip that access the graphics BIOS ROM, optional peripheral chips, and the VAFC.

After the PCI reset signal is asserted, the 21130 is operating with VGA enabled. In VGA mode:

- The VGA feature connector (not VAFC) output can be used.
- ROM can be accessed.
- The lower half of the 64-bit data bus can be used for VGA frame buffer accesses.
- Sixteen-bit VAFC and GPP cycles are not available.

When the 2DA mode with the 64-bit data bus is selected, ROM and GPP cycles are available, and neither 8-bit (feature connector) nor 16-bit VAFC mode is available.

If a 32-bit data bus mode is selected while operating in 2DA mode, either GPP and VAFC modes or ROM and GPP modes are available. (VAFC and ROM are not available simultaneously because they use the same pins.) Table 2–1 summarizes these restrictions and limitations.

Table 2–1 Mode Restrictions

	Functions		
Mode	**Available**	**Not Available**	**Limitations**
VGA	ROM and feature connector (FC)	VAFC, GPP	During VGA mode, only standard FC (not VAFC) output is available and GPP operations are not allowed.
32-bit	VAFC	ROM, GPP	During VAFC mode, ROM reads return undefined data.
32-bit	GPP and VAFC or GPP and ROM	FC	ROM and VAFC are not available simultaneously.
64-bit	GPP and ROM	VAFC, FC	—

2.13 Video Back End

The video timing and cursor control functions provide monitor timing, schedule screen refresh, and provide a 2-bpp cursor during refresh.

2.13.1 Monitor Timing

The video timing function (which is part of the VGA subsystem) provides the timing for the horizontal sync, vertical sync, and blank signals to drive a noninterlaced monitor. The signal edges are specified by the parameters in the video control registers and the VGA CRTC registers. The sync and blank signal output is controlled by the palette and DAC function.

2.13.2 Video Refresh

The video base address is loaded into the refresh address generator at top-of-frame. The refresh address is incremented until the end of the scanline is reached (specified by the scanline width). At the end of the scanline, the scanline increment value is added to the refresh address to determine the starting address of the next scanline. This address is then used with the scanline width to determine the end of the next scanline, and so on.

2.13.3 Pixel Occlusion Bitmap

The 21130's screen refresh logic contains a 256 $\times$ 24 ROM used as a dedicated video palette color LUT and a 256 $\times$ 24 RAM used as a standard graphics palette color LUT. The refresh logic chooses between the graphics and video LUTs on a pixel-by-pixel basis. The selection data is provided by the pixel occlusion bitmap (a 1-bpp choice map) stored in the frame buffer. During horizontal retrace, a scanline's worth of choice information is loaded from the choice map. The choice bits control the palette selection for the pixels on the scanline.

To reduce the amount of off-screen memory required to store the choice map (and also the memory bandwidth required to read the map), the map data for consecutive scanlines are run-length encoded. A field that indicates the number of consecutive scanlines to which the map information is to be applied is stored with each scanline of map information. Therefore, the memory required to store the choice map is scaled according to the complexity of the video window geometry rather than screen size.

2.13.4 Cursor Generation

The video back end monitors which scanline is currently being refreshed. During the horizontal blank time preceding a scanline that intersects the cursor, the video back-end cursor logic generates a request to the memory controller to read a scanline's worth of 2-bpp cursor. Then, at the proper position during the horizontal scan, the video back-end cursor logic drives up to 64 consecutive 2-bit cursor values to the palette and DAC function, synchronously with the video stream and monitor timing.

2.13.5 VAFC Port

The 21130 incorporates an industry-standard VESA advanced feature connector (VAFC) interface, which is an extension of the original VGA feature connector. The original feature connector enables multiple graphics cards to share a common DAC and monitor. The VAFC acts as a local multimedia port for sending or receiving video from another chip or card.

The VAFC port shares pins with the upper half of the frame buffer data bus. Consequently, when moving video data in either direction over this port, the 21130 operates in 32-bit frame buffer mode. Theoretically this reduces frame buffer bandwidth and size, so some display modes that are normally available may not operate while moving video through this port. This is not a problem for applications such as NTSC output, and there is no impact on multimedia video incoming on the PCI. (See Section 2.12.2 for more information about mode restrictions due to shared pins.)

Video arriving through the VAFC port is not stored in the frame buffer, but merges into the video stream before the DAC's video LUT. The video must be in a standard RGB format (rather than YUV).

The VAFC port is controlled by the alternate video control register.

2.13.6 Palette and DAC

The palette and DAC function implements a high-performance, 24-bit, true color RAMDAC with the added capability to select the following as DAC input:

- 24-bit RAM LUT (graphics)
- 24-bit ROM LUT (video)
- 24-bit pixel data (RAM and ROM LUTs are bypassed)

In addition, the palette and DAC function controls the composite blank output; determines whether sync output is separate horizontal sync and vertical sync or composite sync on the green output; and controls cursor output as follows:

- Cursor disabled
- 3-color cursor
- 2-color highlight cursor (Microsoft Windows or XGA cursor)
- 2-color cursor (X Windows cursor)

2.14 Clocks and Clock Control

In addition to the externally supplied PCI clock, the 21130 has two internally generated primary clocks: the memory clock and the pixel clock.

The memory clock is a 66-MHz (nominal) clock to the accelerator section, VGA controller, and memory controller. It is generated by a PLL-based clock generator circuit. The memory clock frequency is programmable and is selected in the PCI clock control register.

The core clock is also used by the accelerator section. It is generated by the memory clock PLL and is one-half the frequency of the memory clock.

The pixel clock is generated by a programmable source, which is based on a second PLL circuit. It can generate pixel clock rates between 8 and 135 MHz. The frequency is selected in the clock control A and B registers (VXCKAR and VXCKBR, Section 8.14.10). Both the memory clock and the pixel clock are derived from the same reference clock, provided by a low-cost 14.31818-MHz crystal.

The pixel clock for video generation can be sourced from an internal PLL circuit or an external ICS2595 device. The pixel clock is driven either by the PLL directly or the VGA controller. The PLL drives the VGA dot clock to the VGA controller where it is divided or not, depending on the specific VGA mode, and returned to the clock generation function as the VGA pixel clock.

2.14 Clocks and Clock Control

The clock register determines whether the VGA dot clock frequency is controlled by the VGA miscellaneous output register or directly by the clock register.

In test mode, either of the two internally generated clocks can be selected as the test clock output. The PCI clock control register (PCCR, Section 8.2.8) selects the pixel clock or memory clock as the test clock source.

3
Pinout

Sections 3.1 through 3.3 list the DECchip 21130 external signals and their associated pins, describe the external signals, and list the signals according to function.

Note

By convention, low-asserted signals carry the suffix **#**. High-asserted signals have no suffix.

3.1 Signal List

Table 3–1 lists the external signals and their associated pins.

Table 3–1 Signal List

Pin	Signal	Pin	Signal	Pin	Signal
Pins 1 through 29					
—	—	10	**memdata<06>**	20	**vsync**
1	**memdata<13>**	11	**memdata<05>**	21	**hsync**
2	**memdata<12>**	12	**memdata<04>**	22	**blank#**
3	**memdata<11>**	13	**memdata<03>**	23	**vafc_vclk**
4	**memdata<10>**	14	**memdata<02>**	24	**grdy**
5	**Vss** (video)	15	**memdata<01>**	25	**vafc_dclk**
6	**Vdd** (video)	16	**memdata<00>**	26	**Vdd** (core0)
7	**memdata<09>**	17	**Vss** (ac0)	27	**Vss** (core0)
8	**memdata<08>**	18	**vafc_en#**	28	**pixclk**
9	**memdata<07>**	19	**evideo#**	29	**pll_test**

(continued on next page)

3.1 Signal List

Table 3–1 (Cont.) Signal List

Pin	Signal	Pin	Signal	Pin	Signal
Pins 30 through 59					
30	**ddc_data**	40	**ref**	50	**dac_Vss**
31	**test_in**	41	**opamp_Vdd**	51	**Vss** (pci0)
32	**pll_Vss**	42	**dac_Vdd**	52	**Vdd** (pci0)
33	**filter**	43	**comp**	53	**pci_inta#**
34	**pll_Vdd**	44	**opamp_Vss**	54	**pci_rst#**
35	**pll_Vss**	45	**blue**	55	**pci_gnt#**
36	**xtal2**	46	**dac_Vss**	56	**pci_req#**
37	**xtal1**	47	**green**	57	**pci_ad<31>**
38	**pll_Vss**	48	**dac_Vss**	58	**pci_ad<30>**
39	**fsadjust**	49	**red**	59	**pci_ad<29>**
Pins 60 through 89					
60	**pci_ad<28>**	70	**Vdd** (pci2)	80	**Vss** (pci3)
61	**Vss** (pci1)	71	**Vss** (pci2)	81	**pci_clk**
62	**Vdd** (pci1)	72	**pci_ad<22>**	82	**Vss** (pci4)
63	**pci_ad<27>**	73	**pci_ad<21>**	83	**pci_cbe<2>#**
64	**pci_ad<26>**	74	**pci_ad<20>**	84	**pci_frame#**
65	**pci_ad<25>**	75	**pci_ad<19>**	85	**pci_irdy#**
66	**pci_ad<24>**	76	**pci_ad<18>**	86	**pci_trdy#**
67	**pci_cbe<3>#**	77	**pci_ad<17>**	87	**pci_devsel#**
68	**pci_idsel**	78	**pci_ad<16>**	88	**pci_stop#**
69	**pci_ad<23>**	79	**Vdd** (pci3)	89	**pci_perr**
Pins 90 through 119					
90	**pci_serr**	100	**pci_ad<10>**	110	**pci_ad<03>**
91	**pci_par**	101	**pci_ad<09>**	111	**pci_ad<02>**
92	**Vdd** (pci4)	102	**pci_ad<08>**	112	**pci_ad<01>**
93	**Vss** (pci5)	103	**Vdd** (pci5)	113	**pci_ad<00>**
94	**pci_cbe<1>#**	104	**Vss** (pci6)	114	**Vdd** (pci6)
95	**pci_ad<15>**	105	**pci_cbe<0>#**	115	**Vss** (pci7)
96	**pci_ad<14>**	106	**pci_ad<07>**	116	**gp_cs#**
97	**pci_ad<13>**	107	**pci_ad<06>**	117	**gp_reset#**
98	**pci_ad<12>**	108	**pci_ad<05>**	118	**gp_int**
99	**pci_ad<11>**	109	**pci_ad<04>**	119	**gp_stb#**

(continued on next page)

Table 3–1 (Cont.) Signal List

Pin	Signal	Pin	Signal	Pin	Signal
Pins 120 through 149					
120	**rom_ce#**	130	**memdata<58>**	140	**memdata<49>**
121	**vafc_data<0>**	131	**memdata<57>**	141	**memdata<48>**
122	**vafc_data<1>**	132	**memdata<56>**	142	**Vss** (dc0)
123	**vafc_data<2>**	133	**Vss** (ac1)	143	**Vdd** (dc0)
124	**Vdd** (ac0)	134	**memdata<55>**	144	**memdata<47>**
125	**memdata<63>**	135	**memdata<54>**	145	**memdata<46>**
126	**memdata<62>**	136	**memdata<53>**	146	**memdata<45>**
127	**memdata<61>**	137	**memdata<52>**	147	**memdata<44>**
128	**memdata<60>**	138	**memdata<51>**	148	**memdata<43>**
129	**memdata<59>**	139	**memdata<50>**	149	**memdata<42>**
Pins 150 through 179					
150	**memdata<41>**	160	**memdata<32>**	170	**memaddr<1>**
151	**memdata<40>**	161	**Vdd** (dc1)	171	**memaddr<0>**
152	**memdata<39>**	162	**Vss** (dc1)	172	**Vss** (ac2)
153	**Vdd** (ac1)	163	**memaddr<8>**	173	**oeb#**
154	**memdata<38>**	164	**memaddr<7>**	174	**cas<7>#**
155	**memdata<37>**	165	**memaddr<6>**	175	**cas<6>#**
156	**memdata<36>**	166	**memaddr<5>**	176	**cas<5>#**
157	**memdata<35>**	167	**memaddr<4>**	177	**cas<4>#**
158	**memdata<34>**	168	**memaddr<3>**	178	**cas<3>#**
159	**memdata<33>**	169	**memaddr<2>**	179	**Vdd** (core1)
Pins 180 through 208					
180	**Vss** (core1)	190	**memdata<30>**	200	**memdata<21>**
181	**cas<2>#**	191	**memdata<29>**	201	**memdata<20>**
182	**cas<1>#**	192	**memdata<28>**	202	**memdata<19>**
183	**cas<0>#**	193	**memdata<27>**	203	**Vss** (ac4)
184	**ras<2>#**	194	**memdata<26>**	204	**memdata<18>**
185	**ras<1>#**	195	**Vdd** (ac2)	205	**memdata<17>**
186	**ras<0>#**	196	**memdata<25>**	206	**memdata<16>**
187	**Vss** (ac3)	197	**memdata<24>**	207	**memdata<15>**
188	**wrb#**	198	**memdata<23>**	208	**memdata<14>**
189	**memdata<31>**	199	**memdata<22>**	—	—

3.2 Signal Descriptions

Table 3–2 describes the function of each external signal in alphabetical order. The following signal type abbreviations are used in Table 3–2:

I	input
I/O	bidirectional
O	output
P	power

Table 3–2 Signal Description

Signal	Type	Description
blank#	O	Blank — when asserted, this output signal indicates that the 21130 monitor timing is in the inactive (blanked) period of the monitor video signal. This signal is generated by the 21130 VGA CRTC.
blue	O	The blue signal to the monitor. Drives a doubly terminated 75-*Ω* cable.
cas<7:0>#	O	Column address strobe — these eight output signals latch the DRAM column addresses into frame buffer memory. They are also used as byte-access controls.
comp	I	Compensation — DAC external compensation. A 0.1-µF ceramic capacitor must be used to bypass this pin to **dac_Vdd**. Place the capacitor as close as possible to the 21130.
dac_Vdd	P	Analog +5-V supply pin, for the DAC analog circuitry.
dac_Vss	P	Three analog ground pins, for the DAC analog circuitry.
ddc_data	I/O	Display data channel data — this is a bidirectional signal to the display monitor. The display data channel (DDC) and other VESA standards specify signal protocols and file formats for transferring information on monitor capabilities to the graphics controller. See the VESA *Display Data Channel Standard, Version 1.0, Revision 0* for more information about the DDC.
evideo#	I	Enable external video data — an external video chip or card asserts this input signal to indicate that the video source is driving the **vafc_p<0:15>** signals into the 21130. The 21130 tristates (disables) the shared **vafc_p<15:00>** data path pins in order to receive video data. When **evideo#** is pulled high (passive, external or internal pullup resistor), the 21130 can drive graphics output data to the video system on the **vafc_p<15:00>** pins. The **evideo#** signal also controls the direction of external transceiver buffers.

(continued on next page)

Table 3–2 (Cont.) Signal Description

Signal	Type	Description
fsadjust	I	Full-scale adjust — DAC external resistor connection. For doubly terminated 75- Ω loads (RS-343A), a value of 147 Ω connected to **dac_Vss** is recommended.
gp_adr<16:0>	O	GPP address — these 17 peripheral address output signals share the **memdata<16:0>** pins.
gp_cs#	O	GPP chip select — this output signal enables external devices. The **gp_cs#** signal also determines what signal is present on the **gp_stb#** pin. If **gp_cs#** is asserted, the **gp_stb#** signal is on the **gp_stb#** pin. If **gp_cs#** is not asserted, the **rom_oe#** signal is on the **gp_stb#** pin.
gp_data<7:0>	I/O	GPP data — these eight bidirectional data path signals are multiplexed on the **memdata<25:18>** pins.
gp_int#	I	GPP interrupt — peripherals that require service from the CPU can asynchronously assert this interrupt input signal.
gp_rdsel#	O	GPP read select — this output signal is asserted low to specify a read cycle. It shares the **memdata<26>** pin.
gp_reset#	O	GPP reset — this output signal is a buffered version of the PCI reset signal (**pci_rst#**), to allow external device reset.
gp_stb#	O	GPP strobe — this output signal is asserted low. It indicates 21130 write data to the GPP is valid or the 21130 is ready to accept GPP read data. The **rom_oe#** signal shares the **gp_stb#** pin. The **gp_cs#** signal determines what signal is present on the **gp_stb#** pin. If **gp_cs#** is asserted, the **gp_stb#** signal is on the **gp_stb#** pin. If **gp_cs#** is not asserted, the **rom_oe#** signal is on the **gp_stb#** pin.
gp_wrsel#	O	GPP write select — this output signal is asserted low to specify a write cycle. It shares the **memdata<27>** pin.
green	O	The green signal to the monitor. Drives a doubly terminated 75-Ω cable.
grdy	O	Graphics ready — this output signal indicates to the external video source that the 21130 is ready to latch the data on the **vafc_p<0:15>** pins.
hsync	O	Horizontal sync — this output signal is sent to an external video chip or card as the horizontal display framing reference signal. This signal is generated by the 21130 VGA CRTC.

(continued on next page)

3.2 Signal Descriptions

Table 3–2 (Cont.) Signal Description

Signal	Type	Description
memaddr<8:0>	O	Memory address — these output signals comprise the multiplexed 9-bit row and 9-bit column memory address.
memdata<63:0>	I/O	Memory data — these signals comprise the bidirectional, 64-bit frame buffer data path.
oeb#	O	Output enable — this output signal is asserted low on a read cycle to enable the DRAM data path outputs.
opamp_Vdd	P	Analog +5-V supply pin, for the DAC op amp circuitry.
opamp_Vss	P	Analog ground pin, for the DAC op amp circuitry.
pci_ad<31:0>	I/O	PCI address and data — these bidirectional signals are multiplexed on the same 32 pins of the PCI bus. A PCI bus transaction consists of an address phase followed by one or more data phases. The **pci_cbe<3:0>#** signals identify the transaction type (for example, read or write) during the address phase. The 21130 supports and fully decodes 32-bit addresses.
pci_cbe<3:0>#	I/O	PCI cycle command and byte enable — these tristate signals are multiplexed on the same four pins of the PCI bus. They define the bus command during the address phase of a transaction, and specify the byte enables during the data phase. The byte enables are valid for all cycles of a data phase, and determine which byte lanes carry meaningful data.
pci_clk	I	PCI clock — the system supplies this 33-MHz (nominal) clock input to PCI peripherals for timing all PCI transactions. All PCI signals, except **pci_rst#** and **pci_inta#**, are sampled on the rising edge of the **pci_clk** signal.
pci_devsel#	I/O	PCI device select — the target of a PCI transaction asserts this signal when it detects an address matching its programmed address space.
pci_frame#	I/O	PCI cycle frame — the PCI master drives this tristate signal to indicate the beginning and duration of an access. The **pci_frame#** signal is asserted to indicate the start of a bus transaction; remains asserted while data transfers continue; and is deasserted to signal the final data phase.
pci_gnt#	I	PCI bus grant — the 21130 monitors this input signal to determine when it has been granted the bus for DMA transfers. The **pci_gnt#** signal is a unique signal between a PCI agent and the central PCI arbiter.

(continued on next page)

Table 3–2 (Cont.) Signal Description

Signal	Type	Description
pci_idsel	I	PCI initialization device select — this input signal is asserted when the 21130 has been selected for a configuration transaction. This unique chip select signal is used during PCI configuration read and write transactions. Typically, the **pci_idsel** signal for each PCI agent is attached (through a load-limiting resistor) to a different address and data line (**pci_ad<31:00>**). For configuration transactions, only one address signal is asserted at a 1 level during the address phase, effectively providing a unique chip select signal for each PCI agent.
pci_inta#	O	PCI interrupt request — the 21130 asserts this output signal to request service from the CPU. Because it is open drain, the **pci_inta#** signal can be shared with other devices on the same module or elsewhere in the system. An external pullup resistor is required to maintain the signal's deasserted state when it is not being driven low. Two 21130 interrupt sources are ORed together internally to generate the **pci_inta#** signal: the video timing generator (end-of-frame interrupt) and the external interrupt signal on the generic peripheral port (**gp_int#**). An interrupt service can identify the interrupt source by reading the 21130 interrupt status register.
pci_irdy#	I/O	PCI initiator ready — this tristate signal indicates the transaction initiator's ability to complete the current data phase of a transaction. The successful transfer of data in any data cycle is indicated when the **pci_irdy#** and **pci_trdy#** signals are asserted together. During a write, the **pci_irdy#** signal indicates valid data is present on the **pci_ad<31:00>** pins; during a read, it indicates the master is ready to accept data. The bus stalls (inserts wait cycles) until **pci_irdy#** and **pci_trdy#** are asserted together. During DMA read cycles, the 21130 is the bus master and controls the **pci_irdy#** signal.
pci_par	I/O	PCI even parity bit — this PCI signal is the even parity bit used across the 36 bits of **pci_ad<31:00>** and **pci_cbe<3:0>#**. For address parity, the tristate **pci_par** signal is asserted one PCI clock after the address phase; for data phases, the **pci_par** signal is stable and valid one PCI clock after either the **pci_trdy#** signal is asserted on a read or the **pci_irdy#** signal is asserted on a write. The master (initiator) drives the **pci_par** signal for address and write data phases, and the target drives the **pci_par** signal for read data phases.
pci_perr#	I/O	PCI parity error — the 21130 asserts this tristate signal when it detects a parity error while it is receiving data. The **pci_perr#** signal is asserted two PCI clocks after the data with the detected error was on the bus.

(continued on next page)

3.2 Signal Descriptions

Table 3–2 (Cont.) Signal Description

Signal	Type	Description
pci_req#	O	PCI request — the 21130 asserts this tristate signal when it requires the PCI bus for a DMA transfer. The **pci_req#** signal is a unique signal between a PCI agent and the central PCI arbiter.
pci_rst#	I	PCI reset — when asserted, this input signal brings the 21130 to a known initialized state. The **pci_rst#** signal resets the PCI interface, 2D accelerator, and internal VGA controller. The **pci_rst#** signal is also passed through the 21130 to the generic peripheral port.
pci_serr#	O	PCI system error — the 21130 asserts this output signal when it detects a parity error during the command/address phase of a transaction. Multiple devices can assert the **pci_serr#** signal on detection of this class of error. An external pullup resistor (to be provided on the system motherboard) is required to return the **pci_serr#** signal to its deasserted state.
pci_stop#	I/O	PCI stop — the target of the current PCI transaction asserts this tristate signal to request that the master stop the transaction.
pci_trdy#	I/O	PCI target ready — this tristate signal indicates the target's ability to complete the current data phase of a transaction. The successful transfer of data in any data cycle is indicated when the **pci_irdy#** and **pci_trdy#** signals are asserted together. The 21130 controls the **pci_trdy#** signal when it is the target of a PCI bus transaction. The 21130 asserts the **pci_trdy#** signal during a write to indicate it is taking the data and during a read when valid read data is present on the bus.
pixclk	I	Backup pixel clock — this input signal is the optional external pixel clock from an ICS2595 device, if a backup pixel clock is used.
pll_filter	I	This pin connects to two external capacitors which are connected to analog Vdd (**pll_Vdd**).
pll_test	O	This clock-test output signal allows external access to PLL clock circuit signals. In test mode, the PCI clock control register determines whether the **pll_test** pin is connected to the memory clock or the pixel clock. This pin can also be used for the **ddc_clk** (see VIVVR bit <10>, Section 8.7.2).
pll_Vdd	P	Analog +5 V for the PLL circuits.
pll_Vss	P	Three analog ground pins for the PLL circuits.
ras<2:0>#	O	Row address strobe — these three output signals latch the DRAM row addresses into frame buffer memory. They are also used as bank-select controls.

(continued on next page)

Table 3–2 (Cont.) Signal Description

Signal	Type	Description
red	O	The red signal to the monitor. Drives a doubly terminated 75-Ω cable.
ref	I	Reference — external voltage reference. This pin must be supplied with a 1.2-V (nominal) source. Decouple this pin to **dac_Vss** with a 0.1-µF capacitor located as close as possible to the 21130.
rom_adr<17:0>	O	ROM address — these 18 output signals share the **memdata<49:32>** pins. These pins are also shared with any peripheral chips attached to the GPP that need an address when accessed.
rom_ce#	O	ROM chip enable — this output signal enables ROM read or write accesses (write accesses are feasible only with a flash ROM).
rom_d<7:0>	I/O	ROM data path — these eight bidirectional signals share the **memdata<57:50>** pins.
rom_oe#	O	ROM output enable — this output signal is asserted on ROM read cycles. The **rom_oe#** signal shares the **gp_stb#** pin. The **gp_cs#** signal determines what signal is present on the **gp_stb#** pin. If **gp_cs#** is asserted, the **gp_stb#** signal is on the **gp_stb#** pin. If **gp_cs#** is not asserted, the **rom_oe#** signal is on the **gp_stb#** pin.
rom_we#	O	ROM write enable — this output signal is asserted during ROM write cycles to change the contents of a location in the ROM. It shares the **memdata<58>** pin.
test_in	I	This input signal places the 21130 in test mode. The test result data is output on the **pll_test** pin.
vafc_dclk	O	VAFC dot clock — this continuous master clock output signal is driven from the 21130 to an external video chip or card. Its frequency is determined by the alternate video control register and is the same as, or a submultiple of, the pixel clock frequency.
vafc_en#	O	VAFC enable — this output signal is asserted low to enable external 16-bit transceivers to drive data from the 21130 to the VAFC connector.
vafc_p<0:15>	I/O	VAFC port — 16-bit, bidirectional, tristate pixel data bus. The **vafc_p<0:2>** pins are not shared. The **vafc_p<3:15>** pins share the **memdata<63:51>** pins (the VAFC pin-number order is deliberately reversed).
vafc_vclk	I	VAFC video clock — this continuous master clock input signal is driven from an external video chip or card to the 21130. This is a delayed version of the **vafc_dclk** signal and is used as a reference for video data and associated handshake signals.

(continued on next page)

3.2 Signal Descriptions

Table 3–2 (Cont.) Signal Description

Signal	Type	Description
Vdd	P	In addition to **dac_Vdd** and **pll_Vdd**, eight Vdd pins supply +5 V to the core logic and I/O pad drivers, as follows: **Vdd** (video) — Video clock 5-V supply **Vdd** (ac<2:0>) — I/O 5-V ac supply **Vdd** (dc<1:0>) — I/O 5-V dc supply **Vdd** (core<1:0>) — Core logic 5-V supply
Vss	P	In addition to **dac_Vss** and **pll_Vss**, ten Vss pins supply ground to the core logic and I/O pad drivers, as follows: **Vss** (video) — Video clock ground **Vss** (ac<4:0>) — I/O ac ground **Vss** (dc<1:0>) — I/O dc ground **Vss** (core<1:0>) — Core logic ground
vsync	O	Vertical sync — this output signal is sent to an external video chip or card as the vertical display framing reference signal. This signal is generated by the 21130 VGA CRTC.
wrb#	O	Write enable — this output signal is asserted low to perform a DRAM write cycle.
xtal1	I	Crystal 1 — this input signal is one of two 14.31818 MHz crystal inputs. It is the reference frequency for the memory clock PLL circuit, and also for the internal video clock PLL circuit that generates the programmable pixel clock and subsequent video timing.
xtal2	I	Crystal 2 — this input signal is one of two 14.31818 MHz crystal inputs. It serves as an input pin for an optional backup memory clock generator. In backup mode, it accepts normal 5-V CMOS signals.

3.3 Signals by Function and Direction

Table 3–3 provides a quick reference to the external signals, which are grouped by function. The following value at reset abbreviations are used in Table 3–3:

NA	Not applicable
TS	Tristate
OD	Open drain
DH	Driven, high
DL	Driven, low
DI	Driven, indeterminate
SH	Shared

Table 3–3 Signals by Function

Signal	Qty	Type	Function	Value at Reset
PCI Interface				
pci_idsel	1	I	PCI initialization device select	NA
pci_gnt#	1	I	PCI DMA grant	NA
pci_rst#	1	I	PCI reset	NA
pci_clk	1	I	PCI clock	NA
pci_ad<31:0>	32	I/O	PCI address or data	TS
pci_cbe<3:0>#	4	I/O	PCI command and byte enable	TS
pci_frame#	1	I/O	PCI frame	TS
pci_irdy#	1	I/O	PCI initiator ready	TS
pci_trdy#	1	I/O	PCI target ready	TS
pci_devsel#	1	I/O	PCI device select	TS
pci_stop#	1	I/O	PCI stop transaction	TS
pci_perr#	1	I/O	PCI parity error	TS
pci_par	1	I/O	PCI parity	TS
pci_req#	1	O	PCI DMA request	TS
pci_inta#	1	O	PCI interrupt	OD
pci_serr#	1	O	PCI system error	OD
Vdd (pci<6:0>)	7	P	PCI I/O 5-V supply	NA

(continued on next page)

3.3 Signals by Function and Direction

Table 3–3 (Cont.) Signals by Function

Signal	Qty	Type	Function	Value at Reset
PCI Interface				
Vss (pci<7:0>)	8	P	PCI I/O ground	NA
Frame Buffer Interface				
memdata<63:0>	64	I/O	Memory data	DI
memaddr<8:0>	9	O	Memory address	DI
cas<7:0>#	8	O	Column address strobe	DH
ras<2:0>#	3	O	Row address strobe	DH
oeb#	1	O	Output enable	DH
wrb#	1	O	Write enable	DL
GPP and ROM Interface				
gp_int#	1	I	Generic peripheral interrupt	NA
gp_data<7:0>[1]	(8)	I/O	Generic peripheral data	SH
gp_adr<16:0>[2]	(17)	O	Generic peripheral address	SH
gp_rdsel#[3]	(1)	O	Generic peripheral read select	SH
gp_wrsel#[4]	(1)	O	Generic peripheral write select	SH
gp_cs#	1	O	Generic peripheral chip select	TS[5]
gp_reset#	1	O	Generic peripheral reset	DL
gp_stb#	1	O	Generic peripheral strobe	TS[5]
rom_d<7:0>[6]	(8)	I/O	ROM data path	SH
rom_adr<17:0>[7]	(18)	O	ROM address	SH
rom_ce#	1	O	ROM chip enable	DH

[1]The **gp_data<7:0>** signals share the **memdata<25:18>** pins.

[2]The **gp_adr<16:0>** signals share the **memdata<16:0>** pins.

[3]The **gp_rdsel#** signal shares the **memdata<26>** pin.

[4]The **gp_wrsel#** signal shares the **memdata<27>** pin.

[5]At reset, the **gp_cs#** and **gp_stb#** signals are inputs and are sampled.

[6]The **rom_d<7:0>** signals share the **memdata<57:50>** pins.

[7]The **rom_adr<17:0>** signals share the **memdata<49:32>** pins.

(continued on next page)

Table 3–3 (Cont.) Signals by Function

Signal	Qty	Type	Function	Value at Reset
GPP and ROM Interface				
rom_oe#[8]	(1)	O	ROM output enable	SH
rom_we#[9]	(1)	O	ROM write enable	SH
VGA and VAFC Video Port Interface				
ddc_data	1	I/O	Display data channel	TS
evideo#	1	I	Enable external video data	NA
vafc_vclk	1	I	VAFC video clock	NA
vafc_p<0:15>[10]	(16)	I/O	Port	AH
vafc_en#	1	O	VAFC data enable	DL
vafc_dclk	1	O	VAFC dot clock	Pixel clock
grdy	1	O	Graphics device ready	DL
blank#	1	O	Composite video blank	DI
RGB-to-Monitor Interface				
hsync	1	O	Horizontal video sync	DI
vsync	1	O	Vertical video sync	DH
red	1	O	Red analog output	DI
green	1	O	Green analog output	DI
blue	1	O	Blue analog output	DI
DAC Interface				
comp	1	I	DAC external compensation	NA
fsadjust	1	I	DAC external resistor	NA
ref	1	I	DAC external voltage reference	NA
dac_Vdd	1	P	DAC 5-V supply	NA
dac_Vss	3	P	DAC ground	NA

[8]The **rom_oe#** signal shares the **gp_stb#** pin.

[9]The **rom_we#** signal shares the **memdata<58>** pin.

[10]The **vafc_p<3:15>** signals share the **memdata<63:51>** pins.

(continued on next page)

3.3 Signals by Function and Direction

Table 3–3 (Cont.) Signals by Function

Signal	Qty	Type	Function	Value at Reset
DAC Interface				
opamp_Vdd	1	P	DAC op amp 5-V supply	NA
opamp_Vss	1	P	DAC op amp ground	NA
Clock Interface				
xtal1	1	I	Crystal input	Reference clock
xtal2	1	I	Crystal input/memory clock	NA
pixclk	1	I	Backup pixel clock	NA
pll_filter	1	I	External filter capacitors	NA
pll_test	1	O	Clock test output	DL
pll_Vdd	1	P	PLL 5-V supply	NA
pll_Vss	3	P	PLL ground	NA
Miscellaneous Test Pins				
test_in	1	I	Test input	NA
Miscellaneous Power Pins				
Vdd (video)	1	P	Video clock 5-V supply	NA
Vss (video)	1	P	Video clock ground	NA
Vdd (ac<2:0>)	3	P	I/O 5-V ac supply	NA
Vss (ac<4:0>)	5	P	I/O ac ground	NA
Vdd (dc<1:0>)	2	P	I/O 5-V dc supply	NA
Vss (dc<1:0>)	2	P	I/O dc ground	NA
Vdd (core<1:0>)	2	P	Core logic 5-V supply	NA
Vss (core<1:0>)	2	P	Core logic ground	NA

Table 3–4 lists external signals and their active levels, which are grouped by direction.

3.3 Signals by Function and Direction

Table 3–4 Signals and Active Levels by Direction

Signal	Active Level	Signal	Active Level	Signal	Active Level
Input Signals					
comp	High	**pci_rst#**	Low	**test_in**	High
evideo#	Low	**pixclk**	High	**vafc_vclk**	High
gp_int#	Low	**pll_filter**	High	**xtal1**	High
pci_clk	High	**ref**	High	**xtal2**	High
pci_idsel	High	**fsadjust**	High		
Input/Output Signals					
ddc_data	High	**pci_devsel#**	Low	**pci_stop#**	Low
gp_data<7:0>	High	**pci_frame#**	Low	**pci_trdy#**	Low
memdata<63:0>	High	**pci_irdy#**	Low	**rom_d<7:0>**	High
pci_ad<31:0>	High	**pci_par**	High	**vafc_p<0:15>**	High
pci_cbe<3:0>#	Low	**pci_perr#**	Low		
Output Signals					
blank#	Low	**grdy**	High	**red**	High
blue	High	**hsync**	High	**rom_adr<17:0>**	High
cas<7:0>#	Low	**memaddr<8:0>**	High	**rom_ce#**	Low
gp_adr<16:0>	High	**oeb#**	Low	**rom_oe#**	Low
gp_cs#	Low	**pci_inta#**	Low	**rom_we#**	Low
gp_rdsel#	Low	**pci_req#**	Low	**vafc_dclk**	High
gp_reset#	Low	**pci_serr#**	Low	**vafc_en#**	Low
gp_stb#	Low	**pll_test**	High	**vsync**	High
gp_wrsel#	Low	**ras<2:0>#**	Low	**wrb#**	Low
green	High				

4

Electrical Specifications

Sections 4.1 through 4.6 specify the following:

- PCI electrical conformance
- Absolute maximum ratings
- Normal operating conditions
- Supply current and power dissipation
- The dc and ac specifications

4.1 PCI Electrical Specification Conformance

The DECchip 21130 PCI pins conform to the basic set of PCI electrical specifications in the *PCI Local Bus Specification, Revision 2.0,* including:

- Standard signaling

  Logic levels follow standard TTL thresholds to accommodate PCI drivers and receivers implemented with existing CMOS and TTL devices and processes.

- 33-10 support

  The 21130 supports a 33-MHz interconnection of up to ten PCI devices.

See the *PCI Local Bus Specification, Revision 2.0* for a complete description of the PCI I/O protocol and pin ac specifications.

4.2 Absolute Maximum Ratings

Table 4–1 lists the absolute maximum ratings for the 21130. These are stress ratings only; extended exposure to the maximum ratings may affect the reliability of the device.

4.2 Absolute Maximum Ratings

Table 4–1 Absolute Maximum Ratings

Parameter	Minimum	Maximum
Storage temperature range	–55°C	+150°C
Supply voltage **Vdd**	–1.0 V	+7.0 V
dc voltage on any pin	–1.0 V	**Vdd** + 1.0 V

4.3 Normal Operating Conditions

Table 4–2 lists the normal operating conditions for the 21130.

Table 4–2 Normal Operating Conditions

Parameter	Minimum	Maximum
Supply voltage **Vdd**	4.5 V	5.5 V
Power dissipation	—	2.5 W *

*Airflow is required when power dissipation exceeds 2.5 W.

4.4 Supply Current and Power Dissipation

The supply current and power dissipation are as follows:

Idd	500 mA (maximum)
Power	2.5 W (maximum with no airflow)

4.4.1 Test Conditions

The supply current and power dissipation test conditions are as follows:

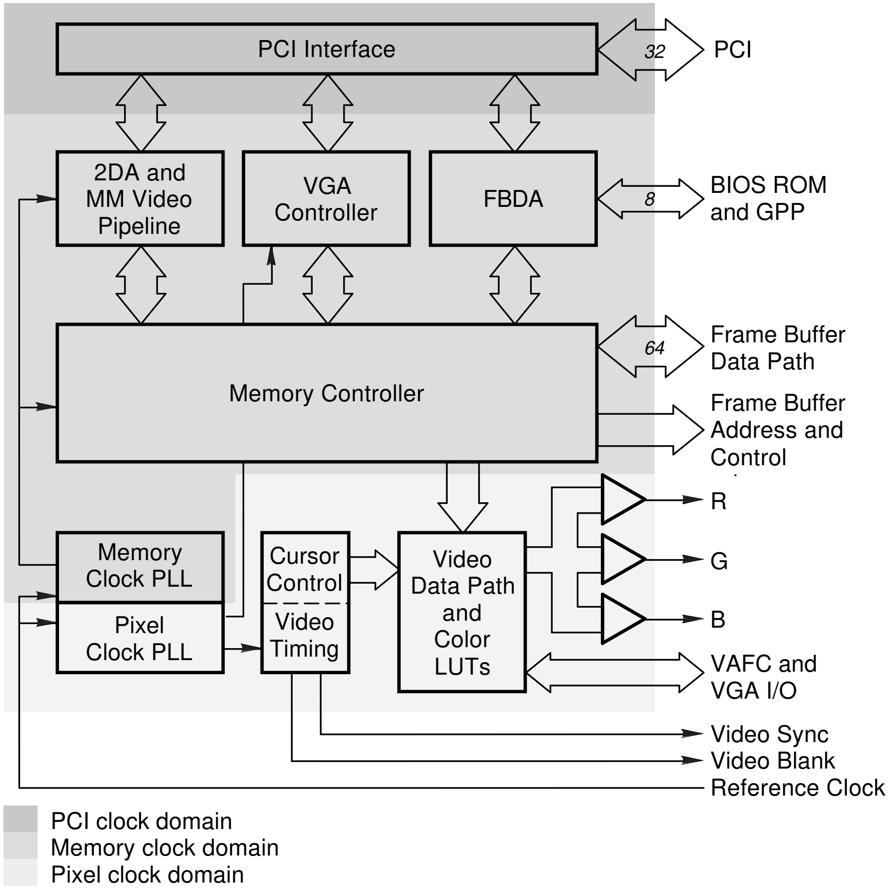
Vdd	5.0 V
Memory clock frequency	80 MHz (55°C ambient rating with no airflow)
Pixel clock frequency	135 MHz (55°C ambient rating with no airflow)

4.5 dc Specifications

Table 4–3 lists the pin characteristics. The pin output drivers are specified to produce TTL signaling levels; however, they are implemented as CMOS drivers and, as a result, actually drive the pins through the full voltage range (rail-to-rail).

Figure 4–1 shows the various clock domains.

Figure 4–1 Clock Domains



4.5 dc Specifications

Table 4–3 Pin Characteristics

Signals	Clock Domain	Type	Signal Level	Notes
gp_int#	Async	I	TTL	—
test_in	Async	I	TTL	—
evideo#	Async	I	TTL	—
vafc_en#	Async	O	TTL	—
pci_rst#	Async PCI	I	TTL	—
pci_inta#	Async PCI	O	TTL	❶
pci_gnt#	**pci_clk**	I	TTL	—
pci_idsel	**pci_clk**	I	TTL	—
pci_clk	**pci_clk**	I	TTL	—
pci_ad<31:0>	**pci_clk**	I/O	TTL	—
pci_cbe<3:0>#	**pci_clk**	I/O	TTL	—
pci_devsel#	**pci_clk**	I/O	TTL	—
pci_frame#	**pci_clk**	I/O	TTL	—
pci_irdy#	**pci_clk**	I/O	TTL	—
pci_par	**pci_clk**	I/O	TTL	—
pci_stop#	**pci_clk**	I/O	TTL	—
pci_trdy#	**pci_clk**	I/O	TTL	—
pci_req#	**pci_clk**	O	TTL	—
xtal1	**mem_clk**	I	CMOS	—
xtal2	**mem_clk**	I	CMOS	—
memdata<63:0>	**mem_clk**	I/O	TTL	—
gp_data<7:0>	**mem_clk**	I/O	TTL	—
memaddr<8:0>	**mem_clk**	O	TTL	—
cas<7:0>#	**mem_clk**	O	TTL	—
gp_adr<16:0>	**mem_clk**	O	TTL	—
gp_rdsel#	**mem_clk**	O	TTL	—
gp_wrsel#	**mem_clk**	O	TTL	—
gp_cs#	**mem_clk**	O	TTL	—
gp_reset#	**mem_clk**	O	TTL	—
gp_stb#	**mem_clk**	O	TTL	—
grdy	**mem_clk**	O	TTL	—
oeb#	**mem_clk**	O	TTL	—
ras<2:0>#	**mem_clk**	O	TTL	—
rom_ce#	**mem_clk**	O	TTL	—
rom_oe#	**mem_clk**	O	TTL	—
rom_we#	**mem_clk**	O	TTL	—
wrb#	**mem_clk**	O	TTL	—

(continued on next page)

Table 4–3 (Cont.) Pin Characteristics

Signals	Clock Domain	Type	Signal Level	Notes
pix_clk	**pix_clk**	I	CMOS	—
pll_test	**pix_clk**	O	TTL	—
vafc_vclk	**pix_clk**	I	TTL	—
ddc_data	**pix_clk**	I/O	CMOS	—
vafc_p<0:15>	**pix_clk**	I/O	TTL	—
vafc_dclk	**pix_clk**	O	TTL	—
blank#	**pix_clk**	O	TTL	—
hsync	**pix_clk**	O	TTL	—
vsync	**pix_clk**	O	TTL	—
red	**pix_clk**	O	—	❷
green	**pix_clk**	O	—	❷
blue	**pix_clk**	O	—	❷

Notes

❶ Pins are driven by an open-drain output driver.

❷ Pins are driven by analog outputs.

4.5.1 Operating Specifications

Table 4–4 lists the functional operating dc parameters for the 21130 under normal operating conditions. The normal operating conditions are specified in Table 4–2.

Note

In Table 4–4, currents into the chip (chip sinking) are denoted as positive (+) current. Currents from the chip (chip sourcing) are denoted as negative (−) current.

4.5 dc Specifications

Table 4–4 dc Parameters

Symbol	Parameter	Minimum	Maximum	Unit	Notes
Vilc	Low-level input voltage for CMOS-level inputs	–0.5	0.3 × **Vdd**	V	—
Vihc	High-level input voltage for CMOS-level inputs	0.7 × **Vdd**	**Vdd** + 0.5 V	V	—
Vilt	Low-level input voltage for TTL-level inputs	–0.5	0.8	V	—
Viht	High-level input voltage for TTL-level inputs	2.0	**Vdd** + 0.5 V	V	—
Vol	Low-level output voltage	—	0.4	V	❶
Voh	High-level output voltage	2.4	—	V	❷
Ioz	Tristate leakage current	–10	+10	µA	—
Cin	Input capacitance	—	6	pF	❸
Co	I/O or output-only pin capacitance	—	7	pF	❸

Notes

❶ Iol = *PCI Local Bus Specification, Revision 2.1* for **pci_clk** domain outputs.
Iol = +8 mA for **memaddr<8:0>**, **oeb#**, **wrb#**, and **vafc_dclk**.
Iol = +4 mA for all other.

❷ Ioh = *PCI Local Bus Specification, Revision 2.1* for **pci_clk** domain outputs.
Ioh = –8 mA for **memaddr<8:0>**, **oeb#**, **wrb#**, and **vafc_dclk**.
Ioh = –4 mA for all other outputs.

❸ Cin = *PCI Local Bus Specification, Revision 2.1* for PCI pins.
Co = *PCI Local Bus Specification, Revision 2.1* for PCI pins.

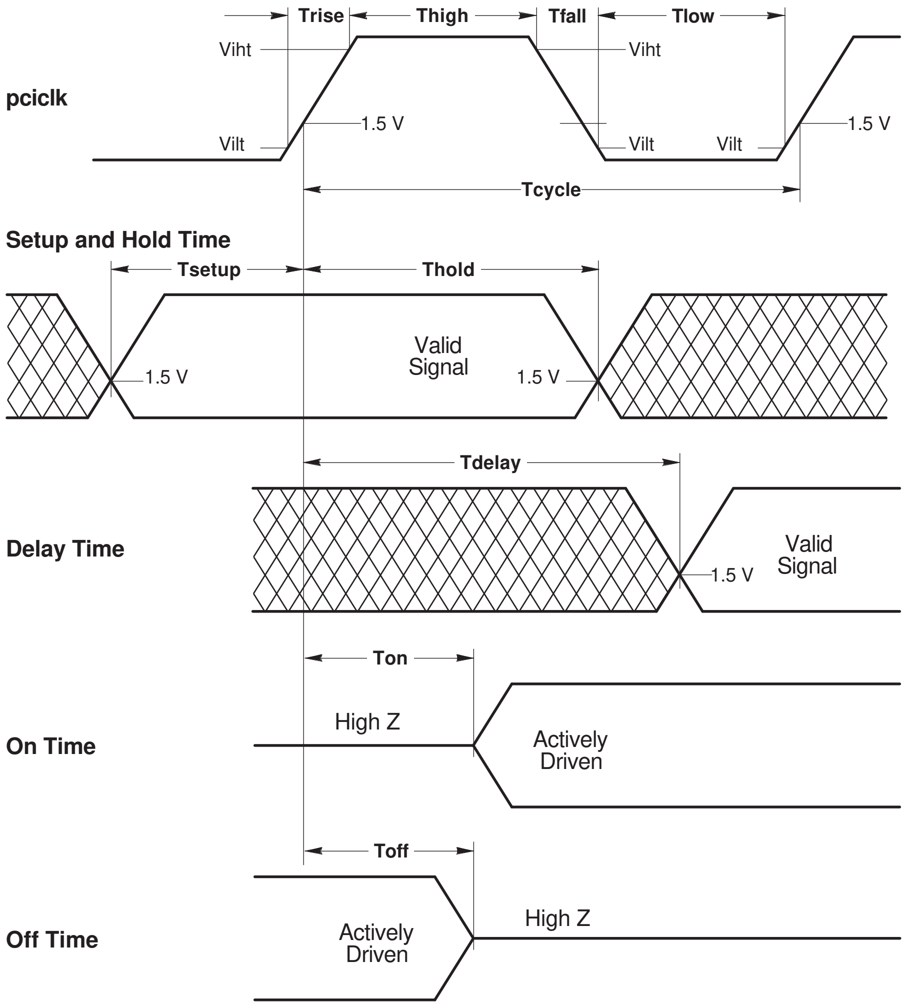
4.6 ac Specifications

The ac specifications consist of input requirements and output responses. The input requirements include setup and hold times, pulse widths, and high and low times. Output responses are delays from clock to signal. The ac specifications are defined separately for each clock domain within the 21130. (See Table 4–3 for a list of signals and their respective clock domains.) All ac specifications apply to the 21130 under normal operating conditions (Table 4–2).

4.6.1 Parameters for PCI Clock Domain Signals

Figure 4–2 shows the ac parameter measurements for signals in the PCI clock domain, and Table 4–5 specifies the parameter values.

Figure 4–2 PCI Clock Domain Signal ac Parameter Measurements



4.6 ac Specifications

Table 4–5 PCI Clock Domain Signal ac Parameters

Symbol	Parameter	Signals	Minimum	Maximum	Unit
Tcycle	Clock cycle time	**pci_clk**	30	—	ns
Thigh	Clock high time	**pci_clk**	12	—	ns
Tlow	Clock low time	**pci_clk**	12	—	ns
Trise	Clock rise time	**pci_clk**	—	2	ns
Tfall	Clock fall time	**pci_clk**	—	2	ns
Trst	Reset low pulse width *	**pci_rst#**	1	—	ms
Trstclk	Clock active time to end of reset	**pci_rst#**	100	—	µs
Tdelay	Clock to signal valid delay†	**pci_ad<31:0>**	2	11	ns
		pci_cbe<3:0>#	2	11	ns
		pci_frame#	2	11	ns
		pci_trdy#	2	11	ns
		pci_irdy#	2	11	ns
		pci_stop#	2	11	ns
		pci_par	2	11	ns
		pci_devsel#	2	11	ns
		pci_req#	2	12	ns
Ton	High-Z to active delay	**pci_ad<31:0>**	2	—	ns
		pci_cbe<3:0>#	2	—	ns
		pci_frame#	2	—	ns
		pci_trdy#	2	—	ns
		pci_irdy#	2	—	ns
		pci_stop#	2	—	ns
		pci_par	2	—	ns
		pci_devsel#	2	—	ns
		pci_req#		—	ns
Toff	Active to high-Z delay	**pci_ad<31:0>**	—	28	ns
		pci_cbe<3:0>#	—	28	ns
		pci_frame#	—	28	ns
		pci_trdy#	—	28	ns
		pci_irdy#	—	28	ns
		pci_stop#	—	28	ns
		pci_par	—	28	ns
		pci_devsel#	—	28	ns
		pci_req#	—	28	ns

* < 0.8 V

†Minimum delay times are specified with unloaded outputs. Maximum delay times are specified with a 50-pF external pin load.

(continued on next page)

Table 4–5 (Cont.) PCI Clock Domain Signal ac Parameters

Symbol	Parameter	Signals	Minimum	Maximum	Unit
Tsetup	Setup time to clock	**pci_ad<31:0>**	7	—	ns
		pci_cbe<3:0>#	7	—	ns
		pci_frame#	7	—	ns
		pci_trdy#	7	—	ns
		pci_irdy#	7	—	ns
		pci_stop#	7	—	ns
		pci_par	7	—	ns
		pci_devsel#	7	—	ns
		pci_idsel	7	—	ns
		pci_gnt#	10	—	ns
Thold	Hold time	**pci_ad<31:0>**	0	—	ns
		pci_cbe<3:0>#	0	—	ns
		pci_frame#	0	—	ns
		pci_trdy#	0	—	ns
		pci_irdy#	0	—	ns
		pci_stop#	0	—	ns
		pci_par	0	—	ns
		pci_devsel#	0	—	ns
		pci_gnt#	0	—	ns
		pci_idsel	0	—	ns
Trstoff	Reset asserted to high-Z delay	**pci_ad<31:0>**	—	40	ns
		pci_cbe<3:0>#	—	40	ns
		pci_frame#	—	40	ns
		pci_trdy#	—	40	ns
		pci_irdy#	—	40	ns
		pci_stop#	—	40	ns
		pci_par	—	40	ns
		pci_devsel#	—	40	ns
		pci_req#	—	40	ns

4.6 ac Specifications

4.6.2 PCI Cycle Timing

Figures 4–3 through 4–8 and Tables 4–6 through 4–11 describe the typical timing for selected PCI cycles with the 21130 as a target.

Figure 4–3 PCI Write — Cycle Start Timing

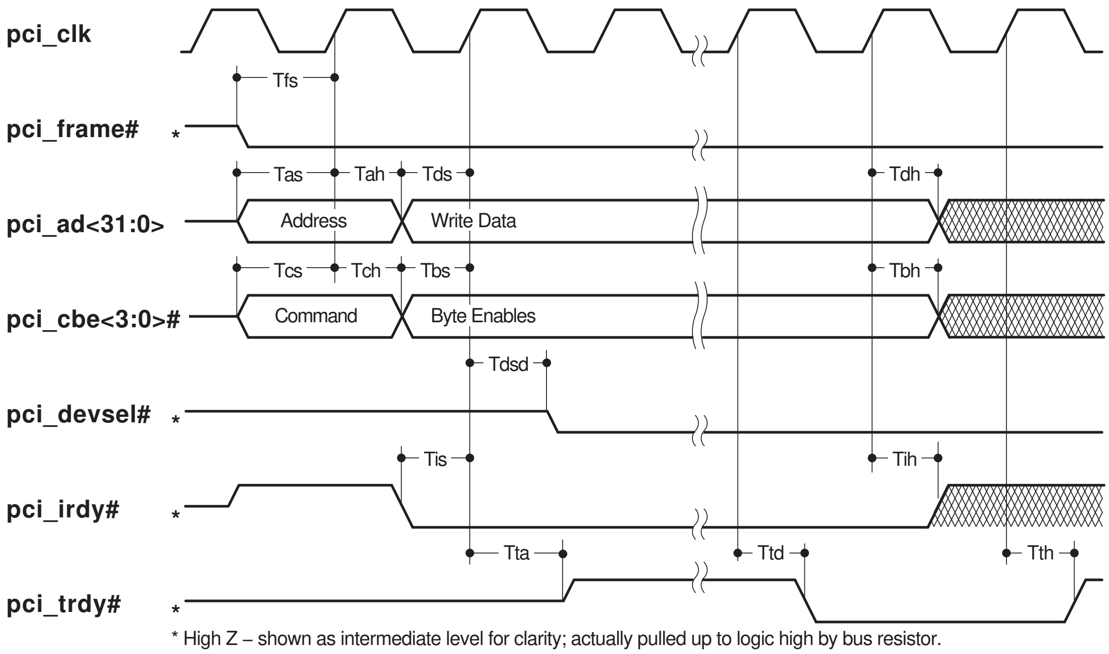


Table 4–6 PCI Write — Cycle Start Timing Parameters

Parameter	Description	Minimum ns	Maximum ns
Tfs	**pci_frame#** setup to clock	7	—
Tas	Address setup to clock	7	—
Tah	Address hold from clock	0	—
Tds	Write data setup to clock	7	—
Tdh	Write data hold from clock	0	—

(continued on next page)

Table 4–6 (Cont.) PCI Write — Cycle Start Timing Parameters

Parameter	Description	Minimum ns	Maximum ns
Tcs	Command setup to clock	7	—
Tch	Command hold from clock	0	—
Tbs	Byte enables setup to clock	7	—
Tbh	Byte enables hold from clock	0	—
Tdsd	**pci_devsel#** clock to signal delay	2	11
Tis	**pci_irdy#** setup to clock	7	—
Tih	**pci_irdy#** hold from clock	0	—
Tta	**pci_trdy#** high Z to active delay	2	—
Ttd	**pci_trdy#** clock to signal delay	2	11
Tth	**pci_trdy#** hold from clock	0	—

4.6 ac Specifications

Figure 4–4 PCI Read — Cycle Start Timing

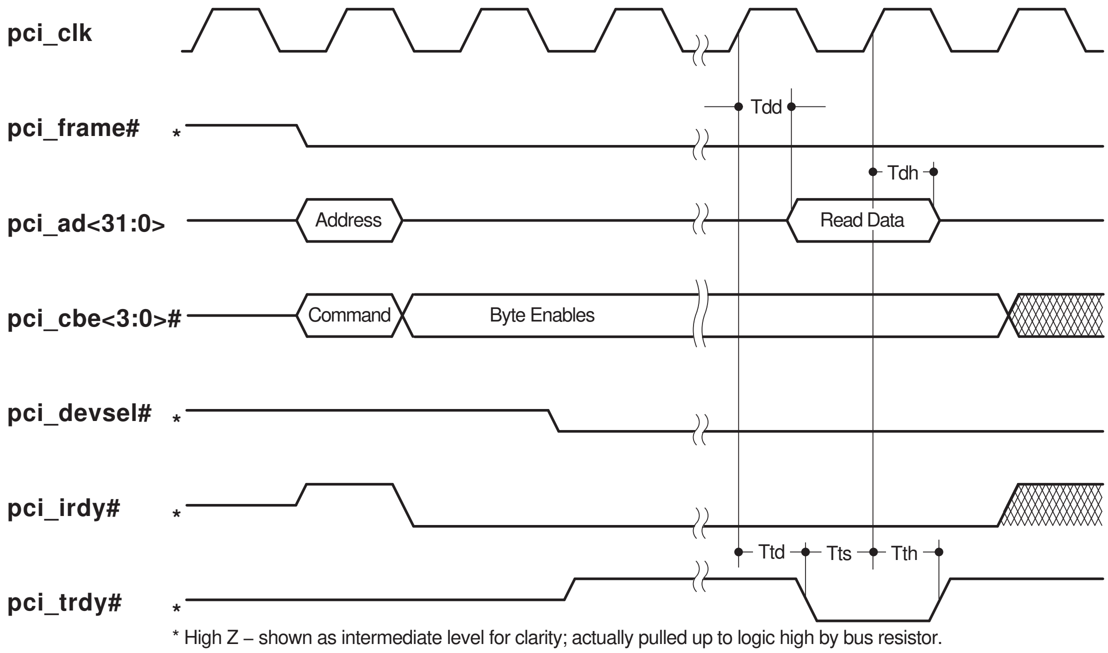


Table 4–7 PCI Read — Cycle Start Timing Parameters

Parameter	Description	Minimum ns	Maximum ns
Tdd	Read data clock to signal delay	2	11
Tdh	Read data hold from clock	0	—
Ttd	**pci_trdy#** clock to signal delay	2	11
Tts	**pci_trdy#** setup to clock	7	—
Tth	**pci_trdy#** hold from clock	0	—

Figure 4–5 PCI Read or Write — Cycle End Timing

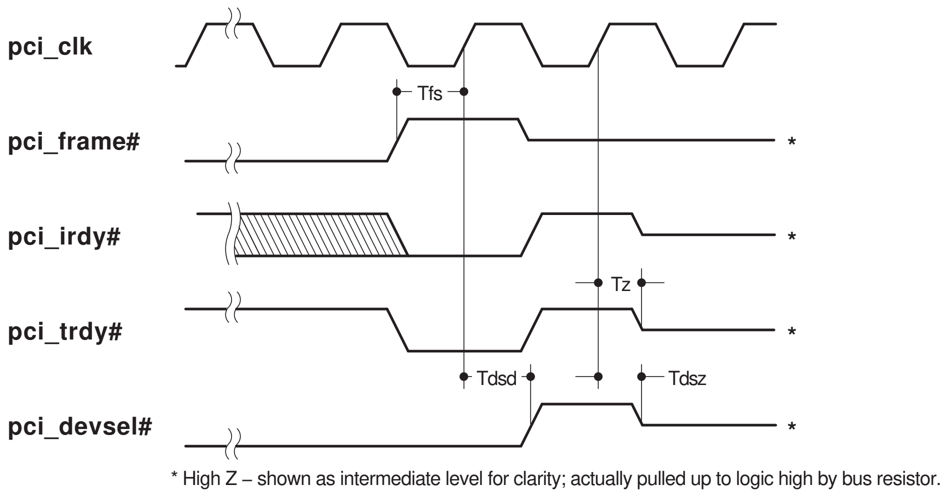


Table 4–8 PCI Read or Write — Cycle End Timing Parameters

Parameter	Description	Minimum ns	Maximum ns
Tfs	**pci_frame#** setup to clock	7	—
Ttz	**pci_trdy#** active to high-Z delay	—	28
Tdsd	**pci_devsel#** clock to signal delay	2	11
Tdsz	**pci_devsel#** active to high-Z delay	—	28

4.6 ac Specifications

Figure 4–6 PCI Target Disconnect or Abort — pci_stop# Timing

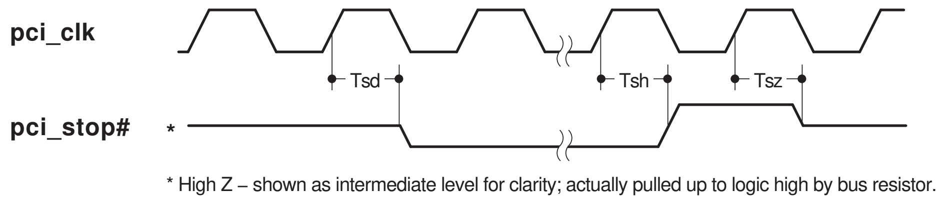


Table 4–9 PCI Target Disconnect or Abort — pci_stop# Timing Parameters

Parameter	Description	Minimum ns	Maximum ns
Tsd	**pci_stop#** clock to signal delay	2	11
Tsh	**pci_stop#** hold from clock	0	—
Tc	**pci_stop#** active to high-Z delay	—	28

Figure 4–7 PCI Configuration Cycle — pci_idsel Timing

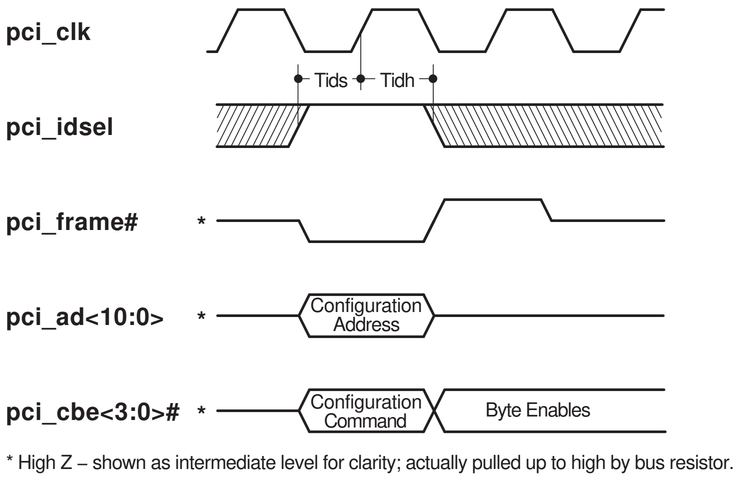


Table 4–10 PCI Configuration Cycle — pci_idsel Timing Parameters

Parameter	Description	Minimum ns	Maximum ns
Tids	**pci_idsel** setup to clock	7	—
Tidh	**pci_idsel** hold from clock	0	—

4.6 ac Specifications

Figure 4–8 and Table 4–11 describe typical parity timing for the 21130 as a target. As a master, the timing is identical, but the 21130 drives address, command, and write data parity, and receives read data parity.

Figure 4–8 PCI Parity — pci_par Timing

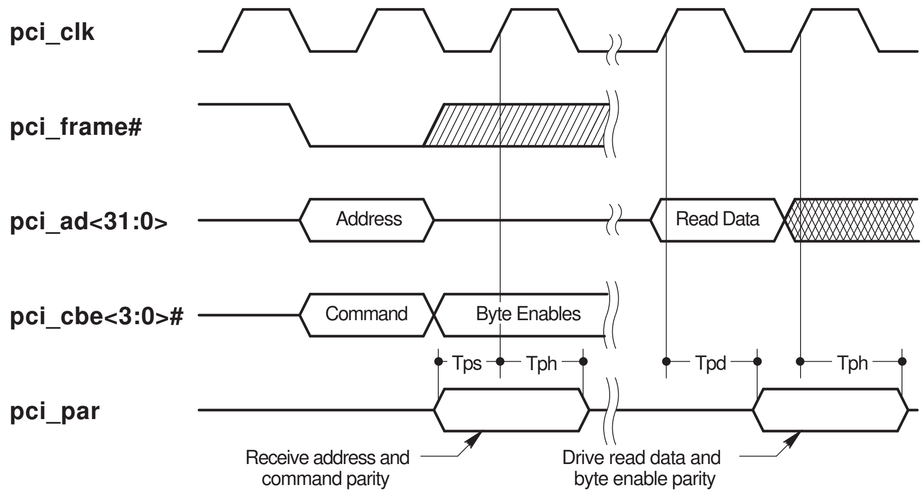


Table 4–11 PCI Parity — pci_par Timing Parameters

Parameter	Description	Minimum ns	Maximum ns
Tps	**pci_par** setup to clock	7	—
Tph	**pci_par** hold from clock	0	—
Tpd	**pci_par** clock to signal delay	2	11

4.6.3 Memory Cycle Timing

Figures 4–9 through 4–12 and Tables 4–12 through 4–15 describe typical memory timing cycles.

Figure 4–9 Hyperpage Mode Memory Write Cycle Timing

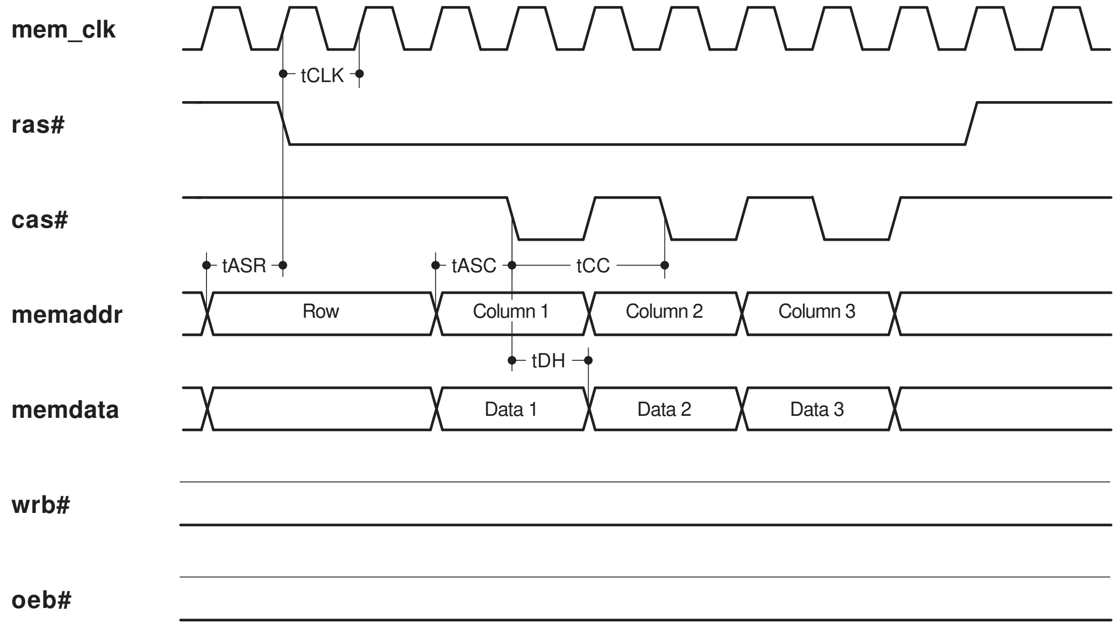


Table 4–12 Hyperpage Mode Memory Write Cycle Timing Parameters

Parameter	Description	21130 Value	DRAM Specification
tCC	Cycle Time	2tCLK	—
tASR	Row address setup time	—	Usually 0
tASC	Column address setup time	—	Usually 0
tDH	Data hold time	—	Must be <(tCLK −2) ns

4.6 ac Specifications

Figure 4–10 Hyperpage Mode Memory Read Cycle Timing

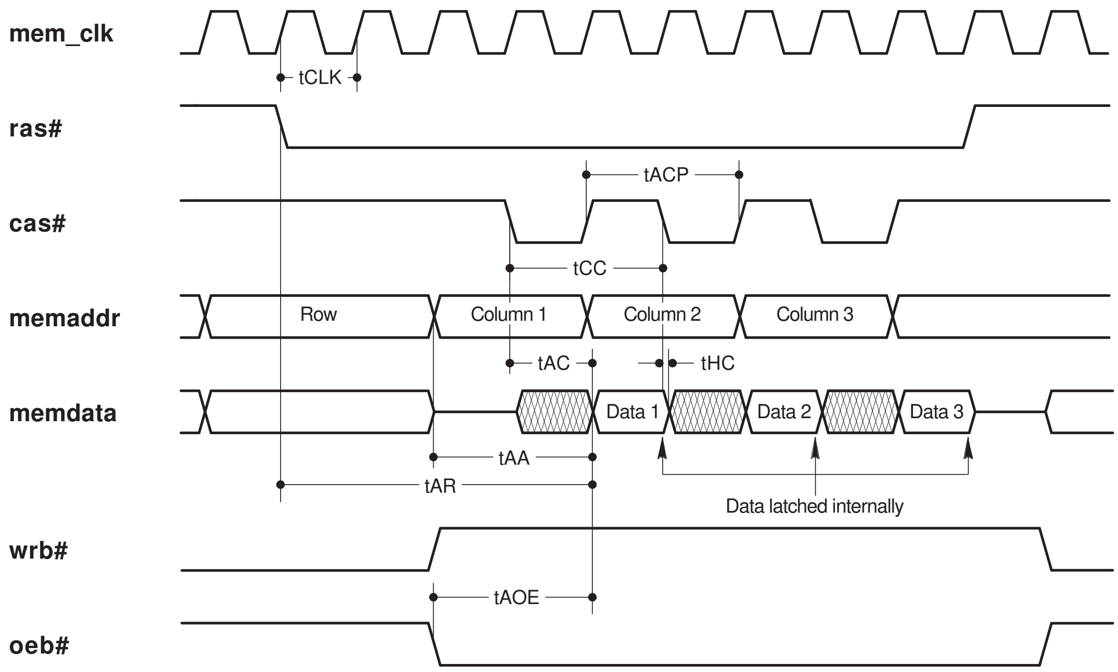


Table 4–13 Hyperpage Mode Memory Read Cycle Timing Parameters

Parameter	Description	21130 Value	DRAM Specification
tCC	Cycle Time	2tCLK	—
tAR	RAS access time	—	<(5tCLK −13) ns
tAA	Address access time	—	<(3tCLK−13.75) ns
tAC	CAS access time	—	<(2tCLK−12.75) ns
tACP	Access time from CAS precharge	—	<(3tCLK−10.25) ns
tHC	Data hold time	—	≥0
tAOE	Output enable access time	—	≤(2tCLK−13.75) ns

Figure 4–11 Read-Modify-Write Memory Cycle Timing

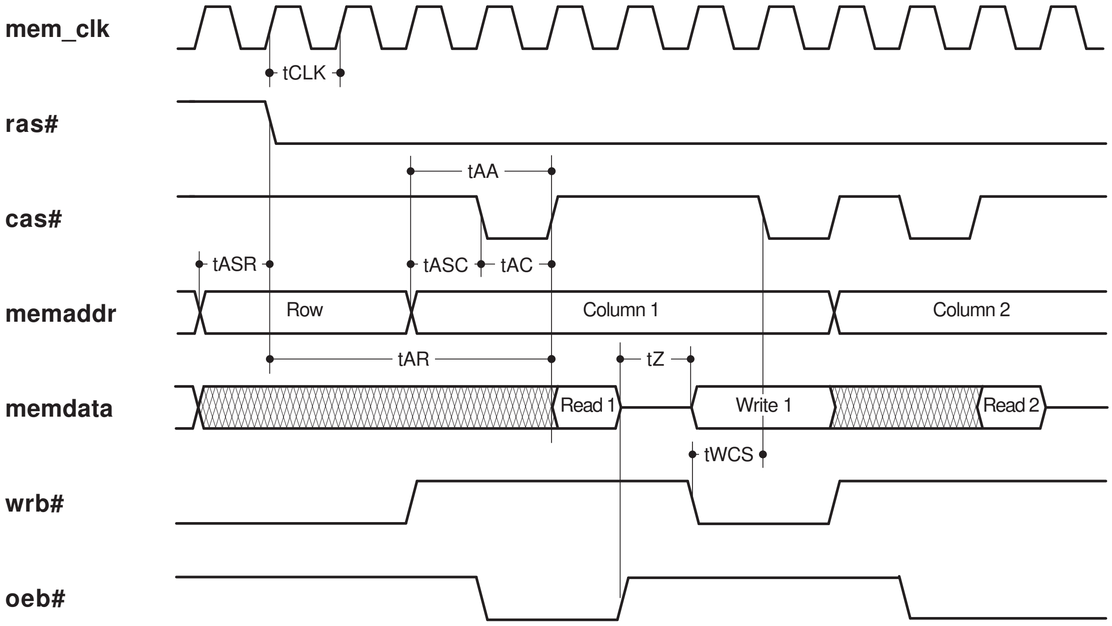


Table 4–14 Read-Modify-Write Memory Cycle Timing Parameters

Parameter	Description	21130 Value	DRAM Specification
tAR	RAS access time	—	<(5tCLK −13) ns
tAA	Address access time	—	<(3tCLK−13.75) ns
tASR	Row address setup time	—	Usually 0
tASC	Column address setup time	—	Usually 0
tAC	CAS access time	—	<(2tCLK−12.75) ns
tZ	Bus turnaround time	= tCLK	—
tWCS	Write enable to CAS setup time	—	≤(tCLK−2) ns

4.6 ac Specifications

Figure 4–12 CAS-Before-RAS Memory Refresh Cycle Timing

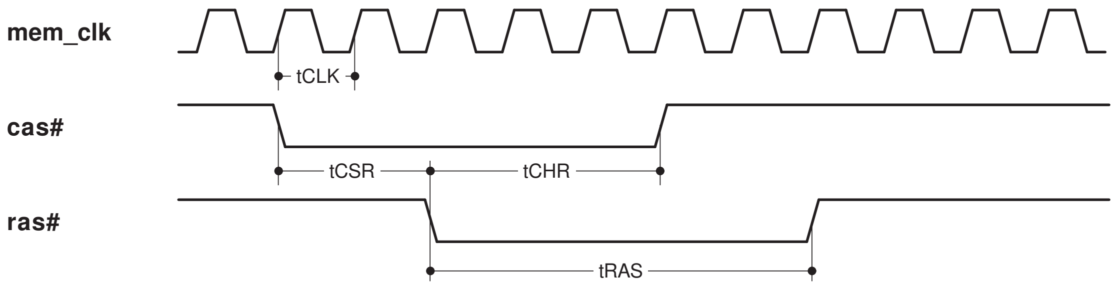


Table 4–15 CAS-Before-RAS Memory Refresh Cycle Timing Parameters

Parameter	Description	21130 Value
tCSR	CAS setup time before RAS	2tCLK
tCHR	CAS hold time	3tCLK
tRAS	RAS assertion time	5tCLK

4.6.4 ROM and GPP Data Cycle Timing

Figure 4–13, Figure 4–14, and Table 4–16 describe typical timing for ROM and GPP data transfer cycles.

Figure 4–13 ROM Data Cycle Timing

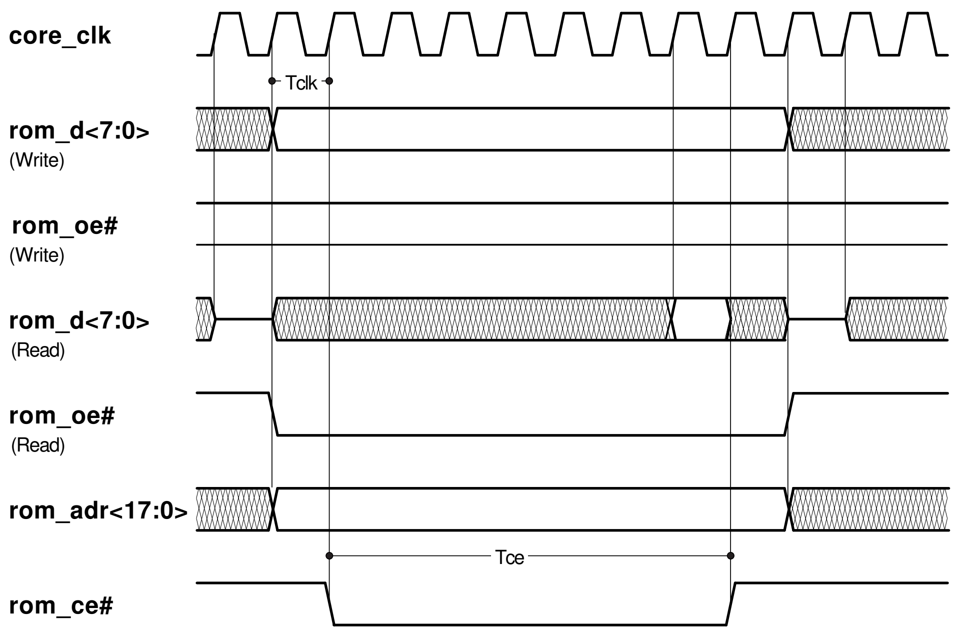
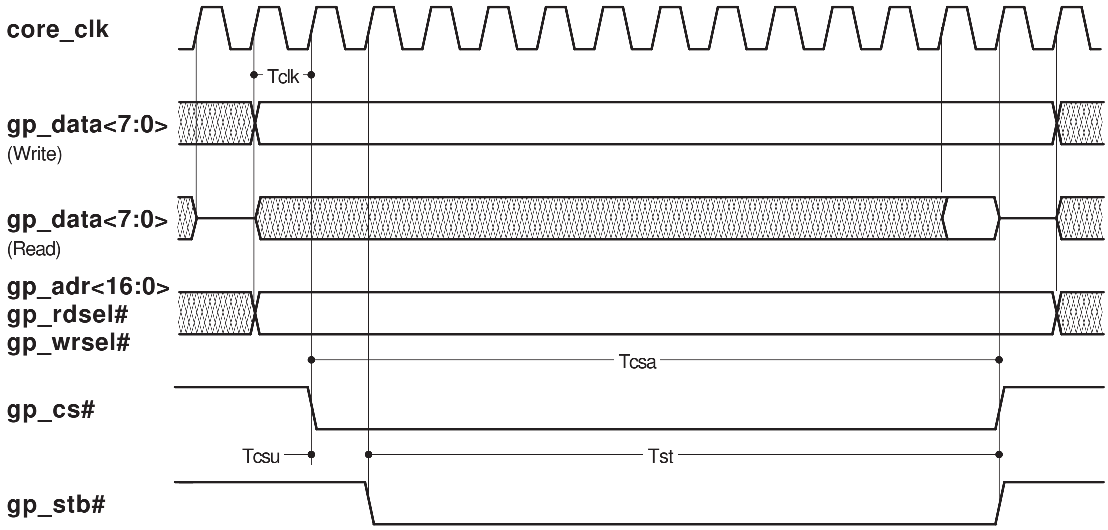


Table 4–16 ROM and GPP Data Cycle Timing Parameters

Parameter	Description	Value
Tclk	Clock cycle time (33 MHz)	30 ns
Tce	Chip enable assertion time	7Tclk
Tcsu	Chip select setup time	1Tclk
Tcsa	Chip select assertion time	12Tclk
Tst	Chip strobe assertion time	11Tclk

4.6 ac Specifications

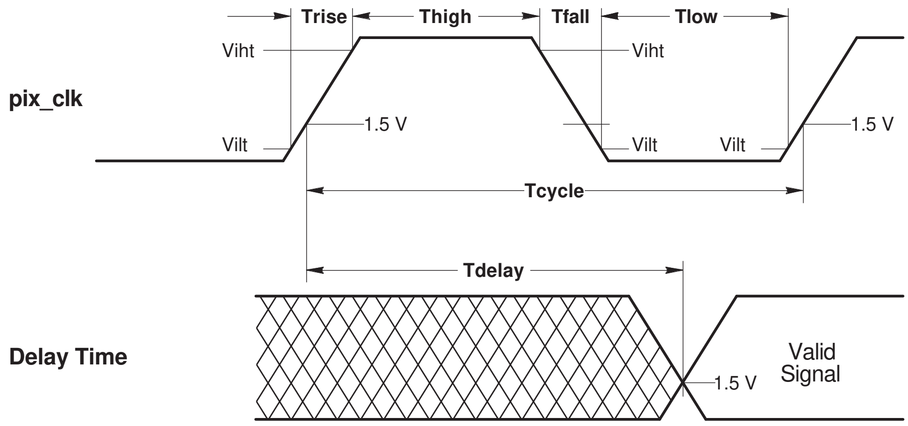
Figure 4–14 GPP Data Cycle Timing



4.6.5 Parameters for Pixel Clock and VAFC Clock Domain Signals

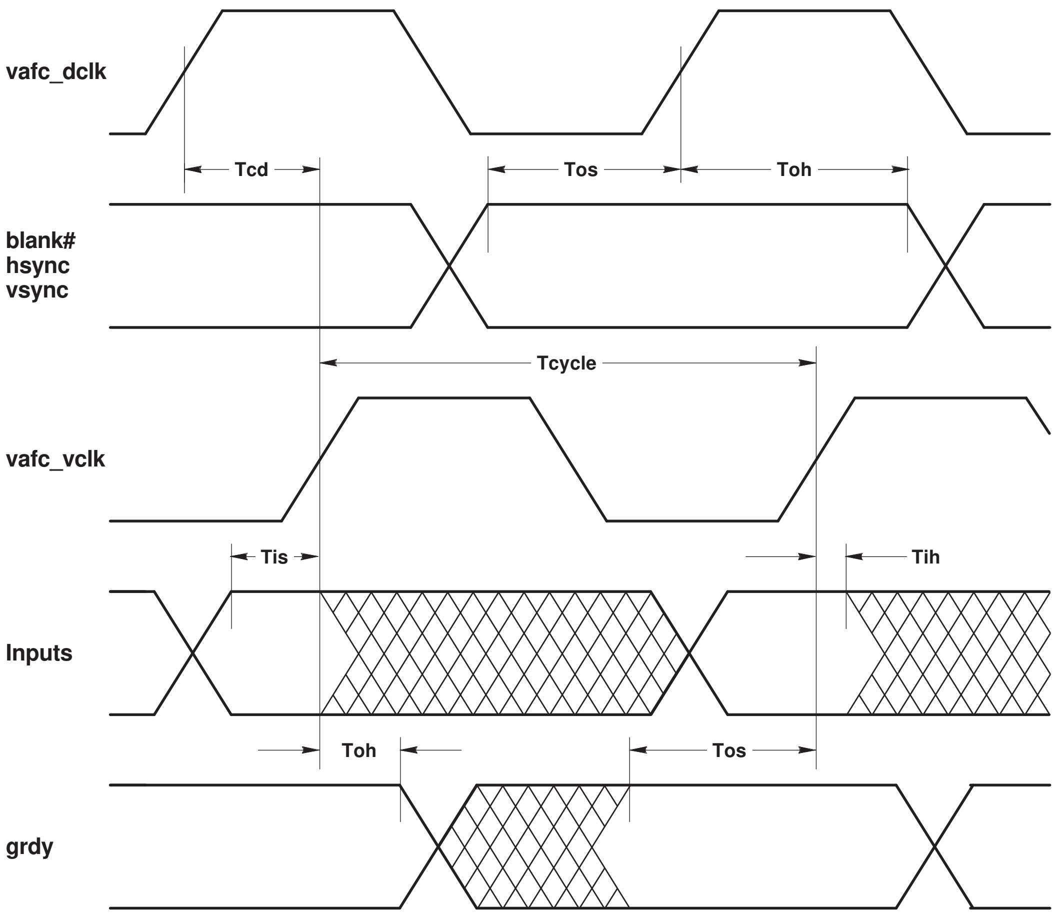
Figure 4–15 shows the ac parameter measurements for signals in the pixel clock (**pix_clk**) domain, and Figure 4–16 shows the ac parameter measurements for signals in the VAFC clock domain (the VAFC clock domain is a subset of the pixel clock domain). Table 4–17 specifies the parameter values.

Figure 4–15 Pixel Clock Domain Signal ac Parameter Measurements



4.6 ac Specifications

Figure 4–16 VAFC Clock Domain Signal ac Parameter Measurements



4.6 ac Specifications

Table 4–17 Pixel Clock and VAFC Clock Domain Signal ac Parameters

Symbol	Parameter	Signals	Minimum ns	Maximum ns	Notes
Tcycle	Clock cycle time (62.5 MHz)	**pix_clk**	16.0	—	❶
	Clock cycle time (37.5 MHz)	**vafc_dclk**	26.6	—	❷
	Clock cycle time (37.5 MHz)	**vafc_vclk**	26.6	—	❷
Thigh	Clock high time	**pix_clk**	6.5	—	—
		vafc_dclk	10.0	—	—
		vafc_vclk	10.0	—	—
Tlow	Clock low time	**pix_clk**	6.5	—	—
		vafc_dclk	10.0	—	—
		vafc_vclk	10.0	—	—
Trise	Clock rise time	**pix_clk**	—	3.0	—
	Rise time	**vafc_en#**	—	3.0	—
	Rise time	**evideo#**	—	3.0	—
Tfall	Clock fall time	**pix_clk**	—	3.0	—
	Fall time	**vafc_en#**	—	3.0	—
	Fall time	**evideo#**	—	3.0	—
Tcd	**vafc_dclk** to **vafc_vclk** delay	**vafc_dclk** **vafc_vclk**	5.0	20.0	❸
Tos	Output setup time	**blank#**	10.0	—	—
		hsync	10.0	—	—
		vsync	10.0	—	—
		grdy	10.0	—	—
		vafc_p<0:15>	10.0	—	—
Toh	Output hold time	**blank#**	2.0	—	—
		hsync	2.0	—	—
		vsync	2.0	—	—
		grdy	2.0	—	—
		vafc_p<0:15>	2.0	—	—
Tis	Input setup time	**vafc_p<0:15>**	10.0	—	—
Tih	Input hold time	**vafc_p<0:15>**	2.0	—	—

(continued on next page)

4.6 ac Specifications

Table 4–17 (Cont.) Pixel Clock and VAFC Clock Domain Signal ac Parameters

Notes

❶ The frequency for **pix_clk** is 62.5 MHz.

❷ The maximum frequency for **vafc_dclk** and **vafc_vclk** is 37.5 MHz (75 MHz ÷ 2, based on 75 MHz for a standard VESA 1024 × 768 70-Hz mode). Operation of graphics or video controllers at higher frequencies is outside of VESA compatibility and correct operation is not guaranteed.

❸ For synchronous transfers, **vafc_vclk** must meet this parameter.

4.6.6 VAFC Cycle Timing

Figures 4–17 and 4–18 and Tables 4–18 and 4–19 describe typical VAFC timing cycles. The measurements are extracted from the *VESA Advanced Feature Connector (VAFC) Proposal, Version 1.0p, Revision 0.4*.

Figure 4–17 VAFC Request Cycle Timing

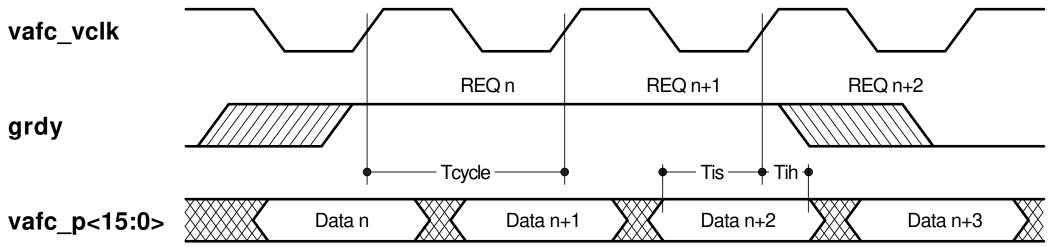


Table 4–18 VAFC Request Cycle Timing Parameters

Parameter	Description	Value
Tcycle	**vafc_vclk** cycle time	≥26.6 ns
Tis	**vafc_p<0:15>** setup time	≥10.0 ns
Tih	**vafc_p<0:15>** hold time	≥2.0 ns

Figure 4–18 VAFC Video Data Transfer Cycle Timing

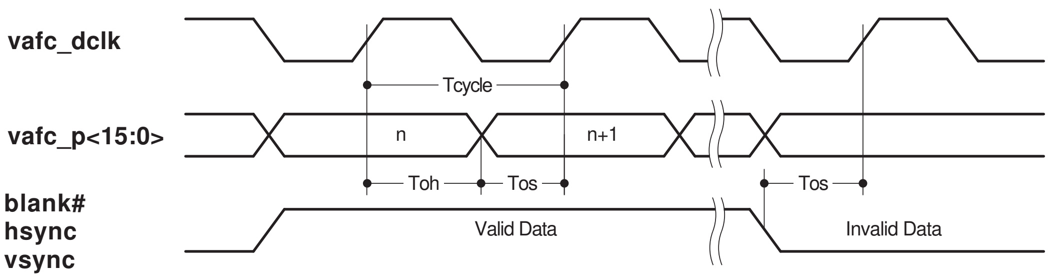


Table 4–19 VAFC Video Data Transfer Cycle Timing Parameters

Parameter	Description	Value
Tcycle	**vafc_dclk** cycle time	$\geq$26.6 ns
Tos	Output setup time	$\geq$10.0 ns
Toh	Output hold time	$\geq$2.0 ns

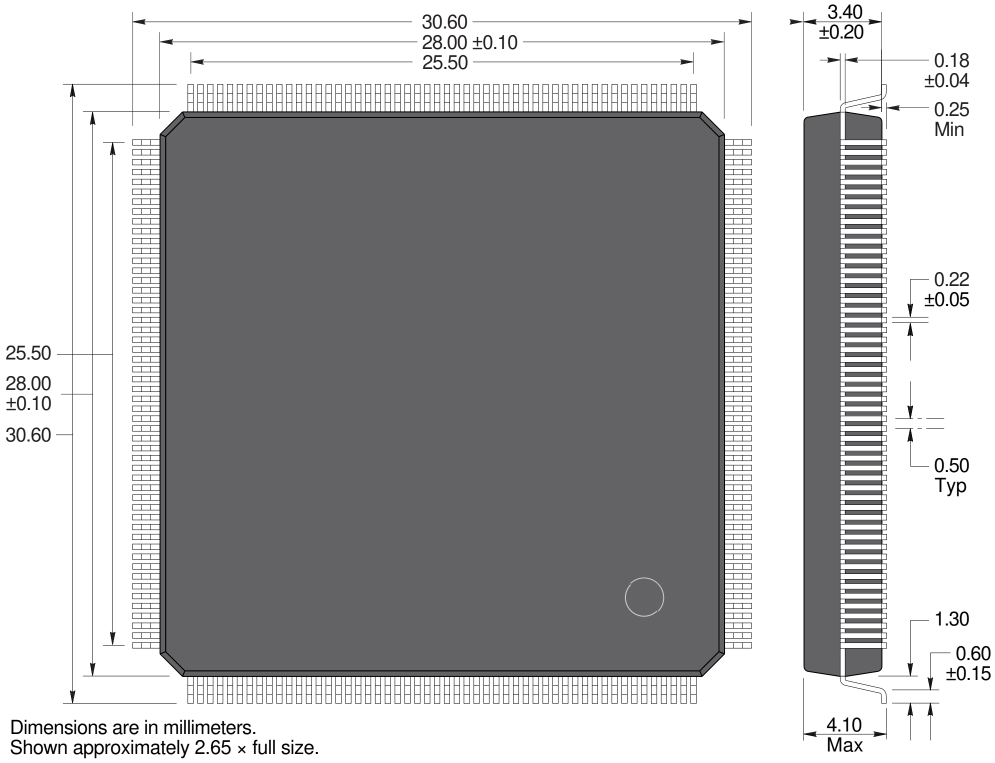
Notes:

- **vafc_dclk** is driven from the graphics source and is typically a submultiple of the pixel clock.
- **blank#** defines the display area.
- During invalid data time, the DAC can send any data.
- **grdy** is not used in output modes.

5
Mechanical Specifications

Figure 5–1 shows the DECchip 21130 208-pin plastic quad flat pack (PQFP) package.

Figure 5–1 DECchip 21130 208-Pin PQFP Package



6 Thermal Specifications

The DECchip 21130 operates reliably in a standard desktop or tower enclosure at enclosure internal ambient temperatures up to 55°C. To remove heat from the device at ambient temperatures greater than 55°C, internal airflow over the 21130 package must be provided.

Table 6–1 shows the airflow, in linear feet/minute (lfm), required for reliable 21130 operation at ambient temperatures (Ta) of 55°C and greater, where the memory clock frequency equals 80 MHz and the pixel clock frequency equals 135 MHz.

Table 6–1 Airflow Versus Temperature

Airflow	Ta
0 lfm	55.0°C
50 lfm	58.0°C
100 lfm	62.5°C
150 lfm	65.0°C
200 lfm	67.5°C

The supported maximum frequencies for the two onchip clocks are 80 MHz for the memory clock and 135 MHz for the pixel clock. When both clocks are at their maximum frequency, and certain worst case patterns are displayed from frame buffer memory, the 21130 approaches its maximum rated power dissipation of 2.5 W.

Lower memory and pixel clock frequencies reduce power dissipation. Lowering the memory clock frequency reduces power dissipation at the rate of 5.1 mW/MHz. For example, slowing the memory clock 10 MHz, from 80 MHz to 70 MHz, reduces power dissipation by 0.051 W. Similarly, lowering the pixel clock frequency reduces power dissipation at an approximate rate of 5.2 mW/MHz.

The relationship between clock speed and power dissipation permits trade-offs between 21130 clock frequencies and ambient operating temperatures. Every reduction of 6.5 MHz in either clock frequency allows an increase of 1°C in maximum ambient temperature.

7

Address Space

This chapter describes the DECchip 21130 address space allocations. The 21130 address space consists of the following discrete spaces:

- Configuration space
- ROM space
- VGA memory space
- VGA I/O space
- 2DA base address 0 and base address 1 mapped memory spaces

7.1 Overview

The 21130 responds to PCI accesses to the following address spaces:

- PCI configuration space
- BIOS ROM (256KB, relocatable)
- VGA I/O space (register accesses)
- Memory space (frame buffer accesses, 512KB range)
- 2DA base address 0 registers and frame buffer (32MB space)
- 2DA base address 1 access to VGA registers, DAC lookup tables (LUTs), and generic peripheral port (GPP) — mapped into 2MB of PCI memory space

All accesses are to PCI memory space (identified by transaction type), except VGA register accesses (see Section 7.5.2).

7.2 Configuration Space

Configuration space includes all the PCI configuration registers described in Section 8.2.

7.3 ROM Space

Note

The 21130 supports one external (E)(E)PROM. It and its associated functions are referred to as the BIOS ROM, EEPROM, flash ROM, PCI expansion ROM (space), and ROM ((sparse) space).ROM (space), and sparse ROM (space).

The location of the ROM in PCI memory space is defined by the PCI expansion ROM base address register (PRBR, Section 8.2.6). See Section 7.5.2.5 for more information about accessing the expansion ROM.

7.4 VGA Memory Space

The VGA memory space is mapped (hardwired) to the standard VGA address range of A0000 through BFFFF. Access to this memory space is enabled in the PCI command and status register (PCSR <1>, Section 8.2.2). The VGA graphics controller miscellaneous register (VGMISR, Section 8.15.9) specifies the address range.

7.5 2DA Memory Space

The 2DA memory space is mapped by PCI device base address registers 0 and 1 (PDBR0 and PDBR1, Section 8.2.5). The 2DA base address 0 memory space includes all of the 2D acceleration registers and the frame buffer. The 2DA base address 1 memory space contains the VGA alternate register space; palette and DAC register space; interrupt status register space; generic peripheral port (GPP) 0 and 1 spaces; and the ROM read and write sparse space.

7.5.1 2DA Base Address 0 Memory Space

The size of the 2DA base address 0 memory space is 32MB. It is mapped into PCI memory space at the base address specified in PDBR0 <31:4> (bits PDBR0 <24:4> are hardwired to zero).

7.5 2DA Memory Space

The 32MB memory space contains up to eight copies of a core space (see Figure 7–1). The core space size can be 4MB or 8MB, depending on the application. The size of a core space is specified by the address mask in the deep register (Table 7–1). The address mask is programmed with different values to tailor core space organization according to the physical size of memory in a particular configuration.

Table 7–1 shows the core space size and the appropriate settings of the deep register address mask field for the supported configurations.

Table 7–1 Core Space per Frame Buffer Option

Configuration	Physical Memory Size	Core Space Size	Address* Mask
1MB	1MB	4MB	000
2MB	2MB	4MB	000
4MB	4MB	8MB	001

*Deep register field binary codes (GDER <4:2>, Section 8.5.2).

Typically, the memory space maps to 32MB of PCI memory space, and the core space is 4MB or 8MB; therefore, the memory space contains four or eight copies of core space. Each copy of core space is identical and maps the frame buffer and the same set of registers. These multiple copies of core space are useful in systems based on CPUs that do not enforce write-ordering (see Section 11.12.1 for more information).

Figure 7–1 shows the memory space mapped as a function of the core space size.

7.5.1.1 Base Address 0 Core Space Organization

The core space maps the 21130 general registers (register space, Section 7.5.1.2), alternate control space (Section 7.5.1.3), and the 21130 frame buffer (frame buffer space). The 21130 frame buffer space can be accessed in any of the drawing modes described in Chapter 10. Figure 7–2 shows the core space maps for various frame buffers.

7.5 2DA Memory Space

Figure 7–1 Memory Space Organization

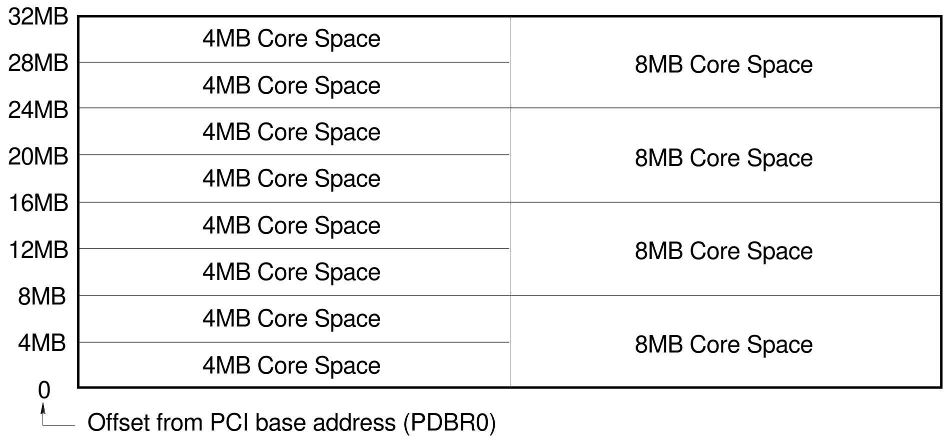
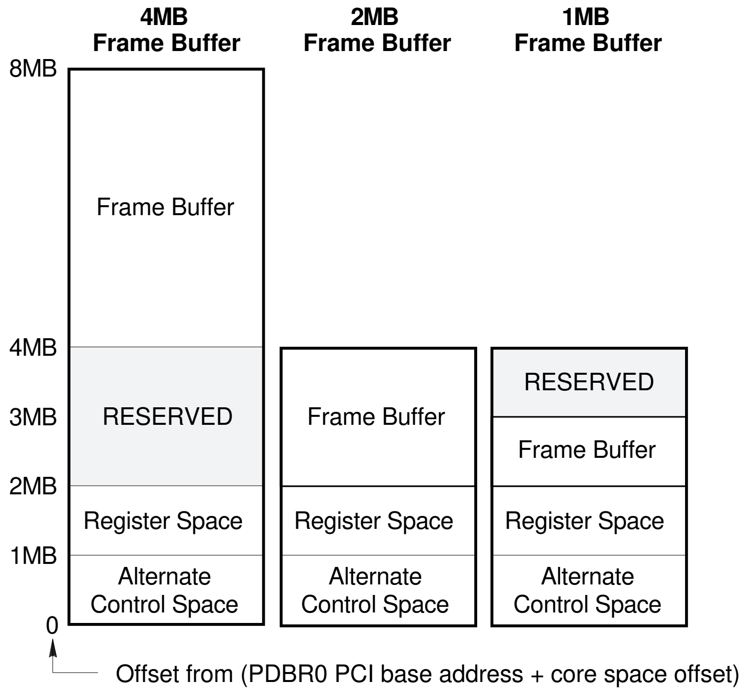


Figure 7–2 Core Space Maps

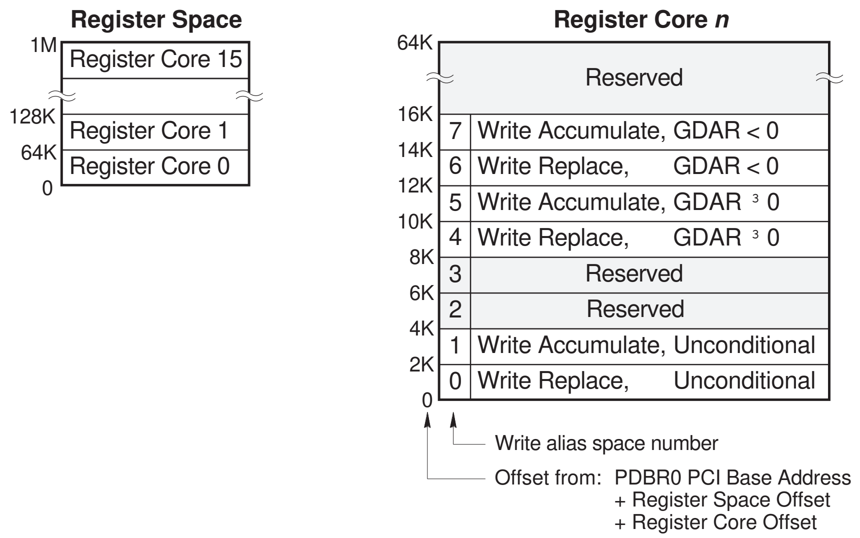


7.5.1.2 Base Address 0 Register Space Organization

The base address 0 (PDBR0, Section 8.2.5) register space contains all the registers except the interrupt status register (MISR), and the PCI configuration, VGA, and palette and DAC registers. The register space size is 1MB. This space is only Dword accessible; that is, all byte enables must be either asserted or deasserted.

Figure 7–3 shows how the register space is divided into sixteen 64KB register core regions. Each register space core is identical and maps several aliases of the 2KB register set. The 21130 registers are mapped by offset into each 2KB alias. (Table 7–3 lists the base address 0 registers in order of offset.)

Figure 7–3 Base Address Register 0 Register Space Organization



The register write alias spaces support the repeat loop mechanism (Section 8.4.6). Note that alias spaces 1, 4, 5, 6, and 7 should be used to write only the registers listed in Table 7–2.

7.5 2DA Memory Space

Table 7–2 Registers Supported by Write Alias Spaces 1, 4, 5, 6, and 7

Offset[1]	Name	Mnemonic	Access
03C	Address register	GADR	RW
080	Data register	GDAR	RW
0B0	Dither row register	GDRR	RW
0B4	Dither column register	GDCR	RW
098	DMA base address register	GDBR	RW

[1]Hexadecimal offset into register-write alias space

Register writes to the various alias spaces cause the following actions to take place when updating the register values.

Alias Space 0: Write Replace, Unconditional

A write to this alias space causes the register value to be replaced with the value being written. This update is performed unconditionally. This is the only alias space that supports all the registers listed in Table 7–3.

Alias Space 1: Write Accumulate, Unconditional

A write to this alias space causes the register to be updated with the sum of the value being written and the current register value. This update is performed unconditionally.

Alias Space 2 and 3: Reserved

Alias Space 4: Write Replace, GDAR $\geq$ 0

A write to this alias space causes a write replace to be performed only if the value of the data register (GDAR), sampled at the time of the last write to the repeat begin register (GRBR) or repeat end register (GRER), is greater than or equal to zero.

Alias Space 5: Write Accumulate, GDAR $\geq$ 0

A write to this alias space causes a write accumulate to be performed if the value of the GDAR, sampled at the time of the last write to the GRBR or GRER, is greater than or equal to zero.

Alias Space 6: Write Replace, GDAR < 0

A write to this alias space causes a write replace to be performed only if the value of the GDAR, sampled at the time of the last write to the GRBR or GRER, is less than 0.

Alias Space 7: Write Accumulate, GDAR < 0

A write to this alias space causes a write accumulate to be performed if the value of the GDAR, sampled at the time of the last write to the GRBR or GRER, is less than zero.

Note

All undefined register locations are reserved and must not be used.

Table 7–3 lists the base address 0 registers in order of offset.

Table 7–3 Base Address Register 0 Register Map

Offset[1]	Name	Mnemonic	Access
000	Copy buffer register 0	GCBR0	RW[2]
004	Copy buffer register 1	GCBR1	RW[3]
008	Copy buffer register 2	GCBR2	RW[2]
00C	Copy buffer register 3	GCBR3	RW[3]
010	Copy buffer register 4	GCBR4	RW[2]
014	Copy buffer register 5	GCBR5	RW[3]
018	Copy buffer register 6	GCBR6	RW[2]
01C	Copy buffer register 7	GCBR7	RW[3]
020	Foreground register	GFGR	RW
024	Background register	GBGR	RW
02C	Pixel mask register (one shot)	GPXR	RW
030	Mode register	GMOR	RW
034	Raster operation register	GOPR	RW
038	Pixel shift register	GPSR	RW
03C	Address register	GADR	RW
040	Bresenham 1 register	GB1R	RW
044	Bresenham 2 register	GB2R	RW
048	Bresenham 3 register	GB3R	RW
04C	Continue register	GCTR	WO
050	Deep register	GDER	RW
05C	Pixel mask register (persistent)	GPXR	WO
060	Cursor base address register	CCBR	RW

[1]Hexadecimal offset into PDBR0 register write alias space

[2]Writes access copy buffer even locations

[3]Writes access copy buffer odd locations

(continued on next page)

Table 7–3 (Cont.) Base Address Register 0 Register Map

Offset[1]	Name	Mnemonic	Access
06C	Video base address register	VIVBR	RW
070	Video valid register	VIVVR	RW
074	Cursor XY register	CXYR	RW
080	Data register	GDAR	RW
098	DMA base address register	GDBR	RW
09C	Bresenham width register	GBWR	WO
0AC[4]	Address register	GADR	WO
0B0	Dither row register	GDRR	RW
0B4	Dither column register	GDCR	RW
0BC	Span width register	GSWR	RW
0C4	Scaled-copy control register	GSCR	RW
0CC	Video scanline increment register	VISIR	RW
0D0	Video line width register	VILWR	RW
0D4	Video pixel format register	VFPFR	RW
0E0	Video pixel occlusion bitmap base address register	VFOBR	RW
0E8	Alternate video control register	VFAVR	RW
0EC	Cursor mode register	CMOR	RW
100	Slope-no-go register 0	GSNR0	WO
104	Slope-no-go register 1	GSNR1	WO
108	Slope-no-go register 2	GSNR2	WO
10C	Slope-no-go register 3	GSNR3	WO
110	Slope-no-go register 4	GSNR4	WO
114	Slope-no-go register 5	GSNR5	WO
118	Slope-no-go register 6	GSNR6	WO
11C	Slope-no-go register 7	GSNR7	WO
120	Slope register 0	GSLR0	WO
124	Slope register 1	GSLR1	WO
128	Slope register 2	GSLR2	WO
12C	Slope register 3	GSLR3	WO
130	Slope register 4	GSLR4	WO
134	Slope register 5	GSLR5	WO
138	Slope register 6	GSLR6	WO
13C	Slope register 7	GSLR7	WO
160	Copy-64 source register	GCSR	WO
164	Copy-64 destination register	GCDR	WO
168[4]	Copy-64 source register	GCSR	WO
16C[4]	Copy-64 destination register	GCDR	WO

[1]Hexadecimal offset into PDBR0 register write alias space

[4]Register alias

(continued on next page)

Table 7–3 (Cont.) Base Address Register 0 Register Map

Offset[1]	Name	Mnemonic	Access
170[4]	Copy-64 source register	GCSR	WO
174[4]	Copy-64 destination register	GCDR	WO
178[4]	Copy-64 source register	GCSR	WO
17C[4]	Copy-64 destination register	GCDR	WO
1F4	Video pixel occlusion bitmap current address register	VFOAR	RO
1F8	Command status register	MCSR	RO
1FC	Video current refresh address register	VFCRR	RO
340	Repeat begin register	GRBR	WO
350	Repeat end register	GRER	WO
360	Copy-64A source register	GCASR	WO
364	Copy-64A destination register	GCADR	WO
Unused locations are reserved.			

[1]Hexadecimal offset into PDBR0 register write alias space

[4]Register alias

The registers in base address 0 register space are described in Sections 8.3.1 through 8.8.5.

7.5.1.3 Base Address 0 Alternate Control Space Writes

Depending on the specific address, writes to alternate control space address either the continue register (GCTR) or the address register (GADR), as shown in Table 7–4.

Table 7–4 Targets for Writes to Alternate Control Space

Alternate Control Space Offset				Write Target
0	$\leq$ Even offset	<	512K	Address register (GADR)
0	< Odd offset	<	512K	Continue register (GCTR)
512K	$\leq$ Offset	<	1M	Continue register (GCTR)

In other words, writes to even addresses in alternate control space below 512K address the GADR; other writes address the GCTR. Sequential access to the GCTR and GADR are useful for 21130 graphics processing in systems based on Alpha microprocessors, as described in Section 11.12.2.

Write access to alternate control space is only by Dword; PCI byte enables are ignored.

7.5 2DA Memory Space

7.5.2 Base Address 1 Memory Space

The base address 1 (PDBR1, Section 8.2.5) memory space is a sparsely populated, longword-aligned, 2MB space. It contains the palette and DAC register space, interrupt status register space, generic peripheral port (GPP) 0 and 1 spaces, and the ROM read and write sparse space. It also contains the VGA alternate register space. (Because they control resources that are shared by the VGA controller and the 2DA, certain addresses, such as the register addresses for the VGA CRTC registers and the palette index registers, must be accessible in both VGA mode and 2DA mode.)

Table 7–5 Base Address Register 1 Memory Space Map

Offset*	Register	Access
1FFFFF:100000	ROM sparse space	RW
0FFFFF:080000	GPP space	RW
07FFFF:040000	Interrupt status register space	RW
03FFFF:001040	Reserved	—
00103F:001000	Palette and DAC register space	RW
000FFF:000000	VGA alternate register space	RW

*Offset from value in PDBR1.

7.5.2.1 Base Address 1 VGA Alternate Register Space

The VGA alternate register space provides sparse space access to VGA registers from PCI memory space. This allows PCI I/O addressing to be disabled with the I/O space enable bit (PCSR <0>, Section 8.2.2), while maintaining access to the CRTC controller and related registers from PCI memory space. Typically, VGA alternate register space is used during 2DA operation to access the following VGA CRTC-related registers:

- VGA CRTC index register (VCINXR, Section 8.13.1)
- VGA CRTC data register (VCDATR, Section 8.13.2)
- VGA miscellaneous output register (VEMISR, Section 8.11.1)

  Vertical sync polarity (VSP, <7>)
  Horizontal sync polarity (HSP, <6>)
- VGA feature control register (VEFCOR, Section 8.11.2)

  Vertical sync select (VSS, <3>)

Bits <31:21> are stripped from addresses that are in the range mapped by PDBR1. If bits <20:12> = 0, the addresses are aliased to PCI VGA register space. PDBR1-mapped addresses in the range 000..FCC are right-shifted 2 bits, and passed to the VGA function (Table 7–6). For example, an address of *(contents of PDBR1) + ED4* accesses address 3B5 in VGA register space. The low byte of the PCI bus (**pci_ad<7:0>**) is used for VGA data.

Table 7–6 lists the directly accessible VGA registers with their I/O and PDBR1 alternate register space addresses.

Table 7–6 Base Address Register 1 VGA Register Map

I/O Address[1]	Memory Address[2]	Name	Mnemonic	Access
3B4	ED0	VGA CRTC index register	VCINXR	RW[3]
3B5	ED4	VGA CRTC data register	VCDATR	RW[3]
3BA	EE8	VGA feature control register	VEFCOR	W[3]
3C2	F08	VGA miscellaneous output register	VEMISR	W
3CA	F28	VGA feature control register	VEFCOR	R
3CC	F30	VGA miscellaneous output register	VEMISR	R
3D4	F50	VGA CRTC index register	VCINXR	RW[4]
3D5	F54	VGA CRTC data register	VCDATR	RW[4]
3DA	F68	VGA feature control register	VEFCOR	W[4]

Unused locations are reserved.

[1]VGA I/O space address
[2]Base address 1 VGA alternate register space address
[3]Monochrome
[4]Color

The VGA registers are described in Sections 8.10 through 8.17.4.

7.5.2.2 Base Address 1 Generic Peripheral Port Space

The 128KB GPP space (512KB sparse PCI address space) provides access to generic peripherals attached to the 21130.

Bits <31:21> are stripped from addresses that are in the range mapped by PDBR1; and, if bits <20:19> = 01_2, the addresses result in a GPP access. The GPP address comprises bits <18:2>. Chip select (**gp_cs#**) is asserted when the GPP space is referenced. The low byte of the PCI bus (**pci_ad<7:0>**) is used for GPP data.

7.5 2DA Memory Space

Note

GPP accesses should be restricted to vertical blank time, or the time when video is disabled by the video valid bit in the video valid register (VIVVR <0>, Section 8.7.2). Byte packing and unpacking is not supported in the GPP space.

See Chapter 3 for GPP signal descriptions and Chapter 4 for GPP timing.

7.5.2.3 Base Address 1 Interrupt Status Register Space

The interrupt status register (MISR, Section 8.3.2) is located in this space, rather than in command buffered space (that is, base address 0 space) with the other core registers, to accommodate interrupts that might occur during a loop operation. This allows the interrupt service routine to clear the interrupt status bits, without disturbing the contents of the command FIFO.

7.5.2.4 Base Address 1 Palette and DAC Register Space

The palette and DAC register space provides sparse space access to the palette and DAC graphics color LUT (RAM), cursor color, and DAC control registers, independently of VGA register space.

Note

The palette and DAC register space must be used to set the cursor color; there is no equivalent function in VGA register space.

Bits <31:21> are stripped from addresses that are in the range mapped by PDBR1; and, if bits <20:12> = 001_{16}, the palette and DAC register space is accessed. For example, an address of *(contents of PDBR1) + 1004*$_{16}$ accesses the palette and DAC color register (Table 7–7).

Table 7–7 lists the palette and DAC registers and their offsets into PDBR1 memory space.

Table 7–7 Base Address Register 1 Palette and DAC Register Map

Offset	Name	Mnemonic	Access
1000	Palette and DAC RAM write address register	DPWR	RW
1004	Palette and DAC RAM color register	DPCR	RW
1008	Palette and DAC pixel mask register	DPMR	RW
100C	Palette and DAC RAM read address register	DPRR	RW
1010	Palette and DAC cursor write address register	DCWR	RW
1014	Palette and DAC cursor color register	DCCR	RW
1018	Palette and DAC command register 0	DCOR0	RW
101C	Palette and DAC cursor read address register	DCRR	RW
1028	Palette and DAC status register	DSTR	RW
1030	Palette and DAC command register 1	DCOR1	RW
1034	Palette and DAC red signature register	DRSR	RW
1038	Palette and DAC green signature register	DGSR	RW
103C	Palette and DAC blue signature register	DBSR	RW
Unused locations are reserved.			

The palette and DAC registers are described in Sections 8.9 through 8.9.9.

7.5.2.5 Base Address 1 ROM Sparse Space Access

The 1MB ROM sparse space is embedded in core space. It provides an alternate map of the EEPROM in addition to the standard PCI expansion ROM space. (The standard PCI expansion ROM space is an independent, 256KB, byte-readable address space. Its location in PCI memory space is defined by the PCI expansion ROM base address register PRBR, Section 8.2.6.)

Unlike the PCI expansion ROM space, ROM sparse space is not byte-contiguous. Each Dword read returns 1 byte of ROM data. ROM sparse space, in which 3 null bytes exist between consecutive valid ROM bytes, is effectively a sparse version of the PCI expansion ROM space.

Because of the ROM sparse space layout, software must effectively multiply the desired byte offset by 4 (left shift 2 bits) to get the correct ROM sparse space address. For example, to operate on the second ROM byte, the offset into ROM sparse space must be 8. The ROM sparse space Dword address is determined as follows:

ROM sparse space address = PCI base address 1
+ ROM sparse space offset
+ (desired byte address × 4)

7.5 2DA Memory Space

The EEPROM is read through the upper-half of the 64-bit memory port. To avoid bus contention, the 21130 disables its drivers and the RAM data bus drivers before reading the EEPROM. Externally, the byte-wide EEPROM must be located on **memdata<57:50>** of the RAM data bus. The EEPROM data must not be driven on **memdata<57:50>** unless the EEPROM chip enable (**rom_ce#**) and output enable (**rom_oe#**) pins are active. (See Section 12.3 for more information about the hardware interface to the external EEPROM.)

Note

When operating in 2DA mode, ROM accesses should be restricted to vertical blank time, or the time when video is disabled by the video valid bit in the video valid register (VIVVR <0>, Section 8.7.2).

The expansion ROM must be written through ROM sparse space. The flash ROM write enable bit must be set in the deep register (GDER <12>, Section 8.5.2) and software must observe the EEPROM write recovery time.

Figure 7–4 shows the assembly and format of the PCI Dword read from ROM sparse space, and Table 7–8 describes the fields.

Figure 7–4 ROM Sparse Space PCI Read Data Format

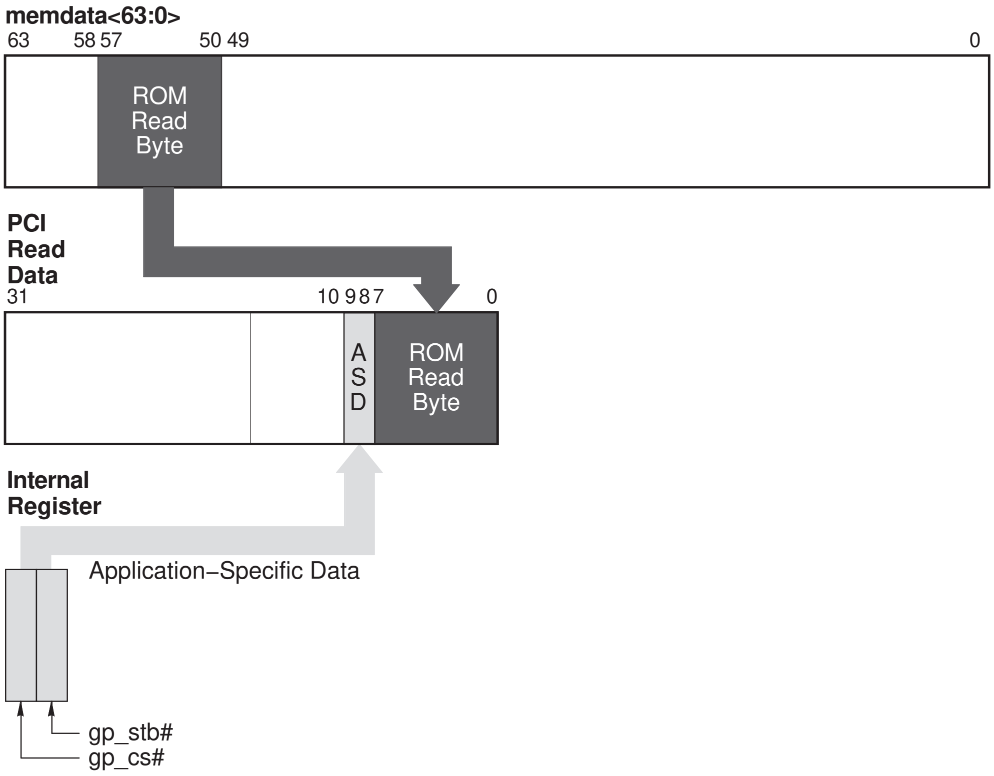


Table 7–8 ROM Sparse Space PCI Read Data Field Description

Bits	Field	Description
9:8	ASD	Application specific data <1:0> — when reset is asserted, the **gp_stb#** and **gp_cs#** pins are sampled and saved in an internal register. This sampled state is returned during a sparse space ROM read.
7:0	ROM Read Byte	The desired byte read from external EEPROM at the ROM sparse space Dword address.

8
Register Descriptions

This chapter describes all of the DECchip 21130 registers.

8.1 Overview

Except as noted in the appropriate register description:

- All 21130 registers can be read and written.
- Reserved fields must be zero (must never be written with a nonzero value and return unpredictable values when read).
- Most registers are cleared when chip reset is asserted.

Note

Abbreviations in the access column of the register field description tables are defined in the Conventions section of the Preface.

Registers are divided into two classes and several subclasses:

- PCI configuration registers — control PCI configuration for the 21130 device.
  - Device-independent registers are required in all PCI devices to implement generic PCI configuration functions.
  - Device-specific registers implement PCI configuration functions specific to the device.
- 21130 device registers — implement the following functions:
  - Miscellaneous registers indicate the current status of chip processing and pending interrupts, enable interrupts, provide a mechanism for scheduling commands, and control the pixel clock.
  - Graphics command registers initiate graphics operations.

8.1 Overview

– Graphics control registers provide the parameters for graphics operations.

– Hardware cursor control registers define the location and display of the chip's 64 $\times$ 64 $\times$ 2 cursor.

– Video control and format registers define the location and display format of a selected portion of the frame buffer.

– Palette and DAC registers control the color LUT, cursor color, and DACs.

– VGA and VGA extended registers set up and control the VGA-compatible subsystem for VGA mode operations.

Table 8–1 lists each 21130 register name, mnemonic, hexadecimal address or index, and the section that describes the register.

Table 8–1 21130 Registers

Name	Mnemonic	Address	Section
Configuration Space Header Block	**PxxR**	**Range[1]**	**8.2**
PCI identification register	PIDR	03..00	8.2.1
PCI command and status register	PCSR	07..04	8.2.2
PCI class and revision register	PCRR	0B..08	8.2.3
PCI latency timer and header type register	PLTR	0F..0C	8.2.4
PCI device base address register 0	PDBR0	13..10	8.2.5
PCI device base address register 1	PDBR1	17..14	8.2.5
Reserved	—	2F..18	—
PCI expansion ROM base address register	PRBR	33..30	8.2.6
Reserved	—	3B..34	—
PCI interrupt line register	PLIR	3F..3C	8.2.7
Device-Specific Configuration Space	**PxxR**	**Range[1]**	**8.2**
PCI clock control register	PCCR	43..40	8.2.8
Reserved	—	FF..40	—

[1] Address = hexadecimal byte address range for PCI registers

(continued on next page)

Table 8–1 (Cont.) 21130 Registers

Name	Mnemonic	Address	Section
Miscellaneous Registers	**MxxR**	**Offset**	**8.3**
Command status register	MCSR	1F8[2]	8.3.1
Interrupt status register	MISR	07FFFF.. 040000[3]	8.3.2
Graphics Command Registers	**GxxR**	**Offset[2]**	**8.4**
Slope register 7	GSLR7	13C	8.4.1
Slope register 6	GSLR6	138	8.4.1
Slope register 5	GSLR5	134	8.4.1
Slope register 4	GSLR4	130	8.4.1
Slope register 3	GSLR3	12C	8.4.1
Slope register 2	GSLR2	128	8.4.1
Slope register 1	GSLR1	124	8.4.1
Slope register 0	GSLR0	120	8.4.1
Span width register	GSWR	0BC	8.4.2
Continue register	GCTR	04C	8.4.3
Copy-64 source register	GCSR	160	8.4.4
Copy-64 destination register	GCDR	164	8.4.4
Copy-64A source register	GCASR	360	8.4.5
Copy-64A destination register	GCADR	364	8.4.5
Repeat begin register	GRBR	340	8.4.6
Repeat end register	GRER	350	8.4.6
Graphics Control Registers	**GxxR**	**Offset[2]**	**8.5**
Mode register	GMOR	030	8.5.1
Deep register	GDER	050	8.5.2
Slope-no-go register 7	GSNR7	11C	8.5.3
Slope-no-go register 6	GSNR6	118	8.5.3
Slope-no-go register 5	GSNR5	114	8.5.3
Slope-no-go register 4	GSNR4	110	8.5.3
Slope-no-go register 3	GSNR3	10C	8.5.3
Slope-no-go register 2	GSNR2	108	8.5.3
Slope-no-go register 1	GSNR1	104	8.5.3
Slope-no-go register 0	GSNR0	100	8.5.3

[2] Address = hexadecimal offset into PDBR0 register space

[3] Address = hexadecimal offset into PDBR1 memory space

(continued on next page)

8.1 Overview

Table 8–1 (Cont.) 21130 Registers

Name	Mnemonic	Address	Section
Graphics Control Registers	**GxxR**	**Offset[2]**	**8.5**
Copy buffer register 7	GCBR7	01C	8.5.4
Copy buffer register 6	GCBR6	018	8.5.4
Copy buffer register 5	GCBR5	014	8.5.4
Copy buffer register 4	GCBR4	010	8.5.4
Copy buffer register 3	GCBR3	00C	8.5.4
Copy buffer register 2	GCBR2	008	8.5.4
Copy buffer register 1	GCBR1	004	8.5.4
Copy buffer register 0	GCBR0	000	8.5.4
Pixel shift register	GPSR	038	8.5.5
Address register	GADR	03C	8.5.6
Data register	GDAR	080	8.5.7
Foreground register	GFGR	020	8.5.8
Background register	GBGR	024	8.5.8
Raster operation register	GOPR	034	8.5.9
Pixel mask register (one shot)	GPXR	02C	8.5.10
Pixel mask register (persistent)	GPXR	05C	8.5.10
Bresenham 1 register	GB1R	040	8.5.11
Bresenham 2 register	GB2R	044	8.5.12
Bresenham 3 register	GB3R	048	8.5.13
Bresenham width register	GBWR	09C	8.5.14
DMA base address register	GDBR	098	8.5.15
Scaled-copy control register	GSCR	0C4	8.5.16
Dither row register	GDRR	0B0	8.5.17
Dither column register	GDCR	0B4	8.5.17
Hardware Cursor Registers	**CxxR**	**Offset[2]**	**8.6**
Cursor mode register	CMOR	0EC	8.6.1
Cursor base address register	CCBR	060	8.6.2
Cursor XY register	CXYR	074	8.6.3
Video Control Registers	**VIxxR**	**Offset[2]**	**8.7**
Video base address register	VIVBR	06C	8.7.1
Video scanline increment register	VISIR	0CC	8.7.1
Video line width register	VILWR	0D0	8.7.1

[2]Address = hexadecimal offset into PDBR0 register space

(continued on next page)

Table 8–1 (Cont.) 21130 Registers

Name	Mnemonic	Address	Section
Video Control Registers	**VIxxR**	**Offset[2]**	**8.7**
Video valid register	VIVVR	070	8.7.2
Video Format Registers	**VFxxR**	**Offset[2]**	**8.8**
Video pixel format register	VFPFR	0D4	8.8.1
Video pixel occlusion bitmap base address register	VFOBR	0E0	8.8.2
Video pixel occlusion bitmap current address register	VFOAR	1F4	8.8.3
Video current refresh address register	VFCRR	1FC	8.8.4
Alternate video control register	VFAVR	0E8	8.8.5
Palette and DAC Registers	**DxxR**	**Offset[3]**	**8.9**
Palette and DAC RAM write address register	DPWR	1000	8.9.1
Palette and DAC RAM read address register	DPRR	100C	8.9.1
Palette and DAC RAM color register	DPCR	1004	8.9.2
Palette and DAC cursor write address register	DCWR	1010	8.9.3
Palette and DAC cursor read address register	DCRR	101C	8.9.3
Palette and DAC cursor color register	DCCR	1014	8.9.4
Palette and DAC pixel mask register	DPMR	1008	8.9.5
Palette and DAC status register	DSTR	1028	8.9.6
Palette and DAC command register 0	DCOR0	1018	8.9.7
Palette and DAC command register 1	DCOR1	1030	8.9.8
Palette and DAC red signature register	DRSR	1034	8.9.9
Palette and DAC green signature register	DGSR	1038	8.9.9
Palette and DAC blue signature register	DBSR	103C	8.9.9
VGA External and General Registers	**VExxxR**	**Index[4]**	**8.11**

[2] Address = hexadecimal offset into PDBR0 register space

[3] Address = hexadecimal offset into PDBR1 memory space

[4] Address = hexadecimal address (3xx) or index for VGA registers

(continued on next page)

8.1 Overview

Table 8–1 (Cont.) 21130 Registers

Name	Mnemonic	Address	Section
VGA External and General Registers	**VExxxR**	**Index[4]**	**8.11**
VGA miscellaneous output register	VEMISR	3C2[5] 3CC[6]	8.11.1
VGA feature control register	VEFCOR	3BA[5,7] 3DA[5,8] 3CA[6]	8.11.2
VGA input status 0 register	VEIS0R	3C2[6]	8.11.3
VGA input status 1 register	VEIS1R	3BA[6,7] 3DA[6,8]	8.11.4
VGA Sequencer Registers	**VSxxxR**	**Index[4]**	**8.12**
VGA sequencer index register	VSINXR	3C4	8.12.1
VGA sequencer data register	VSDATR	3C5	8.12.2
VGA sequencer reset register	VSRESR	0	8.12.3
VGA sequencer clocking mode register	VSCMOR	1	8.12.4
VGA sequencer plane mask register	VSPLMR	2	8.12.5
VGA sequencer character map select register	VSCMSR	3	8.12.6
VGA sequencer memory mode register	VSMMOR	4	8.12.7
VGA CRT Controller Registers	**VCxxxR**	**Index[4]**	**8.13**
VGA CRTC index register	VCINXR	3B4[7] 3D4[8]	8.13.1
VGA CRTC data register	VCDATR	3B5[7] 3D5[8]	8.13.2
VGA CRTC horizontal total register	VCHTOR	00	8.13.3
VGA CRTC horizontal display end register	VCHDER	01	8.13.4
VGA CRTC start horizontal blank register	VCHBSR	02	8.13.5
VGA CRTC end horizontal blank register	VCHBER	03	8.13.5
VGA CRTC start horizontal sync register	VCHSSR	04	8.13.6
VGA CRTC end horizontal sync register	VCHSER	05	8.13.6
VGA CRTC vertical total register	VCVTOR	06	8.13.7
VGA CRTC overflow register	VCOVRR	07	8.13.8

[4]Address = hexadecimal address (3xx) or index for VGA registers

[5]Write access only

[6]Read access only

[7]Monochrome

[8]Color

(continued on next page)

Table 8–1 (Cont.) 21130 Registers

Name	Mnemonic	Address	Section
VGA CRT Controller Registers	**VCxxxR**	**Index[4]**	**8.13**
VGA CRTC preset row register	VCPROR	08	8.13.9
VGA CRTC maximum scanline register	VCMSLR	09	8.13.10
VGA CRTC cursor start register	VCCUSR	0A	8.13.11
VGA CRTC cursor end register	VCCUER	0B	8.13.11
VGA CRTC start address high register	VCSAHR	0C	8.13.12
VGA CRTC start address low register	VCSALR	0D	8.13.12
VGA CRTC cursor location high register	VCCLHR	0E	8.13.13
VGA CRTC cursor location low register	VCCLLR	0F	8.13.13
VGA CRTC start vertical sync register	VCVSSR	10	8.13.14
VGA CRTC end vertical sync register	VCVSER	11	8.13.14
VGA CRTC end vertical display register	VCVDER	12	8.13.15
VGA CRTC offset register	VCOFFR	13	8.13.16
VGA CRTC underline row scan register	VCULRR	14	8.13.17
VGA CRTC start vertical blanking register	VCVBSR	15	8.13.18
VGA CRTC end vertical blanking register	VCVBER	16	8.13.18
VGA CRTC mode control register	VCMODR	17	8.13.19
VGA CRTC line compare register	VCLCMR	18	8.13.20
VGA Extended Registers	**VXxxxR**	**or Index[4,9]**	**8.14**
VGA extended paging control register	VXPCOR	8D	8.14.1
VGA extended host page offset A register	VXHPAR	90	8.14.2
VGA extended host page offset B register	VXHPBR	91	8.14.2
VGA extended split-screen start address low byte register	VXSALR	93	8.14.3
VGA extended split-screen start address high byte register	VXSAHR	94	8.14.3
VGA extended interlace control register	VXICOR	97	8.14.4
VGA extended equalization start register	VXEQSR	9A	8.14.5
VGA extended equalization end register	VXEQER	9B	8.14.5
VGA extended half-line register	VXHLNR	9C	8.14.6
VGA extended timing control A register	VXTCAR	9D	8.14.7
VGA extended timing control B register	VXTCBR	9E	8.14.8
VGA extended video FIFO control register	VXFCOR	A0	8.14.9
VGA extended clock control A register	VXCKAR	A1	8.14.10
VGA extended clock control B register	VXCKBR	A2	8.14.10

[4]Address = hexadecimal address (3xx) or index for VGA registers

[9]Indexed by VGA CRTC index register (VCINXR)

(continued on next page)

8.1 Overview

Table 8–1 (Cont.) 21130 Registers

Name	Mnemonic	Address	Section
VGA Extended Registers	**VXxxxR**	**or Index[4,9]**	**8.14**
VGA extended interface control register	VXEICR	A3	8.14.11
VGA Graphics Controller Registers	**VGxxxR**	**Index[4]**	**8.15**
VGA graphics controller index register	VGINXR	3CE	8.15.1
VGA graphics controller data register	VGDATR	3CF	8.15.2
VGA graphics controller set/reset register	VGSRER	0	8.15.3
VGA graphics controller enable set/reset register	VGESRR	1	8.15.4
VGA graphics controller color compare register	VGCCMR	2	8.15.5
VGA graphics controller data rotate register	VGDROR	3	8.15.6
VGA graphics controller read map select register	VGRMSR	4	8.15.7
VGA graphics controller mode register	VGMODR	5	8.15.8
VGA graphics controller miscellaneous register	VGMISR	6	8.15.9
VGA graphics controller color don't care register	VGCDCR	7	8.15.10
VGA graphics controller bit mask register	VGBMKR	8	8.15.11
VGA Attribute Controller Registers	**VAxxxR**	**Index[4]**	**8.16**
VGA attribute controller index/data register	VAIXDR	3C0[5] 3C1[6]	8.16.1
VGA attribute controller palette register	VAPALR	00:0F	8.16.2
VGA attribute controller mode register	VAMODR	10	8.16.3
VGA attribute controller overscan register	VAOSCR	11	8.16.4
VGA attribute controller color plane enable register	VACPER	12	8.16.5
VGA attribute controller pixel panning register	VAPXPR	13	8.16.6
VGA attribute controller color select register	VACSLR	14	8.16.7

[4] Address = hexadecimal address (3xx) or index for VGA registers

[5] Write access only

[6] Read access only

[9] Indexed by VGA CRTC index register (VCINXR)

(continued on next page)

Table 8–1 (Cont.) 21130 Registers

Name	Mnemonic	Address	Section
VGA Attribute Controller Registers	**VAxxxR**	**Index[4]**	**8.16**
VGA Color Registers	**VPxxxR**	**Index[4]**	**8.17**
VGA color pixel address write mode register	VPPAWR	3C8	8.17.1
VGA color pixel address read mode register	VPPARR	3C7[5]	8.17.1
VGA color DAC state register	VPDSTR	3C7[6]	8.17.2
VGA color pixel data register	VPPDAR	3C9	8.17.3
VGA color pixel mask register	VPPMAR	3C6	8.17.4

[4]Address = hexadecimal address (3xx) or index for VGA registers

[5]Write access only

[6]Read access only

8.2 PCI Configuration Registers

All PCI devices require PCI configuration registers. PCI configuration registers identify the device and vendor, soft-map the device in I/O or memory space, and specify the allowed modes of operation for the device as a PCI master and target.

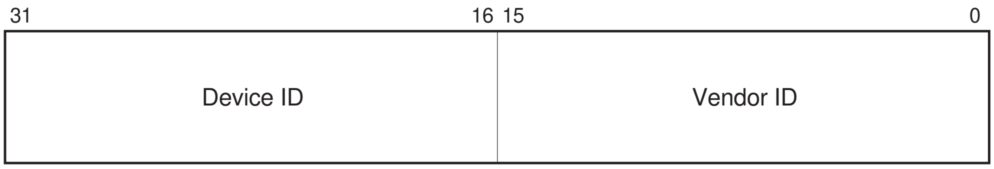
The configuration space occupies 256 bytes, divided as follows:

- 64 bytes (00..3F) for the device-independent, configuration-space header block
- 192 bytes (40..FF) for device-specific registers

All unused configuration space addresses are reserved (Table 8–1). (See the *PCI Local Bus Specification, Revision 2.0* for more information about the PCI configuration space and function codes.)

8.2.1 PCI Identification Register

Mnemonic:	PIDR
Byte address range:	00..03
Reset value:	000C1011



Bits	Field	Access	Description
31:16	Device ID	RO	When read, returns $000C_{16}$ to identify the 21130 as the device.
15:0	Vendor ID	RO	When read, returns 1011_{16} to identify Digital as the vendor.

The read-only PIDR identifies the vendor and device to system software. Writes to this register are ignored.

8.2 PCI Configuration Registers

8.2.2 PCI Command and Status Register

Mnemonic:	PCSR
Byte address range:	04..07
Reset value:	02800000

31	30	29	28	27	26 25	24	23	22 ... 10	9	8	7	6	5	4 3	2	1	0
DPE	SSE	MA	TAM	TAT	DEV	DPD	BBC	RAZ	BBE	SEN	ASE	PER	VPS	RAZ	BM	MS	IO

Bits	Field	Access	Description
31	DPE	R/W1C	Detected parity error—set when the 21130 detects a parity error, regardless of the state of the PER bit (<6>).
30	SSE	R/W1C	Signaled system error—set when the 21130 sets the **pci_serr#** signal.
29	MA	R/W1C	Master abort—set when the 21130 issues a master-abort termination.
28	TAM	R/W1C	Target abort, master—set when the 21130 detects a target-abort termination while acting as a bus master.
27	TAT	R/W1C	Target abort, target—set when the 21130 terminates a transaction with a target-abort.
26:25	DEV	RO	Device select timing—hard-wired to code 01 to indicate that the 21130 has a medium response time to PCI device select.
24	DPD	R/W1C	Data parity error detected—set when the **pci_perr#** signal is asserted, <6> is set, and the 21130 is bus master.
23	BBC	RO	Back-to-back capable—hard-wired to 1 to indicate that the 21130 can handle fast back-to-back PCI transactions as a target.
22:10	RAZ	RO	Reads as zero, ignored on writes, reserved.
9	BBE	RO	Back-to-back enable—hard-wired to 0 to indicate that the 21130 cannot generate fast back-to-back PCI transactions.

Bits	Field	Access	Description
8	SEN	RW	**pci_serr#** enable 0 The **pci_serr#** pin driver is disabled. 1 The **pci_serr#** pin driver is enabled. This bit and <6> must be set to report parity errors.
7	ASE	RO	Address stepping enable—hard-wired to 0 to indicate that the 21130 never does address stepping.
6	PER	RW	Parity error response—enables parity error reporting. 0 Parity error reporting is disabled. 1 Address parity errors are signaled on the **pci_serr#** pin and data parity errors are signaled on the **pci_perr#** pin.
5	VPS	RW	VGA palette snoop 0 The 21130 responds normally to writes to VGA color register space. 1 The 21130 snoops writes to VGA color register space.
4:3	RAZ	RO	Reads as zero, ignored on writes, reserved.
2	BM	RW	Bus master enable 0 The 21130 cannot become bus master. 1 The 21130 is enabled to become bus master.
1	MS	RW	Memory space enable 0 Response to memory space accesses is disabled. 1 Response to memory space accesses is enabled.
0	IO	RW	I/O space enable 0 Response to I/O space accesses is disabled. 1 Response to I/O space accesses is enabled.

The PCSR controls and indicates the status of several PCI functions, including parity error reporting.

The master abort and target abort bits (<29:28>) are set when the 21130 detects or issues the respective transaction terminations. These bits remain set until software explicitly clears them by writing a 1 to the bit (writing a 0 is ignored). The 21130 issues a master abort if a target does not respond by asserting the **pci_devsel#** signal. The 21130 issues a target abort if a VGA register is accessed with an invalid byte mask.

8.2 PCI Configuration Registers

When the back-to-back capable bit (<23>) is set, the 21130 responds to fast back-to-back PCI transactions as a target. As a master, the 21130 does not perform fast back-to-back cycles and the back-to-back enable bit (<9>) is hard-wired to 0.

When a parity error is detected, the 21130 signals the error on the **pci_serr#** pin if the error occurred during an address transaction, or on the **pci_perr#** pin if the error occurred during a data transaction. The 21130 continues to operate normally; that is, if the address is a valid 21130 address, it is used, along with the subsequent data. If a data transaction had the error, the erroneous data will be used for a write.

The VGA palette snoop bit (<5>) determines how the 21130 responds to VGA color register (palette) writes. When the bit is set, the 21130 snoops; that is, it transparently accepts write data but does not explicitly respond to write transactions. When the bit is clear, the 21130 responds normally to VGA color register writes; that is, as it does to any other write to its address space. Palette writes are I/O space writes to addresses 3C6, 3C8, and 3C9. The 21130 never actively responds to an I/O access if the I/O space enable bit (<0>) is clear.

Table 8–2 summarizes the 21130 response according to the value of bits <5,0>.

Table 8–2 Palette Snoop Response

Bit		Writes			
5	**0**	**Palette**	**Other**	**Reads**	**Mode Description**
0	0	NR	NR	NR	Completely shut down
0	1	AR	AR	AR	Fully active
1	0	PS	NR	NR	Shut down and snooping
1	1	PS	AR	AR	Fully active and snooping

Abbreviations

NR No response to the access
AR Active response to the access
PS Passive snoop, no active response to the access

Note

The 21130 does not snoop a transaction that was terminated with a master abort.

The master enable bit (<2>) must be set in order to invoke any 21130 DMA graphics operation.

The 2D accelerator address space can be mapped only into PCI memory space. The 21130 responds to PCI memory accesses in its 32MB address space when the memory space enable bit (<1>) is set.

At reset, the value of the PCSR is 02800000_{16}. The VPS, MS, and IO bits are clear, and the DEV, BBC, BBE, and ASE bits return their hard-wired values.

8.2.3 PCI Class and Revision Register

Mnemonic:	PCRR
Byte address range:	08..0B
Reset value:	03000002

31 — 24	23 — 16	15 — 8	7 — 0
Base Class	Subclass	Programming Interface	Revision ID

Bits	Field	Access	Description
31:24	Base Class	RO	Hard-wired to 03_{16} to indicate that the 21130 device base class is display controller.
23:16	Subclass	RO	Hard-wired to 00_{16} to indicate that the 21130 device subclass is VGA controller.
15:8	Programming Interface	RO	Hard-wired to 00_{16} to indicate that the 21130 supports the VGA programming interface.
7:0	Revision ID	RO	Hard-wired as follows: 00_{16} Revision A devices (DC7538A) 01_{16} Revision B devices (DC7538B) 02_{16} Revision C devices (DC7538C) This field is used when the device revision is not a subfield of the device ID (PIDR <15:0>).

The PCRR identifies the 21130 revision number, device class (base class and subclass), and any compatible register-level programming interfaces.

PCI power-on self-test (POST) code reads the device class information to determine whether the 21130 is suitable as a boot display device. The programming interface (<15:8>) indicates the register-level programming standard supported by the 21130.

At reset, the value of the PCRR is 03000002_{16}. All fields return their hard-wired values.

8.2.4 PCI Latency Timer and Header Type Register

Mnemonic: PLTR
Byte address range: 0C..0F
Reset value: 00000000

31 24	23 16	15 8	7 0
RAZ	Header Type	Latency Timer	RAZ

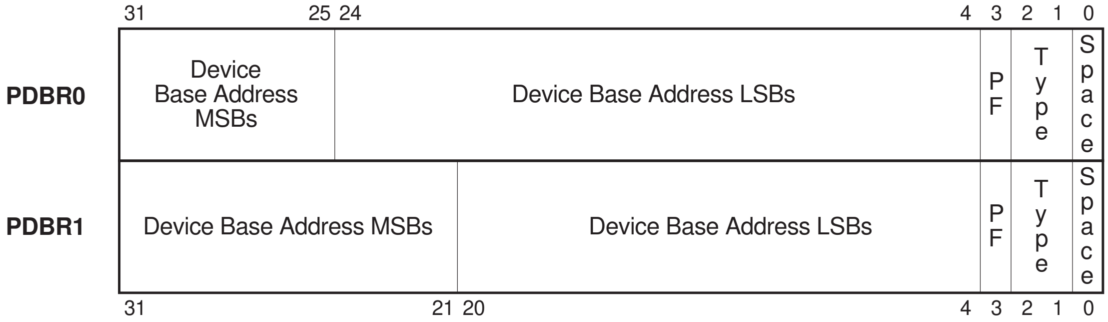
Bits	Field	Access	Description
31:24	RAZ	RO	Reads as zero, ignored on writes, reserved.
23:16	Header Type	RO	Hard-wired to 00_{16} to identify the 21130 as a single-function device.
15:8	Latency Timer	RW	21130 bus ownership is limited to the number of PCI clocks specified in this field.
7:0	RAZ	RO	Reads as zero, ignored on writes, reserved.

The PLTR identifies the type of configuration-space header block in the 21130 configuration space. It also specifies the length of time that the 21130 retains bus ownership in the presence of other bus requests.

At reset, the value of the PLTR is 00000000_{16}. The latency timer field is cleared and the header type field returns its hard-wired value.

8.2.5 PCI Device Base Address Registers

Mnemonic:	PDBR0, PDBR1
PDBR0 byte address range:	10..13
PDBR1 byte address range:	14..17
PDBR0 reset value:	00000008
PDBR1 reset value:	00000000



Bits	Field	Access	Description
PDBR0			
31:25	Device Base Address MSBs	RW	This field can be used to relocate this 21130 address space to any location that is aligned to 32MB.
24:4	Device Base Address LSBs	RO	Hard-wired to 000000_{16} to indicate that this base address must be aligned to 32MB or greater.
3	PF	RO	Prefetchable—hard-wired to 1 to indicate that this 21130 address space is prefetchable.
2:1	Type	RO	Hard-wired to code 00 to indicate that this 21130 address space can be mapped anywhere within the 32-bit address space.
0	Space	RO	Hard-wired to 0 to specify that this 21130 address space can be mapped only into PCI memory space.

Bits	Field	Access	Description
PDBR1			
31:21	Device Base Address MSBs	RW	This field can be used to relocate this 21130 address space to any location that is aligned to 2MB.
20:4	Device Base Address LSBs	RO	Hard-wired to 0000_{16} to indicate that this base address must be aligned to 2MB or greater.
3	PF	RO	Prefetchable—hard-wired to 0 to indicate that this 21130 address space is not prefetchable.
2:1	Type	RO	Same as PDBR0.
0	Space	RO	Same as PDBR0.

8.2.5.1 PDBR0 Functions

The 21130 accelerator-specific address space is mapped to the location in memory space specified in the PDBR0. Configuration firmware can map this address space into any naturally-aligned, contiguous, 32MB (or larger) region.

The value of the prefetchable bit (<3> = 1) indicates that there are no side effects on reads to this 21130 address space. The 21130 returns all bytes on reads regardless of the byte enables. However, host bridges must not collapse or resequence writes into this region, because side effects might cause errors. For example, consider a repeated write to frame buffer memory in simple mode with the XOR Boolean operation. If the writes are collapsed, only one of the writes will occur, leaving the pixel in the wrong state. For another example, consider setting up a line-drawing operation. This involves a specific sequence. If reordering hardware, such as a write buffer, resequences the steps, the line will not be drawn as intended.

At reset, the value of the PDBR0 is 00000008_{16}. The device base address MSB field is cleared and the device base address LSB, PF, type, and space fields return their hard-wired values.

8.2.5.2 PDBR1 Functions

The PDBR1 maps VGA alternate register space (Section 7.5.2.1), generic peripheral port (GPP) register space (Section 7.5.2.2), interrupt status register (MISR) space (Section 7.5.2.3), and DAC register space (Section 7.5.2.4).

The value of the prefetchable bit (<3> = 0) indicates that there are side effects on reads to this 21130 address space. The side effects are due to index registers in the VGA alternate register space and DAC registers that change value when read.

At reset, the value of the PDBR1 is 00000000_{16}. The device base address MSB field is cleared and the device base address LSB, PF, type, and space fields return their hard-wired values.

8.2.6 PCI Expansion ROM Base Address Register

Mnemonic:	PRBR
Byte address range:	30..33
Reset value:	00000000

31 … 18	17 … 11	10 … 1	0
ROM Base Address MSBs	ROM Base Address LSBs	RAZ	DE

Bits	Field	Access	Description
31:18	ROM Base Address MSBs	RW	The ROM base address MSBs.
17:11	ROM Base Address LSBs	RO	Hard-wired to 00_{16} to indicate that the ROM base address must be aligned to 256KB or greater.
10:1	RAZ	RO	Reads as zero, ignored on writes, reserved.
0	DE	RW	Decode enable 1 ROM access decoding is enabled. 0 ROM access decoding is disabled.

The PRBR maps the ROM space and supports a ROM size up to 256KB. The ROM must be mapped on naturally aligned 256KB boundaries. The 21130 responds to all accesses in the 256KB ROM space if the decode enable (<0> in this register) and the memory space enable bit (PCSR <1>, Section 8.2.2) are set.

At reset, the value of the PRBR is 00000000_{16}. The ROM base address MSB field and DE bit are cleared, and the ROM base address LSB field returns its hard-wired value.

8.2.7 PCI Interrupt Line Register

Mnemonic:	PLIR
Byte address range:	3C..3F
Reset value:	00040100

31 — 24	23 — 16	15 — 8	7 — 0
Maximum Latency	Minimum Grant Time	Interrupt Pin	Interrupt Line

Bits	Field	Access	Description
31:24	Maximum Latency	RO	Hard-wired to 00_{16} to indicate that the 21130 has no maximum latency requirements.
23:16	Minimum Grant Time	RO	Hard-wired to 04_{16} to indicate that the 21130 requires a burst length of 1 µs to efficiently use the PCI bandwidth.
15:8	Interrupt Pin	RO	Hard-wired to 01_{16} to indicate that 21130 signals interrupts on the **pci_inta#** pin.
7:0	Interrupt Line	RW	The 21130 **pci_inta#** pin is tied to the system interrupt controller input identified by this field. POST firmware initializes this field.

The PLIR provides hardware support for POST firmware interrupt configuration and identification. The 21130 has only one physical interrupt pin, and POST firmware sets the interrupt line field (<7:0>).

At reset, the value of the PLIR is 00040100_{16}. The interrupt line field is clear and the maximum latency, minimum grant time, and interrupt pin fields return their hard-wired values.

8.2.8 PCI Clock Control Register

Mnemonic:	PCCR
Byte address range:	40..43
Reset value:	00001C0X

31 … 14	13 … 8	7 … 4	3	2	1	0
RAZ	M Term	RAZ	SRES	TCS	PCS	MCS

Bits	Field	Access	Description
31:14	RAZ	RO	Reads as zero, ignored on writes, reserved.
13:8	M Term	RW	Sets the memory clock PLL multiplier (Table 8–3). At reset, the value of this field is 1C and the memory clock is 50.1136 MHz.
7:4	RAZ	RO	Reads as zero, ignored on writes, reserved.
3	SRES	RW	Soft reset—allows software to reset the 21130. 0 21130 normal operation. 1 Resets the 21130. All registers, except the PCI configuration registers, are forced to their reset state.
2	TCS	RW	Test clock source—determines the test clock (**pll_test**) source. 0 Memory clock (see bit <0>) 1 Pixel clock (see bit <1>)
1	PCS	RW	Pixel clock source—determines the pixel clock input source. 0 Internal PLL 1 **pixclk** pin When PCI reset (**pci_rst#**) is asserted, this bit is forced to the inverse of the **gp_int#** signal.
0	MCS	RW	Memory clock source—determines the memory clock input source. 0 Internal PLL 1 **xtal2** pin When PCI reset (**pci_rst#**) is asserted, this bit is forced to the inverse of the **gp_int#** signal.

8.2 PCI Configuration Registers

The PCCR configures the memory clock; selects the memory, pixel, and test clock sources; and enables software to reset the 21130. Note that the clock control A and B registers (VXCKAR and VXCKBR, Section 8.14.10) configure the pixel clock and determine the VGA dot clock source. The video valid register also contains bits that control the test clock (VIVVR <13:12,10>, Section 8.7.2). See Section 12.5 for more information about the clock generation function.

Table 8–3 lists the memory clock frequencies.

Table 8–3 Memory Clock Frequency Select

M[1]	MHz[2]	M	MHz	M	MHz	M	MHz
0F	26.8466	16	39.3750	1D	51.9034	24	64.4318
10	28.6364	17	41.1648	1E	53.6932	25	66.2216[3]
11	30.4261	18	42.9545	1F	55.4830	26	68.0114
12	32.2159	19	44.7443	20	57.2727	27	69.8011
13	34.0057	1A	46.5341	21	59.0625	28	71.5909
14	35.7955	1B	48.3239	22	60.8523	29	73.3807
15	37.5852	1C	50.1136[4]	23	62.6420	2A	75.1705

[1]M term specified in PCCR <13:8>. M values 00..0E and 2B..3F are reserved.

[2]Memory clock frequency in MHz.

[3]Design center.

[4]Reset value.

8.3 Miscellaneous Registers

The miscellaneous registers return information about the current status of chip processing and pending interrupts, and enable interrupts; provide a synchronization mechanism for scheduling commands; and control the pixel clock.

The MCSR is one of the core registers mapped in base address 0 memory space (Section 7.5.1.2) by the PDBR0. The MISR is mapped in base address 1 memory space (Section 7.5.2) by the PDBR1.

8.3.1 Command Status Register

Mnemonic: MCSR
Offset: 1F8
Reset value: Cleared

31 22	21 16	15 14	13 8	7 1	0
RES	FIFO Write Pointer	RES	FIFO Read Pointer	RES	Busy

Bits	Field	Access	Description
31:22	RES	MBZ	Reserved, must be zero. Read value is unpredictable.
21:16	FIFO Write Pointer	RO	Indicates the next command FIFO location to be written. Forced to zero when the 21130 is idle (bit <0> is clear).
15:14	RES	MBZ	Reserved, must be zero. Read value is unpredictable.
13:8	FIFO Read Pointer	RO	Indicates the next command FIFO location to be read. Forced to zero when the 21130 is idle (bit <0> is clear).
7:1	RES	MBZ	Reserved, must be zero. Read value is unpredictable.
0	Busy	RO	Indicates 21130 busy status. 0 The 21130 is idle. 1 The 21130 is processing commands from the command FIFO.

When read, the read-only MCSR returns the state of the busy bit (<0>). The busy bit indicates whether the chip is processing commands or has completed all command processing and the command FIFO is empty.

8.3 Miscellaneous Registers

In addition to the PCI configuration registers and the VGA registers, the MCSR is one of the few registers that is immediately accessible for read (see Section 9.2.2.1). In other words, the command FIFO does not have to be flushed before completing a read of the MCSR.

8.3.1.1 Write Memory Barrier

The 21130 is optimized as a primarily write-only device, and it implements pipelined processing. In typical graphics operations, the driver can stream writes and commands to the chip without overflowing the command FIFO. The 21130's PCI retry mechanism combined with short command processing times prevents most writes from stalling. Hardware retries any writes that do stall and software polling is usually unnecessary.

However, in many cases software should poll the busy bit and wait for the 21130 to become idle before continuing. Although the 21130 provides hardware interlocks to ensure coherency for most operations (such as holding a frame buffer read until the write buffer is flushed), waiting for the 21130 to be idle is necessary for unsupported interlocks and to synchronize hardware and software processing.

The following situations are examples of when it is practical or necessary for software to wait for the 21130 to be idle. The length of time to wait depends on the specific situation.

- To avoid unnecessary retries on the PCI bus while long commands complete.
- On any write to the deep register (GDER).
- When loop commands are used with long graphics commands.

The MCSR acts as a write memory barrier. The 21130 inserts a write to the MCSR into the command FIFO as a flag to ensure that preceding commands and writes are completely processed before subsequent commands and writes are unloaded from the command FIFO. The command parser unloads commands and writes from the command FIFO, performs some initial processing, and then passes graphics processing requests to the pixel pipeline (Section 2.3). The command parser provides a hardware interlock mechanism to ensure that any writes it processes do not affect processing in progress downstream in the pipeline. The MCSR interlock mechanism is an additional precaution, in case the hardware interlock fails or cannot handle a particular operation. (See Section 10.1.4 for more information.)

8.3.2 Interrupt Status Register

Mnemonic:	MISR
Offset:	07FFFF..040000
Reset value:	Cleared

31	30 22	21	20 17	16	15 6	5	4 1	0
INTR	RES	GIEN	RES	EOFEN	RES	GIST	RES	EOFST

Bits	Field	Access	Description
31	INTR	RO	Interrupt status—logical OR of interrupt-enable-masked version of GST and EOFST (<5,0>).
30:22	RES	MBZ	Reserved, must be zero. Read value is unpredictable.
21	GIEN	RW	GPP interrupt enable—set to enable interrupts on the GPP interrupt pin (**gp_int#**).
20:17	RES	MBZ	Reserved, must be zero. Read value is unpredictable.
16	EOFEN	RW	End-of-frame enable—set to enable end-of-frame interrupts.
15:6	RES	MBZ	Reserved, must be zero. Read value is unpredictable.
5	GIST	R/W1C	GPP interrupt status—set when the **gp_int#** pin is asserted. Writing a 1 to this bit clears it.
4:1	RES	MBZ	Reserved, must be zero. Read value is unpredictable.
0	EOFST	R/W1C	End-of-frame interrupt status—set when end-of-frame is reached. Writing a 1 to this bit clears it.

The MISR is located in base address 1 space, rather than in base address 0 space (that is, command buffered space) with the other core registers, to accommodate interrupts that might occur during a loop operation. This allows the interrupt service routine to clear the interrupt status bits, without disturbing the contents of the command FIFO.

When an interrupt occurs, the corresponding interrupt status bit is set whether the corresponding enable bit is set or clear; however, the **pci_inta#** signal is asserted only if both the status and enable bits are set.

End-of-frame interrupts can be enabled in the VGA input status 0 register (VEIS0R <7>, Section 8.11.3) as well as in the MISR. Digital recommends that software use the MISR when operating in modes that are not VGA-compatible, because one read of the MISR returns the source of a 21130 interrupt.

8.3 Miscellaneous Registers

In addition to the PCI configuration registers and the VGA registers, the MISR is one of the few registers that is immediately accessible for read (see Section 9.2.2.1). In other words, the command FIFO does not have to be flushed before completing a read of the MISR.

8.4 Graphics Command Registers

The graphics command registers are part of the core registers mapped in base address 0 memory space (Section 7.5.1.2) by the PDBR0.

The 21130 accelerated graphics operations are selected by specifying a mode in the mode register (GMOR, Section 8.5.1) and initiated by a write to either of the following:

- The frame buffer address space (standard drawing mechanism)

  The chip is set to a specific mode and the frame buffer is written directly. The address and data are interpreted according to the mode.

- Any graphics command register (alternate drawing mechanism)

  The graphics software initiates a drawing operation by writing to a graphics command register.

The graphics command registers are the only 21130 registers that initiate a drawing action when written. They provide a faster and simpler mechanism to draw, extend, and link lines, and copy large spans. They also allow software to indirectly address the frame buffer.

8.4 Graphics Command Registers

8.4.1 Slope Registers

Mnemonic:	GSLR<7:0>
GSLR<7:0> offsets:	13C, 138, 134, 130, 12C, 128, 124, 120
GSLR<7:0> reset value:	Undefined

31 16	15 0
Absolute dy	Absolute dx

Bits	Field	Access	Description
31:16	Absolute dy	WO	An unsigned integer equal to the absolute value of the difference in *y* of the two line endpoints.
15:0	Absolute dx	WO	An unsigned integer equal to the absolute value of the difference in *x* of the two line endpoints.

Note

The Bresenham width register (GBWR, Section 8.5.14) must be written before writing a GSLR.

The write-only GSLRs initialize the internal Bresenham engine for line drawing. On a write to a GSLR, the following Bresenham terms are automatically calculated as a function of absolute *dx* and absolute *dy*.

- Initial error

  The 16-bit signed initial value stored in the Bresenham engine error accumulator.

- Length

  A 4-bit value specifying the number of pixels to be drawn in this line segment.

- Error Increment 1

  The positive value added to the error term when the Bresenham error term is < 0 (a major axis step).

- Address Increment 1

  The signed value added to the current address when the Bresenham error term is < 0 (a major axis step).

- Error Increment 2

  The positive value subtracted from the error term when the Bresenham error term is ≥ 0 (a step along the major and minor axes).

- Address Increment 2

  The signed value added to the current address when the Bresenham error term is ≥ 0 (a step along the major and minor axes).

Each GSLR is associated with one of the drawing octants (Figure 8–1), and each specifies a slope in terms of the absolute values of the rise in *y* (absolute *dy*) and the run in *x* (absolute *dx*). Results are undefined if both absolute *dy* and absolute *dx* are zero. Software must filter out zero-length lines. (Section 10.2.9.2 includes the algorithm for calculating the Bresenham terms.)

On a write to a GSLR, the pixel length of the line segment is initialized to the major axis length MOD 16 (GB3R <3:0>, Section 8.5.13). This means that the 21130 is initialized to draw up to 16 pixels when the GSLR is written. For example, if the major axis length (the greater of absolute *dx* and absolute *dy*) is 19, the GSLR initializes the pixel length to 3. When used with the continue register (GCTR, Section 8.4.3), this feature allows software to draw lines of arbitrary length without monitoring the length of each segment. If the line to be drawn is not an exact multiple of 16 pixels, the shorter line (length MOD 16) is drawn first, and the line is completed with successive writes to the GCTR (which always draws 16 pixels).

Depending on the graphics environment (GMOR <13>, Section 8.5.1), writing a GSLR sets up the Bresenham terms correctly for all X-compliant lines and most lines that comply with Windows NT. The GSLRs create the correct initial terms for lines drawn under Windows NT *only* if the following criteria are met:

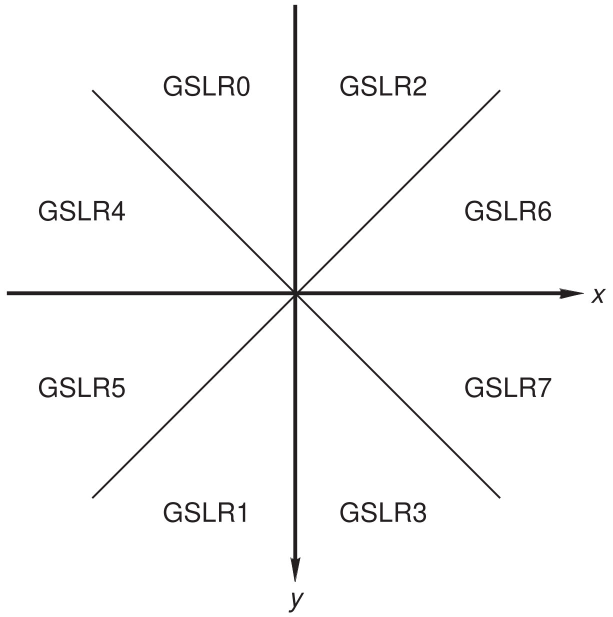
- The endpoint coordinates of the line are integers.
- The length of the line, as measured by the run of the line along the major axis, is limited to 64K -1 pixels.

In general, lines that have subpixel endpoints and clipped lines cannot be drawn with the GSLRs; the slope-no-go registers (GSNR<7:0>, Section 8.5.3) and GCTR can be used to draw such lines.

Figure 8–1 shows the slope register associated with each of the drawing octants.

8.4 Graphics Command Registers

Figure 8–1 Slope Registers and Drawing Octants



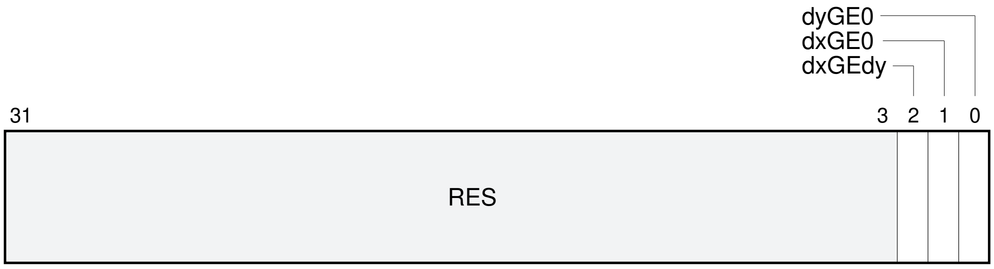
See Section 10.2.9.2 for more information about using the GSLRs to draw lines.

8.4.2 Span Width Register

Mnemonic: GSWR
Offset: 0BC
Reset value: Cleared

The function of the GSWR depends on whether it is being read (Section 8.4.2.1) or written (Section 8.4.2.2).

8.4.2.1 GSWR Read



Bits	Field	Access	Description
31:3	RES	MBZ	Reserved, must be zero. Read value is unpredictable.
2	dxGEdy	RO	Set when the absolute value of the run parameter (*dx*) is greater than or equal to the absolute value of the rise parameter (*dy*); otherwise, clear (*dx* is less than *dy*).
1	dxGE0	RO	Set when the run (*dx*) of the slope is greater than or equal to 0 (*dx* is positive); otherwise, clear (*dx* is negative).
0	dyGE0	RO	Set when the rise (*dy*) of the slope is greater than or equal to 0 (*dy* is positive); otherwise, clear (*dy* is negative).

On a read, the GSWR returns parameters that show the state of the internal Bresenham engine.

The slope parameters are generated by the Bresenham engine on a write to the GSWR, the slope registers, or the slope-no-go registers. (See Section 10.2.9 for more information about the algorithm that generates the slope parameters.)

8.4.2.2 GSWR Write

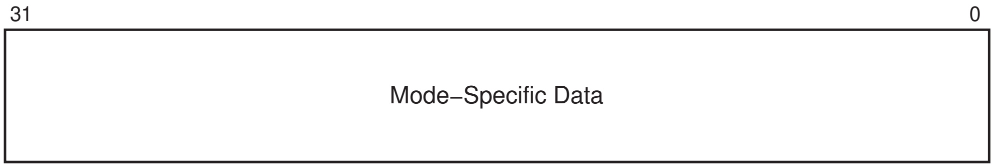
On a write, the GSWR is an alias for slope register 7 (GSLR7), with the same format and field descriptions (Section 8.4.1). The GSWR can be used to draw spans when absolute *dy* is 0.

8.4.3 Continue Register

Mnemonic: GCTR
Offset: 04C
Reset value: Cleared

The function of the GCTR depends on whether it is being written (Section 8.4.3.1) or read (Section 8.4.3.2).

8.4.3.1 GCTR Write



Bits	Field	Access	Description
31:0	Mode-Specific Data	WO	Same as the mode-specific PCI write-data formats described in Chapter 10.

On a write, the two primary functions of the GCTR are to indirectly address the frame buffer and continue a line or span for an additional 16 pixels without recomputing and reloading parameters.

A PCI write to the 21130 frame buffer space usually initiates a drawing action. The address used for the operation is the frame buffer address of the write, and the PCI write data is interpreted according to the drawing mode. Alternatively, software can initiate mode-dependent operations by writing the GCTR, and indirectly specify the frame buffer address. Writes to the GCTR are interpreted exactly the same as writes to the frame buffer.

The GCTR mode-specific data (<31:0>) has the mode-dependent format of the frame buffer PCI write-data except in line mode.

8.4 Graphics Command Registers

GCTR Write in Line Mode

31 – 16	15 – 0
RES	Line Mask

Bits	Field	Access	Description
31:16	RES	MBZ	Reserved, must be zero. Read value is unpredictable.
15:0	Line Mask	WO	Mask or stipple for the next 16-pixel line segment.

The format of GCTR mode-specific data in line mode is the same as the PCI write-data format in opaque-line mode (Section 10.2.9), except that the GCTR data does not include the address LSBs (**pci_ad<1:0>**). These bits are unnecessary because the address is fully contained in the address register (GADR, Section 8.5.6). See Chapter 10 for more information about using the GCTR to extend lines.

Indirect Frame Buffer Addressing

If the GADR was written since the previous operation, the GCTR will take the frame buffer address from the GADR and initiate a graphics operation.

Line or Span Continuation

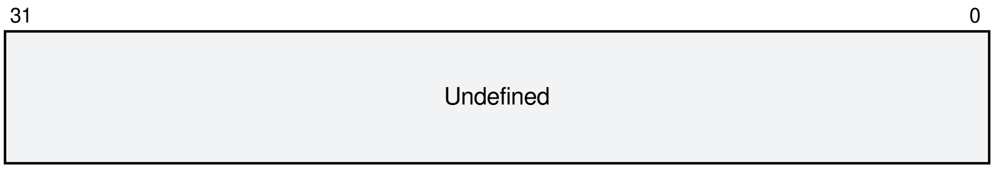
If the GADR is purposely not written before initiating a line mode operation, the GCTR can be written to effectively extend, or continue, the line drawn immediately prior to the current operation, using the address in the 21130 internal addressing hardware.

At the completion of a line or span drawing operation, the 21130 leaves its internal line-drawing hardware in a state that allows a subsequent line-mode operation to continue where the preceding line-mode operation stopped. That state includes at least the frame buffer address, and can also include the Bresenham error terms, depending on the specific line mode. Therefore, the GCTR can quickly and easily extend the previous line-mode operation. For example, a write to a slope register will set up and draw 16 pixels along a line. After the initial write to the slope register, software can simply write the GCTR twice to extend the line or span to a length of 48 pixels.

Writes to Alternate Control Space

The GCTR and GADR are mapped sequentially on writes to the alternate control space in the 21130 PCI memory space. Basically, software can alternately write the GCTR and GADR by writing sequential locations in the otherwise read-only alternate control space. This method of sequential access can help make effective use of the write buffer in an Alpha CPU. (See Sections 7.5.1.3 and 11.12 through 11.12.2 for more information about alternate control space access and mapping.)

8.4.3.2 GCTR Read



Bits	Field	Access	Description
31:0	Undefined	RO	Undefined.

On a read in any mode, the GCTR returns possibly undefined data.

8.4.4 Copy-64 Source and Destination Registers

Mnemonic:	GCSR, GCDR
GCSR offset:	160
GCDR offset:	164
GCSR, GCDR reset value:	Cleared

	31 … 22	21 … 3	2 … 0
GCSR	RES	Frame Buffer Address Source	IGN
GCDR	RES	Frame Buffer Address Destination	IGN

Bits	**Field**	**Access**	**Description**
GCSR			
31:22	RES	MBZ	Reserved, must be zero. Read value is unpredictable.
21:3	Frame Buffer Address Source	WO	Frame buffer byte address of the source. The 8-byte-aligned base address of the 64-byte span to be loaded into the 21130 copy buffer.
2:0	IGN	WO	Ignored when written.
GCDR			
31:22	RES	MBZ	Reserved, must be zero. Read value is unpredictable.
21:3	Frame Buffer Address Destination	WO	Frame buffer byte address of the destination. The 64-byte span will be copied from the 21130 copy buffer to the destination starting at this 8-byte-aligned address.
2:0	IGN	WO	Ignored when written.

The GCSR and the GCDR are used together to perform fast, simple copies of aligned, unmasked, 64-byte spans. Both registers are write-only.

Note

Before writing the copy buffer registers, copy mode must be selected (GMOR <7:0>, Section 8.5.1).

A write to the GCSR must be matched with a write to the GCDR; otherwise, the copy buffer will be left in an undefined state.

A write to the GCSR initiates a fill from the frame buffer to the onchip 64-byte copy buffer, beginning at the frame buffer address source (GCSR <21:3>). A subsequent write to the GCDR unloads the contents of the copy buffer into the frame buffer, beginning at the frame buffer address destination (GCDR <21:3>). The frame buffer source and destination addresses must be aligned to 8 bytes.

Writing the frame buffer address of the source span to the GCSR and then writing the frame buffer address of the destination span to the GCDR effectively copies a 64-byte span from an 8-byte-aligned source to an 8-byte-aligned destination.

Copying 8-bpp bitmaps with the GCSR and GCDR, which copies 64 pixels at a time, is faster than copying with writes to the frame buffer, which copies only 32 pixels at a time. However, the GCSR and GCDR can be used to copy only unmasked spans in which the source and destination are aligned to 8 bytes. Therefore, the GCSR and GCDR are used primarily to copy the interiors of large spans. Given an arbitrary source and destination, addresses are not likely to be aligned to 8 bytes. In such cases, the edges of the span must be copied with writes to the frame buffer in standard copy mode. The GCSR and GCDR can then be used to quickly fill the remaining 8-byte-aligned interior of the span.

Although the 21130 does not support masking when using the GCSR and GCDR, it does shift pixel data to support copies in which the source and destination are unaligned. Pixel data is shifted as specified in the pixel shift register (GPSR, Section 8.5.5).

8.4.5 Copy-64A Source and Destination Registers

Mnemonic:	GCASR, GCADR
GCASR offset:	360
GCADR offset:	364
GCASR, GCADR reset value:	Cleared

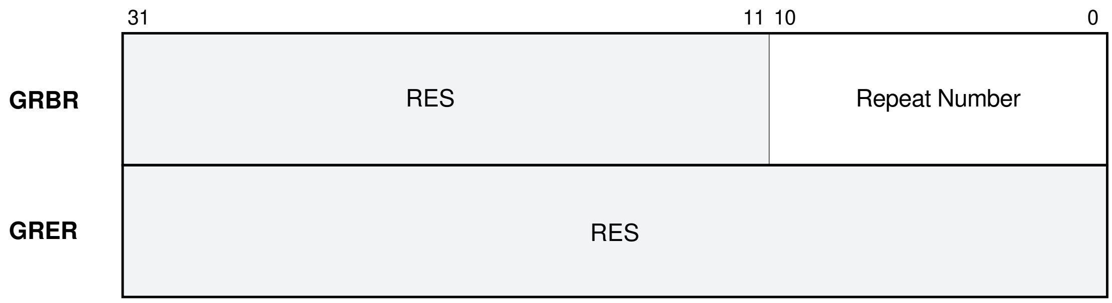
	31 — 0
GCASR	RES
GCADR	RES

Bits	Field	Access	Description
31:0	RES	MBZ	Reserved, must be zero. Read value is unpredictable.

The GCASR and the GCADR are used the same way as the copy-64 source and destination registers (GCSR and GCDR, Section 8.4.4), and both registers are write-only. Writes to the copy-64A registers cause 64 bytes of frame buffer data to be read into and written from the copy buffer in the same way as writes to the copy-64 registers. However, unlike operations initiated by writes to the copy-64 registers, writes to the copy-64A registers use the address register (GADR, Section 8.5.6) to define the starting address for the operation. The copy-64A registers enable the use of 64-byte 8-bpp copy transfers within repeat loops.

8.4.6 Repeat Begin and End Registers

Mnemonic:	GRBR, GRER
GRBR offset:	340
GRER offset:	350
GRBR, GRER reset value:	Cleared



Bits	Field	Access	Description
GRBR			
31:11	RES	MBZ	Reserved, must be zero. Read value is unpredictable.
10:0	Repeat Number	WO	The number of times to repeat the loop defined by the GRBR and GRER. A value of 0 causes one execution of the loop.
GRER			
31:0	RES	MBZ	Reserved, must be zero. Read value is unpredictable.

The GRBR and GRER define a sequence of register space writes that are to be executed multiple times. All register space writes that occur between writes to the GRBR and GRER are part of the repeat loop. The value of the repeat number field +1 specifies the number of iterations through the loop (repeat number = 0 causes one execution of the loop).

The following restrictions are imposed when using the GRBR and GRER repeat looping mechanism:

- The repeat loop must be programmed such that it does not exceed the size of the 64-Dword command FIFO (Section 2.2). The write to the GRER (but not the write to the GRBR) must be included in the repeat-loop size calculation.
- The GRBR and GRER mechanism does not support nested repeat loops.

- The GRBR and GRER writes must not be placed in the command buffer as part of a bursted write sequence. To enforce this restriction, ensure that software never writes to Dword addresses adjacent to the GRBR and GRER (that is, offsets 33C, 344, 34C, and 354).
- Reads of register space must not be done between writes to the GRBR and GRER. Such reads are at risk of completing out of order.

8.5 Graphics Control Registers

The graphics control registers are part of the core registers mapped in base address 0 memory space (Section 7.5.1.2) by the PDBR0.

The graphics control registers control 21130 graphics processing. Reading and writing the graphics control registers does *not* initiate any drawing activity. The register parameters characterize the operations that are initiated by writing to the graphics command registers or frame buffer.

The graphics control registers need not be written for every drawing operation. The number of graphics control registers needed to perform a graphics operation depends on the mode the chip is in and whether drawing is initiated by a write to the frame buffer or to a graphics command register. Additionally, register fields that contain configuration-specific information, such as the width of the frame buffer data path, are written only at initialization time. (Chapter 10 describes the graphics operations and the registers that are required, optional, or ignored for each type of operation.)

Most the graphics control registers can be read and written; however, as noted in the register descriptions, some registers do not read exactly as written, and all the bits in a given register do not necessarily have the same type of access.

8.5.1 Mode Register

Mnemonic: GMOR
Offset: 030
Reset value: 00100000

31:24	23	22	21	20	19:16	15	14	13	12:11	10:8	7:0
RES	PMS	ARS	BR3S	CBS	RES	CE	RES	GE	RES	SB	Mode

Bits	Field	Access	Description
31:24	RES	MBZ	Reserved, must be zero. Read value is unpredictable.
23	PMS	RO	Pixel mask status—indicates whether the pixel mask register (GPXR, Section 8.5.10) is operating in the 1-shot or persistent mode. 0 The GPXR is operating as a 1-shot mask. 1 The GPXR is operating as a persistent mask.
22	ARS	RO	Address register status—indicates the address source for the next line operation. 0 The line operation will use the current internal address. 1 The line operation will use the address in the address register (GADR, Section 8.5.6). This bit is set by writing to the GADR and cleared by doing a line operation.
21	BR3S	RO	Bresenham 3 register status—indicates the source of error and length values for the next line operation. 0 The line operation will use the current internal error and length values. 1 The line operation will use the initial error and length values from the Bresenham 3 register (GB3R, Section 8.5.13). This bit is set by writing to the GB3R and cleared by doing a line operation.

8.5 Graphics Control Registers

Bits	Field	Access	Description
20	CBS	RO	Copy buffer status (copy-direction flag)—indicates the direction of the next copy buffer operation. 0 The next copy mode operation will drain the copy buffer. 1 The next copy mode operation will fill the copy buffer. This bit is set at reset.
19:16	RES	MBZ	Reserved, must be zero. Read value is unpredictable.
15	CE	RW	Cap ends 0 Last pixel write is disabled. 1 Last pixel write is enabled.
14	RES	MBZ	Reserved, must be zero. Read value is unpredictable.
13	GE	RW	Graphics environment 0 The 21130 is operating in an X11 graphics environment. 1 The 21130 is operating in a Win32 graphics environment.
12:11	RES	MBZ	Reserved, must be zero. Read value is unpredictable.
10:8	SB	RW	Source bitmap—specifies the type of source bitmap. 000 8-bpp packed source 001 Reserved 010 Reserved 011 24-bpp source, unpacked, in 32-bpp frame buffer 100 16-bpp packed source, 5:6:5 (R:G:B) organization 101 16-bpp packed source, 1:5:5:5 (α:R:G:B) organization 110 Reserved 111 Reserved **Note:** The destination bitmap field (GOPR <10:8>, Section 8.5.9) is included for DECchip 21030 compatibility. Software must ensure that the source and destination bitmaps are the same type.
7:0	Mode	RW	The code in this field determines the 21130 graphics mode, as shown in Table 8–4.

The GMOR determines how the 21130 responds to writes to the frame buffer space and graphics command registers. Depending on the mode field (<7:0>), the 21130 interprets the address and data differently on any write to the frame buffer, and may interpret the data differently on a write to the graphics command registers. (For more information about the effect of different modes on 21130 graphics processing, see Section 10.2.)

The graphics environment bit (<13>) specifies whether graphics processing must conform to Win32 or X11 specifications. Currently, this field controls how lines are drawn, because Win32 requires a style incompatible with existing X-server drawing code.

The cap ends bit (<15>) determines whether the last pixel in a line is drawn. This bit affects only lines drawn by writing to the slope registers; it has no effect when the frame buffer is accessed in one of the line modes. The host is responsible for adjusting the line length when lines are drawn by writing to the frame buffer in a line mode.

At reset, the value of the GMOR is 00100000_{16}. The copy buffer status bit (<20>) is set.

Table 8–4 lists the graphics modes.

Table 8–4 Graphics Modes

Code*	Graphics Mode
00000000	Simple mode
00000001	Opaque stipple mode
01000001	Opaque bit-reversed stipple mode
00100001	Opaque fill mode
00101001	Opaque extended pattern fill mode
00000010	Opaque line mode
00000101	Transparent stipple mode
01000101	Transparent bit-reversed stipple mode
10000101	Transparent stipple with pixel mask mode
11000101	Transparent bit-reversed stipple with pixel mask mode
00100101	Transparent fill mode
00101101	Transparent extended pattern fill mode
00000110	Transparent line mode
00000111	Copy mode
00010111	DMA-read copy mode, non-dithered
01010111	Scaled-copy mode, enlarge, destination forward
01110111	Scaled-copy mode, reduce, destination forward
11010111	Scaled-copy mode, enlarge, destination backward
11110111	Scaled-copy mode, reduce, destination backward

Unused codes are reserved.

*Code in mode register bits <7:0>.

8.5.2 Deep Register

Mnemonic:	GDER
Offset:	050
Reset value:	0050001C

31 … 23	22	21	20	19 … 13	12	11	10 … 5	4 … 2	1 0
RES	VGAE	GIB	MD32	RES	FRWE	WPG	RES	Addr Mask	RES

Bits	Field	Access	Description
31:23	RES	MBZ	Reserved, must be zero. Read value is unpredictable.
22	VGAE	RW	VGA enable—determines whether VGA or accelerated graphics (2DA) modes are enabled. 0 The 21130 accelerated modes are enabled. 1 The VGA mode is enabled. **Note:** VGA operations with this bit clear and accelerated operations with this bit set have undefined results.
21	GIB	RW	Gib-endian 0 Gib-endian support is disabled. 1 Gib-endian support is enabled.
20	MD32	RW	Mode 32 0 The frame buffer bus width is 64 bits. 1 The frame buffer bus width is 32 bits.
19:13	RES	MBZ	Reserved, must be zero. Read value is unpredictable.
12	FRWE	RW	Flash ROM write enable 0 Writes to the flash ROM are disabled. 1 The flash ROM can be written.
11	WPG	RW	Wrong parity generate 0 Even parity generation is enabled on PCI transactions (normal operation). 1 Odd parity generation is enabled on PCI transactions. Used to test parity generation and reporting logic.
10:5	RES	MBZ	Reserved, must be zero. Read value is unpredictable.

<table>
<tr><th>Bits</th><th>Field</th><th>Access</th><th>Description</th></tr>
<tr><td>4:2</td><td>Addr Mask</td><td>RW</td><td>Address mask—the code in this field determines which bits of the incoming PCI address are masked according to the size of the 21130 address space, as follows:
<table>
<tr><th>Code</th><th>Mask</th><th>Core Map Size</th></tr>
<tr><td>001</td><td>Mask <23:22></td><td>8MB</td></tr>
<tr><td>000</td><td>Mask <24:22></td><td>4MB</td></tr>
<tr><td colspan="3">Unused codes are reserved</td></tr>
</table>
</td></tr>
<tr><td>1:0</td><td>RES</td><td>MBZ</td><td>Reserved, must be zero. Read value is unpredictable.</td></tr>
</table>

The GDER specifies the width of the frame buffer data bus. It is typically written once at initialization time and must be written before any frame buffer access can be performed.

The address mask (<4:2>) determines how incoming PCI address bits <24:22> are masked to index into the 21130's 32MB address space. (See Chapter 7 for more information.)

When the flash ROM write-enable bit (<12>) is clear, flash ROM writes are disabled, but the cycle is externally visible and can be used to implement a write-only parallel port.

8.5.2.1 Gib-Endian Support

The gib-endian bit (<21>) supports big-endian transfers on the PCI. Typically, a big-endian bridge reorders the bytes when transferring data across the PCI. Note that the byte order is maintained on the PCI, as required by the *PCI Local Bus Specification, Revision 2.0*, but byte adjacency is destroyed. This has undesirable side-effects on pixels that span across bytes.

Figure 8–2 shows 1:5:5:5 and 8:8:8:8 pixel format transfers with gib-endian support.

8.5 Graphics Control Registers

Figure 8–2 Gib-Endian Transfers

1:5:5:5 Pixel Format Data Transfer

Big−Endian Data

31	30–26	25–21	20–16	15	14–10	9–5	4–0
a	R	G	B	a	R	G	B

31–24	23–16	15–8	7–0
Byte 0	Byte 1	Byte 2	Byte 3
Pixel *n*		Pixel *n+1*	

Data on PCI Bus (Gib−Endian) After PCI Bridge Chip Transformation

31–29	28–24	23	22–18	17–13	12–8	7	6–2	1–0
G	B	a	R	G	B	a	R	G

31–24	23–16	15–8	7–0
Byte 3	Byte 2	Byte 1	Byte 0
Pixel *n+1*		Pixel *n*	

Data After 21130 Gib−Endian Transformation

31	30–26	25–21	20–16	15	14–10	9–5	4–0
a	R	G	B	a	R	G	B

31–24	23–16	15–8	7–0
Byte 3	Byte 2	Byte 1	Byte 0
Pixel *n+1*		Pixel *n*	

8:8:8:8 Pixel Format Data Transfer

Big−Endian Data

31–24	23–16	15–8	7–0
a	R	G	B
Byte 0	Byte 1	Byte 2	Byte 3
Pixel *n*			

Data on PCI Bus (Gib−Endian) After PCI Bridge Chip Transformation

31–24	23–16	15–8	7–0
B	G	R	a
Byte 3	Byte 2	Byte 1	Byte 0
Pixel *n*			

Data After 21130 Gib−Endian Transformation

31–24	23–16	15–8	7–0
a	R	G	B
Byte 3	Byte 2	Byte 1	Byte 0
Pixel *n*			

The gib-endian bit operates only in the simple, DMA-read copy, and scaled-copy modes. The byte mask is applied to bytes before the gib-endian reordering. In the DMA-read copy and scaled-copy modes, gib-endian reordering is done before the data is shifted.

Because gib-endian reordering is dependent upon the source bitmap field in the mode register (GMOR <10:8>, Section 8.5.1), the value of the source bitmap field should be set before performing gib-endian operations.

At reset, the value of the GDER is 0050001C. VGA is enabled, gib-endian support is disabled, the data bus width is 32 bits, odd parity generation is disabled, and the PCI address is not masked.

8.5.3 Slope-No-Go Registers

Mnemonic:	GSNR<7:0>
GSNR<7:0> offsets:	11C, 118, 114, 110, 10C, 108, 104, 100
GSNR<7:0> reset value:	Undefined

The function of GSNR<7:0> depends on whether the registers are being written (Section 8.5.3.1) or read (Section 8.5.3.2).

8.5.3.1 GSNR<7:0> Write

31 — 16	15 — 0
Absolute dy	Absolute dx

Bits	Field	Access	Description
31:16	Absolute dy	RW	An unsigned integer equal to the absolute value of the difference in *y* of the two line endpoints.
15:0	Absolute dx	RW	An unsigned integer equal to the absolute value of the difference in *x* of the two line endpoints.

On a write, the GSNRs mimic the behavior of the slope registers (GSLR<7:0>, Section 8.4.1), but they do not initiate drawing. That is, they initialize the internal Bresenham engine for line drawing, but do not start Bresenham pixel stepping or draw any pixels.

The GSNRs are primarily used to simplify the drawing of clipped lines and, potentially, to assist in drawing lines with subpixel endpoints. (See Section 11.8.2 for more information about drawing clipped lines.)

Note

The Bresenham width register (GBWR, Section 8.5.14) must be written before writing a GSNR.

8.5.3.2 GSNR<7:0> Read

31	0
Copy Buffer Dword	

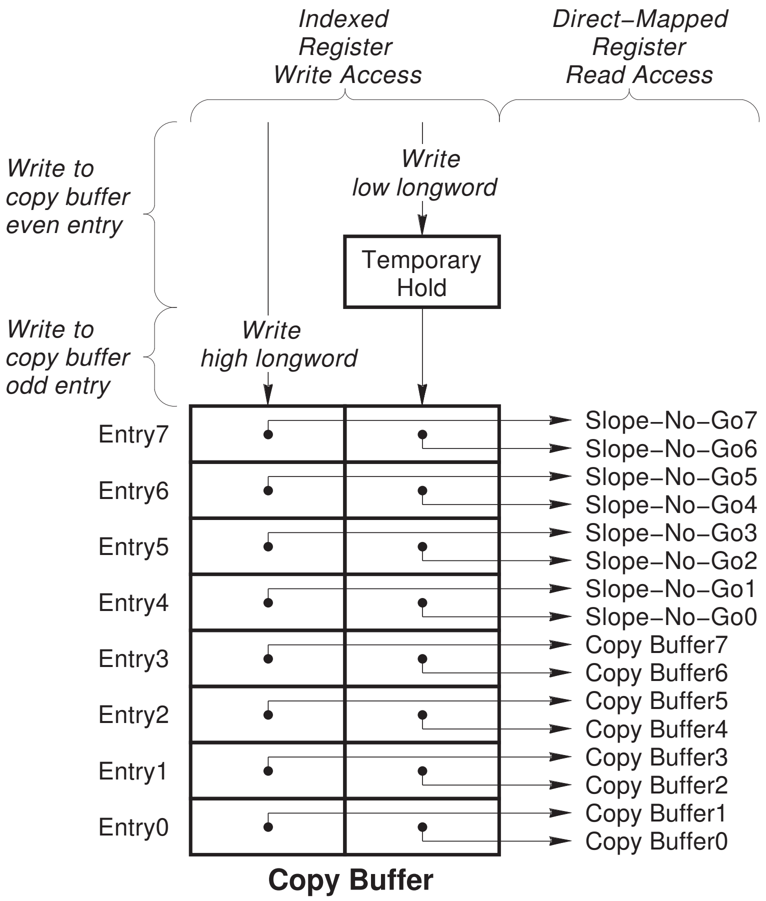
Register	Contents
GSNR7	Copy buffer entry 7 <63:32>
GSNR6	Copy buffer entry 7 <31:0>
GSNR5	Copy buffer entry 6 <63:32>
GSNR4	Copy buffer entry 6 <31:0>
GSNR3	Copy buffer entry 5 <63:32>
GSNR2	Copy buffer entry 5 <31:0>
GSNR1	Copy buffer entry 4 <63:32>
GSNR0	Copy buffer entry 4 <31:0>

On a read, each GSNR returns one Dword of copy buffer entries <7:4> (Figure 8–3). See Sections 8.5.4 and 10.2.6 for more information about programmed I/O access to the copy buffer.

Figure 8–3 shows how the GSNRs and copy buffer registers (GCBR<7:0>) are mapped to the copy buffer entries.

8.5 Graphics Control Registers

Figure 8–3 Copy Buffer Layout



8.5.4 Copy Buffer Registers

Mnemonic:	GCBR
GCBR<7:0> offsets:	01C, 018, 014, 010, 00C, 008, 004, 000
GCBR<7:0> reset value:	Cleared

31	0
Copy Buffer Dword	

The copy buffer registers (GCBR<7:0>) provide read and write access into the internal, 64-byte copy buffer. A read or write to each GCBR returns one Dword from or stores one Dword into the copy buffer.

The copy buffer comprises 8 quadword (64-bit) entries (entry<7:0>). When reading a source bitmap in copy mode, the 21130 fills the copy buffer 1 quadword at a time, from entry0 to entry7. When writing a destination bitmap in copy mode, the 21130 unloads the copy buffer 1 quadword at a time (with mask) in the same sequence (FIFO).

Software can also write the copy buffer in the same order (entry0 to entry7), by alternately writing even-numbered and odd-numbered GCBRs.

Note

Before writing the copy buffer registers, copy mode must be selected in the mode register (GMOR <7:0>, Section 8.5.1).

Writes to the copy buffer must occur in pairs; that is, a write to an even-numbered GCBR must be followed by a write to an odd-numbered GCBR.

Because a write to the first pair of copy-buffer registers (GCBR0 and GCBR1) resets the copy-buffer write pointer, the correct procedure for loading the copy buffer is to write the four even-odd pairs of GCBRs starting with GCBR0.

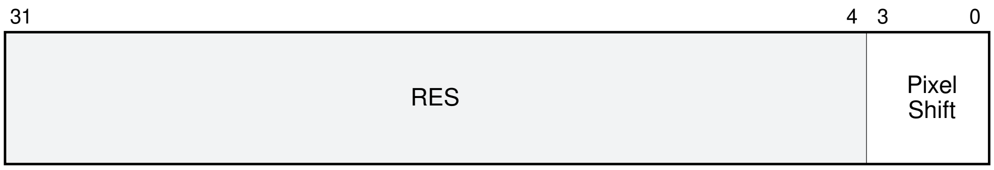
A write to an even-numbered GCBR specifies, but does not load, the low Dword of the next empty copy buffer entry. A subsequent write to an odd-numbered GCBR loads that Dword into the high Dword of the next entry and loads the previously specified Dword into the low Dword of that entry. In other words, writes to GCBRs 0, 2, 4, and 6 go to even-numbered copy buffer locations and writes to GCBRs 1, 3, 5, and 7 go to odd-numbered copy buffer locations. The results of a write to a full copy buffer are undefined.

8.5 Graphics Control Registers

On reads, software directly and randomly accesses individual Dwords of each quadword entry (Figure 8–3). The 8 GCBRs are directly mapped to copy-buffer entries<3:0> and the slope-no-go registers (GSNR<7:0>, Section 8.5.3) are directly mapped to copy-buffer entries<7:4>. The GSNRs are mapped to the copy buffer only in read mode.

8.5.5 Pixel Shift Register

Mnemonic: GPSR
Offset: 038
Reset value: Cleared



Bits	Field	Access	Description
31:4	RES	MBZ	Reserved, must be zero. Read value is unpredictable.
3:0	Pixel Shift	RW	Signed value indicating the number of bytes to shift data on a write into the copy buffer.

The GPSR specifies the number of bytes to shift frame buffer read data in the copy mode. The GPSR is ignored in all other modes. The shift takes place before inserting data into the copy buffer.

As a signed quantity, the pixel shift value (<3:0>) can range from –8 to +7. This allows arbitrary alignment of byte source and destination, and arbitrary copy direction when copying spans. A negative shift value implies a backward copy.

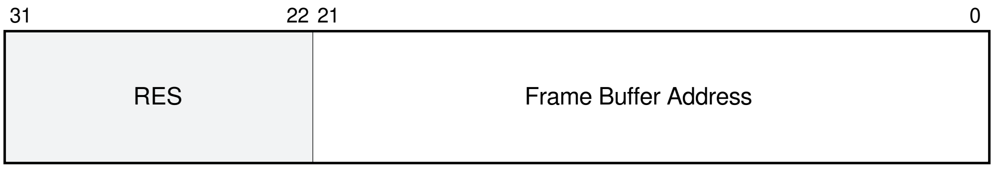
Copy Direction Flag

Writing the GPSR also sets the copy direction flag (the copy buffer status bit, GMOR <20>, Section 8.5.1) to select read-source on the next frame buffer write in copy mode. The flag determines whether the current frame buffer write should read the source into the copy buffer or write the copy buffer into the destination. The flag switches between the read-source and write-destination states on every frame buffer write in copy mode.

(See Sections 10.2.6 and 10.2.7 for more information about using the GPSR and the copy direction flag.)

8.5.6 Address Register

Mnemonic: GADR
Offset: 03C
Reset value: Cleared



Bits	Field	Access	Description
31:22	RES	MBZ	Reserved, must be zero. Read value is unpredictable.
21:0	Frame Buffer Address	RW	The starting address for the next operation initiated by a write to a graphics command register.

The GADR specifies the starting pixel address for a drawing operation. It is used only for operations initiated by writes to the following graphics command registers:

- Slope registers (GSLR<7:0>)
- Span width register (GSWR)
- Continue register (GCTR)
- Copy-64A source and destination registers (GCASR and GCADR)

The frame buffer address (<21:0>) is defined as the offset into the 21130 frame buffer space. A value of 000000_{16} in this field corresponds to the first memory location in frame buffer space. The GADR can be used to access only frame buffer memory; it cannot be used to access the 21130 register space, because that is a separate space. (See Chapter 7 for more information about address mapping.)

In typical operations initiated by a write to the frame buffer, the write address is the starting address of the operation. On the other hand, when an operation is initiated by a write to certain graphics command registers, the starting address is in the GADR. It specifies the address of the first pixel for a drawing operation.

The GADR is used only for operations initiated by writes to the GSLRs, GSWR, GCTR, or GCASR and GCADR. For example, when writing GSLR0 to draw a line, the address in the GADR is the starting address of the drawable. Writing to the GADR does not initiate a drawing operation; it is used on the next write to a GSLR or the GCTR.

When the GADR is written in any of the line modes, the frame buffer address (<21:0>) is used only for the line operation immediately following. If the GADR was not written since the last line operation, the 21130 uses the final address of the most recent operation rather than the address in GADR <21:0>. This feature helps accelerate the drawing of long or linked lines (Section 10.2.9.3). A write to the GADR also sets the address register status bit in the mode register (GMOR <22>, Section 8.5.1) to indicate that GADR <21:0> was written since the last operation.

In repeat-loop copy operations, the GADR specifies the source address first, then the destination address for the copy buffer operation.

The GADR can also be written through the even Dword locations of the first 512KB of alternate control space (Section 7.5.1.3).

8.5.7 Data Register

Mnemonic: GDAR
Offset: 080
Reset value: Cleared

The GDAR format depends on the enabled mode. It specifies a mask for fill mode (Section 8.5.7.1) and some line-mode operations (Section 8.5.7.2).

8.5.7.1 GDAR Opaque-Fill and Transparent-Fill Modes

Bits	Field	Access	Description
31:0	Fill Mask	WO	The mask for each aligned 32-pixel span.

In any fill mode, the GDAR defines a repeating mask, aligned to 4 pixels. Only one mask is specified and used, regardless of the span length. The mask is repeated, or tiled, across the span at 32-pixel intervals. (See Chapter 10 for more information about the fill modes.)

In the transparent-fill and transparent extended-pattern fill modes, the foreground color is written to each pixel of the span that corresponds to a set bit in the fill mask; no color is written to pixels that correspond to clear mask bits. In opaque-fill mode, the foreground color is written to each pixel of the span that corresponds to a set bit in the fill mask; the background color is written to pixels that correspond to clear mask bits.

Note

In any fill mode (except opaque extended-pattern fill mode), the GDAR must be written before the frame buffer, because the write to the frame buffer starts the fill operation (the GDAR is ignored in opaque extended-pattern fill mode).

In both the transparent and opaque extended-pattern fill modes, the pattern data is taken from the copy buffer. (See Sections 10.2.4.1 and 10.2.5.1 for more information about the extended-pattern fill modes.)

8.5.7.2 GDAR Line Mode

31 16	15 0
RES	Line Mask

Bits	Field	Access	Description
31:16	RES	MBZ	Reserved, must be zero. Read value is unpredictable.
15:0	Line Mask	WO	The mask for a 16-pixel line.

In line-mode operations initiated by a write to a slope register (GSLR<7:0>) or the span width register (GSWR), the write data is the slope data, and the GDAR specifies the mask for the 16 pixels of that line segment. (The GDAR is not used when drawing line segments by writing to either the frame buffer in a line mode or to the GCTR. The write data, rather than the GDAR, specifies the line mask.)

The GDAR line mask (<15:0>) is expanded on a per-pixel basis into foreground colors for transparent-line mode and into background or foreground (as specified in the background and foreground registers) in opaque-line mode. In transparent-line mode, the foreground color is written to any pixel in the line that corresponds to a set bit in the line mask. No color is written to pixels that correspond to clear mask bits. In opaque-line mode, the foreground color is written to any pixel in the line that corresponds to a set bit in the line mask, and the background color is written to pixels that correspond to clear mask bits.

Note

The GDAR must be written before the GSLR, because the write to the GSLR starts the drawing operation.

8.5.8 Foreground and Background Registers

Mnemonic:	GFGR, GBGR
GFGR offset:	020
GBGR offset:	024
GFGR, GBGR reset value:	Cleared

	31 … 0
GFGR	Foreground
GBGR	Background

Bits	Field	Access	Description
GFGR			
31:0	Foreground	RW	Defines the foreground color (or set of colors) used in pixel substitution in any of the transparent or opaque stipple, line, or fill modes, except extended-pattern fill modes.
GBGR			
31:0	Background	RW	Defines the background color (or set of colors) used in pixel substitution in any of the transparent or opaque stipple, line, or fill modes, except opaque extended-pattern fill mode.

The GFGR defines foreground pixel colors and the GBGR defines background pixel colors. Foreground color is substituted for 1s in the stipple mask, fill mask, or line mask in any of the transparent or opaque stipple, fill, or line modes; but not in the transparent or opaque extended-pattern fill modes. Background color is substituted for 0s in the stipple mask, fill mask, or line mask in any of the opaque-stipple, opaque-fill, or opaque-line modes; but not in the opaque extended-pattern fill mode. In both the transparent and opaque extended-pattern fill modes, the pattern data is taken from the copy buffer. (See Sections 10.2.4.1 and 10.2.5.1 for more information about the extended-pattern fill modes.)

The mask data can be any of the following:

- PCI write data on a write to the frame buffer or GCTR
- Data in the GDAR on a write to a GSLR or the GSWR
- Data in the GDAR on a write to the frame buffer in a fill mode

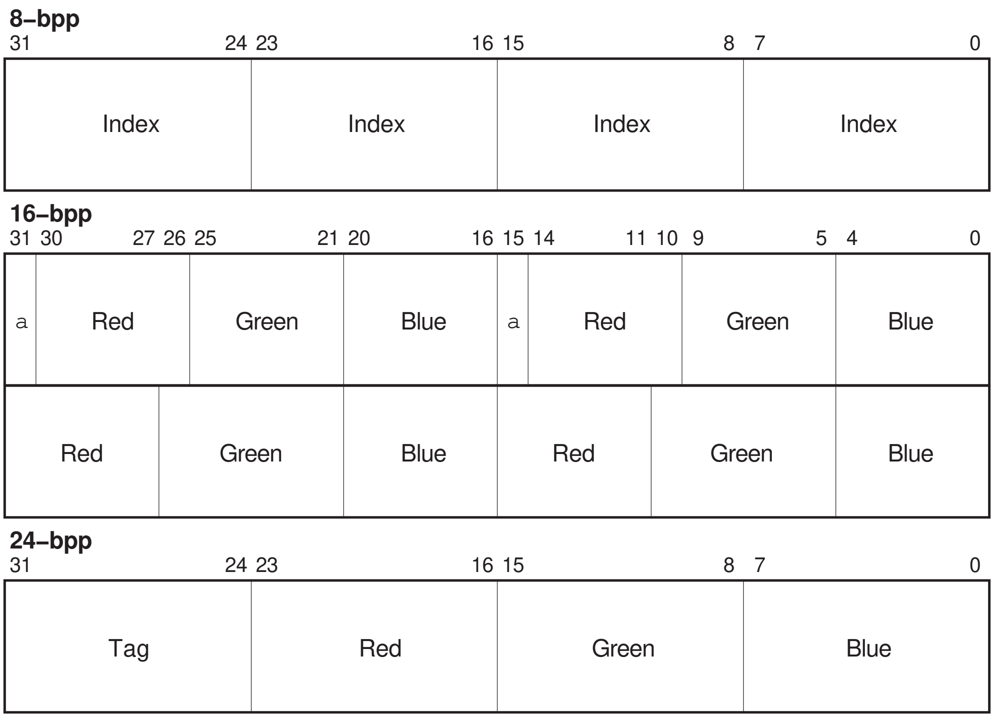
The foreground and background fields are 32-bit quantities regardless of the depth of the bitmap type currently being drawn to. Consequently, software must compensate for the actual depth by replicating the color across the foreground and background fields for bitmap depths less than 32-bpp (Figure 8–4). For example, to present the same color to each possible buffer in 8-bpp mode, the foreground and background colors must be replicated four times across the foreground and background fields. Similarly, in 16-bpp mode, the color must be replicated across both sets of RGB values.

When drawing to 16-bpp bitmaps in a 32-bpp frame buffer, the byte mask (GOPR <19:16>, Section 8.5.9) can be used with the GFGR and GBGR to draw to only the target bitmap while masking off the other bitmap.

Figure 8–4 shows the GFGR and GBGR contents as a function of the bitmap depth in 8-bpp and 32-bpp frame buffers.

8.5 Graphics Control Registers

Figure 8–4 Foreground and Background as a Function of Bitmap Depth



8.5.9 Raster Operation Register

Mnemonic: GOPR
Offset: 034
Reset value: 00000003

31 20	19 16	15 11	10 8	7 4	3 0
RES	Byte Mask	RES	DB	RES	Raster Op

Bits	Field	Access	Description
31:20	RES	MBZ	Reserved, must be zero. Read value is unpredictable.
19:16	Byte Mask	RW	Each bit determines whether eight bit planes are updated, as follows: 0 Updates to the bit's corresponding bit planes are enabled. 1 Updates to the bit's corresponding bit planes are disabled. The bits control the following bit planes: 19 Bit planes 31:24 18 Bit planes 23:16 17 Bit planes 15:8 16 Bit planes 7:0
15:11	RES	MBZ	Reserved, must be zero. Read value is unpredictable.

8.5 Graphics Control Registers

Bits	Field	Access	Description
10:8	DB	RW	Destination bitmap—specifies the type of destination bitmap. 000 8-bpp packed destination 001 Reserved 010 Reserved 011 24-bpp destination, unpacked, in 32-bpp frame buffer 100 16-bpp packed destination, 5:6:5 (R:G:B) organization 101 16-bpp packed destination, 1:5:5:5 (α:R:G:B) organization 110 Reserved 111 Reserved **Note:** This field is included for DECchip 21030 compatibility. Software must ensure that the destination and source bitmaps (GMOR <10:8>, Section 8.5.1) are the same type.
7:4	RES	MBZ	Reserved, must be zero. Read value is unpredictable.
3:0	Raster Op	RW	Raster operation—specifies how the source (src) pixel data and destination (dest) pixel data are logically combined on a write to the destination (Table 8–5).

Table 8–5 lists the Boolean raster operations specified in the GOPR.

Table 8–5 Boolean Raster Operations

Code*	Operation	X	OpenGL	Win32
0000	dest $\Leftarrow$ 0	GXclear	lo_zero	blackness
0001	dest $\Leftarrow$ src AND dest	GXand	lo_and	srcand/mergecopy
0010	dest $\Leftarrow$ src AND (NOT dest)	GXandReverse	lo_andr	srcerase
0011	dest $\Leftarrow$ src	GXcopy	lo_src	srccopy/patcopy
0100	dest $\Leftarrow$ (NOT src) AND dest	GXandInverted	lo_andi	(22_{16})
0101	dest $\Leftarrow$ dest	GXnoop	lo_dst	(AA_{16})
0110	dest $\Leftarrow$ src XOR dest	GXxor	lo_xor	srcinvert/patinvert
0111	dest $\Leftarrow$ src OR dest	GXor	lo_or	srcpaint
1000	dest $\Leftarrow$ (NOT src) AND (NOT dest)	GXnor	lo_nor	notsrcerase
1001	dest $\Leftarrow$ (NOT src) XOR dest	GXequiv	lo_xnor	(99_{16})
1010	dest $\Leftarrow$ NOT dest	GXinvert	lo_ndst	dstinvert

*From GOPR raster operation field <3:0>.

(continued on next page)

Table 8–5 (Cont.) Boolean Raster Operations

Code*	Operation	X	OpenGL	Win32
1011	dest ⇐ src OR (NOT dest)	GXorReverse	lo_orr	(DD_{16})
1100	dest ⇐ NOT src	GXcopyInverted	lo_nsrc	notsrccopy
1101	dest ⇐ (NOT src) OR dest	GXorInverted	lo_ori	mergepaint
1110	dest ⇐ (NOT src) OR (NOT dest)	GXnand	lo_nand	(77_{16})
1111	dest ⇐ 1	GXset	lo_one	whiteness

*From GOPR raster operation field <3:0>.

The 21130 uses the GOPR to support all of the Boolean operations specified under X and OpenGL, and a subset of 2-operand operations specified under Windows (Table 8–5). The source (src) can be used as the source or the pattern to implement the Windows 2-operand raster operations. To update the pixel value, most of these operations require the 21130 to perform read-modify-write cycles to display memory. (The 21130 does not directly support Windows 3-operand operations; for information about handling such operations, see Section 11.3.)

The raster operation field (<3:0>) defines the Boolean operation that is performed on the source and destination pixel data when writing to the destination bitmap in any graphics mode.

At reset, the value of the GOPR is 00000003. The raster operation field is set to *dest* ⇐ *src* (0011_2).

8.5 Graphics Control Registers

8.5.10 Pixel Mask Register

Mnemonic:	GPXR
Address (1-shot):	02C
Address (persistent):	05C
Reset value:	FFFFFFFF

The GPXR format is mode-dependent. It is used to mask pixels in the opaque-stipple modes and transparent-stipple with pixel mask modes (Section 8.5.10.1) and in the simple mode (Section 8.5.10.2).

8.5.10.1 GPXR Stipple Modes

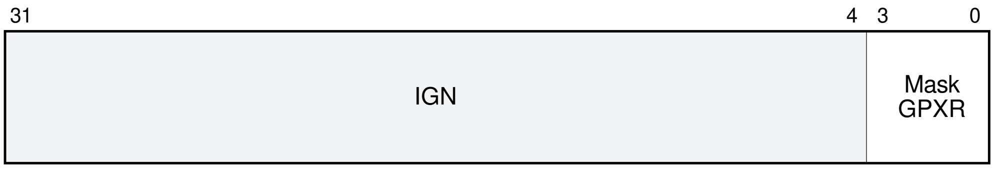
Bits	Field	Access	Description
31:0	Pixel Mask	RW	The mask data for each 32-pixel stippled span. Writes are enabled for pixels that correspond to set mask bits, and disabled for pixels that correspond to clear mask bits.

In the following stipple modes, the frame buffer write data determines whether each of the 32 pixels beginning at the write address should be filled with foreground or background color, but does not determine whether to write the pixels; instead, the GPXR determines which pixels are written.

- Opaque-stipple mode
- Opaque bit-reversed stipple mode
- Transparent-stipple with pixel mask mode
- Transparent bit-reversed stipple with pixel mask mode

Prior to the frame buffer write, the 32-bit mask is written to the GPXR to selectively write-enable each pixel on the subsequent opaque stipple operation.

8.5.10.2 GPXR Simple Mode



Bits	Field	Access	Description
31:4	IGN	RW	Ignored when written, undefined when read.
3:0	Mask GPXR	RW	This is the mask data for each 32-bit frame buffer write. 0 Writes are disabled for bytes that correspond to mask bits = 0. 1 Writes are enabled for bytes that correspond to mask bits = 1.

The mask GPXR field (<3:0>) determines which data bytes are to be written in the next frame buffer write. The field is logically ANDed with the incoming PCI byte mask, to create the byte mask that is ultimately used in simple mode.

Byte mask data for simple mode is primarily useful in systems based on Alpha microprocessors. Because the Alpha instruction set does not support byte granularity, a true PCI byte mask may not be available.

8.5.10.3 GPXR Any Mode

The GPXR is mapped into the 21130 register space twice: as a persistent GPXR and as a 1-shot GPXR. When written as a 1-shot GPXR, the value in the GPXR is used only for the next operation. After that operation is complete, the GPXR reinitializes to an inactive state of FFFFFFFF. When written as a persistent GPXR, the GPXR retains its value until next written at either address. The pixel mask status bit in the mode register (GMOR <23>, Section 8.5.1) indicates the current state of GPXR.

8.5 Graphics Control Registers

8.5.11 Bresenham 1 Register

Mnemonic: GB1R
Offset: 040
Reset value: Cleared

31 16	15 0
Address Increment 1	Error Increment 1

Bits	Field	Access	Description
31:16	Address Increment 1	RW	In line mode, the signed value added to the current address when the Bresenham error term is < 0 (a major axis step).
15:0	Error Increment 1	RW	The positive value added to the error term when the Bresenham error term is < 0 (a major axis step).

The GB1R specifies the address and error increments to be used by the internal Bresenham engine when the cumulative error value is negative. The GB1R can be initialized and used in the following ways:

- Explicitly initialized by software and used during line drawing operations initiated by writing to the frame buffer in line mode.
- Explicitly initialized by software and used during the scaled-copy mode.
- Implicitly initialized and used by 21130 hardware on a write to a slope or slope-no-go register.

8.5.11.1 GB1R Line Mode

Software can initiate a line drawing operation by writing to the frame buffer in line mode at the starting pixel address of the line. Typically, this loads the values from the GB1R into the Bresenham engine, to specify one of the two sets of error and address increment values — the Bresenham 2 register (GB2R) specifies the other set. The engine uses these values to update the cumulative Bresenham error value and addresses as it steps through the line.

When the cumulative error is negative:

- Error increment 1 (<15:0>) is added to the cumulative error.
- Address increment 1 (<31:16>) is added to the current internal address to point to the next pixel address to be written along the line.

This is effectively one step along the major axis of the line.

A write to any slope register causes the 21130 to:

1. Automatically calculate address increment 1 and error increment 1.
2. Unconditionally load them into the Bresenham engine.
3. Initiate the drawing of the first 16 pixels of the line.

Writing to a slope-no-go register has the same effect, but drawing is not initiated.

Section 10.2.9 describes how to draw lines by explicitly writing the GB1R, and how the 21130 hardware initializes the values in the GB1R (as well as the GB2R) on a write to the slope (or slope-no-go) registers.

8.5.11.2 GB1R Scaled-Copy Mode

A scaled-copy is initiated when a DMA command is issued while operating in scaled-copy mode. As the pixels are processed by the YUV pipeline, the Bresenham engine determines whether to replicate (magnify mode) or skip (reduce mode) pixels. This determination is based on the cumulative error register.

When the cumulative error is negative:

- Error increment 1 (<15:0>) is added to the cumulative error.

  When magnifying, the frame buffer pixel pipe, but not the source pixel pipe, is advanced (that is, the source pixel is duplicated). When reducing, the source pixel pipe, but not the frame buffer pixel pipe, is advanced (that is, the source pixel is skipped).

- Address increment 1 (<31:16>) is used to advance to the start of the next span for scaled-copy operations that create multiple destination spans from the same source span. It specifies the number of pixels to advance the frame buffer address in order to move from the last pixel in the current destination span to the first pixel in the next span. This address increment is applied at the beginning of scaled-copy operations for which a new destination address has not been specified (that is, scaled-copy operations initiated through a write to the continue register).

See Section 10.2.8 for more information about scaled-copy mode.

8.5.12 Bresenham 2 Register

Mnemonic: GB2R
Offset: 044
Reset value: Cleared

31 16	15 0
Address Increment 2	Error Increment 2

Bits	Field	Access	Description
31:16	Address Increment 2	RW	In line mode, the signed value added to the current address when the Bresenham error term is ≥ 0 (a step along the major and minor axes).
15:0	Error Increment 2	RW	The positive value subtracted from the error term when the Bresenham error term is ≥ 0 (a step along the major and minor axes).

The GB2R specifies the second set of address and error increments to be used by the internal Bresenham engine. The behavior and use of the GB2R is the same as the GB1R (Section 8.5.11), except that when the cumulative error is greater than or equal to zero:

- Error increment 2 (<15:0>) is subtracted from the cumulative error.
- In line mode, address increment 2 (<31:16>) is added to the current internal address, to point to the next pixel address to be written along the line. Address increment 2 is not used in the scaled-copy mode.
- In the scaled-copy mode, both the frame buffer destination pixel pipe and the source pixel pipe are advanced for magnification and reduction. See Section 10.2.8 for more information about the scaled-copy mode.

8.5.13 Bresenham 3 Register

Mnemonic: GB3R
Offset: 048
Reset value: Cleared

31 15	14 4	3 0
Initial Error	RES	Length

Bits	Field	Access	Description
31:15	Initial Error	RW	The signed initial value stored in the Bresenham error accumulator.
14:4	RES	MBZ	Reserved, must be zero. Read value is unpredictable.
3:0	Length	RW	In line mode, the length, in pixels, of the line segment to be drawn. A value of 0_{16} = 16 pixels. This field is not used in the scaled-copy mode.

The Bresenham error logic uses the initial error value from the GB3R (<31:15>) to determine how to step along the line segment or how to magnify or minify pixels.

The GB3R can be initialized and used in the following ways:

- Explicitly initialized by software and used during line drawing operations initiated by writing to the frame buffer in line mode.
- Implicitly initialized and used by 21130 hardware on a write to one of the slope or slope-no-go registers.

8.5.13.1 GB3R Line Mode

Software can initiate a line drawing operation by writing to the frame buffer in line mode at the starting pixel address of the line. Typically, this loads the values from the GB3R into the Bresenham engine, to specify the initial error term and length of the line to be drawn. The engine updates the error term at each pixel as it steps through the line. After each line segment has been drawn, hardware initializes the length field to 0_{16}, so that all subsequent segments along the line extend its length by 16 pixels.

The GB3R is not written when drawing lines by writing to the slope registers because a write to any slope register causes the 21130 to:

1. Automatically calculate the initial error and initialize length to 16 pixels.
2. Unconditionally load both parameters into the Bresenham engine.
3. Initiate the drawing of the first 16 pixels of the line.

Writing to a slope-no-go register has the same effect, but drawing is not initiated. Therefore, in conjunction with the slope-no-go registers, the GB3R can be useful when drawing clipped lines and certain lines under Win32. Section 10.2.9 describes how the 21130 hardware presets initial error as a function of the slope, octant, and whether the line is being drawn in a Win32 or X11 graphics environment.

8.5.13.2 GB3R Scaled-Copy Mode

A scaled-copy is initiated when a DMA command is issued while operating in scaled-copy mode. When the command is issued, the contents of the GB3R are loaded into the error register to establish the initial error value. The length field is not used in scaled-copy mode.

See Section 10.2.8 for more information about scaled-copy mode.

8.5.14 Bresenham Width Register

Mnemonic: GBWR
Offset: 09C
Reset value: Cleared

31 ... 16	15 ... 0
RES	Bitmap Width

Bits	Field	Access	Description
31:16	RES	MBZ	Reserved, must be zero. Read value is unpredictable.
15:0	Bitmap Width	WO	The width, in pixels, of the destination bitmap.

The GBWR must specify the width, in pixels, of the drawable for all line drawing operations.

The 21130 Bresenham setup hardware uses the bitmap width (<15:0>) to calculate the increment to the pixel's drawable address, on steps along the minor and major axes as the line is drawn. Hardware setup is done only on writes to the slope or slope-no-go registers; therefore, software must initialize the GBWR before writing a slope or slope-no-go register.

The calculated drawable-address increments are stored in, and can be read from, the address increment 1 and 2 fields (GB1R <31:16> and GB2R <31:16>).

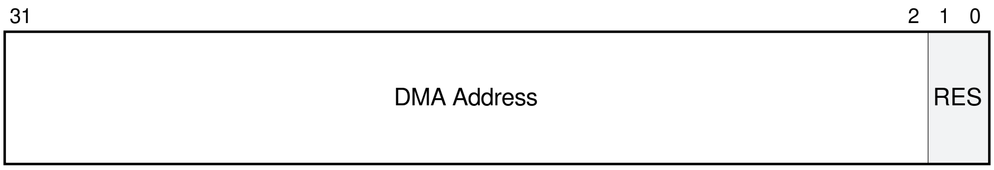
When drawing to the screen in a typical linear-addressed frame buffer, bitmap width is set to the drawable screen width, in pixels; however, the bitmap width is not necessarily constant. The 21130 can draw to arbitrary-sized drawables, whether drawing on screen or off screen.

8.5.15 DMA Base Address Register

Mnemonic: GDBR
Offset: 098
Reset value: Cleared

The GDBR format depends on whether it is being used in the DMA-read copy mode (Section 8.5.15.1) or in the scaled-copy mode (Section 8.5.13.2).

8.5.15.1 GDBR DMA-Read Copy Mode



Bits	Field	Access	Description
31:2	DMA Address	RW	The PCI Dword address pointing to the base address of a drawable bitmap.
1:0	RES	MBZ	Reserved, must be zero. The PCI address must be Dword-aligned.

In the DMA-read copy mode, the GDBR specifies the Dword PCI base address of the source bitmap. A write to the frame buffer in this mode causes the 21130 to begin reading pixels beginning at the DMA address (<31:0>).

Note

To the 21130, the DMA address and other PCI memory addresses are physical addresses. The 21130 has no indication of how the CPU maps system addresses into physical PCI memory addresses, how virtual addresses are translated to physical addresses, or how some systems support scatter-gather mapping from the PCI into main memory. Software must translate these levels of address indirection before writing the GDBR.

8.5.15.2 GDBR Scaled-Copy Mode

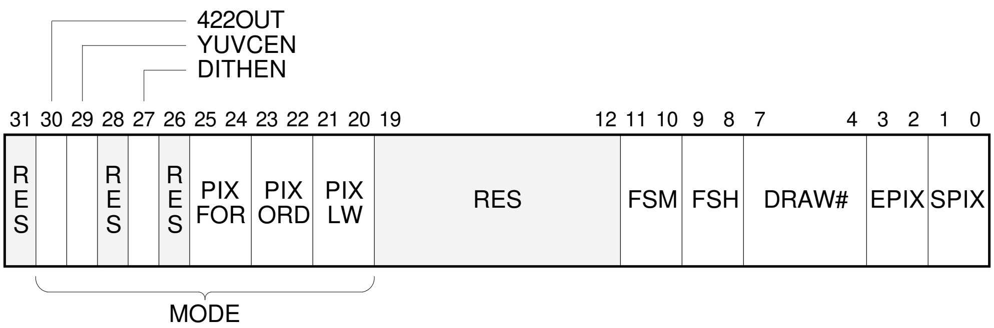
31 – 23	22 – 7	6 – 2	1 – 0
Base Address High	RES	Base Address Low	RES

Bits	Field	Access	Description
31:23	Base Address High	RW	The 9 most significant bits of the Dword-aligned DMA start address.
22:7	RES	MBZ	Reserved, must be zero. Bits <15:0> of the PCI write data (Figure 10–14) specifies DMA start address <22:7>.
6:2	Base Address Low	RW	The 5 least significant bits of the Dword-aligned DMA start address.
1:0	RES	MBZ	Reserved, must be zero. The PCI address must be Dword-aligned.

In the scaled-copy mode (Section 10.2.8), the GDBR and PCI write data specify the PCI Dword address at which a DMA-read copy is to begin. Dividing the DMA start address across the GDBR and the PCI write data streamlines the issue of multiple DMA-read copy video span commands.

8.5.16 Scaled-Copy Control Register

Mnemonic:	GSCR
Offset:	0C4
Reset value:	Cleared



Bits	Field	Access	Description
31	RES	MBZ	Reserved, must be zero. Read value is unpredictable.
30:20	MODE	RW	Specifies the mode for the scaled-copy operation. Table 8–6 describes the subfields and Table 8–7 describes some of the most useful scaled-copy mode operations.
19:12	RES	MBZ	Reserved, must be zero. Read value is unpredictable.
11:10	FSM	RW	Smoothing filter to be used on the Y/G components. When magnifying, the filter is applied after pixel scaling. When reducing, the filter is applied before scaling. This field maps to the following 3-tap filters: 00 0, 1, 0 01 0.5, 0.0, 0.5 10 Reserved 11 Reserved **Note:** The filter should not be used with 8-bpp (3:3:2 or index) source formats.

Bits	Field	Access	Description
9:8	FSH	RW	Sharpening filter to be used on the Y/G components. When magnifying, the filter is applied before pixel scaling. When reducing, the filter is applied after scaling. This field maps to the following 3-tap filters: 00 0, 1, 0 01 –0.5, 2.0, –0.5 10 Reserved 11 Reserved **Note:** The filter should not be used with 8-bpp (3:3:2 or index) source formats.
7:4	DRAW#	RW	Specifies the number of screen pixels to draw before masking pixels in the last Dword of a DMA. Therefore, it specifies an edge mask for the end of video spans. **Note:** A DRAW# value of 0 results in 16 pixels drawn.
3:2	EPIX	RW	End pixel—specifies the position of the last source pixel within the last DMA Dword to be used in a scaled-copy operation. Subsequent pixels are ignored.
1:0	SPIX	RW	Start pixel—specifies the position of the first source pixel within the first DMA Dword to be used in a scaled-copy operation. Preceding pixels are ignored.

The GSCR controls video rendering during scaled-copy operations (Section 10.2.8.1). Table 8–6 describes the subfields contained in the mode field (<30:20>).

Table 8–6 GSCR Mode Field Description

Bits	Field	Access	Description
30	422OUT	RW	4:2:2 output—when set, and the source bitmap field (GMOR <10:8>, Section 8.5.1) is set to 16-bpp, destination pixel writes occur in a 4:2:2 YVYU format.
29	YUVCEN	RW	YUV convert enable—when set, converts pipeline data to color indices through the YUV-to-color-index ROM.
28	RES	MBZ	Reserved, must be zero. Read value is unpredictable.
27	DITHEN	RW	Dither enable 0 Dithering is disabled. 1 Dithering is enabled.

(continued on next page)

8.5 Graphics Control Registers

Table 8–6 (Cont.) GSCR Mode Field Description

Bits	Field	Access	Description
26	RES	MBZ	Reserved, must be zero. Read value is unpredictable.
25:24	PIXFOR	RW	Pixel format—indicates the format of the source pixels. Each channel is MSB-padded up to 8 bits before being sent down the filtering-scaling-dithering pipe. The field is decoded as follows: 00 8:8:8 01 3:3:2 10 5:6:5 11 5:5:5
23:22	PIXORD	RW	Pixel order—indicates how the source pixels are ordered in each DMA Dword. The order depends on the programming of the PIXLW field (<21:20>). The PIXORD field is decoded as follows:

PIXORD Code	PIXLW Code			
	00	**01**	**10**	**11**
00	4:4:4 αYUV	4:2:2 YVYU	8-bpp RGB	Reserved
01	4:4:4 UYVα	4:2:2 UYVY	Reserved	Reserved
10	Reserved	4:2:2 VYUY	Reserved	Reserved
11	32-bpp RGB	16-bpp RGB	Reserved	Reserved

Bits	Field	Access	Description
21:20	PIXLW	RW	Pixels per longword—indicates the number source pixels contained in each DMA Dword. 00 1 pixel in 1 Dword 01 2 pixels in 1 Dword 10 4 pixels in 1 Dword 11 Reserved

Table 8–7 describes some of the most useful scaled-copy mode operations.

Table 8–7 Typical Scaled-Copy Mode Operations

Mode Field[1]	Source Bitmap[2]	Input	Output
Formats with 1 Source Pixel per Dword			
0101 0 00 00 00 0101 0 00 00 00 0101 0 00 00 00	000 100 011	4:4:4 αVYU	Color index into video palette generated through the YUV conversion ROM. For 16- and 32-bpp destinations (specified by GMOR <10:8>) the index is replicated across all byte channels.
0000 0 00 00 00	011	4:4:4 αVYU	4:4:4 VYU (α is dropped).
1000 0 00 00 00	100	4:4:4 αVYU	4:2:2 YVYU.
0001 0 00 01 00 0001 0 00 01 00 0001 0 00 01 00	000 100 101	8:8:8 RGB	Dithered RGB (3:3:2, 5:6:5, or 5:5:5). Dither quantization is determined by GMOR <10:8>.
0000 0 00 01 00	011	8:8:8 αRGB	8:8:8 RGB (α is dropped).
Formats with 2 Source Pixels per Dword			
0101 0 00 *po* 01 0101 0 00 *po* 01 0101 0 00 *po* 01	000 100 011	4:2:2[3]	Color index into video palette. For 16- and 32-bpp destinations (specified by GMOR <10:8>) the index is replicated across all byte channels.
0000 0 00 *po* 01	011	4:2:2[3]	4:4:4 VYU.
1000 0 00 *po* 01	100	4:2:2[3]	4:2:2 YVYU.
0001 0 10 11 01 0001 0 10 11 01	000 101	5:6:5 RGB	Dithered RGB (3:3:2 or 5:5:5). Dither quantization is determined by GMOR <10:8>.
0000 0 10 11 01 0000 0 10 11 01	101 011	5:6:5 RGB	5:6:5 or MSB-padded 8:8:8 RGB. Output format is determined by GMOR <10:8>. Use 5:6:5 to 5:6:5 for 16-bpp index mapping.
0001 0 11 11 01	000	5:5:5 RGB	Dithered 3:3:2 RGB.
0000 0 11 11 01 0000 0 11 11 01 0000 0 11 11 01	100 101 011	5:5:5 RGB	5:5:5 or MSB-padded 5:6:5 or 8:8:8 RGB. Output format is determined by GMOR <10:8>.

[1] GSCR <30:20>
[2] GMOR <10:8>
[3] *po* bits (<23:22>) select YVYU, UYVY, or VYUY

(continued on next page)

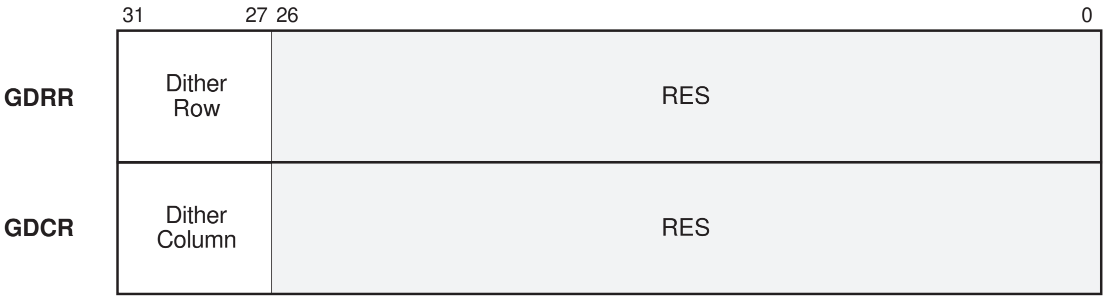
Table 8–7 (Cont.) Typical Scaled-Copy Mode Operations

Mode Field[1]	Source Bitmap[2]	Input	Output
Formats with 4 Source Pixels per Dword			
0000 0 01 00 10 0000 0 01 00 10 0000 0 01 00 10 0000 0 01 00 10	000 100 101 011	3:3:2 RGB	3:3:2 or MSB-padded 5:5:5, 5:6:5, or 8:8:8 RGB. Output format is determined by GMOR <10:8>.
0000 0 01 00 10	000	8-bpp color index	8-bpp color index.

[1]GSCR <30:20>
[2]GMOR <10:8>

8.5.17 Dither Row and Column Registers

Mnemonic: GDRR, GDCR
GDRR offset: 0B0
GDCR offset: 0B4
Reset value: Cleared



Bits	Field	Access	Description
GDRR			
31:27	Dither Row	RW	The row pointer into the 32 × 32 dither matrix. The row index is initialized with this value at the beginning of each scaled-copy operation for which a new destination span address is specified (that is, the scaled-copy operation was not issued through a write to the continue register.)
26:0	RES	MBZ	Reserved, must be zero. Read value is unpredictable.
GDCR			
31:27	Dither Column	RW	The column pointer into the 32 × 32 dither matrix. The dither column index is initialized with this value at the beginning of each destination span.
26:0	RES	MBZ	Reserved, must be zero. Read value is unpredictable.

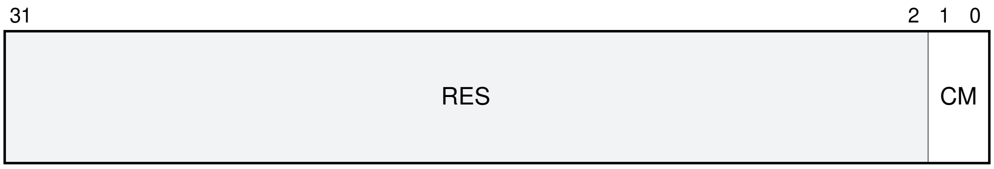
The dither values are initialized at the start of a DMA-dither drawing operation and are then updated by hardware on a per-pixel basis. The dither matrix is used in scaled-copy mode when the dither enable bit (GSCR <27>, Section 8.5.16) is set. The dither row field specifies the row pointer and the dither column field specifies the column pointer into the internal 32 × 32 dither matrix. The pointers address the matrix to produce the dither offsets added to each color before decimation. The GDRR and GDCR are also used in the extended-pattern fill modes.

8.6 Hardware Cursor Registers

The hardware cursor registers are part of the core registers mapped in base address 0 memory space (Section 7.5.1.2) by the PDBR0. See Section 11.11.1 for more information about changing the contents of the cursor registers.

8.6.1 Cursor Mode Register

Mnemonic: CMOR
Offset: 0EC
Reset value: Cleared



Bits	Field	Access	Description
31:2	RES	MBZ	Reserved, must be zero. Read value is unpredictable.
1:0	CM	RW	Cursor mode 00 Cursor disabled 01 3-color cursor 10 Microsoft Windows or XGA cursor 11 X Windows cursor

The 21130 supports a 64 $\times$ 64 hardware cursor. Each cursor pixel is defined by two bits (value bit <1:0>, Table 8–8). The cursor mode (<1:0>) controls the interpretation of the two bits that define each cursor pixel.

Figure 8–5 shows the relationship between the cursor value bits and the cursor pixels.

Figure 8–5 Cursor Value Bits to Pixels Mapping

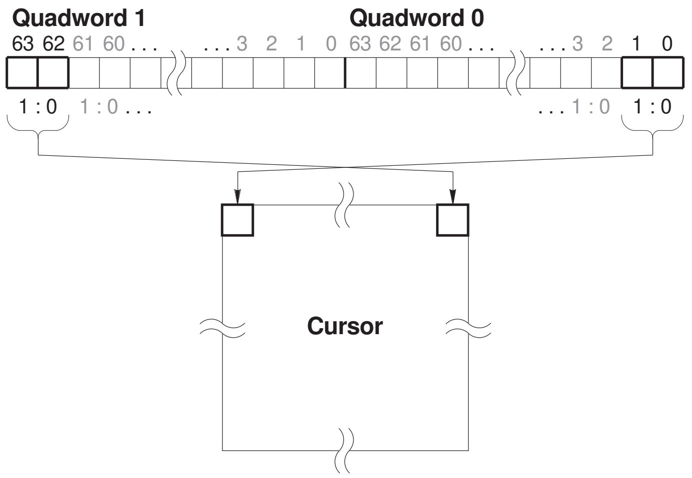


Table 8–8 shows the interpretation of the cursor pixel value bits.

Table 8–8 Cursor Pixel Value Bit Description

Value Bit	GMOR <1:0>		
1:0	01 3-Color Pixel	10 MS Windows Pixel	11 X Windows Pixel
00	Palette data	Cursor color 1	Palette data
01	Cursor color 1	Cursor color 2	Palette data
10	Cursor color 2	Palette data	Cursor color 1
11	Cursor color 3	Palette data inverse	Cursor color 2

8.6.2 Cursor Base Address Register

Mnemonic: CCBR
Offset: 060
Reset value: Cleared

Mode	31–22	21–10	9–0
64–Bit Mode	RES	Cursor Base Address	RES

Mode	31–22	21–11	10–0
32–Bit Mode	RES	Cursor Base Address	RES

Bits	Field	Access	Description
64-Bit Mode			
31:22	RES	MBZ	Reserved, must be zero. Read value is unpredictable.
21:10	Cursor Base Address	RW	The starting address of the 1KB cursor pattern.
9:0	RES	MBZ	Reserved, must be zero. Read value is unpredictable.
32-Bit Mode			
31:22	RES	MBZ	Reserved, must be zero. Read value is unpredictable.
21:11	Cursor Base Address	RW	The left-shifted 64-bit mode value. As a result of the shift, the 64-bit mode MSB is discarded and the 32-bit mode LSB (<10>) must be zero.
10:0	RES	MBZ	Reserved, must be zero. Read value is unpredictable.

The format of the CCBR depends on the frame buffer bus mode (GDER <20>, Section 8.5.2). In 32-bit mode, the 1KB pattern must be aligned to 2KB in frame buffer address space (<10> must be zero).

8.6.3 Cursor XY Register

Mnemonic: CXYR
Offset: 074
Reset value: Cleared

31 24	23 12	11 0
RES	Cursor Y	Cursor X

Bits	Field	Access	Description
31:24	RES	MBZ	Reserved, must be zero. Read value is unpredictable.
23:12	Cursor Y	RW	Determines the displayed location of the bottom-most cursor pattern pixels.
11:0	Cursor X	RW	Determines the displayed location of the right-most cursor pattern pixels.

The CXYR specifies the position of the lower-right cursor pixel. For example, when the cursor X and Y position values = 000_{16}, only the bottom right pixel is displayed on the screen. Conversely, when both values = FFF_{16}, the cursor is off the screen and not displayed. The cursor X and Y position information takes effect on the next top-of-frame. Consequently, if the field is read immediately after being written, the read data might be different than the write data.

8.7 Video Control Registers

The video control registers are part of the core registers mapped in base address 0 memory space (Section 7.5.1.2) by the PDBR0.

The video control registers specify the portion of the frame buffer that is to be displayed and the format of the scanlines composing the display.

See Section 11.11.1 for information about changing the contents of the video control registers.

8.7 Video Control Registers

8.7.1 Video Base Address, Line Increment, and Line Width Registers

Mnemonic:	VIVBR, VISIR, VILWR
VIVBR offset:	06C
VISIR offset:	0CC
VILWR offset:	0D0
VIVBR, VISIR, VILWR reset value:	Cleared

64–Bit Mode

	31 … 22	21 … 3	2 … 0
VIVBR	RES	Video Base Address	RES
VISIR	RES	Scanline Increment	RES
VILWR	RES	Scanline Width	RES

32–Bit Mode

	31 … 22	21 … 4	3	2 … 0
VIVBR	RES	Video Base Address	MBZ	RES
VISIR	RES	Scanline Increment	MBO	RES
VILWR	RES	Scanline Width	MBO	RES

Bits	Field	Access	Mode	Description
VIVBR, VISIR, VILWR				
31:22	RES	MBZ	**64-bit, 32-bit**	Reserved, must be zero. Read value is unpredictable.
VIVBR				
21:3	Video Base Address	RW	**64-bit**	Specifies the start of the visible display in frame buffer memory.
			32-bit	The left-shifted 64-bit mode value. As a result of the shift, the 64-bit mode MSB is discarded and the 32-bit mode LSB (<3>) must be zero.

Bits	Field	Access	Mode	Description
VISIR				
21:3	Scanline Increment	RW	**64-bit**	Specifies the difference between successive scanlines.
			32-bit	The left-shifted 64-bit mode value. As a result of the shift, the 64-bit mode MSB is discarded and the 32-bit mode LSB (<3>) must be one (MBO).
VILWR				
21:3	Scanline Width	RW	**64-bit**	Specifies the number of bytes minus 8 in a scanline.
			32-bit	The left-shifted 64-bit mode value. As a result of the shift, the 64-bit mode MSB is discarded and the 32-bit mode LSB (<3>) must be one (MBO).
VIVBR, VISIR, VILWR				
2:0	RES	MBZ	**64-bit, 32-bit**	Reserved, must be zero. Read value is unpredictable.

The format of the VIVBR, VISIR, and VILWR depends on the frame buffer bus mode (GDER <20>, Section 8.5.2). In 32-bit mode the video base address, scanline increment, and scanline width must be left-shifted 1 bit. As a result of the shift, the 64-bit mode MSB is discarded, bit <3> in the VIVBR must be zero, and bit <3> in the VISIR and VILWR must be one (MBO).

See Section 11.11.3 for more information about calculating the scanline increment and scanline width.

8.7 Video Control Registers

8.7.2 Video Valid Register

Mnemonic: VIVVR
Offset: 070
Reset value: 00001400

31–15	14	13	12	11	10	9	8	7	6	5–4	3–2	1	0
RES	DDCSB	TCLKI	TCLKO	DDCDO	TCLKD	SBLNK	SVV	SBS	DDCDI	DPMS	RES	BLANK	VV

Bits	Field	Access	Description
31:15	RES	MBZ	Reserved, must be zero. Read value is unpredictable.
14	DDCSB	RW	DDC sync bypass—determines how the **ddc_data** pin is used. 0 **ddc_data** is sampled with every **vsync**. 1 **ddc_data** is used directly. This bit is cleared at reset.
13	TCLKI	RO	Test clock input—indicates the state of the test clock (**pll_test**) pin. 0 **pll_test** is low. 1 **pll_test** is high. This bit is forced to 0 when bit <10> is cleared (normal test clock).
12	TCLKO	RW	Test clock output control—controls the test clock (**pll_test**) pin. 0 **pll_test** is floating and can be pulled high or low by an external device. 1 **pll_test** is pulled low. The **pll_test** pin can also be used for **ddc_clk** (see bit <10>). This bit is set at reset and is enabled when bit <10> is set.
11	DDCDO	RW	DDC data output—controls the **ddc_data** pin. 0 **ddc_data** is floating and can be pulled high or low by an external device. 1 The **ddc_data** pin is pulled low.

Bits	Field	Access	Description
10	TCLKD	RW	Test clock output disable—determines whether the test clock output (**pll_test**) pin is controlled by <12>. 0 Bit <13> is forced to 0 (normal test clock). This allows **pll_test** to output the test clock selected by the TCS bit in the PCI clock control register (PCCR <10>, Section 8.2.8). 1 The **pll_test** output is controlled by <12>. When set, this bit allows the **pll_test** pin to be used for the **ddc_clk** signal. This bit is set at reset.
9	SBLNK	RO	Synchronized blank—set when the blank bit (<1>) is set and the current frame pointer is at top-of-frame.
8	SVV	RO	Synchronized video valid—set when the video valid bit (<0>) is set and the current frame pointer is at top-of-frame.
7	SBS	RW	Sync and blank source—indicates whether the source for the **vsync**, **hsync**, and **blank#** signals is external or internal. 0 Internal 1 External Software can write this bit to force the selection of an internal or external source. When the **pci_rst#** signal is asserted, this bit is forced to the inverse of the **gp_int#** signal.
6	DDCDI	RO	DDC data input—when bit <14> is set, indicates the state of the **ddc_data** pin. 0 The **ddc_data** pin is low. 1 The **ddc_data** pin is high. When bit <14> is clear, indicates the state of **ddc_data** sampled with **vsync**. Software can use this field to implement the DDC protocol.
5:4	DPMS	RW	Display power-management signaling—encodes the DPMS states (Section 12.4.4). 00 On 01 Standby 10 Suspend 11 Off
3:2	RES	MBZ	Reserved, must be zero. Read value is unpredictable.

8.7 Video Control Registers

Bits	Field	Access	Description
1	BLANK	RW	Blank (VGA and 2DA modes) 0 Unblank video display. 1 Blank video display. The screen will be unblanked or blanked starting at the next top-of-frame.
0	VV	RW	Video valid (2DA mode only) 0 Soft reset the 2DA video back-end functions. 1 Active display is enabled. **Note:** When this bit is cleared to reset the 2DA video back-end functions, bit <1> should be set to blank the display.

The VIVVR contains several display control bits, as well as bits associated with the test clock, display data channel (DDC), and display power management signaling (DPMS). See Sections 8.2.8 and 12.5 for more information about the test clock, see the VESA *Display Data Channel Standard, Version 1.0, Revision 0* for more information about the DDC, and see Section 12.4.4 and the VESA *Display Power Management Signaling (DPMS) Proposal, Version 1.0p, Revision 0.7p* for more information about DPMS.

Bit <7> indicates the state of the **gp_int#** signal. The signal is sampled at reset. If it is asserted, bit <7> is cleared, indicating that the **vsync**, **hsync**, and **blank#** signals are generated internally by the VGA CRTC registers. If **gp_int#** is deasserted when sampled, bit <7> is set, indicating that the **vsync**, **hsync**, and **blank#** signals are generated by an external source and input to the video back end. Bit <7> can also be written by software to force, as well as indicate, the source selection.

At reset, the value of the VIVVR is 00001400_{16}. Bits <12,10> are set and all other bits are clear.

8.8 Video Format Registers

The video format registers are part of the core registers mapped in base address 0 memory space (Section 7.5.1.2) by the PDBR0.

The 21130 video format registers control pixel formatting and the operation of the external video bus (VAFC). Support is also provided for reading the internal state of the display refresh process.

See Section 11.11.1 for information about changing the contents of the video format registers.

8.8.1 Video Pixel Format Register

Mnemonic:	VFPFR
Offset:	0D4
Reset value:	Cleared

31 16	15	14	13	12	11 10	9 5	4 0
RES	SPOBE	RES	POBM	POBE	FBCD	Outside Pixel Format	Inside Pixel Format

Bits	Field	Access	Description
31:16	RES	MBZ	Reserved, must be zero. Read value is unpredictable.
15	SPOBE	RO	Synchronized pixel occlusion bitmap enable—set when the pixel occlusion bitmap enable bit (<12>) is set and the current frame pointer is at top-of-frame.
14	RES	MBZ	Reserved, must be zero. Read value is unpredictable.
13	POBM	RW	Pixel occlusion bitmap mode 0 Use the pixel occlusion bitmap to select inside or outside pixel format. 1 Use the pixel occlusion bitmap as a monochrome overlay (Section 11.6.2).
12	POBE	RW	Pixel occlusion bitmap enable 0 Disabled—use outside pixel format (<9:5>) as default. 1 Enabled—use the pixel occlusion bitmap (Section 8.8.1.3) to switch between pixel formats. Bit <15> shadows this bit.
11:10	FBCD	RW	Frame buffer color depth—this field controls only how pixels are displayed, not how they are drawn. Drawn pixels are defined by the destination bitmap field in the graphics operation register (GOPR <10:8>, Section 8.5.9). 00 8-bit frame buffer 01 16-bit frame buffer 10 24-bit frame buffer 11 32-bit frame buffer

8.8 Video Format Registers

Bits	Field	Access	Description
9:5	Outside Pixel Format	RW	See Table 8–9.
4:0	Inside Pixel Format	RW	See Table 8–9.

Table 8–9 shows the decoding for the VFPFR outside pixel (<9:5>) and inside pixel (<4:0>) fields.

Table 8–9 Video Pixel Formats

PFS Code	VPFS Code	Interpretation	Notes: ❶
8-bpp			
00000	—	8-bit index (RAM LUT)	
00001	—	8-bit index (ROM LUT)	
00010	—	3/3/2 RGB, direct mapped	
16-bpp			
00000	10000	5/5/5 RGB direct mapped	❷
00001	10001	5/5/5 RGB true color (RAM LUT)	❷
00010	—	5/6/5 RGB direct mapped	
00011	—	5/6/5 RGB true color (RAM LUT)	
01000	11000	8-bit index (RAM LUT)	❸
01001	11001	8-bit index (ROM LUT)	❸
01110	—	8-bpp chroma-keyed overlay (3/3/2 RGB, direct mapped)	❹
01111	—	8-bpp chroma-keyed overlay (ROM LUT)	❹
24-bpp			
00000	—	8/8/8 RGB direct mapped	
00001	—	8/8/8 RGB true color (RAM LUT)	

(continued on next page)

Table 8–9 (Cont.) Video Pixel Formats

PFS Code	VPFS Code	Interpretation	Notes: ❶
32-bpp			
00000	10000	8/8/8 RGB direct mapped	❺
00001	10001	8/8/8 RGB true color (RAM LUT)	❺
01000	11000	8-bit index position 1 (RAM LUT)	❺
01001	11001	8-bit index position 1 (ROM LUT)	❺
01010	11010	8-bit index position 2 (RAM LUT)	❺
01011	11011	8-bit index position 2 (ROM LUT)	❺
01110	11110	8-bpp chroma-keyed overlay (3/3/2 RGB, direct mapped)	❺
01111	11111	8-bpp chroma-keyed overlay (ROM LUT)	❺
Unused codes are reserved.			

Notes for Table 8–9

❶ The pixel format select (PFS) and variable pixel format select (VPFS) codes are contained in VFPFR <9:5> or <4:0>. Unused codes are reserved. See Section 8.8.1.1 for a description of variable pixel formats.

❷ The VPFS codes specify 16-bit pixel formats that are interpreted according to frame buffer bit <15>. If frame buffer bit <15> = 0, bits <14:0> contain pixel data for a 15-bit pixel. If frame buffer bit <15> = 1, bits <15:8> contain a code (Table 8–10) that describes the remaining 8 bits of pixel data.

❸ The VPFS codes specify 16-bit formats in which frame buffer bits <15:8> contain a code (Table 8–10) that describes the remaining 8 bits of pixel data.

❹ The VPFS codes are ignored. The upper and the lower bytes are interpreted as follows:

- If frame buffer bits <15:8> are all zeros, the formats specified in bits <7:0> are 8-bpp chroma-keyed overlay (3/3/2 RGB direct-mapped) and 8-bpp chroma-keyed overlay (ROM LUT).
- If any frame buffer bit <15:8> is not zero (1..255), the format specified in bits <15:8> is 8-bpp chroma-keyed overlay (RAM LUT).

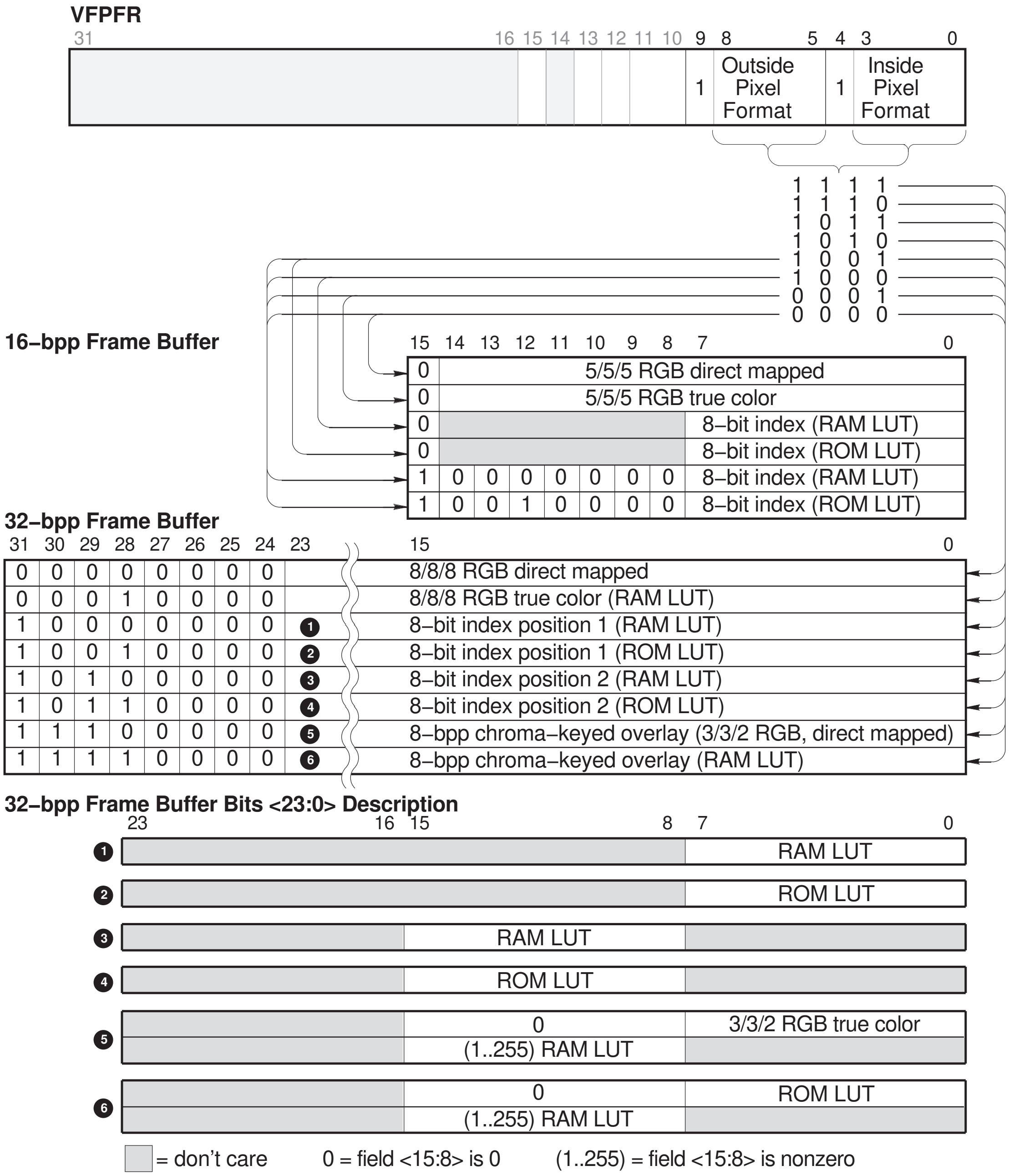
(This is the same as is shown in Figure 8–6, ❺❻, bits <15:0>.)

❺ The VPFS codes specify 32-bit formats in which frame buffer bits <31:24> contain a code (Table 8–10) that describes the remaining 24 bits of data.

Figure 8–6 shows the format of 16- and 32-bit pixels with a variable pixel format.

8.8 Video Format Registers

Figure 8–6 Variable Pixel Formats



8.8.1.1 Pixel Formatting

The video format logic supports the display of multiple pixel formats from the same frame buffer. The default pixel format for the entire displayed region is defined by the outside pixel format field (VFPFR <9:5>). Other pixel formats can be selected through the pixel occlusion bitmap (Section 8.8.1.3) or by using a variable pixel format.

In a variable pixel format, the actual pixel format is encoded in each pixel read from the frame buffer. Variable pixel formats are possible in only 16- and 32-plane frame buffer organizations where extra bits in each pixel are available for pixel format encoding (Figure 8–6).

Table 8–10 describes the upper-byte encoding for 16- and 32-bit pixels with a variable pixel format.

Table 8–10 Variable Pixel Formats

Code[*]	Interpretation	
16-bpp		
0*ppppppp*	When bit <15> = 0, the *ppppppp* field is displayed according to the specified outside or inside pixel format (VFPFR <9:5,4:0>, Section 8.8.1). The following table is extracted from Table 8–9 for reference.	
	10000	5/5/5 RGB direct mapped
	10001	5/5/5 RGB true color (RAM LUT)
	11000	8-bit index (RAM LUT)
	11001	8-bit index (ROM LUT)
10000000	8-bit index (RAM LUT)	
10010000	8-bit index (ROM LUT)	
32-bpp		
00000000	8/8/8 RGB direct mapped	
00010000	8/8/8 RGB true color (RAM LUT)	
10000000	8-bit index position 1 (RAM LUT)	
10010000	8-bit index position 1 (ROM LUT)	
10100000	8-bit index position 2 (RAM LUT)	
10110000	8-bit index position 2 (ROM LUT)	
11100000	8-bpp chroma-keyed overlay (3/3/2 RGB, direct mapped)	
11110000	8-bpp chroma-keyed overlay (ROM LUT)	

*Code in upper byte of frame buffer data. Unused codes are reserved.

8.8 Video Format Registers

In most cases, when VFPR bit <9> or <4> is set, the pixel format is determined by the code in the upper byte of the frame buffer, and VFPR bits <8:5> or <3:0> are ignored. The following cases are exceptions:

- In a 16-bpp frame buffer, when bit <15> is 0, the code in the VFPR specifies the pixel format.
- In a 32-bpp frame buffer, when the upper byte specifies either of the 8-bpp chroma-keyed overlay variable pixel formats (Figure 8–6, ❺❻), the lower bytes are interpreted as follows:
  - If frame buffer bits <15:8> are all zeros, the formats specified in bits <7:0> are 8-bpp chroma-keyed overlay (3/3/2 RGB direct-mapped) and 8-bpp chroma-keyed overlay (ROM LUT).
  - If any frame buffer bit <15:8> is not zero (1..255), the format specified in bits <15:8> is 8-bpp chroma-keyed overlay (RAM LUT).
  - Bits <23:16> are ignored.

8.8.1.2 Addressing the RAM LUT in 15-bpp and 16-bpp True-Color Modes

In true color mode, the 256 $\times$ 24 RAM LUT is separated into 3 256 $\times$ 8 LUTs. Figure 8–7 shows how the pixel data is used to address the LUT. Note that the 2 or 3 upper bits of each pixel color are replicated to form the lower LUT address bits. This results in a sparse use of the 256-entry LUT. The following pseudo code is an example of how the LUT is addressed in these modes.

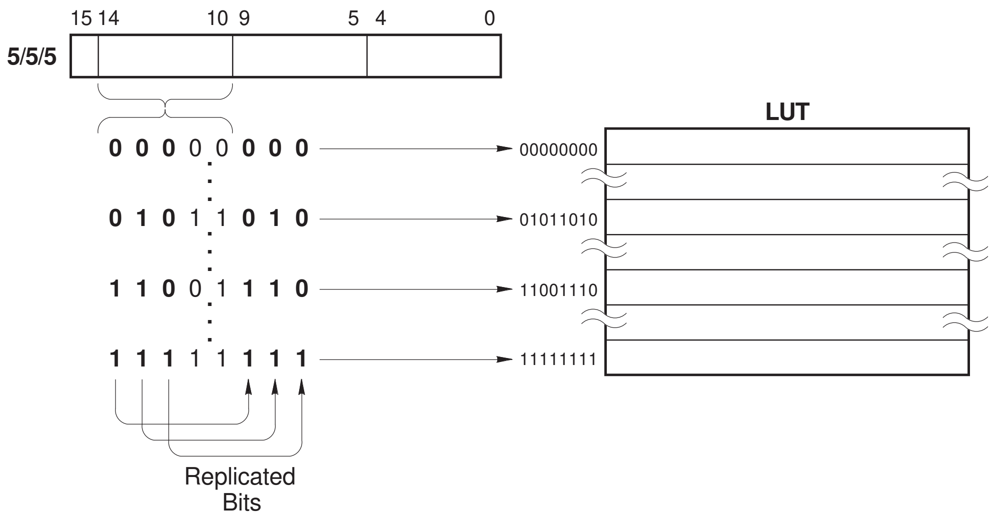
```
Size = power of two size of input table. (e.g. 5 for a 5-bpp color)

Input_array[0..2^Size]

Output_array[0..255], to be used as one of the R, G, or B lookup tables

for (index=0; index==Size; index = index+1) ; loop through entire input table
    temp = index << (8-Size)                ; form most significant index bits
    temp2 = index >> (8-Size)               ; form least significant index bits
    Output_array[temp+temp2] = Input_array[index]
next
```

Figure 8–7 RAM LUT Addressing in 15-bpp and 16-bpp True-Color Modes



8.8 Video Format Registers

8.8.1.3 Pixel Occlusion Bitmap

Figure 8–8 shows the 256-byte pixel occlusion bitmap format in quadword alignment. Table 8–11 describes its fields.

Figure 8–8 Pixel Occlusion Bitmap Format

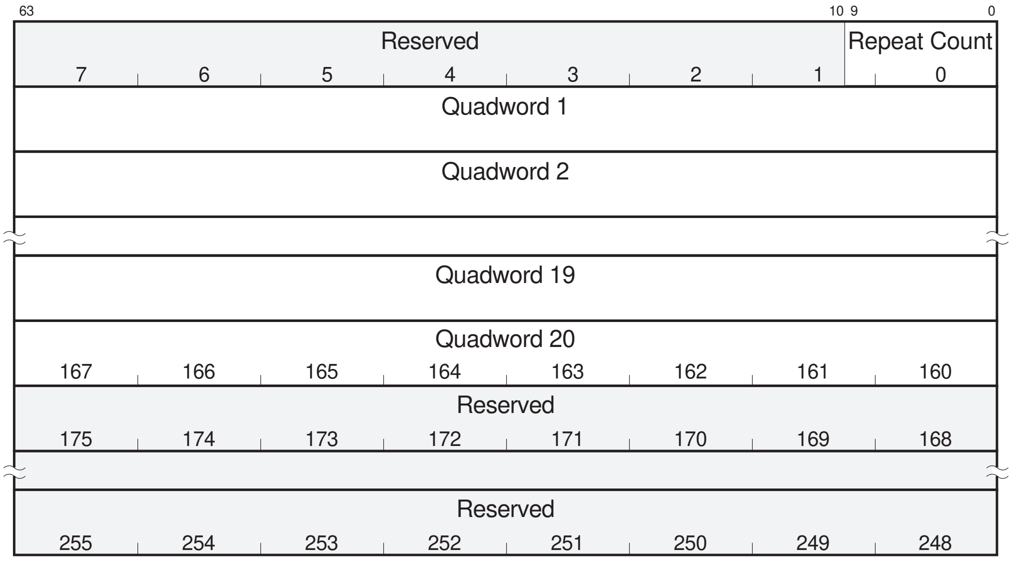


Table 8–11 Pixel Occlusion Bitmap Field Description

Bytes	Field	Description
255:168	Reserved	
167:8	Pixel Occlusion Bitmap	Each bit in these bytes selects the inside or outside pixel format. 0 Outside pixel format 1 Inside pixel format
7:2	Reserved	

(continued on next page)

Table 8–11 (Cont.) Pixel Occlusion Bitmap Field Description

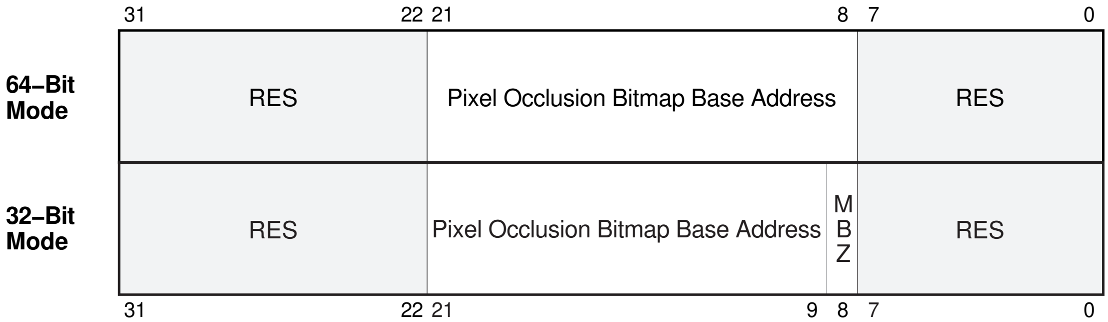
Bytes	Field	Description
1:0	Repeat Count	Bits <9:0> indicate the number of scanlines −1 that will use the pixel occlusion bitmap information in bytes 8 through 167. A zero value specifies that the pixel occlusion bitmap information is valid for only the current scanline. **Note:** The currently supported count range is 0..2047 (bits <63:10> in bytes 7 through 1 are reserved).

The 256-byte pixel occlusion bitmap comprises a 160-byte (20 quadword) per-pixel switch (bytes 167:8). The 21130 video format logic uses the pixel occlusion bitmap to select between two pixel formats or two pixel streams. Typically, it determines whether a given pixel will use the outside pixel format or the inside pixel format. The pixel occlusion bitmap is stored in off-screen frame buffer memory, and is formatted to allow vertical compression. (See Section 2.13.3 for more information.)

8.8 Video Format Registers

8.8.2 Video Pixel Occlusion Bitmap Base Address Register

Mnemonic: VFOBR
Offset: 0E0
Reset value: Cleared



Bits	Field	Access	Mode	Description
31:22	RES	MBZ	**Both**	Reserved, must be zero. Read value is unpredictable.
21:8	Pixel Occlusion Bitmap Base Address	RW	**64-bit**	The starting address of the quadword-aligned pixel occlusion bitmap (Section 8.8.1.3).
			32-bit	The left-shifted 64-bit mode value. As a result of the shift, the 64-bit mode MSB is discarded and the 32-bit mode LSB (<8>) must be zero.
7:0	RES	MBZ	**Both**	Reserved, must be zero. Read value is unpredictable.

The format of the VFOBR depends on the frame buffer bus mode (GDER <20>, Section 8.5.2). In 32-bit mode the pixel occlusion bitmap base address must be left-shifted 1 bit (compared to the address in 64-bit mode) and bit <22> must be zero.

8.8.3 Video Pixel Occlusion Bitmap Current Address Register

Mnemonic: VFOAR
Offset: 1F4
Reset value: Cleared

31 … 22	21 … 3	2 … 0
RES	Pixel Occlusion Bitmap Current Address	RES

Bits	Field	Access	Description
31:22	RES	MBZ	Reserved, must be zero. Read value is unpredictable.
21:3	Pixel Occlusion Bitmap Current Address	RO	A read-only copy of the pixel occlusion bitmap current address. Used only for chip-level testing.
2:0	RES	MBZ	Reserved, must be zero. Read value is unpredictable.

The VFOAR is used only for chip-level testing. The current occlusion bitmap address (<21:3>) is computed by adding the occlusion bitmap base address (VFOBR <21:8>, Section 8.8.2) to either:

$(1280 \div 64) + 1$ in 64-bit mode

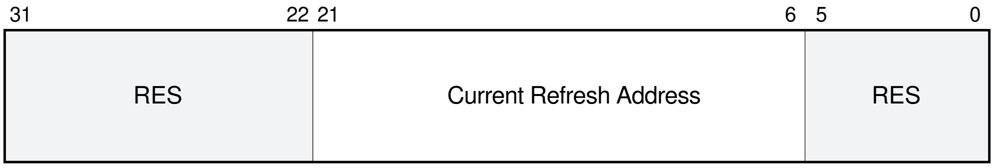
or

$(1024 \div 32) + 1$ in 32-bit mode.

In addition to the PCI configuration registers and the VGA registers, the VFOAR is one of the few registers that is immediately accessible for read (see Section 9.2.2.1). In other words, the command FIFO does not have to be flushed before completing a read of the VFOAR.

8.8.4 Video Current Refresh Address Register

Mnemonic: VFCRR
Offset: 1FC
Reset value: Cleared

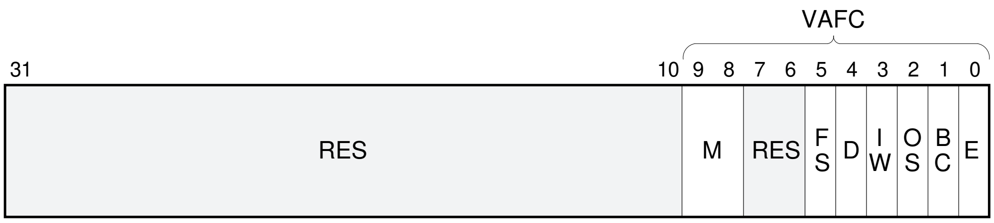


Bits	Field	Access	Description
31:22	RES	MBZ	Reserved, must be zero. Read value is unpredictable.
21:6	Current Refresh Address	RO	A read-only copy of the current refresh address. All of the bits are not provided because they change too quickly at high refresh rates.
5:0	RES	MBZ	Reserved, must be zero. Read value is unpredictable.

In addition to the PCI configuration registers and the VGA registers, the VFCRR is one of the few registers that is immediately accessible for read (see Section 9.2.2.1). In other words, the command FIFO does not have to be flushed before completing a read of the VFCRR.

8.8.5 Alternate Video Control Register

Mnemonic: VFAVR
Offset: 0E8
Reset value: Cleared



Bits	Field	Access	Description
31:10	RES	MBZ	Reserved, must be zero. Read value is unpredictable.
9:8	VAFC M	RW	VAFC mode 00 Unpacked 8-bit pixels (feature-connector-compatible format) 01 Packed 8-bit pixels 10 16-bit pixels 11 Reserved
7:6	RES	MBZ	Reserved, must be zero. Read value is unpredictable.
5	VAFC FS	RW	VAFC frequency select 0 VAFC DCLK equals the pixel clock frequency. 1 VAFC DCLK equals one-half the pixel clock frequency.
4	VAFC D	RO	VAFC direction 0 21130 receives pixels from video system. 1 21130 outputs pixels to video system.
3	VAFC IW	RW	VAFC input window 0 Full-screen VAFC input. 1 VAFC input window is defined by the pixel occlusion bitmap.
2	VAFC OS	RW	VAFC output source 0 VAFC output pixels are derived directly from frame buffer data. 1 VAFC output pixels are derived from DAC pixel input.

8.8 Video Format Registers

Bits	Field	Access	Description
1	VAFC BC	RW	VAFC blank control 0 The **blank#** signal is controlled by the VGA CRTC blanking registers (Sections 8.13.5 and 8.13.18). 1 The **blank#** signal is unconditionally asserted.
0	VAFC E	RW	VAFC enable 0 VAFC enabled 1 VAFC disabled

See Section 2.13.5 for more information about the VAFC port.

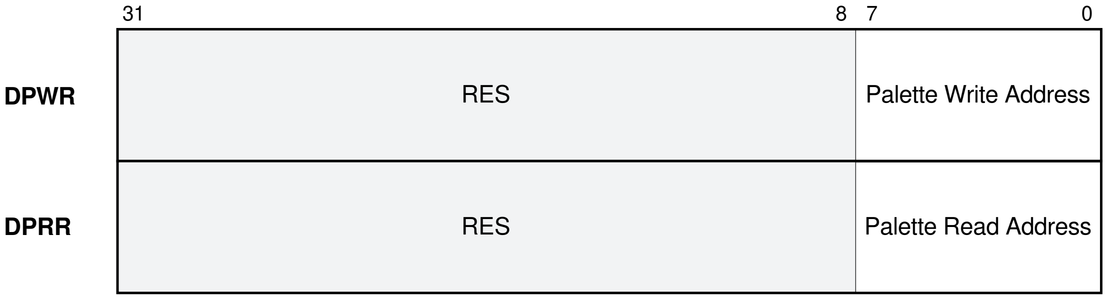
8.9 Palette and DAC Registers

The palette and DAC registers are mapped in base address 1 memory space (Section 7.5.2) by the PDBR1.

The palette and DAC registers control and indicate the status of the onchip graphics color RAM LUT and DACs. They also control the cursor color. The registers are accessed through the palette and DAC register space in the PDBR1 register space (Section 7.5.2.4).

8.9.1 Palette and DAC RAM Write and Read Address Registers

Mnemonic:	DPWR, DPRR
DPWR address:	1000
DPRR address:	100C
DPWR, DPRR reset value:	Undefined

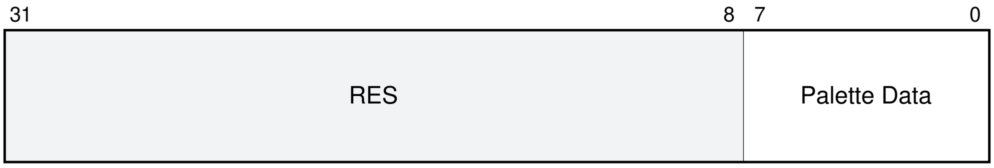


Bits	Field	Access	Description
DPWR			
31:8	RES	MBZ	Reserved, must be zero. Read value is unpredictable.
7:0	Palette Write Address	RW	Specifies the location of the next palette write.
DPRR			
31:8	RES	MBZ	Reserved, must be zero. Read value is unpredictable.
7:0	Palette Read Address	RW	Specifies the location of the next palette read.

The DPWR specifies the address for palette RAM write operations and the DPRR specifies the address for palette RAM read operations. See the palette and DAC RAM color register (DPCR, Section 8.9.2) description for more information.

8.9.2 Palette and DAC RAM Color Register

Mnemonic:	DPCR
Address:	1004
Reset value:	Undefined



Bits	Field	Access	Description
31:8	RES	MBZ	Reserved, must be zero. Read value is unpredictable.
7:0	Palette Data	RW	Palette color data.

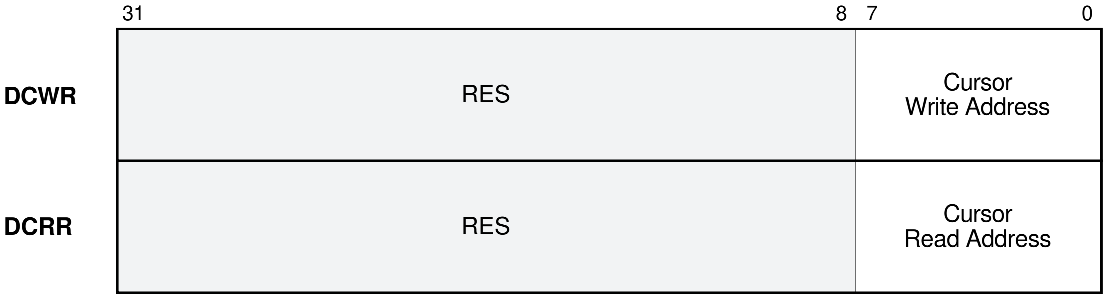
The DPCR accesses the palette location specified by the palette and DAC RAM write address register or the palette and DAC RAM read address register (DPWR or DPRR, Section 8.9.1).

To set palette color values, the desired palette address is loaded into the DPWR; then the red, green, and blue data are written to the DPCR. After the blue write is complete, the palette RAM is updated with the new values. The DPWR is automatically incremented after the blue data is written; consequently, successive palette locations can be updated without reloading the DPWR. Note that the palette address is incremented after the blue value is written to the internal palette holding register.

Palette read operations are similar to palette write operations. The desired palette address is loaded into the DPRR; the address is incremented; then, three successive reads to the DPCR return the red, green, and blue components of the palette RAM entry. As in a write, the palette address is incremented after the internal palette holding register is loaded, not after the blue value is read.

8.9.3 Palette and DAC Cursor Write and Read Address Registers

Mnemonic:	DCWR, DCRR
DCWR address:	1010
DCRR address:	101C
DCWR, DCRR reset value:	Undefined



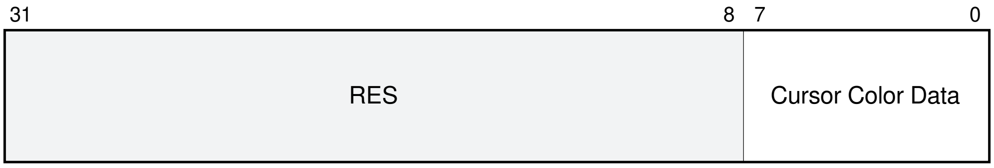
Bits	Field	Access	Description
DCWR			
31:8	RES	MBZ	Reserved, must be zero. Read value is unpredictable.
7:0	Cursor Write Address	RW	Specifies the location of the next cursor color write.
DCRR			
31:8	RES	MBZ	Reserved, must be zero. Read value is unpredictable.
7:0	Cursor Read Address	RW	Specifies the location of the next cursor color read.

The DCWR specifies the address for cursor color write operations and the DCRR specifies the address for cursor color read operations. See the DAC cursor color register (DCCR, Section 8.9.4) description for more information.

8.9 Palette and DAC Registers

8.9.4 Palette and DAC Cursor Color Register

Mnemonic: DCCR
Address: 1014
Reset value: Undefined



Bits	Field	Access	Description
31:8	RES	MBZ	Reserved, must be zero. Read value is unpredictable.
7:0	Cursor Color Data	RW	Cursor color data.

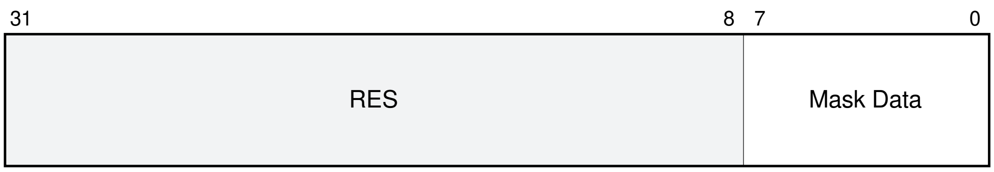
The DCCR accesses the cursor color location specified by the palette and DAC cursor write address register or the palette and DAC cursor read address register (DCWR or DCRR, Section 8.9.3).

To set cursor color values, the desired cursor color address is loaded into the DCWR; then the red, green, and blue data are written to the DCCR. After the blue write is complete, the cursor color location is updated with the new values. The DCWR is automatically incremented after the blue data is written; consequently, successive cursor color locations can be updated without reloading the DCWR. Note that the cursor color address is incremented after the blue value is written to the internal cursor-color holding register.

Cursor color read operations are similar to cursor color write operations. The desired cursor color address is loaded into the DCRR; the address is incremented; then, three successive reads to the DCCR return the red, green, and blue components of the cursor color entry. As in a write, the cursor color address is incremented after the internal cursor-color holding register is loaded, not after the blue value is read.

8.9.5 Palette and DAC Pixel Mask Register

Mnemonic: DPMR
Address: 1008
Reset value: Undefined



Bits	Field	Access	Description
31:8	RES	MBZ	Reserved, must be zero. Read value is unpredictable.
7:0	Mask Data	RW	Palette index mask.

The DPMR specifies the mask for palette index inputs when using indexed color modes.

8.9.6 Palette and DAC Status Register

Mnemonic:	DSTR
Address:	1028
Reset value:	Undefined

31–4	3	2	1–0
RES	SS	RWS	AS

Bits	Field	Access	Description
31:4	RES	MBZ	Reserved, must be zero. Read value is unpredictable.
3	SS	RO	Sense status 0 One or more DAC outputs exceeded the internal voltage reference level (335 mV). 1 No DAC output exceeded the internal voltage reference level.
2	RWS	RO	Read/write state—indicates whether the last palette or cursor color operation was a read or write, as follows: 0 Write—defined as writing the DPWR or the DCWR. 1 Read—defined as writing the DPRR or the DCRR.
1:0	AS	RO	Address state—indicates the color-component address for the next read or write cycle to the DPCR or DCCR, as follows: 00 Red 01 Green 10 Blue 11 Reserved

The DSTR contains several miscellaneous palette and DAC status bits.

8.9.7 Palette and DAC Command Register 0

Mnemonic: DCOR0
Address: 1018
Reset value: Cleared

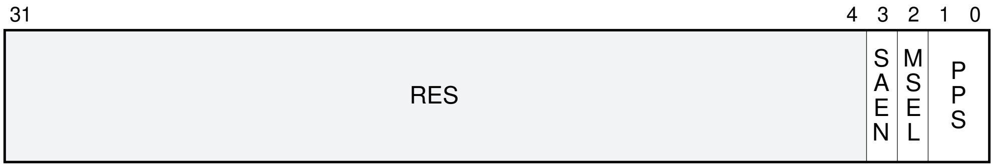
31 ... 6	5	4	3	2	1	0
RES	SE	RES	GSE	RES	DRS	PDE

Bits	Field	Access	Description
31:6	RES	MBZ	Reserved, must be zero. Read value is unpredictable.
5	SE	RW	Setup-enable 0 Blanking pedestal = 0 IRE. 1 Blanking pedestal = 7.5 IRE.
4	RES	MBZ	Reserved, must be zero. Read value is unpredictable.
3	GSE	RW	Green sync enable—when set, enables sync information on the green DAC output.
2	RES	MBZ	Reserved, must be zero. Read value is unpredictable.
1	DRS	RW	DAC resolution select 0 Host 6-bit palette operations are enabled (the 2 LSBs are forced to zero). 1 Host 8-bit palette operations are enabled.
0	PDE	RW	Power-down enable 0 Normal palette and DAC operation is enabled. 1 Removes power to the DACs and palette RAM; host palette operations are not affected.

The DCOR0 allows software to control the palette and DAC function.

8.9.8 Palette and DAC Command Register 1

Mnemonic: DCOR1
Address: 1030
Reset value: Cleared



Bits	Field	Access	Description
31:4	RES	MBZ	Reserved, must be zero. Read value is unpredictable.
3	SAEN	RW	Signature analysis enable 0 Signature analysis is disabled. 1 Signature analysis is enabled. To enable signature analysis, this bit must be set and **blank#** must be deasserted.
2	MSEL	RW	Mode select 0 Accumulate mode 1 Data strobe mode
1:0	PPS	RW	Pixel phase select—signature analysis samples every fourth pixel in a displayed scanline, beginning with the pixel specified in this field, as follows: 00 The first pixel after **blank#** is deasserted. 01 The second pixel after **blank#** is deasserted. 10 The third pixel after **blank#** is deasserted. 11 The fourth pixel after **blank#** is deasserted.

The signature analysis registers (DRSR, DGSR, and DBSR, Section 8.9.9) are enabled to sample pixel data only when the signature analysis enable bit (<3>) is set during active display (**blank#** is deasserted). When **blank#** is asserted, the signature analysis registers retain the last value sampled. The signature analysis registers should not be accessed during the time that **blank#** is asserted plus 5 pixel clocks.

In accumulate mode (<2> = 0), the signature analysis register data changes on every fourth clock. The signature value should be initialized with a specific seed value (other than FF or 00) and a known pixel stream should be sampled. When **blank#** is deasserted, the pixel data causes the signature value to change on every fourth clock. The resulting accumulated value is a function of all the pixels that have been sampled. The known video pattern has a unique signature value that can be used as a functional check.

In data strobe mode (<2> = 1), the signature analysis registers capture and hold the data going to the DACs; there is no accumulated value. For example, when the DACs are at full scale each signature value is FF.

8.9.9 Palette and DAC Signature Analysis Registers

Mnemonic:	DRSR, DGSR, DBSR
DRSR address:	1034
DGSR address:	1038
DBSR address:	103C
DRSR, DGSR, DBSR reset value:	Undefined

	31 … 8	7 … 0
DRSR	RES	Red Signature
DGSR	RES	Green Signature
DBSR	RES	Blue Signature

Bits	Field	Access	Description
31:8	RES	MBZ	Reserved, must be zero. Read value is unpredictable.
DRSR			
7:0	Red Signature	RW	Red color sample.
DGSR			
7:0	Green Signature	RW	Green color sample.
DBSR			
7:0	Blue Signature	RW	Blue color sample.

The palette and DAC signature analysis registers capture every fourth pixel output to the DACs. See Section 8.9.8 for more information.

8.10 VGA Register Overview

For VGA mode operations, the VGA registers are accessed in VGA memory space (Section 7.4). The VGA registers are also mapped to VGA alternate register space in base address 1 memory space (Section 7.5.2.1) by the PDBR1.

Table 8–12 lists the VGA registers according to groups. The registers are either directly accessible (and listed with their port addresses) or indexed, as follows:

- All of the VGA external and general registers are directly accessible.
- None of the VGA extended registers are directly accessible.
- Only the index and data registers in the remaining groups are directly accessible.

The table also lists the I/O port address for direct access and the number of indexed (not directly accessible) registers in each group.

Table 8–12 VGA Register Port Map

VGA Register Group	Directly Accessible Registers	I/O Port Addresses	Indexed Registers
External and general	4	3BA, 3DA, 3CA, 3C2, 3CC	None
Sequencer	Index Data	3C4 3C5	5
CRT controller	Index Data	3B4, 3D4 3B5, 3D5	25
Extended	None	—	13
Graphics controller	Index Data	3CE 3CF	9
Attribute controller	Index Data	3C0 3C1	21
Color	5	3C6, 3C7, 3C8, 3C9	None

8.10 VGA Register Overview

A PCI target abort is signaled if the following VGA register accesses are attempted:

- A longword access with a byte mask that specifies a nonimplemented register.
- A longword or word access that straddles the palette registers or the sequencer registers at address 3C4.

On word accesses to the VGA registers, the least significant byte is used before the most significant byte to ensure that index register operations are performed before data register operations.

8.11 VGA External and General Registers

Each of the external and general registers is directly accessible at its port address, and indexing is not required.

Table 8–13 lists the VGA external and general registers and their read and write access addresses.

Table 8–13 VGA External and General Register Port Map

Register	Write Address	Read Address
VGA miscellaneous output register	3C2	3CC
VGA feature control register	3DA	3CA
VGA input status 0 register	—	3C2
VGA input status 1 register	—	3DA

8.11.1 VGA Miscellaneous Output Register

Mnemonic:	VEMISR
Write address:	3C2
Read address:	3CC
Reset value:	Undefined

7	6	5	4	3	2	1	0
VSP	HSP	PB	RES	CS		ER	IOA

Bits	Field	Access	Description
7	VSP	RW	Vertical sync polarity—sets the polarity of the vertical sync pulse. Used with HSP (<6>) to determine displayed vertical size (Table 8–14). 0 Positive vertical sync pulse 1 Negative vertical sync pulse
6	HSP	RW	Horizontal sync polarity—sets the polarity of the horizontal sync pulse. Used with the VSP bit (<7>) to determine displayed vertical size (Table 8–14). 0 Positive horizontal sync pulse 1 Negative horizontal sync pulse
5	PB	RW	Page bit—selects odd or even display page in modes 0, 1, 2, 3, and 7. 0 Low 64K page of memory 1 High 64K page of memory
4	RES	MBZ	Reserved, must be zero. Read value is unpredictable.
3:2	CS	RW	Clock source select 00 25-MHz clock source 01 28-MHz clock source 10 Reserved 11 Reserved This field is selected by the VGA variable dot clock select field in the clock control A register (VXCKAR <0>, Section 8.14.10). The frequency is determined by hardwired (that is, nonprogrammable) M, L, and N term values (VXCKAR <6:1> and VXCKBR <5:4,3:0>).

8.11 VGA External and General Registers

Bits	Field	Access	Description
1	ER	RW	Enable RAM 0 Disable CPU access to display memory. 1 Enable CPU access to display memory.
0	IOA	RW	I/O address select 0 Address 3Bx for monochrome emulation 1 Address 3Dx for color emulation

The VEMISR controls several VGA functions including the vertical size of the display, dot clock source, display memory access, and monochrome or color I/O address selection.

Table 8–14 shows the displayed vertical size and number of active lines as a function of the vertical and horizontal sync pulse polarities selected by VEMISR bits <7:6>.

Table 8–14 Displayed Vertical Size as Function of HSP and VSP

VSP VEMISR <7>	HSP VEMISR <6>	Vertical Size	Active Lines
0 (+)	0 (+)	Reserved	Reserved
0 (+)	1 (–)	400 lines	414 lines
1 (–)	0 (+)	350 lines	362 lines
1 (–)	1 (–)	480 lines	496 lines

8.11.2 VGA Feature Control Register

Mnemonic:	VEFCOR
Write address (monochrome):	3BA
Write address (color):	3DA
Read address:	3CA
Reset value:	Undefined

7 — 4	3	2	1	0
RES	VSS	RES	FC1	FC0

Bits	Field	Access	Description
7:4	RES	MBZ	Reserved, must be zero. Read value is unpredictable.
3	VSS	RW	Vertical sync select 0 Normal vertical sync output to monitor. 1 Vertical sync is the logical OR of vertical sync and vertical display enable.
2	RES	MBZ	Reserved, must be zero. Read value is unpredictable.
1	FC1	RW	Feature control <1> 0 The value FC1 is 0. 1 The value FC1 is 1.
0	FC0	RW	Feature control <0> 0 The value FC0 is 0. 1 The value FC0 is 1.

VEFCOR bits <1:0> are implemented only for compatibility and do not affect 21130 operation.

8.11 VGA External and General Registers

8.11.3 VGA Input Status 0 Register

Mnemonic: VEIS0R
Read address: 3C2
Reset value: Undefined

7	6–5	4	3–0
VRI	RES	SS	RES

Bits	Field	Access	Description
7	VRI	RO	Vertical retrace interrupt 0 Not in vertical retrace. 1 Vertical retrace is occurring.
6:5	RES	MBZ	Reserved, must be zero. Read value is unpredictable.
4	SS	RO	Sense status 0 One or more DAC outputs exceeded the internal voltage reference level (335 mV). 1 No DAC output exceeded the internal voltage reference level. This bit is included for compatibility and maps to DSTR <4> (Section 8.9.6).
3:0	RES	MBZ	Reserved, must be zero. Read value is unpredictable.

The VEIS0R indicates whether a vertical retrace interrupt is pending.

8.11.4 VGA Input Status 1 Register

Mnemonic:	VEIS1R
Read address (monochrome):	3BA
Read address (color):	3DA
Reset value:	Undefined

7	6	5	4	3	2	1	0
RES		DIA		VR	RES		DE

Bits	Field	Access	Description
7:6	RES	MBZ	Reserved, must be zero. Read value is unpredictable.
5:4	DIA	RO	Diagnostic—indicates the display colors accessed as a function of the VSM field in the color plane enable register (VACPER <3:0>, Section 8.16.5).

VSM	DIA	
<5:4>	<5>	<4>
00	Red	Blue
01	Blue	Green
10	Red	Green

Bits	Field	Access	Description
3	VR	RO	Vertical retrace 0 Display is in display mode. 1 Display is in vertical retrace mode.
2:1	RES	MBZ	Reserved, must be zero. Read value is unpredictable.
0	DE	RO	Display enable 0 Active display time. 1 Display is in horizontal or vertical retrace period.

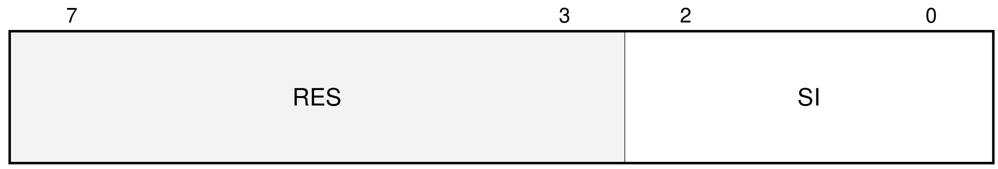
The VEIS1R indicates the status of several display functions.

8.12 VGA Sequencer Registers

The VGA sequencer registers are accessed through the VGA sequencer index register (VSINXR, address 3C4) and the VGA sequencer data register (VSDATR, address 3C5).

8.12.1 VGA Sequencer Index Register

Mnemonic: VSINXR
Address: 3C4
Reset value: Undefined



Bits	Field	Access	Description
7:3	RES	MBZ	Reserved, must be zero. Read value is unpredictable.
2:0	SI	RW	Sequencer index—index to the following VGA sequencer registers: 0 Reset (VSRESR) 1 Clocking mode (VSCMOR) 2 Plane mask (VSPLMR) 3 Character map select (VSCMSR) 4 Memory mode (VSMMOR) Unused index values are reserved.

The VSINXR contains the index used to access the VGA sequencer registers.

8.12.2 VGA Sequencer Data Register

Mnemonic: VSDATR
Address: 3C5
Reset value: Undefined

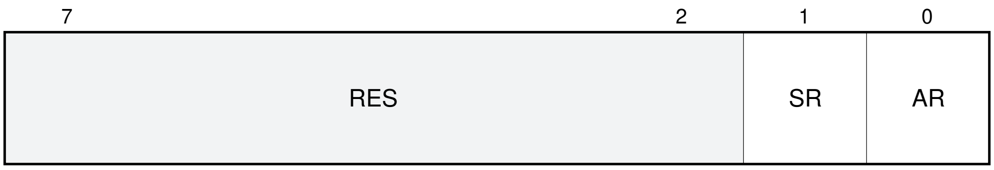
7 … 0
Sequencer Data

Bits	Field	Access	Description
7:0	Sequencer Data	RW	Indexed sequencer register read or write data.

The VSDATR contains the read or write data for the VGA sequencer register indexed by the VSINXR (Section 8.12.1).

8.12.3 VGA Sequencer Reset Register

Mnemonic: VSRESR
Index: 0
Reset value: Undefined



Bits	Field	Access	Description
7:2	RES	MBZ	Reserved, must be zero. Read value is unpredictable.
1	SR	RW	Synchronous reset 0 Hold system in reset state. 1 Release reset if bit <0> = 1.
0	AR	RW	Asynchronous reset 0 Hold system in reset state. 1 Release reset if bit <1> = 1.

VSRESR bits <1:0> are implemented only for compatibility and do not affect 21130 operation.

8.12.4 VGA Sequencer Clocking Mode Register

Mnemonic: VSCMOR
Index: 1
Reset value: Undefined

7	6	5	4	3	2	1	0
RES		SO	S4	DC	SL	BW	8/9

Bits	Field	Access	Description
7:6	RES	MBZ	Reserved, must be zero. Read value is unpredictable.
5	SO	RW	Screen off 0 Screen is turned on (display refresh is active). 1 Screen is turned off (no display refresh).
4	S4	RW	Shift four—in conjunction with bit <2>, determines when video serializers are loaded, as follows:

S4	SL	Load video serializers . . .
0	0	Every character clock.
0	1	Every other character clock.
1	X	Every fourth character clock (SL is ignored).

Bits	Field	Access	Description
3	DC	RW	Divide dot clock by 2 0 Video dot clock frequency = master clock frequency. 1 Video dot clock frequency = (master clock frequency ÷ 2).
2	SL	RW	Shift load—see bit <4> description.
1	BW	RW	Bandwidth—implemented only for compatibility and does not affect 21130 operation. The generic VGA definition is as follows: 0 CRTC controls memory bus on 4 of 5 cycles. 1 CRTC controls memory bus on 2 of 5 cycles.

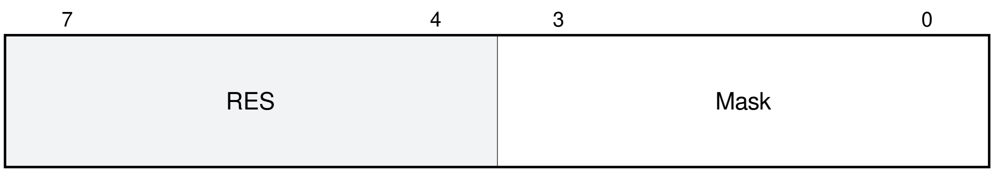
8.12 VGA Sequencer Registers

Bits	Field	Access	Description
0	8/9	RW	Divide dot clock by 8 or 9 0 Character clock period is 9 dots wide. 1 Character clock period is 8 dots wide.

The VSCMOR determines whether display refresh is enabled and controls various sequencer timing functions.

8.12.5 VGA Sequencer Plane Mask Register

Mnemonic:	VSPLMR
Index:	2
Reset value:	Undefined



Bits	Field	Access	Description
7:4	RES	MBZ	Reserved, must be zero. Read value is unpredictable.
3:0	Mask	RW	Each bit masks the corresponding memory plane, as follows: 0 Disable memory plane on CPU write operations. 1 Enable memory plane on CPU write operations.

The VSPLMR determines whether memory planes 0 through 3 can be written.

8.12.6 VGA Sequencer Character Map Select Register

Mnemonic: VSCMSR
Index: 3
Reset value: Undefined

7	6	5	4	3	2	1	0
RES		SAH	SBH	SA		SB	

Bits	Field	Access	Description
7:6	RES	MBZ	Reserved, must be zero. Read value is unpredictable.
5	SAH	RW	Select character generator A (high order).
4	SBH	RW	Select character generator B (high order).
3:2	SA	RW	Select character generator A.
1:0	SB	RW	Select character generator B.

VSCMSR bits <5,3:2> select character set (font) A and bits <4,1:0> select character set B.

8.12.7 VGA Sequencer Memory Mode Register

Mnemonic: VSMMOR
Index: 4
Reset value: Undefined

7 — 4	3	2	1	0
RES	C4	O/E	EM	RES

Bits	Field	Access	Description
7:4	RES	MBZ	Reserved, must be zero. Read value is unpredictable.
3	C4	RW	Chain 4 0 Normal buffer addressing. 1 CPU A0 = plane select bit 0 CPU A1 = plane select bit 1 The VGA graphics controller read map select register (Section 8.15.7) is ignored
2	O/E	RW	Odd or even 0 Even/odd addressing. Even addresses select planes 0 and 2, odd address select planes 1 and 3. 1 Normal buffer addressing. This bit is the opposite of VGMODR <4> (Section 8.15.8).
1	EM	RW	Extended memory 0 No extended memory, display memory $\leq$ 64KB. 1 Extended memory present, display memory > 64KB.
0	RES	MBZ	Reserved, must be zero. Read value is unpredictable.

The VSMMOR determines whether extended memory is enabled and how the planes are addressed.

8.13 VGA CRT Controller Registers

The VGA CRT controller (CRTC) registers control display parameters and are accessed through the VGA CRTC index register (VCINXR) and the VGA CRTC data register (VCDATR), in either monochrome or color display mode. The VCINXR and VCDATR have monochrome and display mode addresses, and the choice of address pair implicitly selects the mode (3B4 and 3B5 for monochrome, and 3D4 and 3D5 for color). The remaining VGA CRTC registers are identical in either mode.

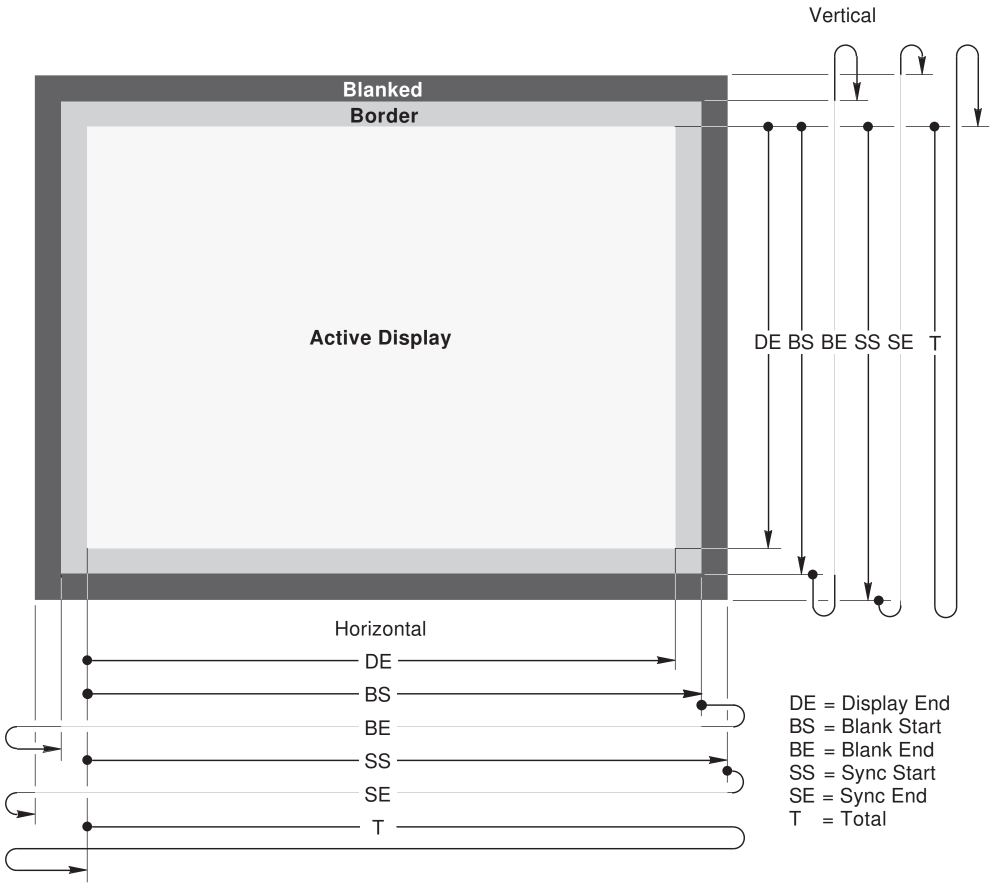
Note

When operating in the accelerated (2DA) modes (GDER <22> is clear, Section 8.5.2), the active display area is derived from the CRTC register values during blank time rather than the register values during display enable time. This also implies that borders are not available in accelerated modes.

The VCINXR and VCDATR also provide access to the VGA extended registers (Section 8.14).

Figure 8–9 shows some of the horizontal and vertical screen parameters controlled by CRTC registers.

Figure 8–9 Screen Parameters



8.13 VGA CRT Controller Registers

8.13.1 VGA CRTC Index Register

Mnemonic:	VCINXR
Monochrome address:	3B4
Color address:	3D4
Reset value:	Undefined

7 … 0
CRTC or Extended Register Index

Bits	Field	Access	Description
7:0	CRTC or Extended Register Index	RW	Selects the VGA CRTC or extended register to be read or written through the VGA CRTC data register, as listed in Table 8–15.

The VCINXR indexes the CRTC registers or the extended registers, depending on the value of <7:5>, as follows:

000 Index CRTC registers.
100 Index extended registers.
Unused codes are reserved.

Table 8–15 lists the registers indexed by the VCINXR.

Table 8–15 VGA CRTC and Extended Register Indices

Index	Register	Mnemonic	Section
00	Horizontal total register	VCHTOR	8.13.3
01	Horizontal display end register	VCHDER	8.13.4
02	Start horizontal blank register	VCHBSR	8.13.5
03	End horizontal blank register	VCHBER	8.13.5
04	Start horizontal sync register	VCHSSR	8.13.6
05	End horizontal sync register	VCHSER	8.13.6
06	Vertical total register	VCVTOR	8.13.7
07	Overflow register	VCOVRR	8.13.8

(continued on next page)

Table 8–15 (Cont.) VGA CRTC and Extended Register Indices

Index	Register	Mnemonic	Section
08	Preset row register	VCPROR	8.13.9
09	Maximum scanline register	VCMSLR	8.13.10
0A	Cursor start register	VCCUSR	8.13.11
0B	Cursor end register	VCCUER	8.13.11
0C	Start address high register	VCSAHR	8.13.12
0D	Start address low register	VCSALR	8.13.12
0E	Cursor location high register	VCCLHR	8.13.13
0F	Cursor location low register	VCCLLR	8.13.13
10	Start vertical sync register	VCVSSR	8.13.14
11	End vertical sync register	VCVSER	8.13.14
12	End vertical display register	VCVDER	8.13.15
13	Offset register	VCOFFR	8.13.16
14	Underline row scan register	VCULRR	8.13.17
15	Start vertical blanking register	VCVBSR	8.13.18
16	End vertical blanking register	VCVBER	8.13.18
17	Mode control register	VCMODR	8.13.19
18	Line compare register	VCLCMR	8.13.20
8D	Extended paging control register	VXPCOR	8.14.1
90	Extended host page offset A register	VXHPAR	8.14.2
91	Extended host page offset B register	VXHPBR	8.14.2
93	Extended split-screen start address low byte register	VXSALR	8.14.3
94	Extended split-screen start address high byte register	VXSAHR	8.14.3
97	Extended interlace control register	VXICOR	8.14.4
9A	Extended equalization start register	VXEQSR	8.14.5
9B	Extended equalization end register	VXEQER	8.14.5
9C	Extended half-line register	VXHLNR	8.14.6
9D	Extended timing control A register	VXTCAR	8.14.7
9E	Extended timing control B register	VXTCBR	8.14.8
A0	Extended video FIFO control register	VXFCOR	8.14.9
A1	VGA extended clock control A register	VXCKAR	8.14.10
A2	VGA extended clock control B register	VXCKBR	8.14.10
A3	Extended interface control register	VXEICR	8.14.11

Unused codes are reserved

8.13.2 VGA CRTC Data Register

Mnemonic:	VCDATR
Monochrome address:	3B5
Color address:	3D5
Reset value:	Undefined

7 … 0
CRTC or Extended Register Data

Bits	Field	Access	Description
7:0	CRTC or Extended Register Data	RW	Indexed VGA CRTC or extended register read or write data.

The VCDATR contains the read or write data for the VGA CRTC or extended register indexed by the VCINXR (Section 8.13.1).

8.13.3 VGA CRTC Horizontal Total Register

Mnemonic: VCHTOR
Index: 00
Reset value: Undefined

7 … 0
Horizontal Total

Bits	Field	Access	Description
7:0	Horizontal Total	RW	The number of character clocks minus 5 from the start of one active display scanline to the start of the next active display scanline.

The VCHTOR specifies the number of character clocks from the start of one scanline to the start of the next scanline; that is, the total number of character clocks minus 5 during horizontal active display, blanking, retrace, and borders (Figure 8–9). The number of pixels per character clock is specified by the VGA sequencer clocking mode register (VSCMOR <0>, Section 8.12.4).

8.13.4 VGA CRTC Horizontal Display End Register

Mnemonic: VCHDER
Index: 01
Reset value: Undefined

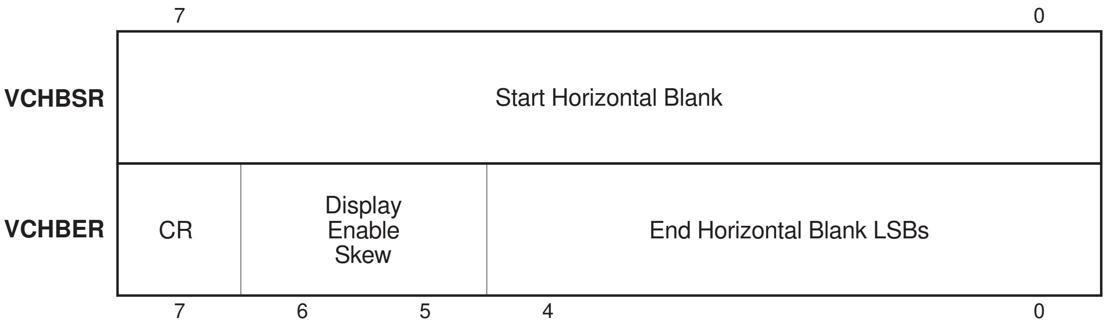
7	0
Horizontal Display End	

Bits	Field	Access	Description
7:0	Horizontal Display End	RW	The number of character clocks minus 1 from the start to the end of horizontal active display.

The VCHDER specifies the number of character clocks minus 1 during the active display of a scanline (see Figure 8–9). The number of pixels per character clock is specified by the VGA sequencer clocking mode register (VSCMOR <0>, Section 8.12.4).

8.13.5 VGA CRTC Start and End Horizontal Blank Registers

Mnemonic:	VCHBSR, VCHBER
VCHBSR index:	02
VCHBER index:	03
VCHBSR, VCHBER reset value:	Undefined



Bits	Field	Access	Description
VCHBSR			
7:0	Start Horizontal Blank	RW	The number of character clocks from the start of an active horizontal display to the start of horizontal blanking. This value must be greater than horizontal display end (Section 8.13.4).
VCHBER			
7	CR	RW	Compatible read—enables access to the VGA CRTC start and end vertical sync registers (VCVSSR and VCVSER, Section 8.13.14). 0 Disable access 1 Enable access
6:5	Display Enable Skew	RW	Specifies the number of characters by which horizontal display enable is delayed. 00 0 characters 01 1 character 10 2 characters 11 3 characters

8.13 VGA CRT Controller Registers

Bits	Field	Access	Description
VCHBER			
4:0	End Horizontal Blank LSBs	RW	The number of character clocks between the start and end of horizontal blanking. This value must not extend the blanking period beyond horizontal total (Section 8.13.3). Bit <5> of this value is in VCHSER <7> (Section 8.13.6).

The VCHBSR and VCHBER specify the start and width of the horizontal blanking period (see Figure 8–9). The number of pixels per character clock is specified by the VGA sequencer clocking mode register (VSCMOR <0>, Section 8.12.4).

8.13.6 VGA CRTC Start and End Horizontal Sync Registers

Mnemonic:	VCHSSR, VCHSER
VCHSSR index:	04
VCHSER index:	05
VCHSSR, VCHSER reset value:	Undefined

Register	7	6	5	4 … 0
VCHSSR	Start Horizontal Sync			
VCHSER	EHB	Horizontal Sync Delay		End Horizontal Sync

Bits	Field	Access	Description
VCHSSR			
7:0	Start Horizontal Sync	RW	The number of character clocks from the start of an active horizontal display to the start of horizontal sync. This value must be greater than or equal to horizontal display end (Section 8.13.4), and less than or equal to horizontal total (Section 8.13.3) minus 4.
VCHSER			
7	EHB	RW	End horizontal blank—EHB bit <5>. Bits <4:0> are specified in VCHBER <4:0> (Section 8.13.5).
6:5	Horizontal Sync Delay	RW	Specifies the number of characters by which horizontal sync is to be delayed. 00 0 characters 01 1 character 10 2 characters 11 3 characters
4:0	End Horizontal Sync	RW	The number of character clocks between the start and end of horizontal sync. This value must not extend the sync period beyond horizontal total (Section 8.13.3) and should not extend the sync period beyond the end of the horizontal blank period (Section 8.13.5).

8.13 VGA CRT Controller Registers

The VCHSSR and VCHSER specify the start and width of the horizontal sync pulse (retrace — see Figure 8–9). The number of pixels per character clock is specified by the VGA sequencer clocking mode register (VSCMOR <0>, Section 8.12.4).

8.13.7 VGA CRTC Vertical Total Register

Mnemonic: VCVTOR
Index: 06
Reset value: Undefined

7 … 0
Vertical Total LSBs

Bits	Field	Access	Description
7:0	Vertical Total LSBs	RW	The number of scanlines minus 2 from the start of one frame to the start of the next frame. Bits <9:8> are in VCOVRR <5,0> (Section 8.13.8).

The VCVTOR specifies the number of scanlines from the start of one frame to the start of the next frame; that is, the total number of scanlines minus 2 during active display, blanking, retrace, and borders (see Figure 8–9).

8.13.8 VGA CRTC Overflow Register

Mnemonic: VCOVRR
Index: 07
Reset value: Undefined

7	6	5	4	3	2	1	0
SVS<9>	EVD<9>	VT<9>	LC<8>	SVB<8>	SVS<8>	EVD<8>	VT<8>

Bits	Field	Access	Description
7	SVS<9>	RW	Start vertical sync bit 9—see Section 8.13.14.
6	EVD<9>	RW	End vertical display bit 9—see Section 8.13.15.
5	VT<9>	RW	Vertical total bit 9—see Section 8.13.7.
4	LC<8>	RW	Line compare bit 8—see Section 8.13.20.
3	SVB<8>	RW	Start vertical blanking bit 8—see Section 8.13.18.
2	SVS<8>	RW	Start vertical sync bit 8—see Section 8.13.14.
1	EVD<8>	RW	End vertical display bit 8—see Section 8.13.15.
0	VT<8>	RW	Vertical total bit 8—see Section 8.13.7.

The VCOVRR holds some of the MSBs of several 10-bit vertical screen parameter values.

8.13.9 VGA CRTC Preset Row Register

Mnemonic: VCPROR
Index: 08
Reset value: Undefined

7	6 5	4 0
RES	BPAN	PROW

Bits	Field	Access	Description
7	RES	MBZ	Reserved, must be zero. Read value is unpredictable.
6:5	BPAN	RW	Byte panning—specifies the number of bytes by which the display is to shift left. Should be combined with the panning pixel count in VAPXPR <3:0> (Section 8.16.6). 00 No panning 01 Pan 1 byte 10 Pan 2 bytes 11 Pan 3 bytes
4:0	PROW	RW	Preset row—the initial scanline for the first character row.

The VCPROR specifies the character-row scanline where scrolling begins. It also specifies the coarse panning value.

8.13.10 VGA CRTC Maximum Scanline Register

Mnemonic: VCMSLR
Index: 09
Reset value: Undefined

7	6	5	4 … 0
SD	LC<9>	SVB<9>	Height

Bits	Field	Access	Description
7	SD	RW	Scan double—when set, enables line-doubling during display.
6	LC<9>	RW	Line compare bit 9—see Section 8.13.20.
5	SVB <9>	RW	Start vertical blanking bit 9—see Section 8.13.18.
4:0	Height	RW	The number of scanlines per character row minus 1.

The VCMSLR specifies the number of scanlines minus 1 in a character row. It also enables scanline-doubling and contains some of the 10-bit vertical screen parameter value MSBs.

8.13.11 VGA CRTC Cursor Start and End Registers

Mnemonic:	VCCUSR, VCCUER
VCCUSR index:	0A
VCCUER index:	0B
VCCUSR, VCCUER reset value:	Undefined

	7	6	5	4 ... 0
VCCUSR	RES		CEN	Cursor Start
VCCUER	RES	Cursor Skew		Cursor End

Bits	Field	Access	Description
VCCUSR			
7:6	RES	MBZ	Reserved, must be zero. Read value is unpredictable.
5	CEN	RW	Cursor enable 0 The cursor is enabled. 1 The cursor is disabled.
4:0	Cursor Start	RW	Specifies the start of the cursor within a character row.
VCCUER			
7	RES	MBZ	Reserved, must be zero. Read value is unpredictable.
6:5	Cursor Skew	RW	Specifies the number of character clocks by which the cursor is to be delayed. 00 No delay 01 1 character clock 10 2 character clocks 11 3 character clocks
4:0	Cursor End	RW	Specifies the end of the cursor within a character row.

8.13 VGA CRT Controller Registers

The VCCUSR and VCCUER specify the scanlines on which the text cursor starts and ends. They also enable the text cursor and specify its skew. The number of pixels per character clock is specified by the VGA sequencer clocking mode register (VSCMOR <0>, Section 8.12.4).

8.13.12 VGA CRTC Start Address High and Low Registers

Mnemonic:	VCSAHR, VCSALR
VCSAHR index:	0C
VCSALR index:	0D
VCSAHR, VCSALR reset value:	Undefined

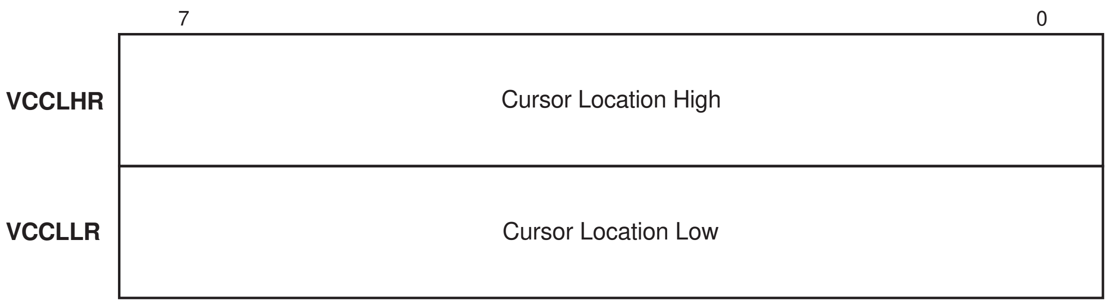
	7 … 0
VCSAHR	Start Address High
VCSALR	Start Address Low

Bits	Field	Access	Description
VCSAHR			
7:0	Start Address High	RW	The 8 MSBs of the display buffer starting address in memory.
VCSALR			
7:0	Start Address Low	RW	The 8 LSBs of the display buffer starting address in memory.

The VCSAHR and VCSALR values are combined to specify the starting address of the display buffer (that is, the upper-left corner of the screen).

8.13.13 VGA CRTC Cursor Location High and Low Registers

Mnemonic:	VCCLHR, VCCLLR
VCCLHR index:	0E
VCCLLR index:	0F
VCCLHR, VCCLLR reset value:	Undefined

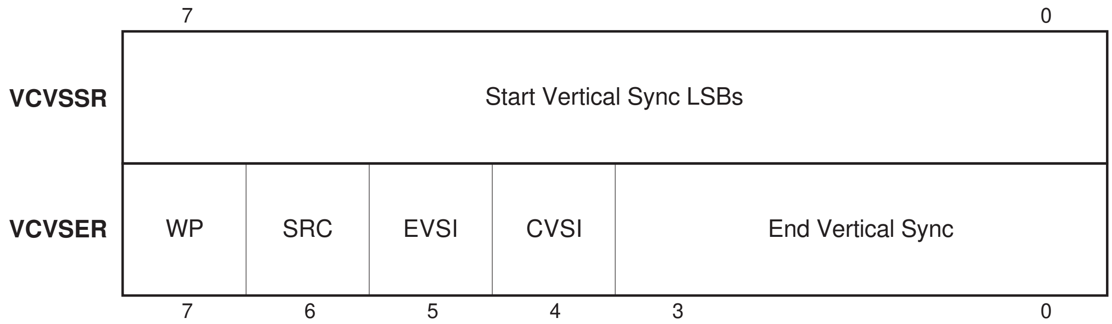


Bits	Field	Access	Description
VCCLHR			
7:0	Cursor Location High	RW	Specifies the 8 MSBs of the address at which the cursor is displayed.
VCCLLR			
7:0	Cursor Location Low	RW	Specifies the 8 LSBs of the address at which the cursor is displayed.

The VCCLHR and VCCLLR values are combined to specify the memory location at which the text cursor is to be displayed. If the display buffer is relocated, the cursor location should be updated.

8.13.14 VGA CRTC Start and End Vertical Sync Register

Mnemonic:	VCVSSR, VCVSER
VCVSSR index:	10
VCVSER index:	11
VCVSSR, VCVSER reset value:	Undefined



Bits	Field	Access	Description
VCVSSR			
7:0	Start Vertical Sync LSBs	RW	The number of scanlines from the start of the active display to the start of the vertical sync pulse. Bits <9:8> are in VCOVRR <7,3> (Section 8.13.8).
VCVSER			
7	WP	RW	Write protect—enables writes to VGA CRTC registers index 7 through 0. This bit does not affect the LC<8> bit in the VGA CRTC overflow register (VCOVRR <4>, Section 8.13.8). 0 Disables protection, enables writes to CRTC index 7 through 0. 1 Enables protection, disables writes to CRTC index 7 through 0.
6	SRC	RW	Select refresh cycles—implemented only for compatibility and does not affect 21130 operation. The generic VGA definition is as follows: 0 Select 5 DRAM refresh cycles per scanline. 1 Select 3 DRAM refresh cycles per scanline.

8.13 VGA CRT Controller Registers

Bits	Field	Access	Description
VCVSER			
5	EVSI	RW	Enable vertical sync interrupt 0 The vertical retrace interrupt is enabled. 1 The vertical retrace interrupt is disabled.
4	CVSI	RW	Clear vertical sync interrupt 0 Clear vertical retrace status. 1 No effect.
3:0	End Vertical Sync	RW	The number of scanlines during which vertical sync is asserted. This value must not extend vertical sync beyond vertical total (Section 8.13.7).

The VCVSSR and VCVSER specify the start and width of vertical sync (see Figure 8–9). The VCVSER also controls the vertical sync interrupt.

Note

- For resolutions with 1024 or more vertical scanlines, vertical sync start and vertical blank start must be specified such that the vertical front porch is a minimum of 2 scanlines.
- To access the VCVSSR and VCVSER, the compatible read bit in the VGA CRTC end horizontal blank register (VCHBER <7>, Section 8.13.5) must be set.

8.13.15 VGA CRTC End Vertical Display Register

Mnemonic: VCVDER
Index: 12
Reset value: Undefined

7	0
End Vertical Display LSBs	

Bits	Field	Access	Description
7:0	End Vertical Display LSBs	RW	The number of scanlines from the start of the active display to the last displayed scanline. Bits <9:8> are in VCOVRR <6,1> (Section 8.13.8).

The VCVDER specifies the last scanline to be displayed (see Figure 8–9).

8.13 VGA CRT Controller Registers

8.13.16 VGA CRTC Offset Register

Mnemonic: VCOFFR
Index: 13
Reset value: Undefined

7	0
Offset	

Bits	Field	Access	Description
7:0	Offset	RW	Specifies the width of the display.

The VCOFFR specifies the width of the display in pixels. The value is computed by dividing the difference between the addresses of two vertically adjacent pixels by 2 or 4, for word- or byte-mode addressing. See the address mode selection bit VCMODR <6> (Section 8.13.19).

8.13.17 VGA CRTC Underline Row Scan Register

Mnemonic: VCULRR
Index: 14
Reset value: Undefined

7	6	5	4 … 0
RES	DW	CB4	Underline Location

Bits	Field	Access	Description
7	RES	MBZ	Reserved, must be zero. Read value is unpredictable.
6	DW	RW	Double-word mode 0 Normal word addressing — VCMODR <6> (Section 8.13.19) selects byte- or word-mode addressing. 1 Double-word addressing. **Note:** Bits <6:5> should be in the same state.
5	CB4	RW	Count by 4 0 Normal clocking. 1 Divide character clock to address counter by 4. **Note:** Bits <6:5> should be in the same state.
4:0	Underline Location	RW	The number of scanlines minus 1 at which the underline is located within a character cell.

The VCULRR specifies the vertical location of the underline in a character cell. It also determines whether display memory is addressed on word (16-bit) or Dword (32-bit) boundaries.

8.13 VGA CRT Controller Registers

8.13.18 VGA CRTC Start and End Vertical Blanking Registers

Mnemonic:	VCVBSR, VCVBER
VCVBSR index:	15
VCVBER index:	16
VCVBSR, VCVBER reset value:	Undefined

	7 … 0
VCVBSR	Start Vertical Blanking LSBs
VCVBER	End Vertical Blanking

Bits	Field	Access	Description
VCVBSR			
7:0	Start Vertical Blanking LSBs	RW	The number of scanlines from the start of the active display to the start of vertical blanking. Bit <9> is in VCMSLR <5> (Section 8.13.10) and bit <8> is in VCOVRR <3> (Section 8.13.8).
VCVBER			
7:0	End Vertical Blanking	RW	The number of scanlines from the start to the end of vertical blanking.

The VCVBSR and VCVBER specify the scanlines at which vertical blanking starts and ends (see Figure 8–9).

Note

For resolutions with 1024 or more vertical scanlines, vertical sync start and vertical blanking start must be specified such that the vertical front porch is a minimum of 2 scanlines.

8.13.19 VGA CRTC Mode Control Register

Mnemonic: VCMODR
Index: 17
Reset value: Undefined

7	6	5	4	3	2	1	0
HR	WB	AW	RES	CB2	HRS	SRS	CMS

Bits	Field	Access	Description
7	HR	RW	Hardware reset 0 Resets and holds all video control signals. 1 Enables horizontal and vertical control signals.
6	WB	RW	Word or byte mode select 0 Word mode selected, addresses shifted left 1 bit—enables bit <5>. 1 Byte mode selected, addresses unshifted.
5	AW	RW	Address wrap—ignored if <6> = 1; otherwise: 0 Select address bit <13> to be sent to LSB of display memory. 1 Select address bit <15> to be sent to LSB of display memory.
4	RES	MBZ	Reserved, must be zero. Read value is unpredictable.
3	CB2	RW	Count by 2 0 Memory address counter is clocked by character clock. 1 Memory address counter is clocked by every other character clock.
2	HRS	RW	Horizontal retrace select 0 Clock scanline counter with every horizontal sync. 1 Clock scanline counter with every other horizontal sync pulse.
1	SRS	RW	Select row scan counter—for Hercules compatibility 0 Sends row scan counter bit <1> to memory address bus bit <14> during active display. 1 Memory addresses are output sequentially.

8.13 VGA CRT Controller Registers

Bits	Field	Access	Description
0	CMS	RW	Compatibility mode support—for CGA compatibility
			0 Substitutes row scan address bit <0> for memory address <13>.
			1 No substitution, memory is sequentially accessed.

The VCMODR contains miscellaneous CRTC control bits.

8.13.20 VGA CRTC Line Compare Register

Mnemonic: VCLCMR
Index: 18
Reset value: Undefined

7 … 0
Line Compare LSBs

Bits	Field	Access	Description
7:0	Line Compare LSBs	RW	The scanline where the top screen ends and the bottom screen begins. Bit <9> is in VCMSLR <6> (Section 8.13.10) and bit <8> is in VCOVRR <4> (Section 8.13.8).

The VCLCMR specifies the scanline where the top screen ends and the bottom screen begins in a vertically split display.

8.14 VGA Extended Registers

To enable the VGA extended registers, the value of VGA CRTC index register (VCINXR, Section 8.13.1) bits <7:6> must be 10_2. The data for the extended registers is contained in the VGA CRTC data register (VCDATR). Table 8–15 in Section 8.13.1 lists the registers indexed by the VCINXR.

8.14 VGA Extended Registers

8.14.1 VGA Extended Paging Control Register

Mnemonic: VXPCOR
Index: 8D
Reset value: Undefined

7	6	5	4	3	2	1	0
VGAAD	LADMD	HPAGE	CURA16	RES	SAA16	RES	VSEG

Bits	Field	Access	Description
7	VGAAD	RW	VGA compatibility address 0 Standard VGA mode video address (default). 1 Enables video address to count above 64KB for 640 × 480 × 256 mode.
6	LADMD	RW	Linear address mode 0 Linear addressing is disabled (default). 1 Enables linearly aligned video memory addressing. This bit should be set when HPAGE (<5>) is set.
5	HPAGE	RW	Host page select 0 Only host page offset A is used (default). 1 Host page offset A and B (Section 8.14.2) are used. Enables 32K windows in the 64K aperture. The host can access the first 32K window at A0000 through A7FFF and the second 32K window at A8000 through AFFFF.
4	CURA16	RW	Cursor address bit 16 0 The value of cursor address bit 16 is 0. 1 The value of cursor address bit 16 is 1.
3	RES	MBZ	Reserved, must be zero. Read value is unpredictable.
2	SAA16	RW	Split-screen address bit 16 0 The value of split-screen address bit 16 is 0. 1 The value of split-screen address bit 16 is 1.
1	RES	MBZ	Reserved, must be zero. Read value is unpredictable.

Bits	Field	Access	Description
0	VSEG	RW	Video address segment 0 Selects the first 256KB of video memory. 1 Selects the second 256KB of video memory. Determines which 256KB video memory is the active display window.

The VXPCOR defines the interface paging (address) controls.

8.14 VGA Extended Registers

8.14.2 VGA Extended Host Page Offset A and B Registers

Mnemonic:	VXHPAR, VXHPBR
VXHPAR index:	90
VXHPBR index:	91
VXHPAR, VXHPBR reset value:	Undefined

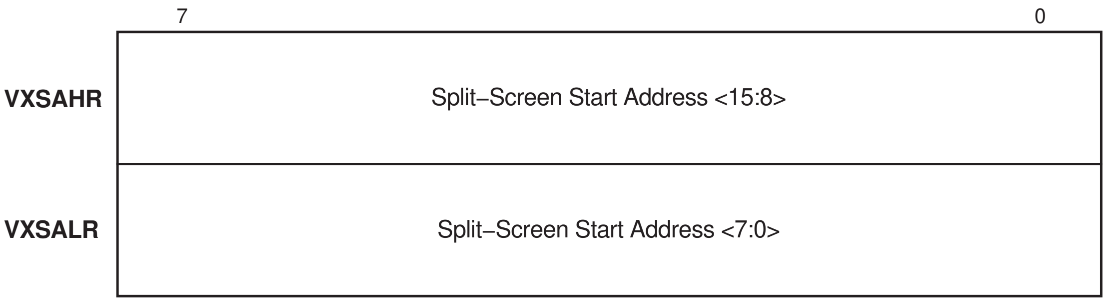
	7	6 … 0
VXHPAR	RES	Host Page Offset A
VXHPBR	RES	Host Page Offset B

Bits	Field	Access	Description
VXHPAR			
7	RES	MBZ	Reserved, must be zero. Read value is unpredictable.
6:0	Host Page Offset A	RW	The A offset within the 32KB windows boundary.
VXHPBR			
7	RES	MBZ	Reserved, must be zero. Read value is unpredictable.
6:0	Host Page Offset B	RW	The B offset within the 32KB windows boundary.

The VXHPAR and VXHPBR define the offset within the 32KB windows boundary. Incrementing the offset value by 1 increases the page address by 4KB.

8.14.3 VGA Extended Split-Screen Start Address High and Low Byte Register

Mnemonic:	VXSAHR, VXSALR
VXSAHR index:	94
VXSALR index:	93
VXSAHR, VXSALR reset value:	Undefined



Bits	Field	Access	Description
VXSAHR			
7:0	Split-Screen Start Address <15:8>	RW	Split-screen starting address high byte.
VXSALR			
7:0	Split-Screen Start Address <7:0>	RW	Split-screen starting address low byte.

The VXSAHR and VXSALR values are combined to define the 16-bit split-screen starting address.

8.14 VGA Extended Registers

8.14.4 VGA Extended Interlace Control Register

Mnemonic: VXICOR
Index: 97
Reset value: Undefined

7	6	5	4	3	2	1	0
Equalization Start <9:8>		CSEN	INLACE	RES		Dot Clock Divisor	

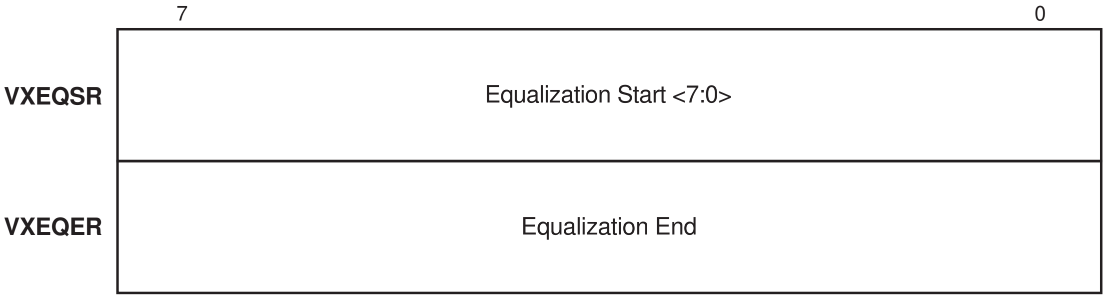
Bits	Field	Access	Description
7:6	Equalization Start <9:8>	RW	Equalization start MSBs—see Section 8.14.5.
5	CSEN	RW	Composite sync enable 0 Composite sync is disabled (default). 1 Composite sync is enabled.
4	INLACE	RW	Interlaced enabled 0 Interlace is disabled (default). 1 Interlace is enabled.
3:2	RES	MBZ	Reserved, must be zero. Read value is unpredictable.
1:0	Dot Clock Divisor	RW	Determines the value of the dot clock divisor, as follows:

Code	Divide dot clock by . . .
00	1 (default)
01	2
10	4
11	Reserved

The VXICOR defines the controls for the interface composite sync, interlace, and dot clock.

8.14.5 VGA Extended Equalization Start and End Registers

Mnemonic:	VXEQSR, VXEQER
VXEQSR index:	9A
VXEQER index:	9B
VXEQSR, VXEQER reset value:	Undefined



Bits	Field	Access	Description
VXEQSR			
7:0	Equalization Start <7:0>	RW	The LSBs of the starting location of the composite sync equalization pulse. The MSBs are contained in VXICOR <7:6> (Section 8.14.4).
VXEQER			
7:0	Equalization End	RW	The ending location of the composite sync equalization pulse.

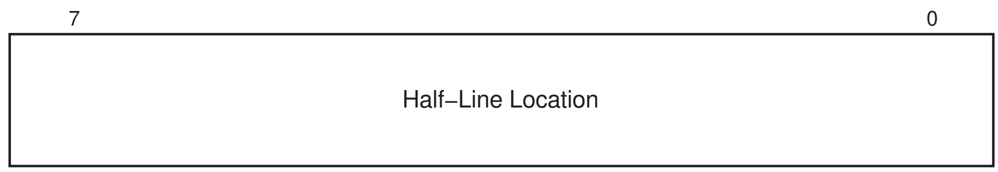
The VXEQSR and VGA extended interlace control register (VXICOR, Section 8.14.4) define the starting location of the composite sync equalization pulse.

The VXEQER defines the ending location of the composite sync equalization pulse.

The VXEQSR and VXEQER are in effect only when composite sync is enabled (VXICOR <5>, Section 8.14.4).

8.14.6 VGA Extended Half-Line Register

Mnemonic: VXHLNR
Index: 9C
Reset value: Undefined



Bits	Field	Access	Description
7:0	Half-Line Location	RW	The half-line location of the composite sync.

The VXHLNR defines the half-line location of the composite sync pulse. This register is in effect only when composite sync is enabled (VXICOR <5>, Section 8.14.4).

8.14.7 VGA Extended Timing Control A Register

Mnemonic: VXTCAR
Index: 9D
Reset value: Undefined

7	6 5	4 0
CAS Precharge Period	CAS Width	Horizontal Sync Width

Bits	Field	Access	Description
7	CAS Precharge Period	RW	Specifies the CAS precharge period in number of SCLKs. 0 The CAS precharge period is 1 SCLK. 1 The CAS precharge period is 2 SCLKs.
6:5	CAS Width	RW	Specifies the CAS width in number of SCLKs. 00 The CAS width is 1 SLCK. 01 The CAS width is 2 SLCKs. 10 The CAS width is 3 SLCKs. 11 The CAS width is 3 SLCKs.
4:0	Horizontal Sync Width	RW	Specifies the horizontal sync width in character clock units. 00000 The HYSNC pulse is 1 character wide. ⋮ ⋮ 11111 The HYSNC pulse is 64 characters wide. This field is in effect only when composite sync is enabled (VXICOR <5>, Section 8.14.4).

The VXTCAR and VGA extended timing control B register (VXTCBR, Section 8.14.8) define the interface timing controls.

8.14.8 VGA Extended Timing Control B Register

Mnemonic: VXTCBR
Index: 9E
Reset value: Undefined

7	6	5	4	3	2	1 0
RES	IRQEN	Burst	MCD	RMD	RAS Setup Period	RAS Precharge Period

Bits	Field	Access	Description
7	RES	MBZ	Reserved, must be zero. Read value is unpredictable.
6	IRQEN	RW	Interrupt enable 0 Vertical sync interrupts to the system are disabled (default). 1 Vertical sync interrupts to the system are enabled.
5	Burst	RW	DRAM burst mode 0 Burst mode is disabled (default). 1 Burst mode is enabled.
4	MCD	RW	Multiplexer-to-CAS delay 0 The multiplexer-to-CAS delay is 1 SCLK (default). 1 The multiplexer-to-CAS delay is 2 SCLKs.
3	RMD	RW	RAS-to-multiplexer delay 0 The RAS-to-multiplexer delay is 1 SCLK (default). 1 The RAS-to-multiplexer delay is 2 SCLKs.
2	RAS Setup Period	RW	Specifies the RAS setup period in number of SCLKs 0 The RAS setup period is 1 SCLK (default). 1 The RAS setup period is 2 SCLKs.
1:0	RAS Precharge Period	RW	Specifies the RAS precharge period in number of SCLKs 00 The RAS precharge period is 3 SCLKs. 01 The RAS precharge period is 4 SCLKs. 10 The RAS precharge period is 5 SCLKs. 11 The RAS precharge period is 5 SCLKs.

The VXTCBR and VGA extended timing control A register (VXTCAR, Section 8.14.7) define the interface timing controls.

8.14.9 VGA Extended Video FIFO Control Register

Mnemonic: VXFCOR
Index: A0
Reset value: Undefined

7 6	5	4	3 0
RES	FIFO Reset	FIFO Enable	FIFO Depth

Bits	Field	Access	Description
7:6	RES	MBZ	Reserved, must be zero. Read value is unpredictable.
5	FIFO Reset	RW	0 Disabled—the FIFO is active (default). 1 Enabled.
4	FIFO Enable	RW	0 Disable FIFO read counter. 1 Normal operation.
3:0	FIFO Depth	RW	Specifies the number of entries in the FIFO. 0001 1 entry 0010 2 entries 0100 4 entries 1000 8 entries

The VXFCOR contains several video FIFO (counter) control bits.

8.14.10 VGA Extended Clock Control A and B Registers

Mnemonic:	VXCKAR, VXCKBR
VXCKAR index:	A1
VXCKBR index:	A2
VXCKAR reset value:	<7:1> undefined, <0> = 0
VXCKBR reset value:	Undefined

	7	6	5	4	3	2	1	0
VXCKAR	RES	M Term (6:1)						VVDS
VXCKBR	RES (7:6)		L Term (5:4)		N Term (3:0)			

Bits	Field	Access	Description
VXCKAR			
7	RES	MBZ	Reserved, must be zero. Read value is unpredictable.
6:1	M Term	RW	Specifies the value of the pixel clock PLL multiplier. The maximum value is 63 and the minimum value is 25. When the **pci_rst#** and **gp_int#** signals are both asserted, this field is set to 42.
0	VVDS	RW	VGA variable dot clock select—specifies how the pixel clock frequency is determined in VGA mode. 0 The clock source field in the VGA miscellaneous output register (VEMISR <3:2>, Section 8.11.1) selects the pixel clock frequency. 1 The L, M, and N term fields determine the pixel clock frequency (Table 8–16). When the **pci_rst#** signal is asserted, this bit is forced to the inverse of the **gp_int#** signal.

Bits	Field	Access	Description
VXCKBR			
7:6	RES	MBZ	Reserved, must be zero. Read value is unpredictable.
5:4	L Term	RW	Specifies the value of the pixel clock PLL VCO output divider: 00 PLL VCO output ÷ 1 01 PLL VCO output ÷ 2 10 PLL VCO output ÷ 4 11 PLL VCO output ÷ 8
3:0	N Term	RW	Specifies the value of the pixel clock PLL divisor. The maximum value is 15 and the minimum value is 4. When the **pci_rst#** and **gp_int#** signals are both asserted, this field is set to 6.

The VXCKAR and VXCKBR configure the pixel clocks and select the VGA pixel clock (dot clock) source. (Note that the PCI clock control register, PCCR, configures the memory clock and specifies the sources for the memory, pixel, and test clocks — see Section 8.2.8.)

The pixel clock frequency is between 8 and 135 MHz, and is determined by the the 14.31818-MHz crystal frequency on the **xtal1** pin and the value in the L, M, and N fields, as follows:

$$pixel\ clock\ =\ 14.31818\left[\frac{M}{N}\left(\frac{1}{L}\right)\right]$$

Table 8–16 lists the values of some typical pixel clock frequencies.

8.14 VGA Extended Registers

Table 8–16 Typical Pixel Clock Frequencies

M[1]	N	L	MHz[2]	Description
41	9	11 (÷8)	8.153	Minimum pixel clock frequency
42	6	10 (÷4)	25.057	VGA1[3]
63	8	10 (÷4)	28.189	VGA2[3]
44	5	10 (÷4)	31.500	640 × 480 @ 75 Hz
62	9	01 (÷2)	49.318	800 × 600 @ 75 Hz
55	10	00 (÷1)	78.750	1024 × 768 @ 75 Hz
47	5	00 (÷1)	134.591	1280 × 1024 @ 75 Hz

[1]M term specified in VXCKAR <6:1>; L and N terms specified in VXCKBR <5:4,3:0>.

[2]Pixel clock frequency in MHz.

[3]VGA dot clock selections (VEMISR <3:2>, Section 8.11.1)

See Section 12.5 for more information about the clock generation function.

8.14.11 VGA Extended Interface Control Register

Mnemonic: VXEICR
Index: A3
Reset value: Undefined

7	6	5	4	3	2	1	0
RES		PLD	TVS	FWSTN	FWRAP	T	

Bits	Field	Access	Description
7:6	RES	MBZ	Reserved, must be zero. Read value is unpredictable.
5	PLD	RW	Parallel load strobe—for diagnostics. 0 Logical 0 1 Logical 1
4	TVS	RW	Vertical sync strobe—strobes the vertical sync pulse. 0 Logical 0 1 Logical 1
3	FWSTN	RW	FIFO wrap reset 0 Reset FWRAP bit (<2>). 1 Normal operation.
2	FWRAP	RW	FIFO wrapped—when read, indicates whether the video FIFO (counter) wrapped around. 0 FIFO did not wrap. 1 FIFO wrapped. Writes to this bit are ignored.
1:0	T	RW	These are test bits and must not be used.

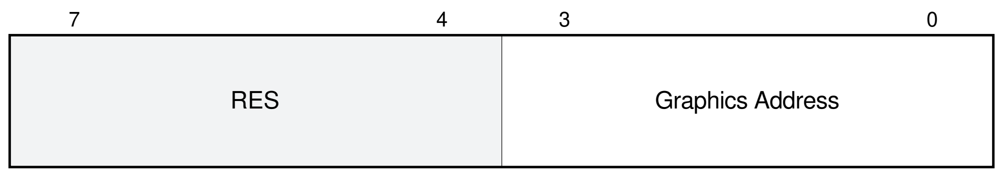
8.15 VGA Graphics Controller Registers

The VGA graphics controller registers are accessed through the VGA graphics controller index register (VGINXR, address 3CE) and the VGA graphics controller data register (VGDATR, address 3CF).

8.15 VGA Graphics Controller Registers

8.15.1 VGA Graphics Controller Index Register

Mnemonic: VGINXR
Address: 3CE
Reset value: Undefined



Bits	Field	Access	Description
7:4	RES	MBZ	Reserved, must be zero. Read value is unpredictable.
3:0	Graphics Address	RW	Index to the following VGA graphics controller registers: 0 Set/reset (VGSRER) 1 Enable set/reset (VGESRR) 2 Color compare (VGCCMR) 3 Data rotate (VGDROR) 4 Read map select (VGRMSR) 5 Mode (VGMODR) 6 Miscellaneous (VGMISR) 7 Color don't care (VGCDCR) 8 Bit mask (VGBMKR)

The VGINXR contains the index used to access the VGA graphics controller registers.

8.15.2 VGA Graphics Controller Data Register

Mnemonic: VGDATR
Address: 3CF
Reset value: Undefined

7 ... 0
Graphics Controller Data

Bits	Field	Access	Description
7:0	Graphics Controller Data	RW	Indexed graphics controller register read or write data.

The VGDATR contains the read or write data for the VGA graphics controller register indexed by the VGINXR (Section 8.15.1).

8.15.3 VGA Graphics Controller Set/Reset Register

Mnemonic: VGSRER
Index: 0
Reset value: Undefined

7 — 4	3 — 0
RES	Set/Reset Plane

Bits	Field	Access	Description
7:4	RES	MBZ	Reserved, must be zero. Read value is unpredictable.
3:0	Set/Reset Plane	RW	The data from each bit is written to the corresponding plane for write mode 0 or write mode 3. For write mode 0, the plane must be enabled through the VGESRR (Section 8.15.4).

The VGSRER is loaded with the pattern to be written to the display planes in write mode 0 or write mode 3. There is a one-to-one correspondence between the bits in this register and the display planes. The write modes are specified in the VGMODR (Section 8.15.8).

8.15.4 VGA Graphics Controller Enable Set/Reset Register

Mnemonic: VGESRR
Index: 1
Reset value: Undefined

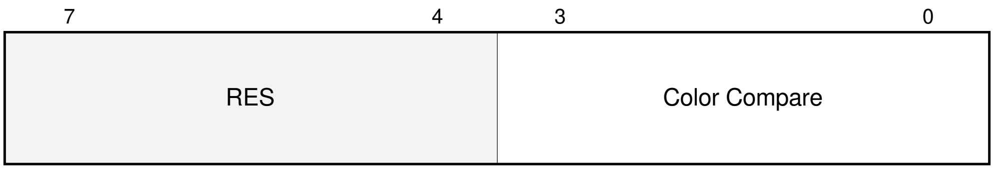
7 4	3 0
RES	Enable Set/Reset Plane

Bits	Field	Access	Description
7:4	RES	MBZ	Reserved, must be zero. Read value is unpredictable.
3:0	Enable Set/Reset Plane	RW	Each bit in this field determines the write data source for the corresponding plane, as follows: 0 Write CPU data to the plane. 1 Write VGSRER (Section 8.15.3) data to the plane.

The VGESRR specifies the source of write data for planes 3 through 0 in write mode 0. The write modes are specified in the VGMODR (Section 8.15.8).

8.15.5 VGA Graphics Controller Color Compare Register

Mnemonic: VGCCMR
Index: 2
Reset value: Undefined



Bits	Field	Access	Description
7:4	RES	MBZ	Reserved, must be zero. Read value is unpredictable.
3:0	Color Compare	RW	The value of each bit is compared to all 8 pixels in the corresponding memory plane.

In read mode 1, the value in the VGCCMR is compared to 8 adjacent pixels in memory planes 3 through 0. The VGCDCR (Section 8.15.10) selects the planes to be compared. The read returns a 1 in the bit positions corresponding to the pixels in all 4 planes that match the value in this field. The read modes are defined in the VGMODR (Section 8.15.8).

8.15.6 VGA Graphics Controller Data Rotate Register

Mnemonic: VGDROR
Index: 3
Reset value: Undefined

7 5	4 3	2 0
RES	Function Select	Rotate Count

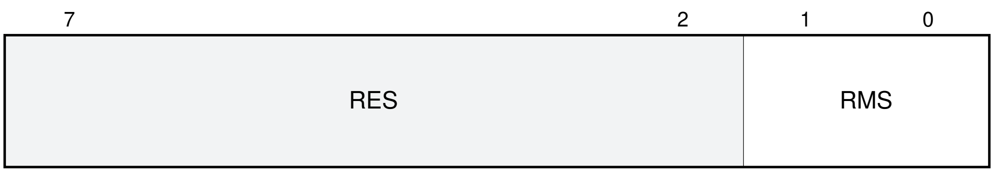
Bits	Field	Access	Description
7:5	RES	MBZ	Reserved, must be zero. Read value is unpredictable.
4:3	Function Select	RW	Specifies the logical operation that is to be performed using write data and latched memory data. 00 Write data is unmodified. 01 Write data is ANDed with latched memory data. 10 Write data is ORed with latched memory data. 11 Write data is XORed with latched memory data.
2:0	Rotate Count	RW	Specifies the number of bits that write data is to be rotated to the right.

The VGDROR function select field (<4:3>) specifies a logical operation for the write data in write mode 0. The write data can be new data from the host or data from the set/reset logic. The write modes are specified in the VGMODR (Section 8.15.8).

The rotate count field (<2:0>) can be used in any write mode. It specifies the number of bit positions that write data from the host is to be rotated to the right before it is modified by the set/reset logic.

8.15.7 VGA Graphics Controller Read Map Select Register

Mnemonic: VGRMSR
Index: 4
Reset value: Undefined



Bits	Field	Access	Description
7:2	RES	MBZ	Reserved, must be zero. Read value is unpredictable.
1:0	RMS	RW	Read map select—specifies the plane to be read in read mode 0. 00 Plane 0 01 Plane 1 10 Plane 2 11 Plane 3

The VGRMSR specifies the display memory plane to be read in read mode 0. The read modes are specified in the VGMODR (Section 8.15.8).

8.15.8 VGA Graphics Controller Mode Register

Mnemonic: VGMODR
Index: 5
Reset value: Undefined

7	6	5	4	3	2	1	0
RES	256CM	SR	O/E	RM	RES	WM	

Bits	Field	Access	Description
7	RES	MBZ	Reserved, must be zero. Read value is unpredictable.
6	256CM	RW	256-color mode 0 Configure shift registers for 2, 4, or 16 colors. 1 Configure shift registers for 256 colors.
5	SR	RW	Shift register 0 Configure shift registers for EGA or VGA compatibility. 1 Configure shift registers for CGA compatibility.
4	O/E	RW	Odd or even 0 Normal buffer addressing. 1 Even/odd addressing. This bit is the opposite of VSMMOR <2> (Section 8.12.7).
3	RM	RW	Read mode 0 Read mode 0—read data from the planes selected by the VGRMSR (Section 8.15.7). 1 Read mode 1—read comparison of the planes and the VGCCMR (Section 8.15.5).
2	RES	MBZ	Reserved, must be zero. Read value is unpredictable.
1:0	WM	RW	Write mode—specifies how planes enabled in the VGA sequencer plane mask register (VSPLMR, Section 8.12.5) are to be written. See Table 8–17.

The VGMODR controls several graphics controller functions including the read and write modes (Table 8–17).

8.15 VGA Graphics Controller Registers

Table 8–17 VGA Graphics Controller Write Modes

VGMODR <1:0>	Write Mode	Description
00	0	If set/reset is not enabled (VGESRR, Section 8.15.4), each plane is written with the CPU data rotated by the number of bits specified in the data rotate register (VGDROR, Section 8.15.6). If set/reset is enabled, each enabled plane is written with the value contained in the set/reset register (VGSRER, Section 8.15.3). Write data is modified as specified by the VGDROR function select field, and masked according to the bit mask register (VGBMKR, Section 8.15.11).
01	1	The data contained in the CPU latches is written to each plane.
10	2	CPU data is masked according to the VGMBPR and written to the selected plane.
11	3	Each plane is written with the value contained in the VGSRER, regardless of the value in the VGESRR. Rotated CPU data is ANDed with the VGMBPR to form a mask that serves the same function as the VGMBPR in write modes 0 and 2.

8.15.9 VGA Graphics Controller Miscellaneous Register

Mnemonic: VGMISR
Index: 6
Reset value: Undefined

7 … 4	3 … 2	1	0
RES	Memory Map <1:0>	COE	GM

Bits	Field	Access	Description
7:4	RES	MBZ	Reserved, must be zero. Read value is unpredictable.
3:2	Memory Map <1:0>	RW	Specifies memory mapping. 00 128KB (A0000:BFFFF) 01 64KB (A0000:AFFFF) 10 32KB (B0000:B7FFF) 11 32KB (B8000:BFFFF)
1	COE	RW	Chain odd/even 0 CPU A14 = video buffer A0 for map0 or map2. CPU A16 = video buffer A0 for map1 or map3. 1 CPU A0 = 0 selects map2 or map0. CPU A0 = 1 selects map3 or map1.
0	GM	RW	Graphics mode 0 Text mode 1 Graphics mode

The VGMISR controls the VGA display mode, monochrome graphics emulation, and memory mapping.

8.15.10 VGA Graphics Controller Color Don’t Care Register

Mnemonic: VGCDCR
Index: 7
Reset value: Undefined

7 4	3 0
RES	Color Don’t Care

Bits	Field	Access	Description
7:4	RES	MBZ	Reserved, must be zero. Read value is unpredictable.
3:0	Color Don’t Care	RW	Each bit in this field determines whether color comparison is enabled for the corresponding plane. 0 Disable color comparison. 1 Enable color comparison.

The VGCDCR specifies the planes to be compared in a color compare operation.

8.15.11 VGA Graphics Controller Bit Mask Register

Mnemonic: VGBMKR
Index: 8
Reset value: Undefined

7	0
Bit Mask	

Bits	Field	Access	Description
7:0	Bit Mask	RW	A mask for modifying displayed pixels in which set bits allow corresponding pixels to change. Bit <0> corresponds to the right-most pixel in the displayed group of 8 pixels.

The VGBMKR determines which pixels are modified in write modes 0, 2, and 3 (see Table 8–17 in Section 8.15.8).

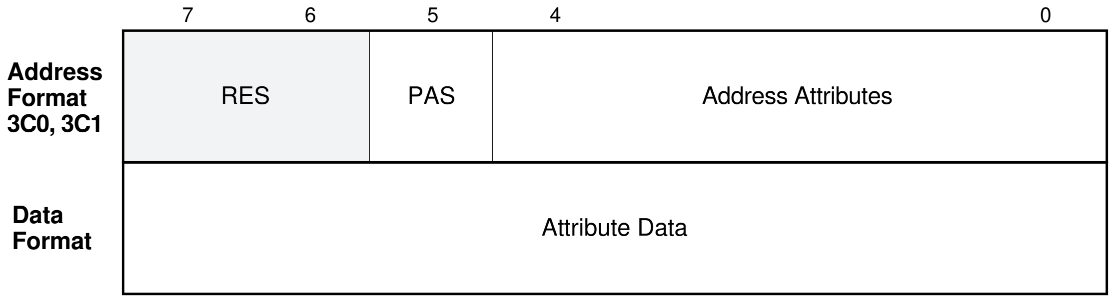
8.16 VGA Attribute Controller Registers

The VGA attribute controller registers are accessed through the VGA attribute controller index/data register (VAIXDR). The index and data values are written sequentially to address 3C0 and read from address 3C1. Writing and reading are implicitly controlled by the address.

8.16 VGA Attribute Controller Registers

8.16.1 VGA Attribute Controller Index/Data Register

Mnemonic: VAIXDR
Write address: 3C0
Read address: 3C1
Reset value: Undefined



Bits	Field	Access	Description
Address Format			
7:6	RES	MBZ	Reserved, must be zero. Read value is unpredictable.
5	PAS	RW	Palette address source — determines whether the color palette address is supplied by the host or a location in display memory. 0 The host supplies the palette address. 1 Display memory supplies the palette address (normal operation).
4:0	Address Attributes	RW	To write an attribute controller register, the register index is first written to this field at address 3C0, followed by the write data to the same address. To read an attribute controller register, the register index is first written to this field at address 3C0, followed by a read at address 3C1. The indexed registers are as follows: 00:0F Palette registers (VAPALRs) 10 Mode register (VAMODR) 11 Overscan register (VAOSCR) 12 Color plane enable register (VACPER) 13 Pixel panning register (VAPXPR) 14 Color select register (VACSLR) Unused index values are reserved.

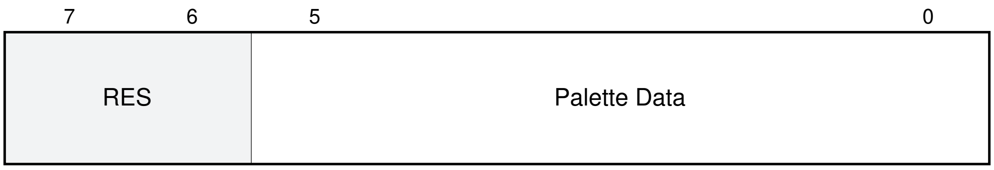
8.16 VGA Attribute Controller Registers

Bits	Field	Access	Description
Data Format			
7:0	Attribute Data	RO	Attribute controller register write data (3C0) or read data (3C1).

The VAIXDR contains the read or write index and data for the VGA attribute controller registers. It also determines the palette address source.

8.16.2 VGA Attribute Controller Palette Registers

Mnemonic: VAPALR
Indices: 0F:00
Reset value: Undefined



Bits	Field	Access	Description
7:6	RES	MBZ	Reserved, must be zero. Read value is unpredictable.
5:0	Palette Data	RW	Maps a pixel value to a color.

Each VAPALR maps the 6 LSBs of a pixel value to a palette address. The PSEL bit in the VGA attribute controller mode register (VAMODR <7>, Section 8.16.3) determines whether bits <5:4> are used. The MSBs are contained in the VGA attribute controller color select register (VACSLR, Section 8.16.7).

8.16.3 VGA Attribute Controller Mode Register

Mnemonic: VAMODR
Index: 10
Reset value: Undefined

7	6	5	4	3	2	1	0
PSEL	PW	PAN	RES	BIA	GCC	CME	GAM

Bits	Field	Access	Description
7	PSEL	RW	P<5:4> select—selects source for pixel data bits <5:4>. 0 Palette register <5:4> (Section 8.16.2). 1 VGA attribute controller color select register <1:0> (Section 8.16.7).
6	PW	RW	Pixel width 0 One pixel is 1 dot clock (all modes except 13_{16}). 1 One pixel is 2 dot clocks (mode 13_{16}).
5	PAN	RW	Pixel panning 0 Normal operation. 1 Successful line compare forces VGA attribute controller pixel panning register (VAPXPR, Section 8.16.6) output to zero.
4	RES	MBZ	Reserved, must be zero. Read value is unpredictable.
3	BIA	RW	Blink or intensity attribute 0 Select background intensity. 1 Enable blink.
2	GCC	RW	Graphics character codes 0 Ninth dot not enabled. 1 Ninth dot same as eighth dot.
1	CME	RW	Color or monochrome emulation 0 Color 1 Monochrome
0	GAM	RW	Graphics or alphanumeric mode 0 Alphanumeric 1 Graphics

The VAMODR contains miscellaneous attribute control bits.

8.16.4 VGA Attribute Controller Overscan Register

Mnemonic: VAOSCR
Index: 11
Reset value: Undefined

7 … 0
Overscan Color

Bits	Field	Access	Description
7:0	Overscan Color	RW	Specifies the overscan color.

The VAOSCR determines the color of the display border (see Figure 8–9).

8.16.5 VGA Attribute Controller Color Plane Enable Register

Mnemonic: VACPER
Index: 12
Reset value: Undefined

7 4	3 0
RES	Color Plane Enable

Bits	Field	Access	Description
7:4	RES	MBZ	Reserved, must be zero. Read value is unpredictable.
3:0	Color Plane Enable	RW	Set bits enable the corresponding plane.

The VACPER determines which of the four color planes (3 through 0) are enabled. The planes can be enabled in any combination.

8.16.6 VGA Attribute Controller Pixel Panning Register

Mnemonic:	VAPXPR
Index:	13
Reset value:	Undefined

7 4	3 0
RES	Pixel Panning

Bits	Field	Access	Description
7:4	RES	MBZ	Reserved, must be zero. Read value is unpredictable.
3:0	Pixel Panning	RW	Specifies the number of pixels by which video data is to be left shifted. The following table lists the number of pixels shifted, according to the register value and the current mode.

	Modes		
Code	**0+,1+,2+,3+,7,7+**	**13_{16}**	**All Others**
0000	1	0	0
0001	2	NA	1
0010	3	1	2
0011	4	NA	3
0100	5	2	4
0101	6	NA	5
0110	7	3	6
0111	8	NA	7
1000	0	NA	NA

Unused codes are reserved
NA = Not applicable

The VAPXPR specifies the number of pixels panned (smooth panning). The number of bytes panned (coarse panning) is specified in the VGA CRTC preset row register (VCPROR <6:5>, Section 8.13.9).

8.16.7 VGA Attribute Controller Color Select Register

Mnemonic: VACSLR
Index: 14
Reset value: Undefined

7 4	3 2	1 0
RES	SC<7:6>	SC<5:4>

Bits	Field	Access	Description
7:4	RES	MBZ	Reserved, must be zero. Read value is unpredictable.
3:2	SC<7:6>	RW	Select color <7:6>—palette data bits <7:6> in all color modes except 256-color mode (13_{16}).
1:0	SC<5:4>	RW	Select color <5:4>—palette data bits <5:4> if the PSEL bit is set in the VGA attribute controller mode register (VAMODR <7>, Section 8.16.3).

The VACSLR contains the conditional MSBs of the palette data. The LSBs are contained in the VGA attribute controller palette registers (VAPALR, Section 8.16.2).

8.17 VGA Color Registers

The VGA color registers are included only for compatibility. They map to equivalent palette and DAC registers as follows:

VGA Color Register	Palette and DAC Register
Pixel address write mode register (VPPAWR)	RAM write address register (DPWR)
Pixel address read mode register (VPPARR)	RAM read address register (DPRR)
DAC state register (VPDSTR)	Status register (DSTR)
Pixel data register (VPPDAR)	RAM color register (DPCR)
Pixel mask register (VPPMAR)	Pixel mask register (DPMR)

8.17 VGA Color Registers

8.17.1 VGA Color Pixel Address Write Mode and Read Mode Registers

Mnemonic:	VPPAWR, VPPARR
VPPAWR address:	3C8
VPPARR address:	3C7
VPPAWR, VPPARR reset value:	Undefined

7 … 0
Color Register Address

Bits	Field	Access	Description
VPPAWR			
7:0	Color Register Address	RW	The next palette address to be written.
VPPARR			
7:0	Color Register Address	RW	The next palette address to be read.

The VPPAWR and VPPARR are included only for compatibility. The VPPAWR maps to the palette and DAC RAM write address register (DPWR, Section 8.9.1) and the VPPARR maps to the palette and DAC RAM read address register (DPRR, Section 8.9.1).

8.17.2 VGA Color DAC State Register

Mnemonic: VPDSTR
Read address: 3C7
Reset value: Undefined

7 … 4	3	2	1 … 0
RES	SS	RWS	AS

Bits	Field	Access	Description
7:4	RES	MBZ	Reserved, must be zero. Read value is unpredictable.
3	SS	RO	Sense status 0 One or more DAC outputs exceeded the internal voltage reference level (335 mV). 1 No DAC output exceeded the internal voltage reference level.
2	RWS	RO	Read/write state—indicates whether the last palette or cursor color operation was a read or write, as follows: 0 Write—defined as writing the DPWR or the DCWR. 1 Read—defined as writing the DPRR or the DCRR.
1:0	AS	RO	Address state—indicates the color-component address for the next read or write cycle to the palette or cursor color register, as follows: 00 Red 01 Green 10 Blue 11 Reserved

The VPDSTR is implemented only for compatibility. It maps to the palette and DAC status register (DSTR, Section 8.9.6).

8.17.3 VGA Color Pixel Data Register

Mnemonic: VPPDAR
Address: 3C9
Reset value: Undefined

7	0
Color Register Data	

Bits	Field	Access	Description
7:0	Color Register Data	RW	Data read from or written to the addressed color register.

The VPPDAR is implemented only for compatibility. It maps to the palette and DAC RAM color register (DPCR, Section 8.9.2).

8.17.4 VGA Color Pixel Mask Register

Mnemonic: VPPMAR
Address: 3C6
Reset value: Undefined

7 … 0
Mask

Bits	Field	Access	Description
7:0	Mask	RW	Bitwise ANDed with 8-bit VGA pixel stream.

The VPPMAR is implemented only for compatibility. It maps to the palette and DAC pixel mask register (DPMR, Section 8.9.5).

9
PCI Operations

This chapter describes the PCI functions supported by the DECchip 21130. The PCI signals are described in Chapter 3. See the *PCI Local Bus Specification, Revision 2.0* for more information about the PCI bus transactions described in this chapter.

9.1 Configuration Operations

Prior to normal device operation, configuration firmware must write several configuration registers to define the following:

- Device address space
- Expansion ROM address space
- Bus access privilege
- Bus ownership duration

To allow system configuration software to access the configuration registers, the 21130 supports the following PCI configuration transactions:

- Configuration write
- Configuration read

When the PCI interface detects a PCI configuration write or read operation while the **pci_idsel** signal is asserted and **pci_ad<1:0>** = 00, it asserts **pci_devsel#** and uses address bits **pci_ad<7:2>** to index into the PCI configuration space header block (Table 8–1). Writes to reserved configuration addresses are ignored, and reads return zeros. The 21130 will not abnormally terminate a configuration cycle.

9.2 Memory Reads and Writes

The PCI interface decodes all memory read and write transactions. All accesses are to PCI memory space except VGA register-access transactions. The interface detects accesses to:

- PCI configuration space
- VGA I/O space (register accesses)
- PCI ROM space
- Frame buffer memory space (512KB)
- Base address 0 32MB memory space (2DA registers and frame buffer)
- Base address 1 alternate VGA register space
- Base address 1 palette and DAC space
- Base address 1 GPP space
- Base address 1 interrupt status register space
- Base address 1 sparse ROM space (256KB)

9.2.1 Memory Write to Core Space

The 32MB base address 0 memory space contains up to eight copies of a core space (Section 7.5.1). Each copy of core space is identical and maps the frame buffer and the same set of registers. On a memory write to core space, the PCI interface loads the write address and data into the command FIFO (Section 2.2). On a memory write burst, the interface loads the starting address and successive Dwords of data into the command FIFO.

If the command FIFO is full at the start of a memory write, the interface waits for up to 8 PCI clock cycles for an entry to become free. If an entry is still not free, the PCI interface terminates the transaction with a RETRY on the following PCI clock cycle.

If the command FIFO has free entries at the start of a memory write burst, the interface can load at least the address and one Dword of data into the command FIFO. If the command FIFO is then full, the interface waits for up to 8 PCI clock cycles for an entry to become free. If an entry is still not free, the PCI interface terminates the transaction with a DISCONNECT (with **pci_trdy#** not asserted) on the following PCI clock cycle.

9.2.2 Memory Read of Core Space

On a memory read of core space, the PCI interface fetches data from one of the following:

- A core register
- The frame buffer, through the frame buffer and device access (FBDA) function
- The alternate ROM space, through the FBDA function

The interface drives the read data on the PCI and asserts **pci_trdy#**. If **pci_irdy#** is asserted at that time, the interface tests for an attempted burst read (**pci_frame#** asserted). If **pci_irdy#** is not asserted when **pci_trdy#** is first asserted, the interface continues to assert **pci_trdy#** and waits until **pci_irdy#** is asserted before testing for an attempted burst read.

If a burst read is attempted, the interface terminates the transaction with a DISCONNECT (with **pci_trdy#** not asserted) on the following PCI clock cycle. If a burst read is not attempted, the transaction completes normally.

On a memory read of alternate ROM space, the PCI interface shifts the address left 2 bits, to map to the expansion ROM space, and then forwards the request to the FBDA.

9.2.2.1 Read Interlock

The PCI interface provides a read interlock for the 21130 core registers, frame buffer, and all external devices. A read of these objects cannot complete until the 21130 is idle; that is, the busy bit is clear in the command status register (MCSR, Section 8.3.1).

For core register reads, the PCI interface waits up to 8 PCI clock cycles for the chip to become idle. If the chip is still not idle within 8 PCI clock cycles, the PCI interface terminates the transaction with a RETRY on the following PCI clock cycle.

For frame buffer or external device reads, the PCI interface waits up to 8 PCI clock cycles for the chip to become idle. If the chip becomes idle in that time, an internal request for data from an external resource is made. Depending on possible priority collisions with refresh requests, the data can return as early as 16 cycles or as late as 50 cycles after the request is made. If the chip is still not idle within 8 PCI clock cycles, an internal request is not made and the PCI interface terminates the transaction with a RETRY on the following PCI clock cycle.

9.2 Memory Reads and Writes

Note that the following registers are exceptions:

- PCI configuration registers
- VGA registers
- Command status register (MCSR)
- Interrupt status register (MISR)
- Video current refresh address register (VFCRR)
- Video occlusion bitmap current address register (VFOAR)

Read data is returned from these registers whether the busy bit is set or clear.

9.3 Target Operations

As a target, the 21130 responds to the following PCI memory transactions:

- Memory read
- Memory write

It responds to any memory read or memory write cycle in which the address falls within the address space defined by either of the two PCI device base-address registers (PDBR0 or PDBR1, Section 8.2.5). Additionally, the 21130 responds to any memory cycle in which the address falls within the address space defined by the PCI expansion ROM base-address register (PRBR, Section 8.2.6). Writes to expansion ROM address space are treated as writes to a reserved location.

If the 21130 detects a write to a reserved location in the 21130 address space, it responds to and completes the bus cycle, but ignores the data. Similarly, the 21130 responds to and completes a read transaction of a reserved location, but returns zeros.

The 21130 also responds to the following types of memory transactions, treating them as one of the simpler supported types:

- Memory write and invalidate (operates as memory write)
- Memory read line (operates as memory read)
- Memory read multiple (operates as memory read)

9.3.1 Access Granularity

As a target, the 21130 supports arbitrary, subDword (less than 32 bits) read and write accesses. The 21130 handles all possible permutations of byte masks presented on the **pci_cbe<3:0>#** pins during both read and write accesses, with the following restrictions:

- Writes to 21130 registers are limited to Dword access. Byte masks are ignored. That is, all 32 bits are written unless **pci_cbe<3:0>#** = F, in which case no bits are written.
- Expansion ROM reads, through the alternate ROM space (Sections 7.3 and 7.5.2.5), return only Dword-aligned data.

9.3.2 Transaction Termination

As a target, the 21130 supports arbitrary burst-length, memory-write cycles to the base address 0 (PDBR0) PCI memory space. If the internal command FIFO fills during a burst write, the 21130 disconnects to avoid losing write data.

The 21130 does not support burst memory-read cycles or any burst transactions to PCI configuration space. The 21130 disconnects such transactions after one successful transfer.

The PCI interface loads all base address 0 writes into the internal 64-entry command FIFO. If the 21130 detects a memory-write cycle to its base address 0 address space and no command FIFO entries are available, it stalls for up to 8 PCI clocks, waiting for an entry to become available. If an entry is still not available, the 21130 issues a target-disconnect termination. The 21130 does not initiate a target-abort termination in response to a base address 0 or 1 (2DA) access.

Tables 9–1 through 9–4 summarize the 21130 response as a target to various PCI transactions and conditions.

9.3 Target Operations

Table 9–1 PCI Transactions to 2DA Memory Space

	Read PDBR0 Space	Read PDBR1 Space	Write PDBR0 Space	Write PDBR1 Space
Transaction	**Response**			
Burst	Automatic disconnect	Automatic disconnect	Allowed if command FIFO is not full	Automatic disconnect
Null data phase	Not applicable	Not applicable	Supported	NO-OP — immediately assert **pci_trdy#**
End of boundary	Automatic disconnect	Automatic disconnect	Disconnects at appropriate time	Automatic disconnect
Read side effects	None	Not marked as prefetchable	Not applicable	Not applicable
Stalls by master	Assert **pci_trdy#** and wait for master	Assert **pci_trdy#** and wait for master	Assert **pci_trdy#** and wait for master	Wait for **pci_irdy#** and assert **pci_trdy#**
Response	**Transaction or Condition**			
Retry (first data phase)	After 8 cycles either **pci_trdy#** or miscellaneous access read request is not asserted	Never issued	Command FIFO remains full after 8 cycles	Never issued
Disconnect, **pci_trdy#** not asserted	Burst attempted	Burst attempted	Command FIFO remains full after 8 cycles	Burst attempted
Target abort	The 21130 does not issue target aborts for these transactions.			

Table 9–2 PCI Transactions to Configuration Space and Expansion ROM Space

	Read Configuration Space	Write Configuration Space	Read Expansion ROM Space	Write Expansion ROM Space
Transaction	**Response**			
Burst	The 21130 automatically disconnects on burst transactions to these spaces.			
Null data phase	Not applicable	NO-OP — immediately assert **pci_trdy#**	Not applicable	NO-OP — immediately assert **pci_trdy#**
End of boundary	The 21130 automatically disconnects on end-of-boundary transactions to these spaces.			
Read side effects	None	Not applicable	None	Not applicable
Stalls by master	Assert **pci_trdy#** and wait for master	Wait for **pci_irdy#** and assert **pci_trdy#**	Assert **pci_trdy#** and wait for master	Assert **pci_trdy#** and wait for master
Response	**Transaction or Condition**			
Retry	The 21130 does not issue retry on transactions to these spaces.			
Disconnect, **pci_trdy#** not asserted	The 21130 will disconnect and not assert **pci_trdy#** on burst transactions to these spaces.			
Target abort	The 21130 does not issue target aborts for these transactions.			

9.3 Target Operations

Table 9–3 PCI Transactions to VGA Memory and I/O Space

	VGA Read*	VGA Write*	I/O Space Read DAC	I/O Space Write DAC
Transaction	**Response**			
Burst	The 21130 automatically disconnects on burst transactions to these spaces.			
Null data phase	Supported	Supported	Not applicable	Supported
End of boundary	The 21130 automatically disconnects on end-of-boundary transactions to these spaces.			
Read side effects	Not marked prefetchable	Not applicable	Not marked prefetchable	Not applicable
Stalls by master	Assert **pci_trdy#** and wait for master	Wait for **pci_irdy#** and assert **pci_trdy#**	Assert **pci_trdy#** and wait for master	Wait for **pci_irdy#** and assert **pci_trdy#**
Response	**Transaction or Condition**			
Retry	Issued	Issued	Never issued	Never issued
Disconnect, **pci_trdy#** not asserted	The 21130 will disconnect and not assert **pci_trdy#** on burst transactions to these spaces.			
Target abort	The 21130 issues target abort for illegal combinations of VGA I/O address and byte enables.		The 21130 issues target abort if **pci_cbe<3:0>#** do not match byte address or fail to map only to bytes owned by the 21130.	

*VGA memory and I/O space

Table 9–4 Snooped DAC Write PCI Transactions to VGA Space

Transaction	Response
Burst	Unsupported
Null data phase	Supported
End of boundary	Not applicable
Read side effects	Not applicable
Stalls by master	Wait until data is transferred, then snoop
Mix of null data phases or bursts	Unsupported

Response	Transaction or Condition
Retry	Not applicable
Disconnect, **pci_trdy#** not asserted	Not applicable
Target abort	Not applicable

9.4 Master Operation

The 21130 masters the PCI to move image data from system memory to display memory. To support this function, the PCI interface initiates PCI transactions.

In response to a host read request, the 21130 attempts to read in bursts of arbitrary length, according to the command it received. The 21130 responds in a DMA-read copy mode. The specified length can be between 1 and 2K PCI longword transfers (that is, burst read between 1 byte and 8KB). The 21130 attempts to string together the largest burst possible, but allows the PCI target to regulate the access through its target-disconnect mechanism.

The 21130 monitors the number of transfers remaining to complete the DMA request, making the number of separate burst transfers transparent to the driver. If the initial attempt to transfer the entire burst length is disconnected, the 21130 attempts to remaster the bus as many times as necessary to complete the request without driver assistance. For example, if a DMA read requests 100 bytes, the 21130 attempts one burst read of 100 bytes. However, depending on the speed of the target (for example, a bridge to system memory), the transfer might comprise 10 bursts averaging 10 bytes each, or 20 bursts averaging 5 bytes each, and so on.

9.4 Master Operation

9.4.1 DMA Read Transfer

The command parser can request a DMA read transfer over the PCI. While a DMA operation is in progress, the PCI interface retries all target accesses except those to the command status register (MCSR) or PCI configuration space.

If the command parser requests a DMA read transfer, the PCI interface requests the PCI bus. When the bus is granted, the PCI interface attempts to read from the specified address until the request is completed and as long as the DMA read FIFO is not full.

If the DMA read FIFO becomes full, the interface attempts to terminate the transaction as soon as possible by deasserting **pci_frame#** in the cycle following the completion of the current data phase. Depending on the status of the DMA read-data pipe stages at the time of the termination, the PCI interface can reread up to 3 address locations when it remasters the PCI in an attempt to complete the DMA read operation.

9.4.2 Transaction Termination

The 21130 supports the PCI-master latency timer mechanism that limits a master's tenure in the presence of other bus requests. The 21130 limits its bus ownership to the number of PCI clocks programmed in the PCI latency timer register (PLTR, Section 8.2.4). The timer is cleared and disabled when the 21130 is not asserting **pci_frame#**. While **pci_frame#** is asserted, the timer counts. If the count equals the value in the PLTR and **pci_gnt#** is deasserted (that is, another agent needs the bus), the 21130 attempts to terminate the transaction as soon as possible; that is, it deasserts **pci_frame#** and enters the final data phase as soon as the current data phase is completed (signaled when the target asserts **pci_trdy#**). The 21130 does not relinquish the bus until this final data phase is completed.

When initiating a memory transaction, the 21130 issues a master abort if it does not detect the assertion of **pci_devsel#** within 6 PCI clocks after it asserts **pci_frame#**. In such cases, the 21130 terminates the transaction, relinquishes PCI bus ownership, and sets the master-abort bit in the PCI command and status register (PCSR <29>, Section 8.2.2).

Cycles terminated by a target abort are handled similarly. If a target signals target abort, the 21130 immediately terminates the cycle, relinquishes bus ownership and sets the target-abort bit (PCSR <28>).

9.4.3 Aborted DMA Transaction Termination

The 21130 treats an aborted DMA-read copy transaction as a successfully completed transaction, but it sets the appropriate abort-bit status in the PCSR. The 21130 immediately completes all subsequent DMA transfers internally (no PCI activity) until the abort bit is cleared.

As a master, the 21130 supports all types of target-initiated terminations defined by the *PCI Local Bus Specification, Revision 2.0*.

9.5 Parity

The 21130 generates and drives parity on the **pci_par** pin. It also does parity-error checking and notification on the **pci_serr#** and **pci_perr#**, as described in the *PCI Local Bus Specification, Revision 2.0*.

As a master, the 21130:

- Generates parity across 36 bits (**pci_ad<31:0>** and **pci_cbe<3:0>#**) for all address cycles
- Checks parity received on **pci_par** during read-data cycles and reports errors if the PER bit is set in the PCI command and status register (PCSR, Section 8.2.2)

As a target, the 21130:

- Generates parity for all read-data cycles
- Checks parity received on **pci_par** during address and write-data cycles and reports errors if the PER bit is set in the PCI command and status register (PCSR, Section 8.2.2)

When a parity error is detected, the 21130 signals the error on either the **pci_serr#** pin if the error occurred during an address transaction, or the **pci_perr#** pin if the error occurred during a data transaction. The 21130 continues to operate normally; that is, if the address is a valid 21130 address, it is used, along with the subsequent data. If a data transaction had the error, the erroneous data will be used for a write.

9.6 Bus Parking

The 21130 supports PCI bus parking. The central PCI arbitration resource can select the 21130 to actively drive much of the PCI bus to a known state while the bus is idle, to prevent the bus from floating. When the arbiter asserts the **pci_gnt#** input, the 21130 drives pins **pci_ad<31:0>**, **pci_cbe<3:0>#**, and, at least 1 clock later, **pci_par**, to an arbitrary state. The 21130 can enable these drivers over several PCI clocks. When **pci_gnt#** is deasserted, the 21130 tristates **pci_ad<31:0>** and **pci_cbe<3:0>#** on the next clock, and tristates **pci_par** 1 clock later.

9.7 Functions Not Supported

The 21130 does not support and ignores special cycle and interrupt acknowledge PCI transactions. The 21130 does not do address or data stepping. As a target, the 21130 does not support the following:

- Exclusive accesses (LOCK cycles) for any of its registers or for display memory
- Burst memory-read cycles
- Burst transactions to PCI configuration space

As a master, the 21130 does not do write transactions or request exclusive access. Also see Tables 9–1 through 9–4.

10

Graphics Operations

This chapter describes the DECchip 21130 general graphics functions and specific graphics modes.

10.1 Overview

The accelerated graphics operations are specified by mode and initiated by a write to either of the following:

- The frame buffer address space (standard)
- Any graphics command register (alternative)

10.1.1 Frame Buffer Writes

Writing to the frame buffer address space is the standard way to invoke a graphics function. In general, the 21130 responds to and interprets write data according to the mode specified in the mode register (GMOR, Section 8.5.1). When the 21130 detects a write to its frame buffer address space, it starts the mode-specified graphics operation at the specified address using control parameters passed in the write data, and possibly, one or more graphics control registers.

In several graphics modes, writing to the frame buffer to initiate an operation does not take full advantage of the 21130's speed or range. For example, a write to the frame buffer in copy mode uses only half of the 64-byte onchip copy buffer for an 8-bpp span. For another example, the 21130 memory interface supports very fast line-drawing rates, but writing to the frame buffer in line mode burdens the CPU with processing Bresenham-style setup code.

Table 10–1 describes the graphics functions that can be invoked in each mode on a write to the frame buffer. (Table 8–4 lists all the modes.)

10.1 Overview

Table 10–1 Mode-Dependent Frame Buffer Write Operations

Mode	Action Initiated on Frame Buffer Write
Simple	Write pixels.
Opaque stipple	Draw patterned, bitonal 32-pixel spans.
Opaque bit-reversed stipple	Draw patterned, bitonal 32-pixel spans.
Transparent stipple	Draw patterned, monotone 32-pixel spans.
Transparent bit-reversed stipple	Draw patterned, monotone 32-pixel spans.
Transparent stipple with pixel mask	Draw patterned, masked, monotone 32-pixel spans.
Transparent bit-reversed stipple with pixel mask	Draw patterned, masked, monotone 32-pixel spans.
Opaque fill	Fill bitonal span up to 2K pixels.
Opaque extended-pattern fill	Fill patterned span up to 2K pixels.
Transparent fill	Fill solid span up to 2K pixels.
Transparent extended-pattern fill	Fill patterned span up to 2K pixels.
Opaque line	Draw patterned, bitonal 16-pixel lines.
Transparent line	Draw patterned, monotone 16-pixel lines.
Copy	Fill the copy buffer with a 32-pixel, masked, 8-bpp span or a 16-pixel, masked, 32-bpp span; or, empty the copy buffer to a 32-pixel, masked, 8-bpp span or a 16-pixel, masked, 32-bpp span.†
DMA-read copy	Transfer an unaligned, edge-masked span up to 8KB from PCI addressable memory to display memory.
Scaled-copy	Transfer and scale an unaligned, edge-masked span up to 4KB from PCI addressable memory to display memory.

†Whether the copy buffer is filled or emptied depends on the state of the copy hardware.

10.1.2 Graphics Command Register Writes

For better performance, the graphics command registers can be used to initiate graphics operations. They give software a faster and simpler way to invoke graphics operations.

Similar to writing directly to the frame buffer, writing to a graphics command register invokes a mode-dependent graphics operation, but the frame-buffer address is provided in a register rather than on the write. Writes to graphics command registers cannot invoke all mode operations, but can and should be used to generate the graphics functions listed in Table 10–2.

Table 10–2 describes the graphics operations that can be initiated by writing to a graphics command register.

Table 10–2 Graphics Command Register Write Operations

Register	Mode	Action Initiated on Register Write
Slope<7:0> (GSLR<7:0>) Span width (GSWR)	Line*	Initializes the Bresenham engine and then draws a mode-dependent 16-pixel 2D line (Table 10–1).
Slope-no-go<7:0> (GSNR<7:0>)†	Line*	Initializes the Bresenham engine.
Continue (GCTR)	Line*	Continues the current line another 16 pixels.
	Other	In any mode other than a line mode, initiates an operation based on the specified mode (Table 10–1), conditionally using the address from the GADR.
Copy 64 source (GCSR)	Copy	Fills up to 64 bytes of the copy buffer from the specified frame buffer address.
Copy 64 destination (GCDR)	Copy	Empties up to 64 bytes from the copy buffer to the specified frame buffer address.
Copy-64A source (GCASR)	Copy	Fills up to 64 bytes of the copy buffer from the frame buffer address specified in the GADR.
Copy-64A destination (GCADR)	Copy	Empties up to 64 bytes from the copy buffer to the frame buffer address specified in the GADR.

*Any line mode.

†The GSNRs are included because they initialize the Bresenham engine, but they do *not* initiate line drawing and are not graphics command registers.

10.1.3 Invoking Graphics Operations

To invoke a graphics function in any supported mode, the basic sequence is as follows:

1. Set the mode for the desired operation.
2. Write the required mode-specific parameters to the appropriate graphics control registers.
3. Initiate the operation with a write to the frame buffer or to a graphics command register.

10.1 Overview

This sequence of writes is grouped as one command packet. Each packet typically contains none to several control parameters, followed by the operation that initiates the write. Software streams command packets to the 21130 where they are stored in the 64-entry command FIFO. The 21130 unloads the packets from the FIFO one at a time, and executes them as specified by the mode.

The order of the control parameters is usually not important, but they all must be written before the final write that initiates the operation. All 21130 drivers must maintain this level of ordering. In particular, Alpha drivers present special problems because the CPU write buffer does not enforce write ordering. (See Section 11.12.1 for more information about 21130 support for the write buffer in Alpha CPUs.)

The 21130 uses a different set of control parameters for each operating mode. The parameters are provided by the graphics control registers and also by the data that the operation-initiating write passes to the frame buffer or to the graphics command registers. In each mode, the 21130 can operate on a variety of on-screen and off-screen visual bitmaps.

10.1.4 Register Load Synchronization

In general, software can write the frame buffer, any graphics control register, or any graphics command register, without regard to the internal state of the chip. The order of the writes within each command packet is important to the extent that all control registers must be set before the frame buffer or graphics command registers are written to initiate the graphics operation. However, in all but a few cases, software need not send register data or command packets in synchronism with the previous operation's completion.

The 21130 does not schedule a command packet for execution until the previous command has finished executing. Most of the graphics registers are double-buffered, such that, while one set is being loaded from a command packet, the other set can be used for graphics processing without interference. Therefore, software can usually issue register and packet writes indefinitely, without polling the state of the 21130 graphics processing hardware or registers. However, the chip must be idle (that is, processing complete with the command buffer empty) before writing to the deep register (GDER) in any mode.

When it is required, register-load synchronization can be done in either of the following ways:

- Software can poll the busy bit in the command status register (MCSR <0>, Section 8.3.1) and write the register only when the value of busy is zero.

- Software can insert a synchronization barrier into the command stream. A write to the MCSR effectively causes the 21130 to wait for the busy bit to go to logical zero. A write to the MCSR goes into the command buffer along with all other writes. But when the MCSR write is removed from the command buffer for processing, the operation stalls until all previous graphics processing is completed. For example, before writing the GDER, software can first write the MCSR and then write the GDER, rather than polling the busy bit and waiting for the chip to become idle.

10.1.5 Source and Destination Operands

The 21130 references a source and a destination operand for every graphics operation. Table 10–3 shows the specific source and destination operands according to mode.

Table 10–3 Source and Destination Operands According to Mode

Mode	Source	Destination
Simple	PCI write data	Frame buffer bitmap
Opaque stipple Opaque line Transparent stipple Transparent line	GFGR or GBGR	Frame buffer bitmap
Opaque fill Opaque extended-pattern fill	PCI write data	Frame buffer bitmap
Transparent fill Transparent extended-pattern fill	PCI write data	Frame buffer bitmap
Copy	Frame buffer bitmap	Frame buffer bitmap
DMA-read copy	PCI memory bitmap	Frame buffer bitmap
Scaled-copy	PCI memory bitmap	Frame buffer bitmap

In most cases, the source and destination operands are simply pixel values that are read from or written to a bitmap. For example, a copy mode operation reads a pixel value from a source bitmap and writes it to a destination bitmap.

10.2 Graphics Modes

Sections 10.2.1 through 10.2.10 describe the graphics modes. Each section describes the mode's invocation, required parameter sets, and functional operation. The descriptions include the standard invocation mechanism (directly writing the frame buffer), and for applicable modes, the alternate graphics command register mechanism.

Note

The functional-algorithm pseudo-code examples in the following sections are for descriptive purposes and do not describe the exact logic implementation.

10.2.1 Simple Mode

In the simple mode, a PCI write to the frame-buffer address space writes 4 independently masked bytes of data to the frame buffer at the Dword-aligned write address. The 21130 performs the write as a function of the parameters listed in Table 10–4.

```
Simple Write (Frame Buffer Address, Frame Buffer Data, Mask PCI, Byte Mask,
              Mask GPXR, Raster Op, Destination Bitmap, GIB Endian);
```

Table 10–4 Simple Mode Parameters

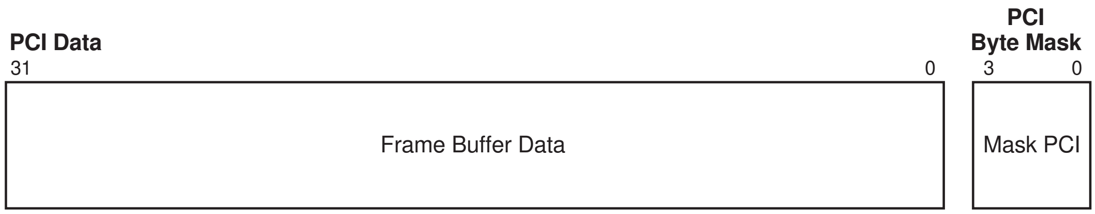
Parameter	Source		Section
`Frame Buffer Address`	PCI write address	—	—
`Frame Buffer Data`	PCI write data	<31:0>	—
`Mask PCI`	PCI data byte mask	<3:0>	—
`Mask GPXR`	Pixel mask register	GPXR <31:0>	8.5.10

`Byte Mask`
`Destination Bitmap`
`Raster Op`
	Raster operation register	GOPR <19:16>
		GOPR <10:8>
		GOPR <3:0>
			8.5.9

`GIB Endian`	Deep register	GDER <21>	8.5.2

Figure 10–1 shows the PCI write data format for the `Frame Buffer Data` and `Mask PCI`.

Figure 10–1 Simple Mode PCI Write-Data Format



In the simple mode, the 21130 acts as a generic 32-bit memory controller with the following exceptions:

- The GPXR specifies a byte mask (`Mask GPXR`) to be used in addition to the mask passed over PCI (`Mask PCI`).
- The `Raster Op` programmed in the GOPR is applied.
- The `Byte Mask` specified in the GOPR is applied.

10.2 Graphics Modes

For every write in the simple mode, the 21130 ANDs `Mask PCI` and `Mask GPXR` to generate the final byte mask that determines whether to write each byte. As specified by the `Raster Op`, the write conditionally combines the `Frame Buffer Data` and the data stored at the `Frame Buffer Address`. Only the bit-planes that are enabled by the `Byte Mask` are written.

The `Destination Bitmap` parameter allows access to all types of destination bitmaps. The `Frame Buffer Address` must be aligned to 4 bytes (Dword-aligned) for all destinations.

The following pseudo-code represents the basic algorithm for the simple mode:

```
Write Mask = Mask PCI & Mask GPXR;
Write Pixel (Frame Buffer Address, Frame Buffer Data, Raster Op, Byte Mask,
             Write Mask, Destination Bitmap);
```

The 21130 always uses the GPXR to specify which Dword bytes are to be written, but software does not always write the GPXR. Because hardware resets the GPXR to FFFFFFF (all bytes unmasked) after every operation when operating in 1-shot mode, software must write the GPXR only if a different value is required. If a persistent mask is desired, software writes to the persistent version of the GPXR (Section 8.5.10.3). Additionally, the 21130 always performs the `Raster Op` specified in the GOPR when writing the frame buffer data (the `Raster Op` retains its value from operation to operation).

The `GIB Endian` bit must be set to enable gib-endian byte swapping during simple writes and reads, DMA-read copy operations, and scaled copy operations with 16-bpp and 32-bpp RGB sources.

The simple mode can also be used to write arbitrary data to the frame buffer.

10.2.2 Opaque-Stipple Mode

In the opaque-stipple mode, a PCI write to the frame buffer address space draws a bitonal, masked span of 32 contiguous pixels starting at that address. The 21130 draws the span as a function of the parameters listed in Table 10–5.

```
Opaque Stipple Span (Frame Buffer Address, Stipple Mask, Byte Mask,
                     Pixel Mask, Raster Op, Foreground, Background,
                     Destination Bitmap);
```

Table 10–5 Opaque-Stipple Mode Parameters

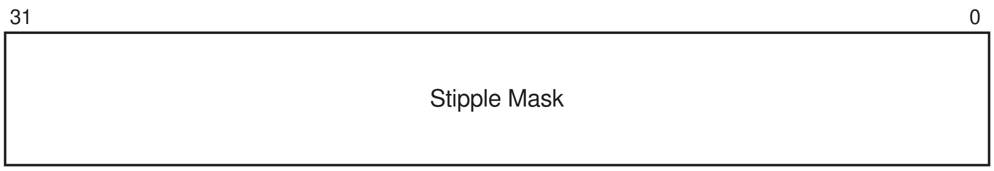
Parameter	Source		Section
`Frame Buffer Address`	PCI write address	—	—
`Stipple Mask`	PCI write data	<31:0>	—
`Pixel Mask`	Pixel mask register	GPXR <31:0>	8.5.10

`Byte Mask`
`Destination Bitmap`
`Raster Op`
	Raster operation register	GOPR <19:16>
		GOPR <10:8>
		GOPR <3:0>
			8.5.9

`Foreground`	Foreground register	GFGR <31:0>	8.5.8
`Background`	Background register	GBGR <31:0>	8.5.8

The PCI write cycle to the `Frame Buffer Address` initiates the drawing operation and specifies the `Stipple Mask` in the format shown in Figure 10–2.

Figure 10–2 Opaque-Stipple Mode PCI Write-Data Format



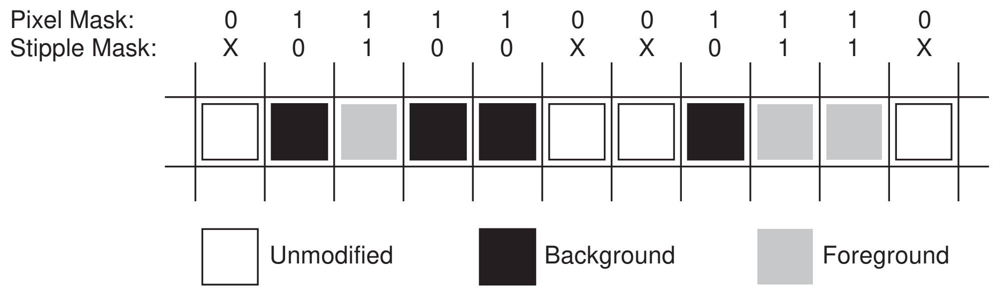
10.2 Graphics Modes

The 21130 expands the 32-bit `Stipple Mask` to 32 pixels, masking each pixel according to the `Pixel Mask` (Figure 10–3), as follows:

Pixel Mask Bit Value	Corresponding Pixel
0	Unmodified
1	Write enabled
Stipple Mask Bit Value	**Write to Corresponding Write-Enabled Pixel**
0	Background color
1	Foreground color

The opaque-stipple operation writes a bitonal pattern to a bitmap. Figure 10–3 is an example of drawing in opaque-fill mode.

Figure 10–3 Opaque-Stipple Mode Operation



The 21130 applies the specified `Raster Op` and `Byte Mask` on the write to the frame buffer.

The `Destination Bitmap` parameter allows access to all types of destination bitmaps. The `Frame Buffer Address` must be aligned to 4 bytes for 8-bpp destinations or 8 bytes for 16-bpp and 32-bpp destinations.

The following pseudocode represents the basic algorithm for the opaque-stipple mode:

```
for (n = 0; n <= 31; n++)
{
if (Pixel Mask<n> = 1)
  {
  Pixel = (Stipple Mask<n> ? Foreground : Background;
  Write Frame Buffer (Frame Buffer Address, Pixel, Raster Op, Byte Mask,
                      Destination Bitmap);
  }
Increment Pixel Address (Frame Buffer Address);
}
```

The 21130 optimizes the algorithm by writing 64 bits at a time; that is, up to 8 pixels to 8-bpp bitmaps, 4 pixels to 16-bpp bitmaps, or 2 pixels to 24-bpp bitmaps. The 21130 also increases performance by skipping over leading and trailing strings of zeros in the `Pixel Mask`. In stipple mode, unlike line mode, the 21130 does not update the internal pixel-processing address during a stipple operation. Therefore, a continue operation cannot be used to extend a stipple operation; instead, a new address must be specifies for each stipple operation.

The 21130 requires address alignments of 4 bytes for 8-bpp bitmaps or 8 bytes for 16-bpp and 32-bpp bitmaps and it does not implicitly mask span edges—software must align addresses and mask left and right span edges. Therefore, before delivering the `Stipple Mask` and `Pixel Mask` parameters to the 21130, software must:

- Align the `Stipple Mask`
- Align and logically combine the intended `Pixel Mask` with the desired left and right edge masks

The 21130 does the following operations in the opaque-stipple mode:

- Under X, does opaque stippling and tiling operations
- Under Windows, paints a region with an arbitrary bitonal brush
- In certain cases, draws text
- Quickly fills a solid region

(See Section 11.7 for more examples of opaque- and transparent-stipple mode applications.)

10.2.2.1 Opaque Bit-Reversed Stipple Mode

In the opaque bit-reversed stipple mode, the bits in `Stipple Mask` and `Pixel Mask` are reversed before being used, such that <31> corresponds to the first pixel drawn and <0> corresponds to the last pixel drawn.

10.2.3 Transparent-Stipple Mode

In the transparent-stipple mode, a PCI write to the frame buffer address space draws a solid, masked span of 32 contiguous pixels starting at that address. The 21130 draws the span as a function of the parameters listed in Table 10–6.

```
Transparent Stipple Span (Frame Buffer Address, Stipple Mask, Byte Mask,
                          Raster Op, Foreground, Destination Bitmap)
```

Table 10–6 Transparent-Stipple Mode Parameters

Parameter	Source		Section
`Frame Buffer Address`	PCI write address	—	—
`Stipple Mask`	PCI write data	—	—

`Byte Mask`
`Destination Bitmap`
`Raster Op`
	Raster operation register	GOPR <19:16>
		GOPR <10:8>
		GOPR <3:0>
			8.5.9

`Foreground`	Foreground register	GFGR <31:0>	8.5.8

The PCI write cycle to the `Frame Buffer Address` initiates the drawing operation and specifies the `Stipple Mask` in the format shown in Figure 10–4.

Figure 10–4 Transparent-Stipple Mode PCI Write-Data Format

31	0
Stipple Mask	

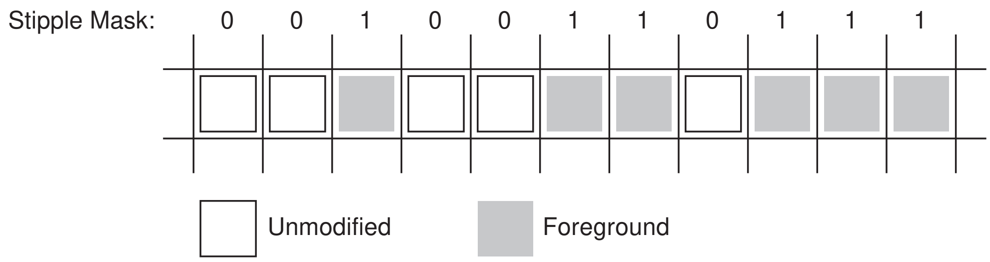
Transparent-stipple operations are, in effect, a simpler version of opaque-stipple operations, and operate in the same way with the following exceptions:

- The `Pixel Mask` is not specified.
- The `Stipple Mask` bits determine whether foreground color is written or write is disabled for the corresponding pixels, rather than determining whether foreground or background color is written.

These exceptions do not apply to the bit-reversed modes. See Section 10.2.3.1.

Figure 10–5 is an example of drawing in the transparent-stipple mode.

Figure 10–5 Transparent-Stipple Mode Operation



The transparent-stipple mode basic algorithm differs slightly from the opaque-stipple mode basic algorithm, and is represented by the following pseudo-code:

```
for (n = 0; n <= 31; n++)
{
if (Stipple Mask<n> = 1)
    Write Frame Buffer(Frame Buffer Address, Foreground, Raster Op,
                       Byte Mask, Destination Bitmap);
Increment Pixel Address(Frame Buffer Address);
}
```

The transparent-stipple mode does the following operations:

- Fills regions in X transparent-stipple mode
- Under Windows, paints a region with a monochrome brush
- In many cases, draws text
- In some cases, fills solid regions

(See Section 11.3 for more examples of opaque-stipple and transparent-stipple mode applications.)

10.2.3.1 Transparent-Stipple with Pixel Mask Modes

In the transparent-stipple with pixel mask modes, the pixel mask register (GPXR, Section 8.5.10) is used to mask pixels in the same way as in the opaque-stipple modes. In the transparent-stipple with pixel mask modes, pixels are drawn with the `Foreground` color if the corresponding bits in the `Stipple Mask` and `Pixel Mask` are both set.

In the transparent bit-reversed stipple with pixel mask mode, the bits in `Stipple Mask` and `Pixel Mask` are reversed before being used, such that <31> corresponds to the first pixel drawn and <0> corresponds to the last pixel drawn.

10.2.4 Opaque-Fill Mode

In the opaque-fill mode, a PCI write to the frame buffer address space writes a bitonal, unmasked span of up to 2K contiguous pixels starting at that address. The 21130 draws the span as a function of the parameters listed in Table 10–7.

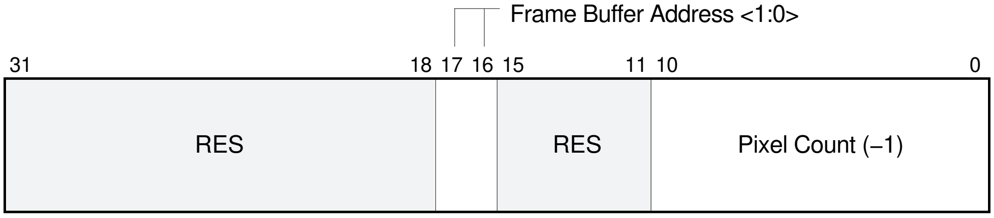
```
Opaque Fill Span (Frame Buffer Address, Pixel Count,
                  Frame Buffer Address <1:0>, Foreground, Background,
                  Raster Op, Byte Mask, Fill Mask, Destination Bitmap);
```

Table 10–7 Opaque-Fill Mode Parameters

Parameter	Source		Section
Frame Buffer Address	PCI write address	—	—
Frame Buffer Address <1:0> Pixel Count (-1)	PCI write data	<17:16> <10:0>	—
Foreground	Foreground register	GFGR <31:0>	8.5.8
Background	Background register	GBGR <31:0>	8.5.8
Byte Mask Destination Bitmap Raster Op	Raster operation register	GOPR <19:16> GOPR <10:8> GOPR <3:0>	8.5.9
Fill Mask	Data register	GDAR <31:0>	8.5.7

The PCI write cycle to the Frame Buffer Address initiates the drawing operation and specifies the Pixel Count and Frame Buffer Address <1:0> in the format shown in Figure 10–6.

Figure 10–6 Opaque-Fill Mode PCI Write-Data Format

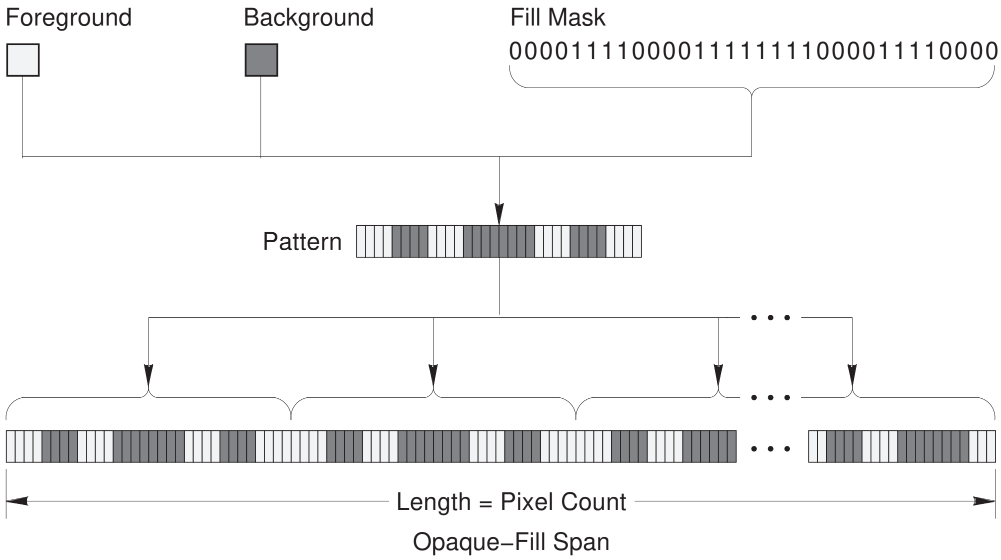


The opaque-fill mode fills a span of `Pixel Count` (up to 2K) pixels with a repeating, bitonal, 32-pixel pattern, as follows:

1. The pattern is defined by the `Foreground`, `Background`, and `Fill Mask` parameters.
2. The 21130 writes the `Foreground` color to each pixel that corresponds to a `Fill Mask` bit = 1, and writes the `Background` color to each pixel that corresponds to a `Fill Mask` bit = 0.
3. The 32-pixel pattern is repeated as many times as necessary to fill up to `Pixel Count` pixels.

Figure 10–7 is an example of drawing in the opaque-stipple mode.

Figure 10–7 Opaque-Fill Mode Operation



The `Frame Buffer Address` must be aligned to 1 pixel (1 byte in 8-bpp frame buffers) and `Fill Mask` must be aligned to 4 pixels. The `Frame Buffer Address` `<1:0>` parameter (two LSBs) provides byte granularity on 8-bpp frame buffers.

10.2.4.1 Opaque Extended-Pattern Fill Mode

The opaque extended-pattern fill mode uses the 64-byte copy buffer to perform $n \times n$ multicolor pattern fills. This mode is similar to the opaque-fill mode (Section 10.2.4) except that the data register is ignored and the copy buffer, rather than the foreground and background registers, provides the pattern data.

Before the extended-pattern fill is performed, the copy buffer is loaded with up to 64 bytes of pattern data. If the pattern is cached in off-screen memory, the copy buffer can be loaded by writing to the copy-64 source register (GCSR, Section 8.4.4); or, if the pattern is to be down-loaded from the host, the copy buffer can be loaded by writing to the copy buffer registers (GCBR<7:0>, Section 8.5.4).

The programmer cannot control which byte in a copy buffer quadword is chosen to go to a given byte in a destination quadword because copy buffer data is not shifted on output. However, the programmer can use the dither row and dither column registers (GDRR and GDCR) to control which copy buffer quadword is output to a given destination address. The following example shows how the hardware selects one of the eight copy buffer quadwords for a given destination address:

```
CopyQWORD =
   (((ByteAddress >> 3) & DitherColumn) | (DitherColumn & ~DitherColumn)) & 7
```

Typically, this mode is used to fill an area with an 8×8 color brush. This is easily done in 8-bpp mode by setting `DitherColumn` to 0 and incrementing `DitherRow` for each line.

In 16-bpp and 32-bpp modes the entire brush does not fit in the copy buffer. Therefore, two loops are required for 16-bpp and four loops are required for 32-bpp.

In a typical 16-bpp mode operation, every other brush line is loaded into the copy buffer and the `DitherColumn` is set to 1. The loop outputs 1 line, then increments the address register by 2 lines and the `DitherRow` by 2.

Similarly, in a typical 32-bpp mode operation, every fourth line is loaded into the copy buffer, the `DitherColumn` is set to 3, and the `DitherRow` is incremented by 4.

10.2.5 Transparent-Fill Mode

In the transparent-fill mode, a PCI write to the frame buffer address space writes a solid, masked span of up to 2K contiguous pixels starting at that address. The 21130 draws the span as a function of the parameters listed in Table 10–8.

```
Transparent Fill Span (Frame Buffer Address, Pixel Count,
                       Frame Buffer Address <1:0>, Foreground,
                       Raster Op, Byte Mask, Fill Mask,
                       Destination Bitmap);
```

Table 10–8 Transparent-Fill Mode Parameters

Parameter	Source		Section
`Frame Buffer Address`	PCI write address	—	—
`Frame Buffer Address <1:0>` `Pixel Count (-1)`	PCI write data	<17:16> <10:0>	—
`Foreground`	Foreground register	GFGR <31:0>	8.5.8

`Byte Mask`
`Destination Bitmap`
`Raster Op`
	Raster operation register	GOPR <19:16>
		GOPR <10:8>
		GOPR <3:0>
			8.5.9

`Fill Mask`	Data register	GDAR <31:0>	8.5.7

The transparent-fill mode fills a span of `Pixel Count` (up to 2K) pixels with a repeating, bitonal, 32-pixel pattern defined by the `Foreground` and `Fill Mask` parameters.

The transparent-fill mode is almost identical to the opaque-fill mode (Section 10.2.4). The `Foreground` color is written to each pixel that corresponds to a `Fill Mask` bit = 1; but in the transparent-fill mode, pixels that correspond to a `Fill Mask` bit = 0 are unmodified (rather than being written with the `Background` color).

As in the opaque-fill mode, the 32-pixel pattern is repeated as many times as necessary to fill up to `Pixel Count` pixels.

10.2.5.1 Transparent Extended-Pattern Fill Mode

The transparent extended-pattern fill mode uses the 64-byte copy buffer to perform $n \times n$ multicolor pattern fills. This mode is similar to the transparent-fill mode (Section 10.2.5) except that the copy buffer, rather than the foreground register, provides the pattern data.

Before the extended-pattern fill is performed, the copy buffer is loaded with up to 64 bytes of pattern data. If the pattern is cached in off-screen memory, the copy buffer can be loaded by writing to the copy-64 source register (GCSR, Section 8.4.4); or, if the pattern is to be down-loaded from the host, the copy buffer can be loaded by writing to the copy buffer registers (GCBR<7:0>, Section 8.5.4). When the fill operation is initiated, the pattern data in the copy buffer is used to color pixels as in the following pseudo-code:

```
if (8bpp)
{ iteration = (pixelAddress - startPixelAddress & 0x3f) / 8
  QWIndex   = ditherRow & ~ditherColumn | iteration & ditherColumn
  pixel     = copyBufferAsBytes[(QWIndex & 0x07 << 3) | pixelAddress & 0x07]
}
if (16bpp)
{ iteration = (pixelAddress - startPixelAddress & 0x3f) / 4
  QWIndex   = ditherRow & ~ditherColumn | iteration & ditherColumn
  pixel     = copyBufferAsWords[(QWIndex & 0x07 << 2) | pixelAddress & 0x03]
}
if (32bpp)
{ iteration = (pixelAddress - startPixelAddress & 0x3f) / 2
  QWIndex   = ditherRow & ~ditherColumn | iteration & ditherColumn
  pixel     = copyBufferAsLWs[(QWIndex & 0x07 << 1) | pixelAddress & 0x01]
}
```

10.2.6 Copy Mode

In the copy mode, a set of two consecutive PCI writes to the frame buffer address space copies a contiguous span of up to 64 bytes from the first address to the second address. The span can be copied in either direction: left-to-right (increasing addresses) or right-to-left (decreasing addresses). The 21130 performs the copy as a function of the parameters listed in Table 10–9.

```
Copy Span (Frame Buffer Address Source, Frame Buffer Address Destination,
          Mask Source, Mask Destination, Byte Mask, Raster Op,
          Source Bitmap, Destination Bitmap);
```

Table 10–9 Copy Mode Parameters

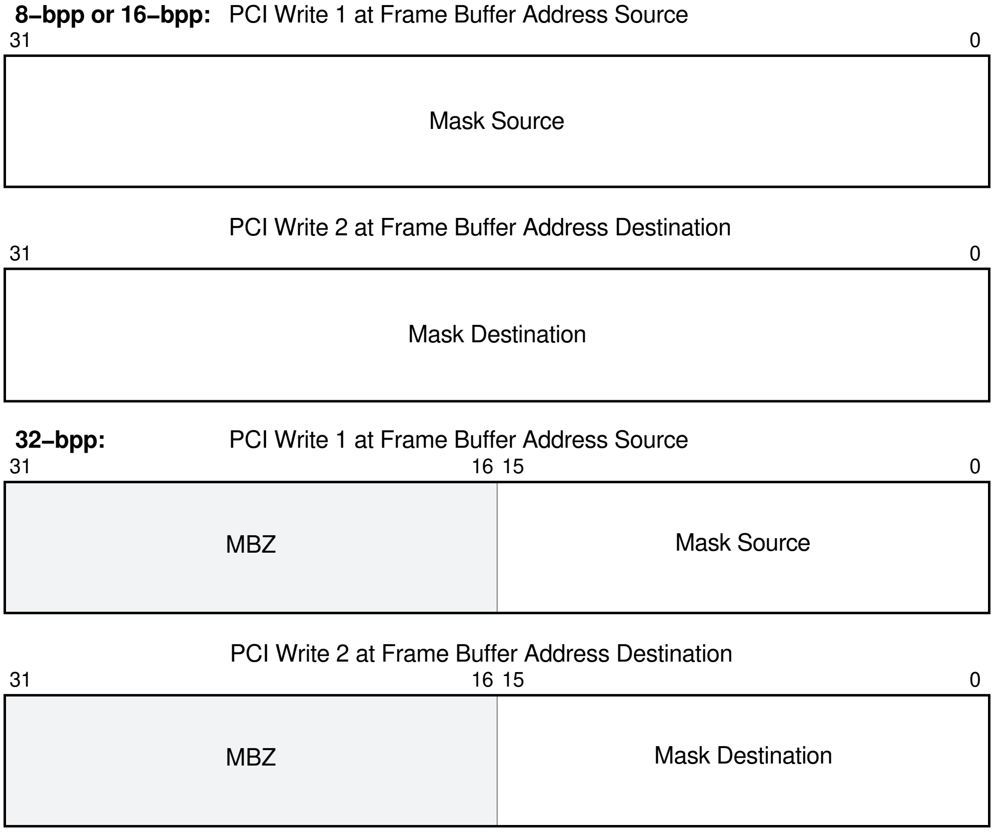
Parameter	Source		Section
`Frame Buffer Address Source`	PCI write address 1	—	—
`Mask Source`	PCI write data 1	—	—
`Frame Buffer Address Destination`	PCI write address 2	—	—
`Mask Destination`	PCI write data 2	—	—
`Pixel Shift`	Pixel shift register	GPSR <3:0>	8.5.5

`Byte Mask`
`Destination Bitmap`
`Raster Op`
	Raster operation register	GOPR <19:16>
		GOPR <10:8>
		GOPR <3:0>
			8.5.9

`Source Bitmap`	Mode register	GMOR <10:8>	8.5.1

Two PCI write cycles are necessary to copy one span locally in the frame buffer. The first PCI write addresses the location of the source span (`Frame Buffer Address Source`) and passes a read mask for the source (`Mask Source`) as data. The second PCI write addresses the location of the destination span (`Frame Buffer Address Destination`) and passes a write mask for the destination (`Mask Destination`) as data. Figure 10–8 shows the format of the PCI write operations.

Figure 10–8 Copy Mode PCI Write Data Formats



The maximum span limit of 64 bytes is set by the depth of the internal copy buffer and is independent of the pixel depth (Table 10–10). Consequently, the maximum span size is 16 pixels for 32-bit pixels. For 16-bit pixels, the size of the copy buffer allows only 32-pixel spans and each PCI write can supply only 32 bits to specify the mask. For 8-bit pixels, the size of the copy buffer allows 64-pixel spans, but each PCI write can supply only 32 bits to specify the mask. Consequently, the maximum span is 32 pixels when copying a 16-bpp span or 8-bpp masked span. (A different 21130 mechanism allows 64-pixel 8-bpp unmasked spans to be copied, Section 10.2.6.5.)

Table 10–10 shows the copy mode span limits according to pixel depth.

Table 10–10 Copy Mode Span Limits

Pixel Depth	Span Limit (Masked)	Span Limit (Unmasked)
8-bpp	32 pixels	64 pixels
16-bpp	32 pixels	32 pixels
32-bpp	16 pixels	16 pixels

Basically, the 21130 performs a masked, span copy operation in two stages, as follows:

1. On the first PCI write, the 21130 reads up to 64 bytes from the `Frame Buffer Address Source`, selectively aligning and depositing those bytes into the copy buffer, starting at the bottom and filling upward. Only pixels that correspond to a `Mask Source` bit = 1 are read.
2. On the second PCI write, the 21130 unloads up to 64 bytes from the copy buffer, starting at the bottom and draining upward. Each pixel is conditionally stored as a function of the `Mask Destination`, starting at the `Frame Buffer Address Destination`. Only pixels that correspond to a `Mask Source` bit = 1 are written.

   On the final write to the destination, the `Byte Mask` and the Boolean operation specified by the `Raster Op` parameter are applied.

Copy mode can handle any span including the following:

- Copies with aligned or unaligned source and destination
- Copies that require backward (right-to-left) processing in addition to forward (left-to-right) processing

Arbitrarily aligned sources and destinations require the 21130 to shift source data as it is processed. Backward processing is necessary in certain alignments of overlapping copies, and requires the 21130 to increment and decrement its addresses as it steps through the span.

10.2.6.1 Source and Destination Alignment

To copy a 32-pixel span, the 21130 reads up to 4 successive quadwords from the `Frame Buffer Address Source` masked by `Mask Source`; and writes up to 4 successive quadwords to the `Frame Buffer Address Destination` masked by `Mask Destination`. Consequently, copies are simple when both the `Frame Buffer Address Source` and the `Frame Buffer Address Destination` lie on natural quadword boundaries. However, graphics software (graphics applications, window managers, and so on) is not limited to specifying only quadword-aligned source and destination addresses. Therefore, the 21130 display driver must handle arbitrarily aligned source and destination addresses. The 21130 driver and hardware share the responsibility for ensuring that all possible combinations of desired source and destination are correctly handled.

Software must first adjust (that is, decrement) the desired source-pixel and destination addresses to quadword boundaries, such that the adjusted addresses can be passed as the `Frame Buffer Address Source` and `Frame Buffer Address Destination`. In addition, `Mask Source` and `Mask Destination` must be bit-shifted by the number of bytes that the addresses were decremented. That number is defined as `Source Align` and `Destination Align` for the source and destination. They are calculated as follows:

```
Source Align      = (desired source address) & Align Mask;
Destination Align = (desired destination address) & Align Mask;
```

In the previous equations,

`Align Mask` = 00000007_{16}

In general, the specified source and the destination alignment is random, with `Source Align` and `Destination Align` taking values from 0 to 7. In an aligned copy, `Source Align` and `Destination Align` take the same value; however, the 21130 display driver must usually process an unaligned copy in which `Source Align` and `Destination Align` take different values.

To process unaligned spans, the 21130 includes a hardware byte-shifter that aligns quadword source read data to the destination prior to filling the copy buffer. The `Pixel Shift` parameter is a signed 4-bit value ($-8 \leq$ `Pixel Shift` ≤ 7) that specifies the number of bytes to shift. Embedded in the byte-shifter is a 64-bit residue register that stores the previous quadword read from the copy source. (The residue register cannot be directly read or written.) The byte-shift function, in conjunction with the residue register, allows the 21130 to process all possible combinations of the `Source Align` and `Destination Align` values.

In an unaligned copy, at least 1 pixel from each of 2 successive quadwords read from the source must be merged into 1 quadword in the destination. Consequently, in each quadword the 21130 writes to the destination, it must extract some subset of pixels from source quadword *n* and merge them with a complementary subset of pixels from source quadword *n–1*. This amounts to a 1-stage source-read pipeline, in which the residue register always stores the last quadword read.

Starting at the `Frame Buffer Address Source`, the 21130 does the following:

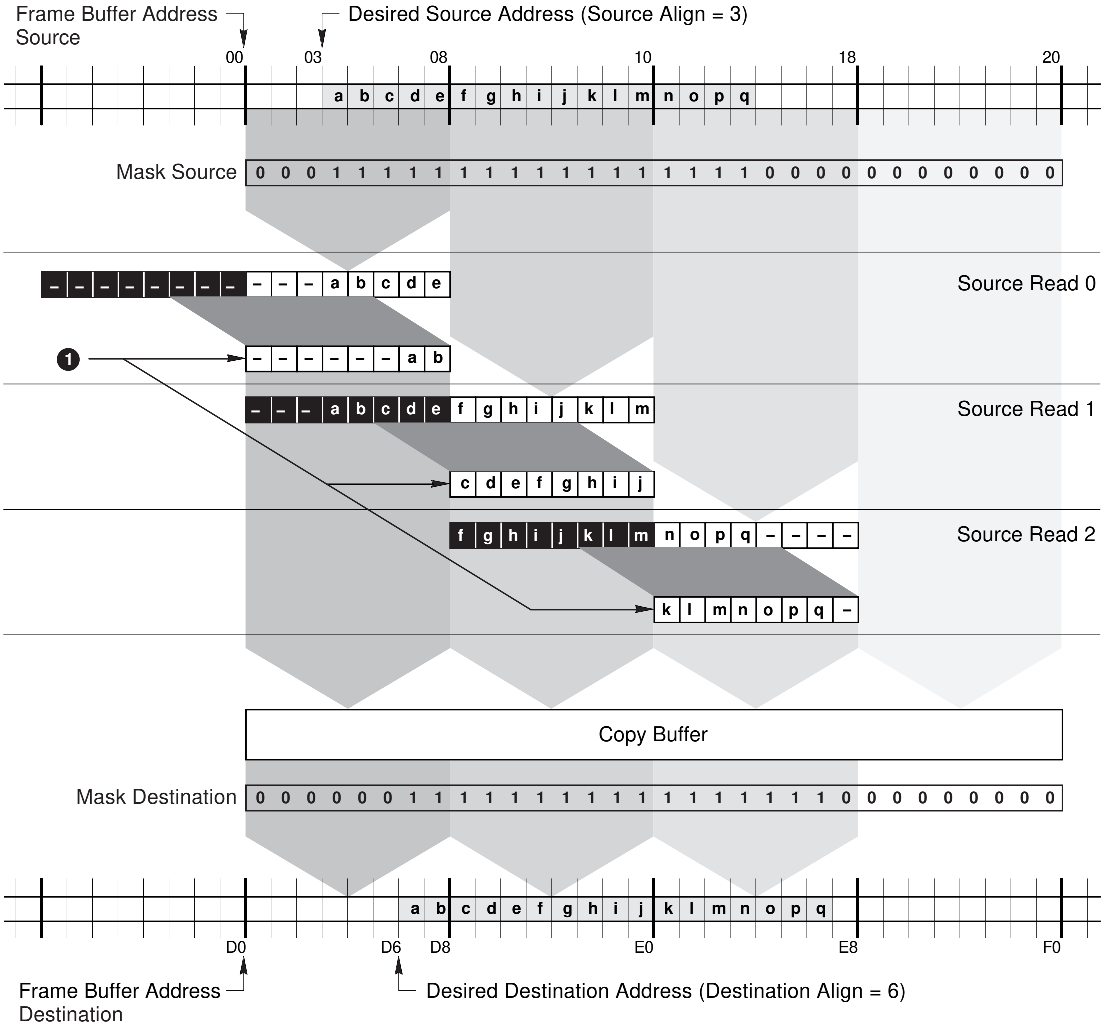
1. Reads 1 to 4 quadwords, depending on the value in `Mask Source`.
2. Concatenates the 64 bits read from each quadword with the residue register.
3. Rotates the resulting 128-bit quantity through the byte shifter by the amount specified by the `Pixel Shift` value (a negative value rotates left, and a positive value rotates right).
4. Extracts a quadword (now properly aligned with the destination) from the bit positions corresponding to the position of the read data before rotation.
5. Loads the extracted quadword into the copy buffer in the next available quadword entry.
6. Stores the last quadword read into the residue register.
7. Moves on to the next source quadword and repeats the process until the span is complete.

Figure 10–9 is an example of an unaligned forward (left-to-right) copy with an 8-bpp packed source and destination. The individual pixels are labeled **a** through **q**. Three quadwords are read through `Mask Source` and three are written through `Mask Destination`. For each read, the figure shows a ‘‘snapshot’’ of the contents of the residue register and the resultant byte-shifter output quadword. An example of a copy with a 24-bpp source and destination would be almost the same, with the following exceptions:

- Letters **a** through **q** would correspond to bytes within a pixel.
- The `Source Align` and `Destination Align` values would be 0 or 4.

10.2 Graphics Modes

Figure 10–9 Forward Span Copy



1 Pixel Shift = (Destination Align – Source Align) = +3
Residue Register

10.2.6.2 Backward Copies

In addition to arbitrary alignments, the 21130 must process forward (left-to-right) and backward (right-to-left) copies. Spans that overlap require the graphics server to pick a direction, to avoid corrupting a portion of the source before it is read. Consequently, the 21130 selectively increments (forward) or decrements (backward) source and destination quadword addresses in order to step through the span. The sign of the `Pixel Shift` value determines the direction of the span copy, as follows:

$-8 \leq$ `Pixel Shift` ≤ -1 for backward copies
$0 \leq$ `Pixel Shift` ≤ 7 for forward copies

For a negative `Pixel Shift` value, the 21130 does the following:

- Begins reading at the `Frame Buffer Address Source` and writing at the `Frame Buffer Address Destination`.
- Decrements the `Frame Buffer Address Source` after each quadword is read.
- Decrements the `Frame Buffer Address Destination` after each quadword is written.

For a positive `Pixel Shift` value, the 21130 also begins at `Frame Buffer Address Source` and `Frame Buffer Address Destination`, but it increments the respective addresses as it steps through the span.

The sign of the `Pixel Shift` value also determines the direction that the byte shifter rotates incoming source data (with residue): negative for rotate left and positive for rotate right. Therefore, the assignment of the `Pixel Shift` value must take into account that all incoming source data is rotated to the right in a forward copy and to the left in a backward copy.

Table 10–11 shows how the `Pixel Shift` value is calculated as a function of alignment and copy direction.

Table 10–11 Assigning the Pixel Shift Value

Direction	Destination Align $\geq$ Source Align
Forward	Destination Align $-$ Source Align
Backward	(Destination Align $-$ Source Align) $-$ 8
Direction	**Source Align > Destination Align**
Forward	8 $-$ (Source Align $-$ Destination Align)
Backward	Destination Align $-$ Source Align

10.2.6.3 Priming and Flushing the Residue Register

Certain combinations of alignment and copy direction require one additional adjustment to be made prior to starting the copy mode operation. Two types of copies fall into this category:

- Forward copies when `Source Align` > `Destination Align`
- Backward copies when `Destination Align` > `Source Align`

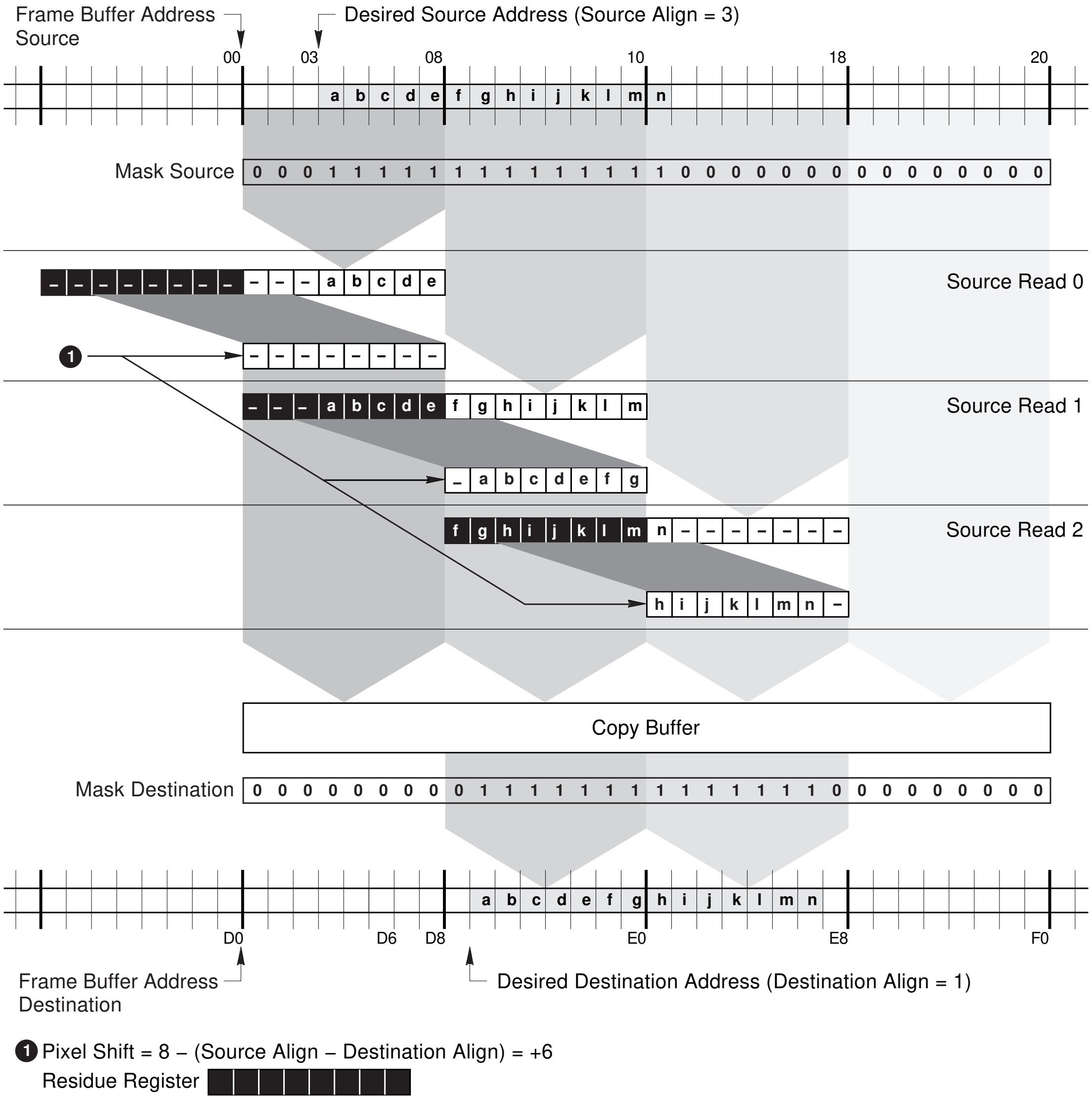
In either case, the first quadword written to the destination takes some bytes from both the first and second quadwords read from the source. The `Pixel Shift` value is set such that none of the valid pixels from the first read are rotated into the first quadword generated by the byte shifter. The byte shifter does not generate the proper quadword for the first destination quadword until the second source quadword is read. In effect, the first read primes the residue register, and every subsequent source read generates a valid destination quadword to store in the copy buffer. This amounts to a 1-stage, read data path pipeline.

To compensate for priming the residue register, software must adjust the `Frame Buffer Address Destination` and `Mask Destination` by an additional quadword. The `Frame Buffer Address Destination` must be decremented (forward copies) or incremented (backward copies) by 8 and the `Mask Destination` must be bit-shifted 8 bits to the right (forward) or left (backward) (Figure 10–10).

In some cases, including those in which priming is not required, the pipeline delay introduced by the residue register has a side effect. In such cases, the residue register must be flushed after the last unmasked quadword has been read, because it may contain leftover valid destination-pixels. Consider a span copy similar to that shown in Figure 10–9, but with the source span extended 2 pixels to the right. In that case, the 21130 hardware flushes the residue register when necessary, to generate the last destination-quadword written into the copy buffer — software need not do anything special. However, residue-register priming and flushing must also be considered in the DMA-read mode. In that mode, flushing requires software intervention (see Section 10.2.7 for more information).

Figure 10–10 is an example of a forward copy in which priming is necessary.

Figure 10–10 Primed Forward Span Copy



The following pseudo-code represents the basic algorithm for copying a span in copy mode, including source and destination alignment:

On PCI write 1:

```
for (i = 0; i <= 3; i++)
{
  Read Quadword (Frame Buffer Address Source, Mask Source<i*8+7:i*8>,
                 Data Source, Source Bitmap);
  Byte Rotate (Data Source, Residue, Pixel Shift, Data Shift);
  Load Copy Buffer (Data Shift);
  Residue = Data Source;
  Frame Buffer Address Source += 8*Sign(Pixel Shift);
}
```

On PCI write 2:

```
for (i = 0; i <= 3; i++)
{
  Unload Copy Buffer (Data Out);
  Store Quadword (Frame Buffer Address Destination,
                  Mask Destination<i*8+7:i*8), Data Out, Destination Bitmap);
  Frame Buffer Address Destination += 8*Sign(Pixel Shift);
}
```

The algorithm is for descriptive purposes and does not address all details of the copy-mode operation. For example, the 21130 does not necessarily read or write all 4 quadwords. The 21130 monitors leading and trailing zeros in `Mask Source` and `Mask Destination` to save time when copying. The 21130 jumps to the first unmasked pixel to start reading, and terminates the read and copy-buffer fill after the last unmasked pixel.

10.2.6.4 Copy Direction Flag

In the copy-mode process, software does not pass an explicit parameter to indicate whether the address and mask parameters passed on a PCI write to the frame buffer correspond to the source or the destination. In the copy mode, the 21130 requires a strict ordering of alternating source reads and destination-writes to the frame buffer, and uses the copy direction flag to indicate the next operation. The copy direction flag is a 2-state, internal, hardware pointer (GMOR <20>). The flag state, `Source Next` or `Destination Next`, determines whether the next incoming PCI write to the `Frame Buffer Address` space should trigger a read to, or a write from, the copy buffer.

Software neither reads the copy direction flag nor writes it directly. The flag is initialized to `Source Next` on a write to the GPSR or copy buffer. Therefore, when software sets the `Pixel Shift` parameter before starting the copy, the hardware is ready to read the first span. Each time software writes the frame buffer in the copy mode, the copy direction flag changes state. As long as software properly initializes the GPSR and alternates source-reads and destination-writes, the hardware always does the appropriate operation without explicit software control. If necessary, software can rewrite the GPSR to reset the copy direction flag.

10.2.6.5 64-Byte Unmasked Span Copies

The 32-bit masks passed in the copy mode limit the span to 32 bytes in the packed 8-bpp format. This uses only half of the 64-byte copy buffer. To overcome this limitation, the 21130 has a separate mechanism for copying spans of 64, unmasked, contiguous bytes (masked copies are not supported). In other words, this mechanism cannot be used to copy any span segment in which either the source or destination includes an edge that is not naturally aligned to an 8-byte boundary, because any such span must be masked.

Copying 64-byte spans involves a 2-stage operation: one PCI write to load the copy buffer starting at a specified `Frame Buffer Address Source` aligned to 8 bytes; and a second PCI write to unload the copy buffer starting at the `Frame Buffer Address Destination`. However in this case, rather than writing to the frame buffer, software writes the copy-64 source register (GCSR) to load the copy buffer, using the `Frame Buffer Address Source` as data. Similarly, to write the copy buffer contents to the frame buffer, software writes the copy-64 destination register (GCDR), using the `Frame Buffer Address Destination` as data.

Loading and unloading the copy buffer in this way always moves 64 bytes. Although the GCSR and GCDR are not normally used for span segments containing an edge, they can be used to fill interior span segments in a large copy operation, where the edge segments are copied using alternating writes to the frame buffer source and destination addresses.

10.2.6.6 Copy Buffer Operation

The 21130 copy buffer contains 8 quadword entries (Figure 10–11). The 21130 loads and unloads the copy buffer in copy-mode operations as follows:

- A write to the frame buffer in the copy mode with the copy direction flag pointing to `Source Next`

  For this operation, the 21130 loads the copy buffer with up to 8 quadwords from a 32-pixel span, starting at copy buffer entry0 and filling contiguously up to entry7. For 8-bpp frame buffers, a 32-pixel span consists of 32 bytes and only entries 0 through 3 are filled. The `Mask Source` parameter specifies which pixels in the span are enabled to be loaded into the copy buffer, but does not affect how each pixel is mapped to a copy-buffer entry. In effect, each pixel in the quadword-aligned source span is mapped to a specific byte (8-bpp) or Dword (32-bpp) of a specific entry. Zeros in the `Mask`

Source parameter affect only the leading and trailing ends of the span; the 21130 saves time by not reading pixels that will be masked.

For example, on a write to an 8-bpp frame buffer in the copy mode with a Mask Source value of FFFFFFFF, the 21130 loads all bytes in all copy-buffer entries, with the first pixel of the span loaded in the least significant byte of entry0 and the last pixel in the most significant byte of entry7. On the other hand, on a write with a Mask Source value of 00FFFF00, only entries 1 and 2 are filled.

- A write to the frame buffer in copy mode with the copy direction flag pointing to Destination Next

  For this operation, the 21130 unloads up to 8 quadwords from the copy buffer to a 32-pixel span, starting with entry0 and draining contiguously up to entry7. For 8-bpp frame buffers, a 32-pixel span consists of 32 bytes and only entries 0 through 3 are drained. On the write, Mask Destination bits enable each byte in an 8-bpp frame buffer and each Dword in a 32-bpp frame buffer. On the drain, the 21130 uses the Mask Destination to optimize frame buffer accesses, skipping leading and trailing zeros. The pixel masking does not affect how each copy buffer entry is mapped to the quadword; for example, entry6 is always mapped to the starting quadword-address + 7.

- A write to the copy-64 source register (GCSR)

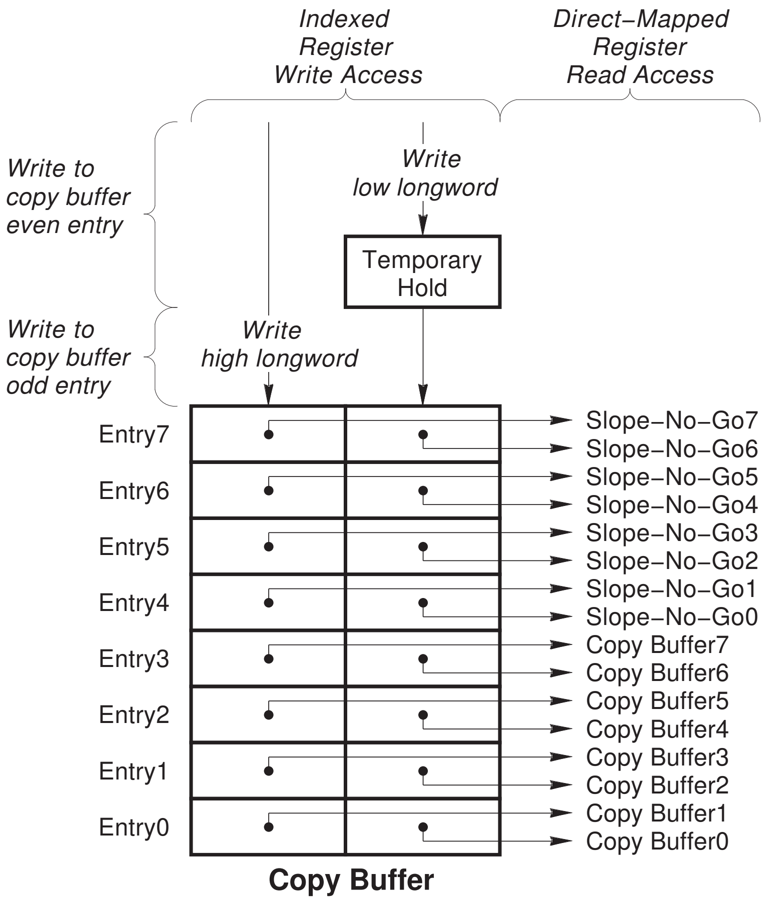
  For this operation, the 21130 fills the copy buffer with exactly 8 quadwords from a quadword-aligned address, starting with entry0 and filling contiguously up to entry7. Masking to read fewer than 8 quadwords is not done.

- A write to the copy-64 destination register (GCDR)

  For this operation, the 21130 drains exactly 8 quadwords from the copy buffer to a quadword-aligned address, starting with entry0 and draining contiguously up to entry7. Masking to write fewer than 8 quadwords is not done.

Figure 10–11 shows how the copy buffer registers (GCBR<7:0>) and slope-no-go registers (GSNR<7:0>) are mapped to the copy buffer entries.

Figure 10–11 Copy Buffer Layout



Programmed I/O Copy Buffer Operation

The copy buffer is also available for programmed I/O read and write operations. The host can sequentially fill or random-access-read all entries of the copy buffer through the GCBRs and GSNRs. See Section 8.5.4 for more information.

10.2.6.7 Fast Frame Buffer Access Using the Copy Buffer Registers

The best way to copy back-and-forth between host and screen is to use the DMA copy modes. While DMA should provide the best performance for large operations, it incurs an appreciable amount of overhead that can make it inefficient for small regions. On the other hand, simple mode, particularly when reading, is too slow for extended, dumb frame buffer access or image transfer between the host and screen. However, the copy mode, in conjunction with direct software access to the copy buffer, allows localized regions in the frame buffer to be quickly read and written.

Standard screen-to-screen copies involve groups of two alternating PCI writes, either to the source and destination frame buffer addresses (the standard copy mechanism) or to the GCSR and GCDR. However, each write can individually load or unload the copy buffer from or to the frame buffer. Additionally, software can directly read and write the copy buffer (Section 10.2.6.6). For example, by interleaving writes to the GCSR and GCDR with programmed I/O reads and writes, software can rapidly transfer image data between host memory and the frame buffer.

To transfer a bitmap from host memory to the frame buffer, software can do the following:

1. Write the GCBRs.
2. Write the resulting contents of the copy buffer to the frame buffer with either a write to the destination address with the copy direction flag set to write `Destination Next`, or a write to the GCDR.

Similarly, to transfer a bitmap from the frame buffer to host memory, software can load the copy buffer with either a write to the source address with the copy direction flag set to read `Source Next`, or a write to the GCSR.

In either transfer, using the copy direction flag can be awkward, because the copy direction flag can be manipulated only through writes to the frame buffer and the GPSR.

10.2.7 DMA-Read Copy Mode

In the DMA-read copy mode, a PCI write to the frame buffer address space copies a contiguous span of up to 2K Dwords (8KB) from external PCI memory to the frame buffer. The 21130 copies the span as a function of the parameters listed in Table 10–12.

```
DMA Read Copy Span (DMA Address, Frame Buffer Address Destination,
                    Read Count (-1), Mask Left <1:0>, Mask Right <1:0>,
                    Byte Mask, Pixel Shift, Raster Op, Destination Bitmap,
                    GIB Endian)
```

Table 10–12 DMA-Read Copy-Mode Parameters

Parameter	Source		Section
`Frame Buffer Address Destination`	PCI write address	—	—

`Read Count (-1)`
`Mask Right 1`
`Mask Right 0`
`Mask Left 1`
`Mask Left 0`
	PCI write data	<31:16>
		<15:12>
		<11:8>
		<7:4>
		<3:0>
			—

`DMA Address`	DMA base address register	GDBR <31:0>	8.5.15
`Pixel Shift`	Pixel shift register	GPSR <3:0>	8.5.5

`Byte Mask`
`Destination Bitmap`
`Raster Op`
	Raster operation register	GOPR <19:16>
		GOPR <10:8>
		GOPR <3:0>
			8.5.9

`Source Bitmap`	Mode register	GMOR <10:8>	8.5.1
`GIB Endian`	Deep register	GDER <21>	8.5.2

The PCI write cycle initiates a DMA-read copy of one span from PCI external memory into the frame buffer. The PCI write addresses the location of the destination span (`Frame Buffer Address Destination`). The PCI write data consists of a Dword `Read Count` and four read masks for the destination. Figure 10–12 shows the format of the PCI write data.

Figure 10–12 DMA-Read Copy-Mode PCI Write-Data Format

31–27	26–16	15–12	11–8	7–4	3–0
MBZ	Read Count (–1)	Mask Right 1	Mask Right 0	Mask Left 1	Mask Left 0

On the PCI write, the 21130 requests and then masters the PCI bus. It then reads `Read Count` Dwords from PCI external memory starting at the `DMA Address` and writes the Dwords to the frame buffer, starting at the `Frame Buffer Address`. On each successful transfer, the 21130 reads and writes 1 full Dword as follows:

- First Dword
  1. Reads the Dword from PCI external memory.
  2. Writes the Dword to the frame buffer at the `Frame Buffer Address Destination`. On the write, `Mask Left 0` masks the individual bytes of the first Dword.
  3. Decrements the `Read Count`.
- Second Dword
  4. Reads the Dword from PCI external memory.
  5. Writes the Dword to the frame buffer at the next frame buffer address. On the write, `Mask Left 1` masks the individual bytes of the second Dword.
  6. Updates the frame buffer address.
  7. Decrements the `Read Count`.
- Third Dword through next-to-last Dword
  8. Reads the Dword from PCI external memory.
  9. Writes the Dword to the frame buffer at the next frame buffer address. No bytes are masked.
  10. Updates the frame buffer address.
  11. Decrements the `Read Count`.
- Last Dword
  12. Reads the Dword from PCI external memory.

13. Writes the Dword to the frame buffer at the next frame buffer address. On the write, `Mask Right 0` masks the individual bytes of the last Dword.

14. Updates the frame buffer address.

15. Decrements the `Read Count`.

- After the last Dword is read and written, the 21130 writes the contents of the residue register to the frame buffer at the next frame buffer address, masked by `Mask Right 1`.
- For each write to the frame buffer destination, the 21130 also executes the specified `Raster Op` and filters data through the `Byte Mask`.

The DMA-read copy mode is functionally similar to the copy mode. However, the DMA-read copy mode differs in the following ways:

- Addresses are aligned to Dword (4 bytes) rather than quadword (8 bytes).
- The copy source is located in PCI external memory.
- External memory does not support all bitmap formats.
- The span can contain as many as 2K Dwords.
- The PCI write data passes two sets of mask data to mask the span's left and right edges (per-pixel masking is not allowed).

The DMA-read copy operation can be considered to be a copy-mode operation in which the source is accessed across the PCI bus and the granularity of the operation is 32 bits rather than 64 bits. Both the `Frame Buffer Address Destination` and `DMA Address` must be aligned to 4 bytes. The process of reading the source, rotating using the residue register, and writing the destination occurs in groups of 4 bytes rather than 8 bytes.

Because the `Frame Buffer Address Destination` and the `DMA Address` must be aligned to 4 bytes, software must adjust the desired source and destination addresses and masks for unaligned copies to the next whole Dword. However, all bitmaps, except packed 8-bpp bitmaps, are naturally aligned to 4 bytes.

Each Dword read from PCI external memory is concatenated with a 32-bit version of the residue register, and rotated by `Pixel Shift` bytes to produce the destination Dword written to the frame buffer. In the DMA-read copy mode, the `Pixel Shift` is calculated as in the copy mode (Table 10–11). However, unlike the copy mode, the `Pixel Shift` value range is 0 to +3, because backward copies are unnecessary and the granularity is 4 bytes rather than 8 bytes.

In the DMA-read copy mode, the copy buffer is not used, and the destination Dword is written directly to the frame buffer, using the specified `Raster Op` and `Byte Mask`. Residue-register priming and flushing is similar to the copy mode (Section 10.2.6).

The `GIB Endian` bit must be set to enable gib-endian byte swapping during simple writes and reads, DMA-read copy operations, and scaled-copy operations with 16-bpp and 32-bpp RGB sources.

10.2.7.1 Priming and Flushing the Residue Register

The two left-edge masks compensate for residue-register priming. For the copy alignments that require residue-register priming, the following occur:

- The `Frame Buffer Address Destination` is decremented one Dword (that is, the destination span's left edge is extended 4 bytes).
- `Mask Left 0` masks out the additional Dword.
- `Mask Left 1` contains the desired edge mask.

For alignments that do not require residue-register priming, `Mask Left 0` usually contains the desired edge mask and `Mask Left 1` is set to 1111_2.

The two right-edge masks compensate for copy alignments that require residue-register flushing. As in the copy mode, the pipelined nature of the source-read data path causes valid source data to remain in the residue register under certain conditions. Depending on the alignment and location of the span's right edge, this also applies to the DMA-read copy mode. But unlike the copy mode, the DMA-read copy mode requires explicit software attention to flush the residue register. Specifically, explicit residue-register flushing is required for alignments in which `Source Aligned` and `Destination Aligned` are the desired address alignments of the end of the span and one of the following is true:

`Source Align > Destination Align` and
`Source Aligned < Destination Aligned`

or

`Source Align < Destination Align` and
`Source Aligned > Destination Aligned`

To flush the residue register in such cases, `Mask Right 1` contains the desired edge mask and `Mask Right 0` is set to 1111_2.

Table 10–13 shows how the four edge-mask parameters are set according to the requirement to prime and flush the residue register.

Table 10–13 Edge Mask Settings in DMA-Read Copy Mode

	Mask Left		Mask Right	
Residue Register	**0**	**1**	**0**	**1**
Prime	0000	Left-edge mask	—	—
No prime	Left-edge mask	1111	—	—
Flush	—	—	1111	Right-edge mask
No flush	—	—	Right-edge mask	0000

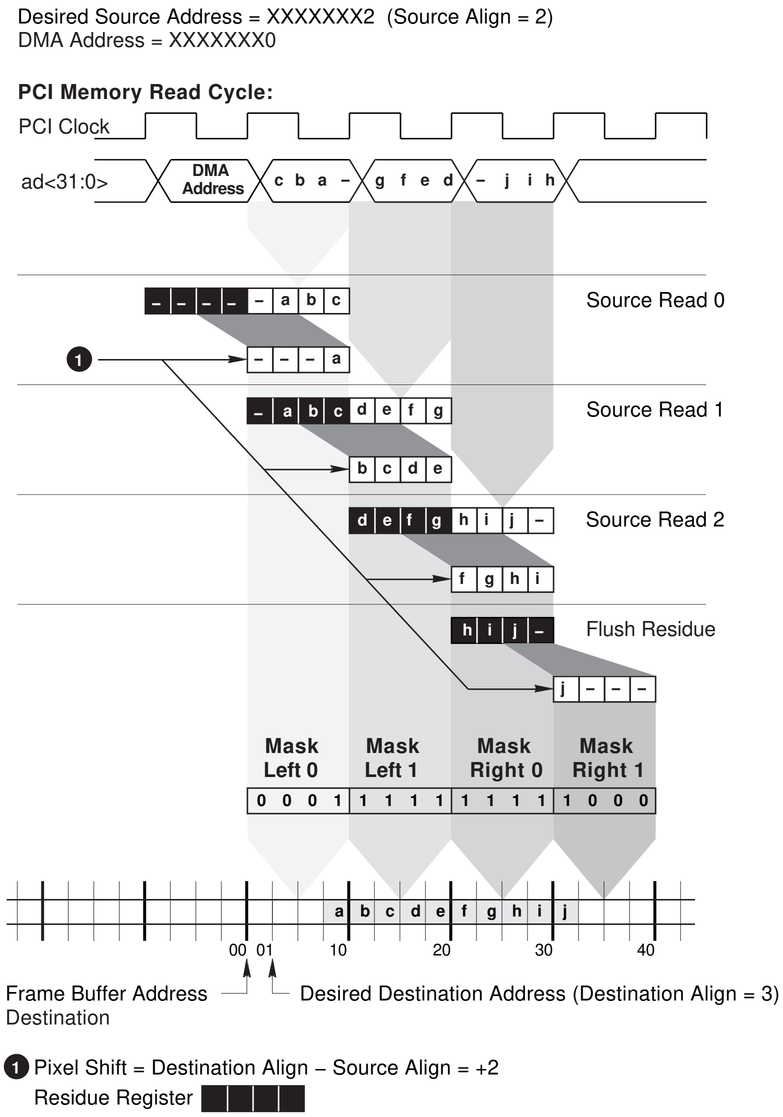
For short spans, in which fewer than 3 Dwords are read across the PCI, all edge masks are not used. (However, the DMA copy modes are seldom used to copy such small spans and the limitation can usually be ignored.) Table 10–14 lists the masks used for such spans.

Table 10–14 Edge Mask for Short Spans in DMA-Read Copy Mode

	Mask Left		Mask Right	
Read Count	**0**	**1**	**0**	**1**
≥ 3	Yes	Yes	Yes	Yes
2	Yes	No	Yes	Yes
1	No	No	Yes	Yes

Figure 10–13 is an example of a packed, 8-bpp, short span copied to the frame buffer over the PCI bus in DMA-read copy mode. The alignment requires an extra frame buffer write to flush the residue register. For descriptive purposes, the PCI cycle shown assumes a fast target response with no read latency.

Figure 10–13 DMA-Read Copy



10.2.8 Scaled-Copy Mode

In scaled-copy mode, a PCI write to the frame buffer address space copies (with a start and end mask) a contiguous span of up to 1K Dwords (4KB) from external PCI memory to the frame buffer. The 21130 performs the scaled-copy as a function of parameters specified Table 10–15.

```
Scaled-copy (Base Address, Frame Buffer Address Destination,
                           Repeat Count (-1), Read Count (-1), Start Pixel,
                           End Pixel, Draw#, Filter Sharp, Filter Smooth,
                           Pixel In Longword, Pixel Order, Pixel Format,
                           Dither, YUV Convert, 422 Out, Address Increment 1,
                           Initial Error, Error Increment 1, Error Increment 2,
                           Dither Row, Dither Column, GIB Endian)
```

Table 10–15 Scaled-Copy Mode Parameters

Parameter	Source		Section
Frame Buffer Address Destination	PCI write address	—	—
Repeat Count (-1) Read Count (-1) Base Address Mid	PCI write data	<31:28> <26:16> <15:0>	—
Base Address Hi Base Address Lo	DMA base address register	GDBR <31:23> GDBR <6:2>	8.5.15.2
Start Pixel End Pixel Draw# Filter Sharp Filter Smooth Pixel In Longword Pixel Order Pixel Format Dither YUV Convert 422 Out	Scaled-copy control register	GSCR <1:0> GSCR <3:2> GSCR <7:4> GSCR <9:8> GSCR <10:9> GSCR <21:20> GSCR <23:22> GSCR <25:24> GSCR <27> GSCR <29> GSCR <30>	8.5.16
Address Increment 1 Error Increment 1	Bresenham 1 register	GB1R <31:16> GB1R <15:0>	8.5.11.2
Error Increment 2	Bresenham 2 register	GB2R <15:0>	8.5.12
Initial Error	Bresenham 3 register	GB3R <31:15>	8.5.13.2
Dither Row	Dither row register	GDRR <31:27>	8.5.17

(continued on next page)

10.2 Graphics Modes

Table 10–15 (Cont.) Scaled-Copy Mode Parameters

Parameter	Source		Section
`Dither Column`	Dither column register	GDCR <31:27>	8.5.17
`GIB Endian`	Deep register	GDER <21>	8.5.2

The PCI write cycle initiates a scaled-copy operation. The PCI write addresses the location of the destination span (`Frame Buffer Address Destination`) and passes as data `Repeat Count`, `Read Count`, and `Base Address Mid`. Figure 10–14 shows the format of the PCI write data, and Table 10–16 describes its fields.

Figure 10–14 Scaled-Copy Mode PCI Write Data Format

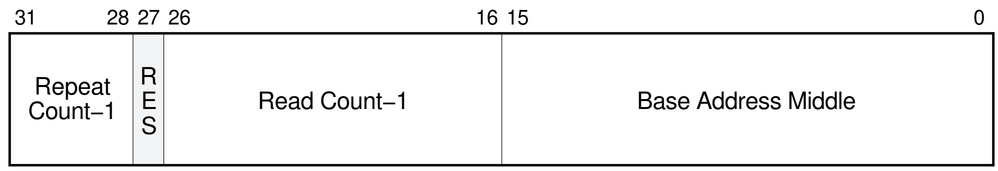
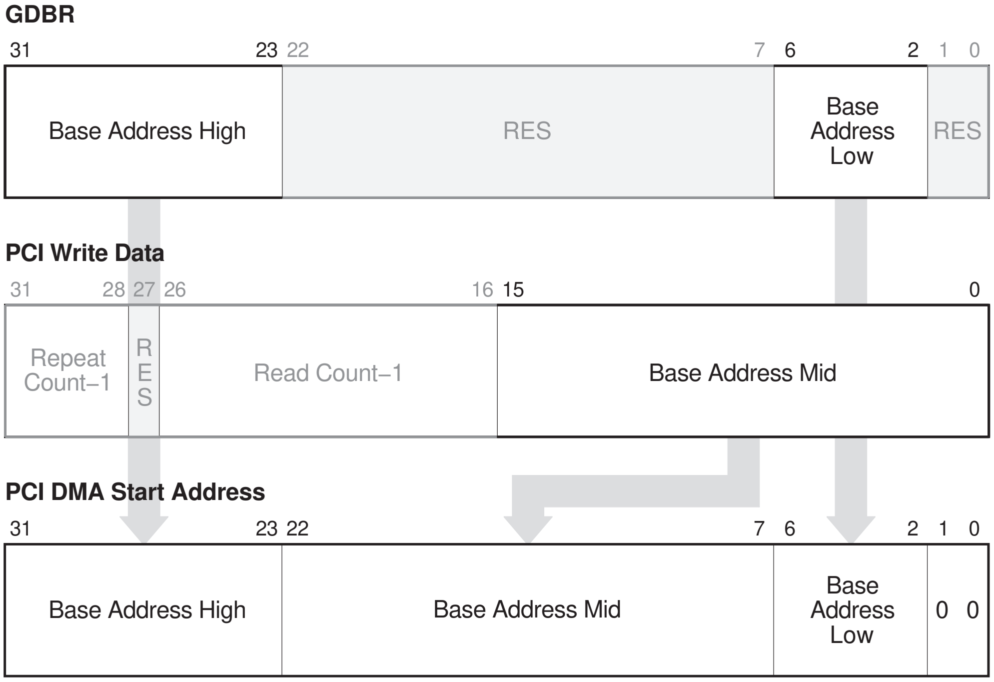


Table 10–16 Scaled-Copy Mode PCI Write Data Field Description

Bits	Field	Description
31:28	Repeat Count–1	Specifies the number of destination spans –1 to be created from the specified source span. The `Address Increment 1` field is used to advance from the end of one destination span to the start of the next. The `Dither Row` index is incremented by 1 for each destination span created.
27	RES	Reserved
26:16	Read Count–1	Specifies the number of Dwords –1 to be transferred in rendering each destination span specified by a scaled-copy command.
15:0	Base Address Middle	In scaled-copy mode this field is substituted for GDBR bits <22:7> to form the PCI DMA start address (Figure 10–15).

Figure 10–15 shows how the `Base Address Mid` field of the PCI write data is combined with the `Base Address Hi` and `Base Address Lo` fields to form the source-span starting address. (This method of specifying the source span starting address requires that the address stride between the start of consecutive source spans is a multiple of 128 bytes.)

Figure 10–15 Scaled-Copy PCI DMA Start Address



The DMA-read logic implements the scaled-copy mode. In this mode, as in normal DMA-read copy mode, the 21130 does a DMA operation to transfer a span from host memory to its frame buffer. However, the scaled-copy mode includes the following functions that can be applied to the source span during the transfer:

- Arbitrary up or down scaling
- Support for YUV source formats
- Sharpening and smoothing filters
- RGB color depth conversion
- Arbitrary remapping of 8-bpp indices

10.2.8.1 Video Rendering Pixel Flow

The scaled-copy mode supports video rendering. Using a sequence of scaled-copy operations, a to-be-displayed YUV-space video image resident in host memory (or any other PCI source) can be transferred to a window in the 21130's frame buffer.

Each DMA command issued in scaled-copy mode causes the 21130 to obtain mastership of the PCI bus and transfer a single source span to one or multiple destination spans in the frame buffer. During the transfer, the span is filtered, color-space converted, and scaled in the *X* axis. A repeat count (indicating the number of destination spans to be created from the source span) assists in doing *Y*-axis scaling.

Figure 10–16 shows the flow of pixels through the 21130 during video rendering. The span is rendered as follows.

❶ Input DMA Dword pixel data.

❷ Byte-lane multiplexing is done to extract the Y, U, and V components from the appropriate positions in each DMA Dword (multiple source YUV formats are supported). 8-bpp (treated as 3:3:2) is MSB-replicated to 8:8:8.

❸ Decoded pixel data is 24-bit (YUV or RGB) data and pixel *x, y* location.

❹ The pixels pass through a selectable Y/G prescaling filter. The filter can be sharpening (–0.5, 2.0, –0.5), smoothing (0.5, 0.0, 0.5), or bypassed (0, 1, 0).

Note

The filter should not be used with 8-bpp (3:3:2 or index) source formats.

---

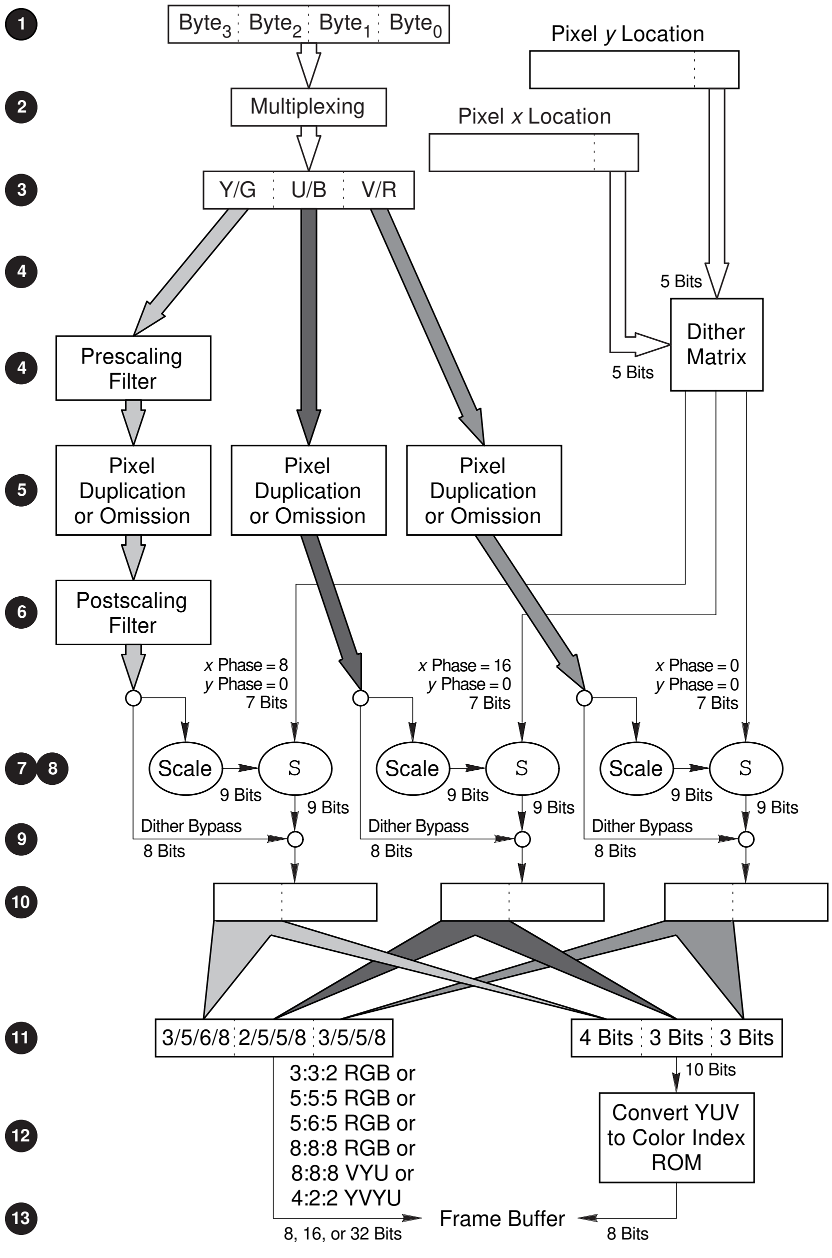
❺ The pixels are duplicated or omitted (to effect magnification or minification) through a Bresenham-style scaler.

❻ The pixels pass through a selectable Y/G postscaling filter. The filter can be sharpening (–0.5, 2.0, –0.5), smoothing (0.5, 0.0, 0.5), or bypassed (0, 1, 0).

Note

The filter should not be used with 8-bpp (3:3:2 or index) source formats.

Figure 10–16 Host-to-Screen Scaled-Copy and Video Rendering Pixel Flow



The prescaling and postscaling filters operate only on the luminance components of the pixels. Duplication and omission occur across all components of the YUV triplets.

After passing through the postscaling filter, pixels are dithered and quantized to form 4:3:3 YUV values. The 32 × 32 dither matrix 7-bit output is ‘‘tiled’’ throughout the image. The U/B and V/R outputs are phase-shifted with respect to the Y/G output.

❼ The YUV/RGB data is shifted and scaled for dithering quantizations.

❽ The ‘‘controlled noise’’ from the dither matrix is added to the YUV/RGB data.

❾ Either the dithered value or the original value (which bypasses the dither logic) is selected.

❿ The selected values are quantized (LSBs are truncated to form 8:8:8, 5:6:5, 5:5:5, or 3:3:2 RGB, or 8:8:8 or 4:4:3 YUV).

⓫ The quantized values are concatenated into 8:8:8, 5:6:5, 5:5:5, or 3:3:2 RGB; or 8:8:8 or 4:4:3 YUV; or 4:2:2 YVYU.

⓬ A 1K × 8 ROM converts the quantized YUV values into 8-bit color indices. (The color indices are translated into RGB values by a dedicated video palette ROM LUT in the video back end.)

⓭ The pixel data is saved to the frame buffer. In 16-bpp or 32-bpp modes, the 8-bit indices are replicated across all byte channels of the target pixels. Certain mode selections also allow 4:2:2 or 4:4:4 YUV formats to be written to the frame buffer (useful when driving an external NTSC encoder connected to the 21130’s VAFC port).

10.2.8.2 YUV Pixel Formats

4:4:4 αVYU Format

31 24	23 16	15 8	7 0
a	V	Y	U

In the 4:4:4 αVYU format, each Dword represents a single YUV pixel. During scaled-copy operations the α channel is not propagated from source to destination pixels. In other words, the α channel in αYUV format destination pixels is indeterminate. This format matches PCI multimedia (MM) guidelines (*PCI Multimedia Design Guide, Revision 1.0*).

4:2:2 YVYU Format

31 – 24	23 – 16	15 – 8	7 – 0
Y_1	V	Y_0	U

In the 4:2:2 YVYU format, each Dword represents two YUV pixels. The two pixels have independent Y values, but share U and V components. This format matches the PCI MM guidelines for the little-endian 4:2:2 YUV format. If the 422OUT bit is set (GSCR <30>, Section 8.5.16), the 21130's video logic can also write pixels in this format to the frame buffer.

4:2:2 YVYU Destination Pixel Format

In a 16-bpp output mode, a 4:2:2 YVYU format destination is produced by setting the 422OUT bit (GSCR <30>). This destination format is supported to interface with an external NTSC encoder device through the VAFC connector.

Each 4:2:2 YVYU Dword is formed from 2 consecutive destination pixels as created by the scaled-copy pipeline. The Y_0 and U values are taken from the destination pixel's write address that points to the low word in the destination Dword. Similarly, the Y_1 and V values are taken from the destination pixel's write address that points to the high word in the destination Dword. Consequently, the U and V values are sub-sampled by alternately ignoring one and then the other U or V component. Note that U and V values are not interpolated in this process.

4:2:2 UYVY Format

31 – 24	23 – 16	15 – 8	7 – 0
U	Y_0	V	Y_1

In the 4:2:2 UYVY format, each Dword represents two YUV pixels. The two pixels have independent Y values, but share U and V components.

4:2:2 VYUY Format

31–24	23–16	15–8	7–0
V	Y_1	U	Y_0

In the 4:2:2 VYUY format, each Dword represents two YUV pixels. The two pixels have independent Y values, but share U and V components. This format matches PCI MM guidelines.

10.2.8.3 16-bpp and 32-bpp RGB Formats

Both the 16-bpp and 32-bpp RGB formats require byte swapping that effectively performs an endian-swap of bytes within each pixel. The `GIB Endian` bit must be set to enable gib-endian byte swapping during simple writes and reads, DMA-read copy operations, and scaled-copy operations with 16-bpp and 32-bpp RGB sources. (The gib-endian byte swapping implemented in the 21130 is based on the *PCI Multimedia Design Guide, Revision 1.0* and the Apple seed note *Designing PCI Cards for Macintosh Computers.*)

10.2.8.4 Rendering Full Frames

The programming interface for scaled-copy rendering is optimized for the case of an unoccluded or trivially occluded target window. This performance-critical case requires only one PCI write per source video-frame span, enabling maximum use of the 21130 command FIFO. (The command FIFO queues the write commands used to initiate scaled-copy renderings. By queuing enough scaled-copy commands, the 21130 can buffer enough work to keep it busy while the CPU spends a few cycles on other tasks, such as video decompression.) As the occlusion of a target window becomes more complex, additional PCI writes and CPU supervision are required, which might degrade overall rendering performance. Such performance degradation is acceptable for video rendering applications, because complexly occluded windows are expected to lack user focus.

10.2.8.5 Unoccluded or Trivially Occluded Target Windows

Unoccluded or trivially occluded target windows typically require only one PCI write per span of the source video frame.

The PCI write address specifies the `Frame Buffer Address Destination` for the start of the destination span. The destination span address can also be specified indirectly, by a write to the address register (GADR, Section 8.5.6); or the destination span address can be accumulated through writes to the continue register (GCTR, Section 8.4.3), so that a new address is not required for each operation.

The PCI write data specifies the source span starting address (`Base Address Mid`), the number of Dwords to be transferred (`Read Count`), and the number of consecutive destination spans to be created from the source span (`Repeat Count`). The `Address Increment 1` value is used to advance from the end of one destination span to the start of the next.

Unoccluded Target Window

To render into an unoccluded target window, software first writes to the `Base Address Hi` and `Base Address Lo` fields. Software then writes the `Frame Buffer Address Destination` to the GADR. Subsequent writes to the GCTR initiate one scaled-copy operation for each PCI write. The `Base Address` (DMA start address) for each scaled-copy operation is formed as shown in Figure 10–15.

During the scaled-copy operation, the source span at the `Base Address` is transferred to one or multiple destination spans. After rendering each destination span, the `Dither Row` is incremented by 1 and the `Frame Buffer Address` is incremented by `Address Increment 1` in preparation for rendering the next destination span.

Trivially Occluded Target Window

If occlusions exist such that each screen span has identical *x*-extents and is composed of only a single unoccluded region (for instance, a situation in which only the left side of a target window is exposed) the window is trivially occluded. Trivially occluded windows require only one PCI write per source span. The `Base Address Lo` field can be adjusted to align the source span starting address with the edge of the occlusion.

Note

The scaled-copy mode supports operation in which source frame data is not arranged specially in PCI memory. In this case, two writes (rather than one) are required per screen span rendered. In addition to the frame buffer write, the GDBR must be written for each span in order to specify arbitrary values for the `Base Address Hi` and `Base Address Lo` fields.

10.2.8.6 Nontrivially Occluded Windows

For windows that are occluded such that there exist regions of differing *x*-extents, additional writes are required to the registers controlling scaled-copy operations. Upon crossing into a region of different *x*-extent, the `Base Address Lo`, `Dither Column`, `Initial Error`, `Start Pixel`, `End Pixel`, and `Draw#` fields might require updating, depending on the situation. To minimize the number of additional PCI writes required, software can render spans with identical *x*-extents as a group.

10.2.8.7 Determining the Command FIFO Entry Availability

When rendering a video frame using scaled-copy operations, it is useful to know the number of entries that are available in the 21130 command FIFO. By issuing only the number of commands needed to fill the FIFO, the CPU can avoid stalling until enough scaled-copy operations complete to accommodate the remaining operations.

To determine command FIFO entry availability, the current FIFO read and write pointers are included in the data returned on a command status register (MCSR) read. The CPU can subtract and mask to determine the number of entries available (*minimum entries available = write pointer − read pointer*). The command status register returns 00000000_{16} when the 21130 is idle.

Figure 10–17 shows the MCSR format (read and write).

Figure 10–17 MCSR Format

31–22	21–16	15–14	13–8	7–1	0
RES	FIFO Write Pointer	RES	FIFO Read Pointer	RES	Busy

10.2.8.8 Scaling

The 21130 uses a Bresenham scaler to scale spans horizontally. The precision of the `Initial Error`, `Error Increment 1`, and `Error Increment 2` allows the magnification or reduction of source lines from 0 to 65535 (0 to 2^{16}–1) pixels, in 1-pixel increments, to frame buffer spans of 0 to 65535 pixels, in 1-pixel increments.

The CPU must do vertical scaling. Magnification can be done with multiple scaled-copy operations, to place a given source frame line into multiple 21130 frame buffer spans. Similarly, reduction can be done by selectively omitting source frame lines. For arbitrary vertical scaling, a software Bresenham scaler (similar to the 21130 internal scaler) can be used. The 21130's dithering function helps to reduce *banding* caused by vertical span duplication during magnification.

10.2.8.9 Programming the Bresenham Scaler for Unoccluded Spans

The following sections summarize how to program the horizontal Bresenham scaler to effect reduction, magnification, and unity scaling. The method of calculating the initial Bresenham errors in this section conforms to the Windows NT DrvStretchBlt source and destination alignment constraints.

Reduction

To map a source frame line of `numSource` pixels to a frame buffer span of `numDest` pixels (where `numSource > numDest`), the following Bresenham increments and initial error can be used:

```
Bresenham1.Increment1 = numDest;
Bresenham2.Increment2 = numSource - numDest;
Bresenham3.InitialError = (Increment1 >> 1) - Increment2;
Mode.mode = Scaled-copy_reduce;
```

Iteration within the 21130's Bresenham scaler proceeds as follows:

```
error = InitialError;
repeat until no more source pixels
        if (error < 0)
                move to next source pixel;
                error = error + Increment1;
        else
                create destination pixel from current source pixel;
                move to next source pixel;
                error = error - Increment2;
```

Magnification

To map a source frame line of `numSource` pixels to a frame buffer span of `numDest` pixels (where `numSource < numDest`), the following Bresenham increments and initial error can be used:

```
Bresenham1.Increment1 = numSource;
Bresenham2.Increment2 = numDest - numSource;
Bresenham3.InitialError = (Increment1 >> 1) - Increment2;
Mode.mode = Scaled-copy_magnify;
```

Iteration within the 21130's Bresenham scaler proceeds as follows:

```
error = InitialError;
repeat until no more source pixels
        if (error < 0 && (Increment1 != 0))
                create destination pixel from current source pixel;
                error = error + Increment1;
        else
                create destination pixel from current source pixel;
                move to next source pixel;
                error = error - Increment2;
```

Unity Scaling

To map a source frame line of numSource pixels to a frame buffer span of numDest pixels (where numSource = numDest) the following Bresenham increments and initial error can be used:

```
Bresenham1.Increment1 = 0;
Bresenham2.Increment2 = 0;
Bresenham3.InitialError = 0;
Mode.mode = Scaled-copy_reduce or _magnify;
```

10.2.8.10 Scaling of Occluded Spans

When rendering an occluded span, Bresenham scaler initialization is slightly different than for nonoccluded or trivially occluded spans.

The effective scaling factor of an occluded span is identical to that of its parent unoccluded span, such that the Error Increment 1 and Error Increment 2 fields are programmed as described in the preceding sections. (When calculating the increments, the values of numSource and numDest for the full, unoccluded span are used.) On the other hand, it might be necessary to modify the Initial Error term if the first frame buffer pixel to be rendered does not originate from the first pixel in the source span. When this is the case, the Initial Error value should be set to the accumulated error value that would have resulted if Bresenham stepping was done from the first Dword in the source line up to the pixel that is the origin of the first frame buffer pixel to be rendered. If the error is not initialized in this way, the target image might be skewed in the *x*-direction at boundaries between regions of unlike occlusion. Software can calculate the Initial Error value using the same algorithm as the 21130 scaling hardware. Note that the Initial Error calculation needs to be done only once per region of similar *x*-extent.

10.2.8.11 Specifying Span Starting and Trailing Edges

The `Initial Error`, `Start Pixel`, `End Pixel`, and `Draw#` values position the starting and trailing edges for spans generated by scaled-copy operations.

The first pixel generated by a scaled-copy operation is placed at the `Frame Buffer Address Destination` specified by the PCI write which initiated the operation (or was programmed into the GADR, or was advanced by `Address Increment 1` from the end of the previous span). The `Initial Error` and `Start Pixel` values map the destination span starting edge to the appropriate position on the source span.

The `Start Pixel` value indicates which pixel in the first source Dword is the first pixel to be used. For example, if the source format is 4:2:2 Y_1VY_0U and the `Start Pixel` value is 1, then the source pixel associated with Y_0 of the first Dword will be skipped; instead, the first source pixel used will be the one associated with Y_1. The `Initial Error` value establishes the number of destination pixels to be created from the first source pixel.

The trailing edge of a destination span is specified with the `End Pixel` and `Draw#` values. The `End Pixel` value indicates which pixel within the last DMA Dword is the last pixel to be considered from the source span. The `Draw#` value indicates the number of destination pixels to be generated from the last source pixel. The 4-bit wide `Draw#` value supports span trailing edge masking for magnifications up to approximately 16 ×.

Note

Nonsensical combinations of `Start Pixel` and `End Pixel` (`Start Pixel` > `End Pixel` for DMA-count of 1; or `Start Pixel` or `End Pixel` > the number of pixels in a Dword) produce undefined results, but will not hang the chip.

10.2.8.12 Required Software Interlock

Scaled-copy mode requires a software interlock when updating the GB1R, GB2R, GB3R, GDCR, or GDRR so that register updates do not corrupt an ongoing scaled-copy operation. Digital recommends an interlock mechanism that writes the GSCR before writing to the GB1R, GB2R, GB3R, GDCR, or GDRR. This causes the register updates to stall until an ongoing scaled-copy mode operation is complete. This software interlock should not impose performance constraints because the registers should require updating only when transitioning to a swath of different *x*-extent.

10.2.9 Opaque-Line Mode

Section 10.2.9.1 describes opaque-line mode operations initiated by the standard frame buffer write mechanism, and it is included for continuity. However, the same functionality is more efficiently implemented with the alternate slope register write mechanism described in Section 10.2.9.2.

10.2.9.1 Drawing Lines with Frame Buffer Writes

In the opaque-line mode, a PCI write to the frame buffer address space draws a masked, 16-pixel, bitonal line segment starting at the specified address. For this description, a *line segment* is defined as a string of 16 contiguous pixels drawn along an arbitrary slope; a *line* is made up of multiple segments and its length is arbitrary.

The 21130 draws the line segment as a function of the parameters listed in Table 10–17.

```
Opaque Line (Frame Buffer Address, Frame Buffer Address <1:0>, Line Mask, Raster Op,
             Byte Mask, Foreground, Background, Address Increment 1,
             Address Increment 2, Error Increment 1, Error Increment 2, Initial Error,
             Length, Destination Bitmap, Cap Ends);
```

Table 10–17 Opaque-Line Mode Parameters

Parameter	Source		Section
`Frame Buffer Address`	PCI write address	—	—
`Frame Buffer Address <1:0>` `Line Mask`	PCI write data	<17:16> <15:0>	—

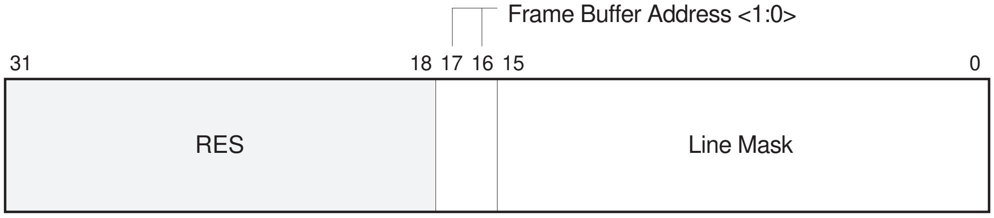
`Byte Mask`
`Destination Bitmap`
`Raster Op`
	Raster operation register	GOPR <19:16>
		GOPR <10:8>
		GOPR <3:0>
			8.5.9

`Foreground`	Foreground register	GFGR <31:0>	8.5.8
`Background`	Background register	GBGR <31:0>	8.5.8
`Address Increment 1` `Error Increment 1`	Bresenham 1 register	GB1R <31:16> GB1R <15:0>	8.5.11
`Address Increment 2` `Error Increment 2`	Bresenham 2 register	GB2R <31:16> GB2R <15:0>	8.5.12
`Initial Error` `Length`	Bresenham 3 register	GB3R <31:16> GB3R <3:0>	8.5.13
`Cap Ends`	Mode register	GMOR <15>	8.5.1

The PCI write cycle initiates the line segment drawing operation to the 21130 frame buffer. The PCI write addresses the start of the segment (`Frame Buffer`

Address) and passes as data the two address LSBs (Frame Buffer Address <1:0>) and a 16-bit Line Mask to pattern the line. Figure 10–18 shows the format of the PCI write data.

Figure 10–18 Opaque-Line Mode PCI Write-Data Format



The Frame Buffer Address must be aligned to 1 pixel. For drawing to packed 8-bpp bitmaps, the two LSBs of the frame buffer address (Frame Buffer Address <1:0>) are part of the PCI write data. For drawing to any other bitmap, the address is Dword-aligned (by default) and Frame Buffer Address <1:0> is ignored.

Before writing to the frame buffer in a line mode, software must ensure that the Address Increment, Error Increment, Length, and Initial Error values stored in the Bresenham registers are appropriate to the slope, octant, and length of the line segment. Software can write these parameters directly or initialize them indirectly by writing the GSLRs or GSNRs (Section 10.2.9.2).

Before starting to draw the line segment, the 21130 uses the Frame Buffer Address (concatenated with Frame Buffer Address <1:0>, if necessary) to initialize the address stored in its Bresenham engine.

The 21130 draws a line segment as follows:

1. To draw the first pixel, the 21130 checks the bits from the Line Mask as follows:
   - If the first Line Mask bit = 1, Foreground color is written.
   - If the first Line Mask bit = 0, Background color is written.
2. On any write in opaque-line mode, the 21130 does the specified Raster Op and uses the Byte Mask to mask the writes to individual pixel bits.
3. The Bresenham engine then takes one step along the line, as follows:
   - If the current error term is <0, the engine adds Address Increment 1 to the current address and adds Error Increment 1 to the current error term to take one step along the major axis of the line segment.

- If the current error term is ≥ 0, the engine adds Address Increment 2 to the current address and subtracts Error Increment 2 from the current error term to take one step along the major and minor axes of the line segment.

4. The 21130 then decrements Length and repeats the process for each pixel along the line, until the segment Length = 0. Once initialized by Frame Buffer Address <1:0>, the 21130 internally monitors which Dword-byte is to be written to a packed 8-bpp bitmap as it steps through the line.

The following pseudo-code represents the basic algorithm for opaque-line mode:

```
while (Length > 0)
{
    Pixel = (Extract Bit(Line Mask,Length)) ? Foreground : Background;
    Write Frame Buffer(Frame Buffer Address, Pixel, Raster Op, Byte Mask,
                       Destination Bitmap);
    /* Bresenham step along line */
    if (Error < 0)
    {
        Frame Buffer Address += Address Increment 1;
        Error += Error Increment 1;
    }
    else
    {
        Frame Buffer Address += Address Increment 2;
        Error -= Error Increment 2;
    }
    Length --;
}
```

10.2.9.2 Drawing Lines with the Slope Registers

Drawing lines as described in Section 10.2.9.1 results in a bottleneck for the following reasons:

- Overall line throughput in the CPU or I/O is slow, due to the software overhead incurred in setting up and writing all of the Bresenham address and error terms for each line.
- The 21130's high-performance, 64-bit memory bus can draw at a rate faster than a CPU can supply commands and data.

To avoid bottlenecks, the GSLRs are a faster and simpler mechanism for drawing lines with less computation and fewer writes.

Table 10–18 is the modified list of parameters used in drawing lines with the GSLRs.

Table 10–18 Opaque-Line Mode Parameters Using Slope Registers

Parameter	Source		Section
`Absolute Dy` `Absolute Dx`	Slope register	GSLR*n* <31:16> GSLR*n* <15:0>	8.4.1
`Frame Buffer Address`	Address register	GADR <31:0>	8.5.6
`Line Mask`	Data register	GDAR <15:0>	8.5.7

`Byte Mask`
`Destination Bitmap`
`Raster Op`
	Raster operation register	GOPR <19:16>
		GOPR <10:8>
		GOPR <3:0>
			8.5.9

`Foreground`	Foreground register	GFGR <31:0>	8.5.8
`Background`	Background register	GBGR <31:0>	8.5.8
`Bitmap Width`	Bresenham width register	GBWR <15:0>	8.5.14
`Cap Ends`	Mode register	GMOR <15>	8.5.1

Drawing lines with the GSLRs is similar to the standard line drawing mechanism, with the following exceptions:

- A write to a GSLR, rather than to the frame buffer, initiates the drawing operation.
- The address and line-mask data are specified in registers.
- Software must initialize the Bresenham width register (GBWR, Section 8.5.14) instead of the GB1R, GB2R, and GB3R registers.

Each GSLR corresponds to one drawing octant (Figure 8–1) and contains two 16-bit fields, one for the absolute value of the slope rise (`Absolute Dy`) and the other for the absolute value of the slope run (`Absolute Dx`).

On a write to a GSLR, the 21130 calculates the Bresenham terms and then starts the standard Bresenham line drawing algorithm. Because the PCI write that initiates the drawing operation addresses a GSLR and passes slope information as data, the `Frame Buffer Address` and `Line Mask` parameters are specified in the GADR and GDAR, rather than in the PCI write data.

Given the slope and octant information, the 21130 does all of the Bresenham setup. It calculates all of the Bresenham error and address terms and stores them in the appropriate Bresenham register fields. The 21130 implements a slightly different setup algorithm depending on whether the line must comply with Win32 or be compatible with existing Digital conventions for lines drawn under X.

The following pseudo-code represents the basic hardware setup algorithm:

```
dxGEdy = (Absolute Dx >= Absolute Dy);
dxGE0 = (Absolute Dx > 0);
dyGE0 = (Absolute Dy > 0);
dmajor = (dxGEdy ? Absolute Dx : Absolute Dy)
dminor = (dxGEdy ? Absolute Dy : Absolute Dx)
majorGE0 = (dxGEdy ? dxGE0 : dyGE0);
minorGE0 = (dxGEdy ? dyGE0 : dxGE0);
amajor = (dxGEdy ? PixelBytes : BitmapWidth);
aminor = (dxGEdy ? BitmapWidth : PixelBytes);
if Graphics Environment
     {
     errinc = (dxGEdy ? dyGE0 : !dxGE0);
     }
else
     {
    errinc = majorGE0;
     }
/* Initial Bresenham terms */
Length = dmajor + Cap Ends mod16;
Error Increment 1 = dminor;
Error Increment 2 = dmajor + ~dminor + 1;
Initial Error = ((dminor<<1) + ~dmajor + errinc) >>1;
Address Increment 1 = (majorGE0 ? amajor : ~amajor + 1);
Address Increment 2 = (minorGE0 ? aminor : ~aminor + 1) + Address Increment 1;
```

Cap Ends is an additional parameter specified in the GMOR. Results are undefined if Absolute Dx and Absolute Dy are both set to zero.

Note that Length is set to the major axis length MOD 16. Therefore, the 21130 draws up to, but not necessarily exactly, 16 pixels when a GSLR is written. For example, if the major axis length is 19, writing a GSLR causes a 3-pixel line to be drawn (assuming Cap Ends is set).

Because Win32 has strict requirements on which pixels must be illuminated for a particular line, while X does not, the Initial Error term is calculated differently for Win32 display drivers than for Digital X servers. Win32 lines must comply with Microsoft's grid intersect quantization (GIQ) specification:

> ''That is, the geometric line from the starting point to the ending point is imagined as drawn on a grid with p(ix)els at the grid intersections. Whenever the geometric line crosses the grid, the nearest p(ix)el is illuminated. In the case where two p(ix)els are equidistant, the upper or left p(ix)el is illuminated, unless the slope of the line is exactly one, in which case the upper or right p(ix)el is illuminated.''

While Win32 lines are generally X-compliant, they do not comply with Digital's traditional way of drawing lines under X. Traditionally, Digital's X servers draw X-compliant lines that are not always Win32-compliant; specifically, the upper or left pixel is not always illuminated in accordance with the GIQ specification. Consequently, the 21130 line setup compensates for the difference by setting `Initial Error` as a function of the graphics environment. If there is no need to adhere to traditional practice, the 21130 draws X-compliant lines, including when the graphics environment is Win32.

In any line mode, drawing lines by writing to the GSLRs is almost always faster than drawing lines by writing to the frame buffer. However, additional restrictions imposed when drawing Win32-compliant lines prevent using the GSLRs. Therefore, some lines can be drawn only by directly writing to the frame buffer. These restrictions affect the 21130 display driver rather than the hardware (Section 11.8).

The slope-no-go registers (GSNR<7:0>) mimic the behavior of the GSLRs, but they do not initiate drawing. That is, on a write to a GSNR, the 21130 processes the slope data, generates the Bresenham terms, and loads them into the Bresenham registers, but the line is not drawn. The GSNRs are useful for drawing clipped lines, in which some portion of the line is not drawn.

Figure 10–19 is an example opaque-line mode operation.

10.2.9.3 Extending and Linking 2D Lines

The 21130 processes up to 16 pixels per line-segment drawing operation, but graphics applications do not limit line drawing requests to lines that are 16 or fewer pixels. Additionally, applications can request a string of lines, with each subsequent line starting at the end of the preceding line. The line drawing hardware supports two ways of linking 16-pixel line segments:

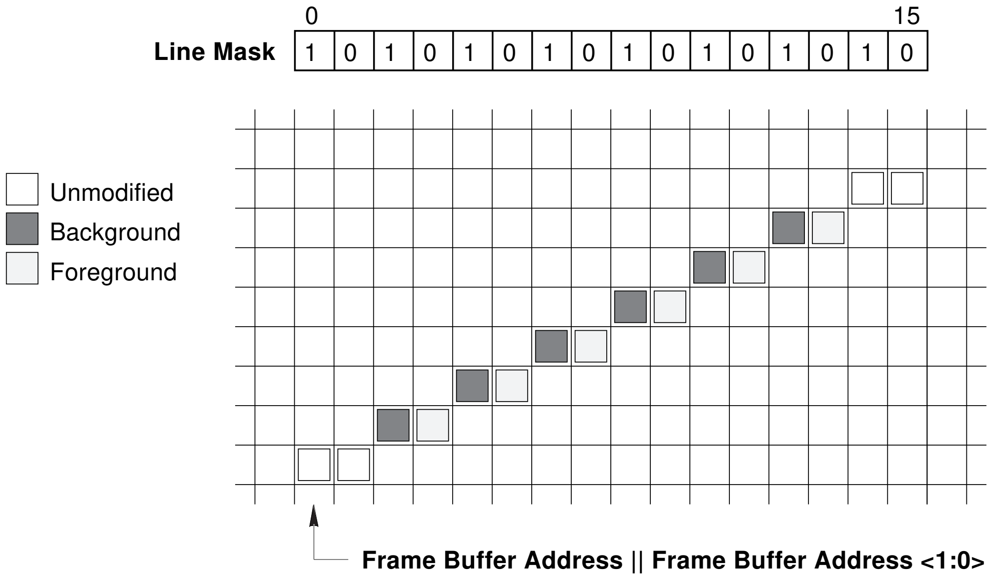
- A previously drawn line can be extended up to 16 pixels along the same slope.
- A new line drawing can start at the end of a previously drawn line.

For example, to draw a 50-pixel line, the 21130 software must link four segments along the same line. The 21130 allows multiple segments of the same line and multiple lines to be drawn without software reassigning the address and other parameters for each segment. The Bresenham engine has several features to facilitate such operations.

10.2 Graphics Modes

Figure 10–19 is an example opaque-line mode operation.

Figure 10–19 Opaque Line Drawing



The Bresenham engine contains a working set of the parameters `Initial Error`, `Length`, and `Frame Buffer Address`. When a line-segment drawing operation is initiated, the Bresenham engine conditionally loads the primary parameter values into its working set. (The drawing operation can be initiated by a write to a GSLR or the GCTR.) During the line-stepping process, the Bresenham engine operates only on the working set.

On completion of the segment drawing operation, the Bresenham engine leaves its working set in a state suitable for linking to the next segment or line. Specifically, the Bresenham engine's working set of parameters is managed as follows:

- If a GSLR was written since the last line segment was drawn, the Bresenham engine updates its working copies of `Length` and `Initial Error` from the register before drawing a line segment. Otherwise, the line segment is drawn without updating the working set parameters.

- If a new address was specified in the GADR since the last line segment was drawn, the Bresenham Engine updates its working copy of the `Frame Buffer Address` before drawing a line segment. Otherwise, the line segment is drawn without updating the working copy of the `Frame Buffer Address`.
- On completion of a line-segment drawing operation, the Bresenham engine does the following:
  - Leaves its working copy of the `Frame Buffer Address` at the address of the next pixel along the line.
  - Resets the value of its working copy of `Length` to 16.
  - Sets its copy of `Initial Error` to the error term for the next pixel along the line.

In other words, the Bresenham engine uses new values of `Initial Error` and `Length` only if a GSLR was reloaded after the last line segment was drawn; otherwise, the engine does not sample either parameter, but uses the current working set values.

Similarly but independently, the Bresenham engine uses a new address only if a new address was specified by a write to the GADR after the last line segment was drawn.

Extending a Single Line

By taking advantage of the Bresenham engine behavior, software can extend the current opaque line up to 16 pixels by writing the `Line Mask` for the next segment to the GCTR. Because software does not write a GSLR, the Bresenham engine's working parameters correspond to the next pixel in the line, with `Length` reset to 16. Given the new line mask, the 21130 extends the line 16 pixels.

In summary, the fastest way for software to extend the current line by one more 16-pixel segment is to write the following:

1. Any relevant registers, except the GADR and GSLR
2. The segment's `Line Mask` to the GCTR

This process can be repeated as many times as necessary to draw lines of arbitrary length. Usually, the first segment is drawn by writing to a GSLR, and all subsequent segments along the same line are drawn by writing to the GCTR.

Figure 10–20 shows a typical sequence for drawing a line of length *n* by drawing the first segment and then drawing as many extending segments as necessary.

Note

Other than `Length`, `Initial Error`, and `Frame Buffer Address`, all relevant opaque-line mode parameters (such as `Foreground` and `Background`) are sampled every time a line segment drawing operation is initiated.

Linking Multiple Lines

The Bresenham engine behavior also allows software to link multiple lines. Each line can have different color, mask, or slope attributes, but the lines must be drawn end-to-end (a polyline). In this case, software writes a GSLR, rather than the GCTR, and does not write the GADR. This effectively reinitializes all of the engine's slope parameters, including `Initial Error`, but does not change the working copy of the `Frame Buffer Address`.

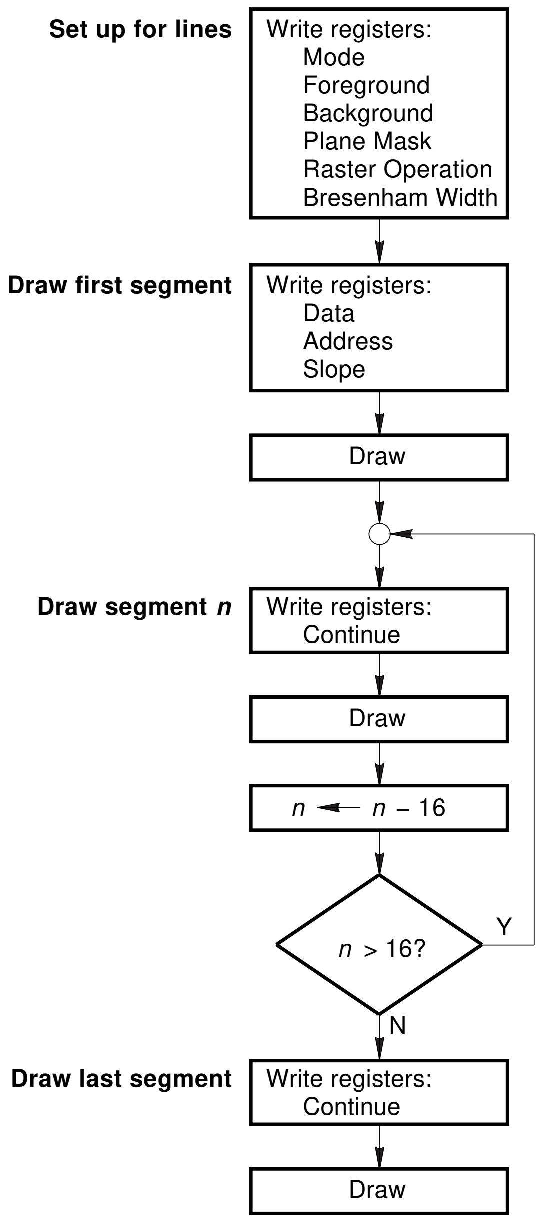
In summary, to write the first segment of a new line where the previous line terminated, software writes the following:

1. Any relevant registers except the GADR
2. A GSLR

Specifying Cap Ends

Whether extending or linking lines, software must specify the appropriate value for `Cap Ends` (GMOR <15>). When the value of `Cap Ends` = 0, the last pixel in the line is not drawn; otherwise, the last pixel is drawn. Therefore, to extend or link line segments as described above, software must set `Cap Ends` = 0, so that the last pixel in the previous line segment is not drawn. If the value of `Cap Ends` = 1, the last pixel in the previous line segment and the first pixel in the next line segment will be drawn at the same place, possibly with undesired results.

Figure 10–20 Opaque-Line Drawing Sequence



10.2.10 Transparent-Line Mode

In the transparent-line mode, a PCI write to the frame buffer address space draws a masked, 16-pixel solid-line segment starting at the specified address. The 21130 draws the line segment as a function of the parameters listed in Table 10–19.

```
Transparent Line (Frame Buffer Address, Frame Buffer Address <1:0>, Line Mask,
                  Raster Op, Byte Mask, Foreground, Address Increment 1,
                  Address Increment 2, Error Increment 1, Error Increment 2,
                  Initial Error, Length, Destination Bitmap, Cap Ends);
```

Table 10–19 Transparent-Line Mode Parameters

Parameter	Source		Section
`Frame Buffer Address`	PCI write address	—	—
`Frame Buffer Address <1:0>` `Line Mask`	PCI write data	<17:16> <15:0>	—

`Byte Mask`
`Destination Bitmap`
`Raster Op`
	Raster operation register	GOPR <19:16>
		GOPR <10:8>
		GOPR <3:0>
			8.5.9

`Foreground`	Foreground register	GFGR <31:0>	8.5.8
`Address Increment 1` `Error Increment 1`	Bresenham 1 register	GB1R <31:16> GB1R <15:0>	8.5.11
`Address Increment 2` `Error Increment 2`	Bresenham 2 register	GB2R <31:16> GB2R <15:0>	8.5.12
`Initial Error`	Bresenham 3 register	GB3R <31:16>	8.5.13
`Length`		GB3R <3:0>	
`Cap Ends`	Mode register	GMOR <15>	8.5.1

The transparent-line mode works in the same way as the opaque-line mode (Section 10.2.9), and is similarly more efficient when operations are initiated by writing a slope register rather than the frame buffer. Transparent-line mode differs in that `Line Mask` determines whether the `Foreground` color is written (`Line Mask` bit = 1) or write is disabled (`Line Mask` bit = 0), rather than determining whether foreground or background color is written.

11
Programming

This chapter contains additional programming information. It includes information about the DECchip 21130 configuration firmware, graphics drivers and servers, video support functions, and functions to support Alpha systems.

11.1 PCI Configuration Firmware

The 21130 hardware implements the full set of required PCI configuration registers, and is fully configurable by generic PCI-compliant system firmware. The 21130 is not limited to motherboard applications, but behaves as a generic plug-and-play PCI option for all PCI-compliant systems independent of operating system. (Section 11.12 addresses systems that require dedicated support for the 21130 in the base system firmware.)

11.1.1 Device Address Mapping

Configuration firmware can map the 21130 device and enable response to that mapping by manipulating fields in the PCI device base address registers (PDBR0 and PDBR1, Section 8.2.5) and PCI command and status register (PCSR, Section 8.2.2). Table 11–1 describes the fields to be manipulated.

Table 11–1 21130 Base Address and Memory Space Enable Fields

Field	Register	Bits	Field Description
Device base address	PDBR0 PDBR1	<31:4> <31:4>	The PCI memory address defined as base address 0 and 1 of the 21130 address space.
Memory space enable	PCSR	<1>	When set, enables the 21130 to respond to memory space accesses.

The PCI device base address and command and status registers are written in the following sequence:

11.1 PCI Configuration Firmware

1. Configuration firmware probes the device base address registers to determine where the 21130 is to be mapped and the amount of space to allocate to it; that is, firmware writes all ones to the registers and then reads back the value. The 21130 returns the following values:
   - PDBR0:
     - <24:4> = 000000_{16} to indicate that base address 0 must be aligned to 32MB or greater.
     - <0> = 0 to indicate that this address space must be mapped to PCI memory space.
   - PDBR1:
     - <20:4> = 00000_{16} to indicate that base address 0 must be aligned to 2MB or greater.
     - <0> = 0 to indicate that this address space must be mapped to PCI memory space.
2. Firmware allocates:
   - 32MB of naturally aligned PCI memory space and writes the base address 0 MSBs to PDBR0 <31:25>
   - 2MB of naturally aligned PCI memory space and writes the base address 1 MSBs to PDBR1 <31:21>
3. Firmware sets the memory space enable bit (PCSR <1>) to enable device response. (Usually, memory space enable should not be set until PDBR0 and PDBR1 have been properly initialized as described in steps 1 and 2.)

After the PDBR0 and PDBR1 are written and memory space enable is set in the PCSR, the 21130 can respond as a normal PCI target (Section 9.3).

11.1.2 Bus Mastering

The 21130 supports DMA operations to rapidly transfer image data from PCI-accessible memory to display memory. To invoke 21130 DMA operations, the 21130 must be able to master the PCI bus. Configuration firmware must write fields in the PCI latency timer register (PLTR, Section 8.2.4) and PCSR, to enable the 21130 to be a PCI bus master. Table 11–2 describes the fields to be written.

Table 11–2 PCI Latency Timer and Bus Master Enable Fields

Field	Register	Bits	Field Description
Latency timer	PLTR	<15:8>	21130 bus ownership is limited to the number of PCI clocks specified in this field.
Bus master	PCSR	<2>	When set, enables the 21130 to become bus master. It must be set to enable DMA operations, but should not be set until the PLTR is initialized.

DMA operations usually involve a long (hundreds of bytes) burst transfer. Therefore, a high latency-timer value helps improve performance. However, the benefit of a high latency-timer value depends on the PCI bridging structure, and is limited, for example, by PCI bridges that terminate transfers on cache-line boundaries.

11.1.3 Interrupt Routing

Configuration firmware is also responsible for mapping system interrupt lines to PCI devices that require interrupt services (as does the 21130). After the interrupt lines are mapped, configuration firmware must write the routing information to the interrupt line field in the PCI interrupt line register (PILR <7:0>, Section 8.2.7). During subsequent normal graphics operation, display drivers or the operating system can determine interrupt vectors and priorities either by reading the line interrupt registers or through the GET_DEVICE_INTERRUPT BIOS routine.

11.1.4 Expansion ROM

The 21130 supports an external EEPROM that conforms to PCI expansion ROM specifications. See the *PCI Local Bus Specification, Revision 2.0* for more information.

11.2 Mode Switching

Sections 11.2.1 and 11.2.2 describe programming considerations for switching the 21130 from VGA mode to 2DA mode and from 2DA mode to VGA mode.

11.2.1 VGA-to-2DA Mode Switching

Digital recommends the following procedure to efficiently switch the 21130 from VGA mode to 2DA mode. The first five steps place the VGA cell in a known state, and then halt video and graphics memory cycles issued by the VGA cell. The remaining steps set up 2DA video, switch the memory pins to the 2DA memory controller, and enable 2DA video.

1. Call the video BIOS to set up mode 3.

2. Clear the synchronous reset bits (1:0) in the reset register (VSRESR <1:0>, Section 8.12.3). The VSRESR is index 0 in the VGA sequencer register set.
3. Clear the VGA enable bit (<22>) and update the mode 32 bit (<20>) as appropriate in the graphics deep register (GDER, Section 8.5.2).
4. Set the desired pixel clock values and set the VGA variable dot clock select bit (<0>) in the clock control A register (VXCKAR, Section 8.14.10).
5. Set the synchronous reset bits (1:0) in the reset register (VSRESR <1:0>).
6. Load the 21130-specific video control and format registers (Sections 8.7 through 8.8.5).
7. Set the DAC resolution select bit in the palette and DAC command register 0 (DCOR0 <1>, Section 8.9.7) to enable 8-bit DAC resolution.
8. Set the video valid bit in the video valid register (VIVVR <0>, Section 8.7.2). Active video will start on the next top-of-frame.

Frame buffer data is not preserved across the mode switch. Digital recommends that the software doing the mode switch initialize the frame buffer after the switch is completed.

11.2.2 2DA-to-VGA Mode Switching

Digital recommends the following procedure to efficiently switch the 21130 from 2DA mode to VGA mode. The first step halts video cycles issued by the 2DA. The remaining steps switch the memory controller pins to the VGA memory controller and enable VGA operation.

1. Clear the video valid bit (<0>) and blank bit (<1>) in the video valid register (VIVVR, Section 8.7.2).
2. Read the video valid register and wait for synchronized video valid to deassert (VIVVR <8>).

   The video back end is idle.
3. Read the command status register and wait for the busy bit (MCSR <0>, Section 8.3.1) to deassert.
4. Set the mode-32 bit in the graphics deep register (GDER <20>, Section 8.5.2).
5. Reset the video control and format registers (Sections 8.7 through 8.8.5) to their initial power-up states.
6. Clear the synchronous reset bits (1:0) in the reset register (VSRESR <1:0>, Section 8.12.3). The VSRESR is index 0 in the VGA sequencer register set.

7. Clear the VGA variable dot-clock select bit in the clock control A register (VXCKAR <0>, Section 8.14.10).
8. Clear the screen-off bit in the clocking mode register (VSCMOR <5>, Section 8.12.4). The VSCMOR is index 1 in the VGA sequencer register set.
9. Clear the DAC resolution select bit in the palette and DAC command register 0 (DCOR0 <1>, Section 8.9.7) to enable 6-bit DAC resolution.
10. Read the command status register and wait for the busy bit (MCSR <0>) to deassert.
11. Set the VGA enable bit in the graphics deep register (GDER <22>).
12. Read the command status register and wait for the busy bit (MCSR <0>) to deassert.
13. Set the synchronous reset bits (1:0) in the reset register (VSRESR <1:0>).
14. Call the video BIOS to set the desired mode.

Frame buffer data is not preserved across the mode switch. Digital recommends that the software doing the mode switch initialize the frame buffer after the switch is completed.

11.2.2.1 Expected 2DA Operation During VGA Mode

The graphics engine and memory controller are not disabled during VGA mode. This means that any attempted 2DA graphics operations will complete, but actual frame buffer operations will not be done. This behavior prevents inadvertent programming from hanging the 21130.

11.3 Bit-Block Transfers

Bit-block transfers (BitBlts) can be implemented as screen-to-screen copies and host-to-screen copies.

11.3.1 Screen-to-Screen Copy

For high performance, screen-to-screen copy is the most important function to accelerate. The 21130 copy mode, 64-bit memory port, and 64-byte copy buffer all contribute to the high speed of screen-to-screen copies.

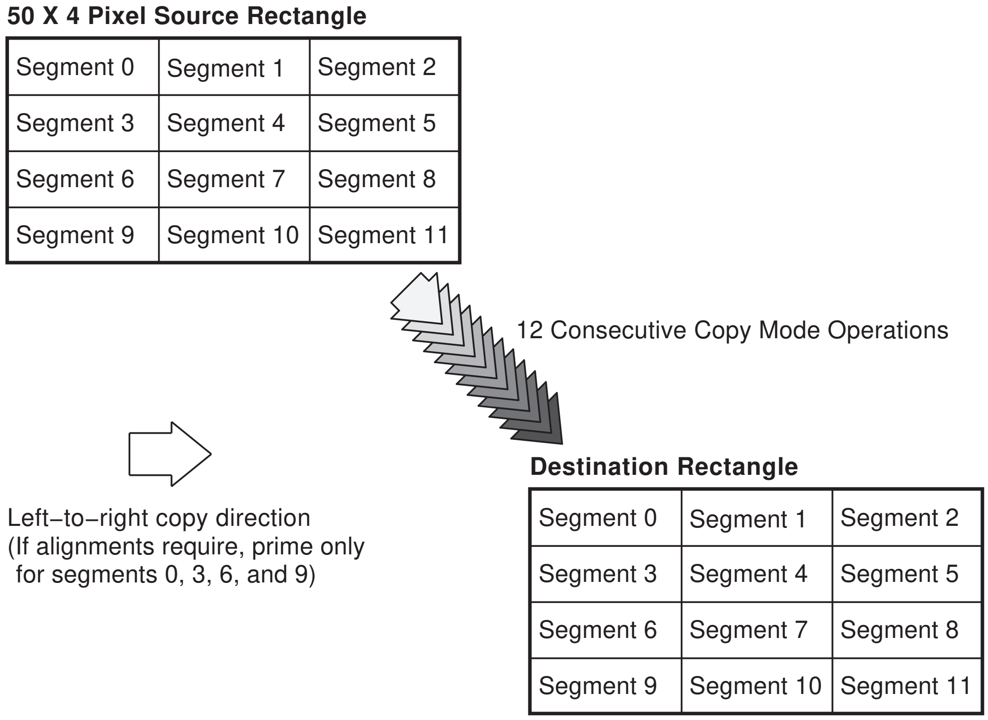
Typically, driver-level calls move a rectangular source region to a destination region, and, possibly, apply a Boolean raster operation to the source and destination. Because the copy mode (Section 10.2.6) supports only span copies, software must break the rectangle into as many individual spans as necessary, with the width of each span equal to the width of the rectangle. Furthermore, it must break each arbitrary-width span into as many individual segments as necessary, with the length of each segment equal to 16, 32, or 64 pixels, depending on the frame buffer, bitmap, and masking used (Figure 11–1).

For overlapping source and destination spans, software must choose the proper copy direction (right-to-left or left-to-right), so that the source is not corrupted before it is read. The copy mode supports both directions, and maintains the internal state of the residue register for unaligned copies. Therefore, software must prime the residue register for only the first span segment (if necessary), rather than for each segment copy. Priming for subsequent segments occurs on the last read of the previous segment.

For the most efficient copying of 8-bpp spans, the copy-64 source and destination registers (GCSR and GCDR, Section 8.4.4) use the entire copy buffer. Although the GCSR and GCDR can copy only aligned, unmasked span segments, they can be used to copy the interior of a large unaligned copy, where the left and right edges are copied by direct writes to the frame buffer.

Figure 11–1 shows how an arbitrary rectangle can be broken into segments and where priming and flushing occur, if necessary.

Figure 11–1 BitBlt Using Copy Mode Example



The 21130 provides 16 raster-operation encodings to support the full set of 2-operand Boolean operations specified by X and OpenGL, but not the full set of Win32 graphics operations. (Win32 supports 256 ternary operations, two of which can be specified in a particular operation by a mask operand.) However, the 21130 raster operation encodings do include the most commonly used Win32 Boolean operations (such as srccopy and patcopy). (See the raster operation register description, Section 8.5.9.) Therefore, under Win32, if the Boolean operation passed in the DrvBitBlt call is not supported by the 21130, it can be broken into supported operations (if possible and desirable), or handled by the graphics device interface (GDI). Note that handling unsupported raster operations is not specific to BitBlts — raster operations are called into every Win32 device-driver interface (DDI) graphics call.

11.3.2 Host-to-Screen Copy

An image or bitmap can be copied between host memory and the 21130 frame buffer with X PutImage or GetImage calls or a Win32 DrvCopyBits call. Ideally, the DMA-read copy mode can be used. If DMA-read copy mode cannot be used, the image or bitmap can be burst-written directly into the 21130 frame buffer space using simple mode and standard programmed I/O. A final (and likely slower than simple mode) option is to write the copy buffer in standard programmed I/O, then unload the copy buffer with a write to the GCDR.

To use the DMA-read copy mode, the source rectangle must be broken into spans (as in the case of local frame buffer copies). The residue register is primed and flushed as necessary, with appropriate address and pixel-count values. (See Section 10.2.7 for more information about the DMA-read copy mode.)

11.3.3 Scaled-Copy

In the scaled-copy mode, as in the DMA-read copy mode, the 21130 does a DMA operation to transfer a span from host memory to its frame buffer. However, the scaled-copy mode includes the following functions that can be applied to the source span during the transfer:

- Arbitrary up or down scaling
- Support for YUV source formats
- Sharpening and smoothing filters
- RGB color depth conversion
- Arbitrary remapping of 8-bpp indices

See Sections 11.4, 11.5, and 10.2.8 for more information.

11.4 Dither Mathematics

The proprietary dithering logic in the 21130 is inherited from the DECchip 21030, with minor modifications for different output quantizations. The dithering operation is a two step process:

1. The input value is scaled by a simple shift and subtract operation. The scaling is performed in order to range-limit the dither output according to the desired output quantization.
2. A noise value obtained from a lookup into a fixed 32×32 ROM matrix is added to the scaled input. Lookups into the noise matrix are coordinated so as to tile the 32×32 matrix across the target image.

The output of the dithering process is the most significant *n* bits of the noise addition result. In the 21130, output quantizations (values of *n*) of 2, 3, 4, 5, and 6 are used.

The following pseudo-code represents the exact operations the 21130 performs during the dithering process, for different output quantizations. In the 21130, the input value is 8 bits and noise values are 7 bits. During the subtraction and addition, 9 bits of intermediate result are carried.

```
n=2: {dataOut(1:0),ignored(6:0)} =
     {dataIn(7:0),1'b0} - ({dataIn(7:0),1'b0}>>2) + (matrix(6:0) >> 0);
n=3: {dataOut(2:0),ignored(5:0)} =
     {dataIn(7:0),1'b0} - ({dataIn(7:0),1'b0}>>3) + (matrix(6:0) >> 1);
n=4: {dataOut(3:0),ignored(4:0)} =
     {dataIn(7:0),1'b0} - ({dataIn(7:0),1'b0}>>4) + (matrix(6:0) >> 2);
n=5: {dataOut(4:0),ignored(3:0)} =
     {dataIn(7:0),1'b0} - ({dataIn(7:0),1'b0}>>5) + (matrix(6:0) >> 3);
n=6: {dataOut(5:0),ignored(2:0)} =
     {dataIn(7:0),1'b0} - ({dataIn(7:0),1'b0}>>6) + (matrix(6:0) >> 4);
```

Dither Phases

Dither noise from different locations in the 32 $\times$ 32 matrix are applied to the RGB (VYU) channels. For hardware simplicity, the same row (*y*-direction) index is used for all three channels. However, the column indices (*x*-direction) use phase offsets 0 (R/V), 8 (G/Y), and 16 (B/U).

11.5 Scaling Filters

The 21130 supports two 3-tap filters for the Y/G channel: a smoothing filter, with coefficients 0.5, 0.0, and 0.5; and a sharpening filter, with coefficients –0.5, 2.0, and –0.5. The sharpening filter is clamped to prevent overflow and underflow. The following pseudo-code represents the exact operations the 21130 performs in applying the 3-tap filters.

```
smoothing:
(0.5, 0, 0.5):  {dataOut(7:0),ignored} =
pixelLeft(7:0) +  pixelRight(7:0);
```

```
sharpening:
(-0.5,2,-0.5):  {dataOut(7:0),ignored} =
(4*pixelCenter(7:0) - pixelLeft(7:0) - pixelRight(7:0) < 0) ?
9'b000000000 :
(4*pixelCenter(7:0) - pixelLeft(7:0) - pixelRight(7:0) > 511) ?
9'b111111111 :
(4*pixelCenter(7:0) - pixelLeft(7:0) - pixelRight(7:0));
```

Special treatment is given to the first and last pixels of the source and destination spans. For these pixels, the filters are forced to operate in pass-through mode. More specifically, the first and last pixels in the source span after the application of the `Start Pixel` and `End Pixel` values pass through the prescaling filter unmodified; and the first and last pixels in the destination span after scaling, but before the application of the `Draw#` value, pass through the postscaling filter unmodified. All other pixels are modified according to the current filter settings.

11.6 Overlays

Sections 11.6.1 through 11.6.3 describe the use of overlays.

11.6.1 Flicker-Free Monochrome Overlay Support

Certain multimedia applications combine text, graphics, and video in a single window. Typically, this is done by specifying a base image (perhaps a video frame) along with an overlay image (that is, text or graphics). The two images are combined to form the image displayed in the window. This style of overlay support is included in a recent Microsoft and Intel display control interface (DCI) specification.

In a 21130 system, the process of rendering with an overlay is likely to be done as follows:

1. One (or multiple) scaled-copy operations transfer base image spans to the target window.
2. Simple, stipple, or other graphics modes selectively place pixels from the overlay image on top of the base image spans.

Because these two operations do not occur instantaneously, it is possible that the screen refresh process will ‘‘snapshot’’ the target window spans before they are fully updated. This can cause pixel color errors that, if they occur frequently enough, cause motion sequences to ‘‘flicker’’ or ‘‘sparkle.’’

To provide a mechanism by which software can prevent overlay flicker or sparkle, the 21130’s current screen refresh address is readable (VFCRR, Section 8.8.4). By checking the screen refresh address before rendering video and overlay spans, software can avoid issuing commands at a time likely to collide with the screen refresh operation. Unlike most reads (which are stalled until the chip goes idle), reads of the screen refresh address are serviced immediately.

11.6.2 True Monochrome Overlay with Pixel Occlusion Bitmap Hardware

The 21130's pixel occlusion bitmap hardware can be used as a true monochrome overlay surface (as well as specifying regions of inside and outside pixel format — see Section 8.8.1.1). This overlay mechanism provides a true overlay for 8-bpp depths. (True overlay surfaces for 16-bpp and 32-bpp pixel depths can be implemented using the mechanism described in Section 11.6.3.)

Note

The pixel occlusion bitmap hardware cannot be used to select between RAM and ROM LUTs at the same time that it is being used as a monochrome overlay. When using the pixel occlusion bitmap hardware as a monochrome overlay, software must coordinate the sharing of a single LUT by graphics and video applications.

To implement the monochrome overlay function, the onchip palette and DAC is forced to display cursor color 3 when a lit overlay pixel is encountered. This imposes the following limitations on the use of the monochrome overlay.

- The overlay is merged with the graphics pixel stream only after the stream has passed through the palette and DAC. Therefore, modes that output pixels to the VAFC before palette and DAC processing cannot use the overlay.
- Because cursor color 3 is the overlay color, and to maintain hardware simplicity, the monochrome overlay is supported only when the cursor mode is 00_2 (cursor disabled) or 10_2 (MS Windows mode). Cursor mode is set in the cursor mode register (CMOR <1:0>, Section 8.6.1).

To use the pixel occlusion bitmap hardware as a monochrome overlay:

- Set the pixel occlusion bitmap mode bit in the video pixel format register (VFPFR <13>, Section 8.8.1).
- Set the cursor mode to disabled or MS Windows in the CMOR.

Table 11–3 shows the displayed pixel value for all combinations of monochrome overlay and cursor data. The first two table entries apply to cursor disabled mode and the remainder of the table assumes MS Windows mode is specified. Note that the table shows the palette and DAC cursor mode (not necessarily the same as the value programmed in the CMOR).

Table 11–3 Cursor Color Displayed with Monochrome Overlay

Pixel* Occlusion Bitmap Value	Cursor* Data	Cursor† Mode	Cursor† Data	Color Displayed
0	Not in range	00	Don't care	Palette data
1	Not in range	01	11 (color 3)	Overlay color
0	00 (color 1)	10	00 (color 1)	Cursor color 1
0	01 (color 2)	10	01 (color 2)	Cursor color 2
0	10 (transparent)	10	10 (transparent)	Palette data
0	11 (invert)	10	11 (invert)	Inverse palette data
1	00 (color 1)	10	00 (color 1)	Cursor color 1
1	01 (color 2)	10	01 (color 2)	Cursor color 2
1	10 (transparent)	01	11 (color 3)	Overlay color
1	11 (invert)	01	11 (color 3)	Overlay color

*From frame buffer
†To palette and DAC

11.6.3 True 8-bpp Overlay in 16-bpp or 32-bpp Frame Buffers

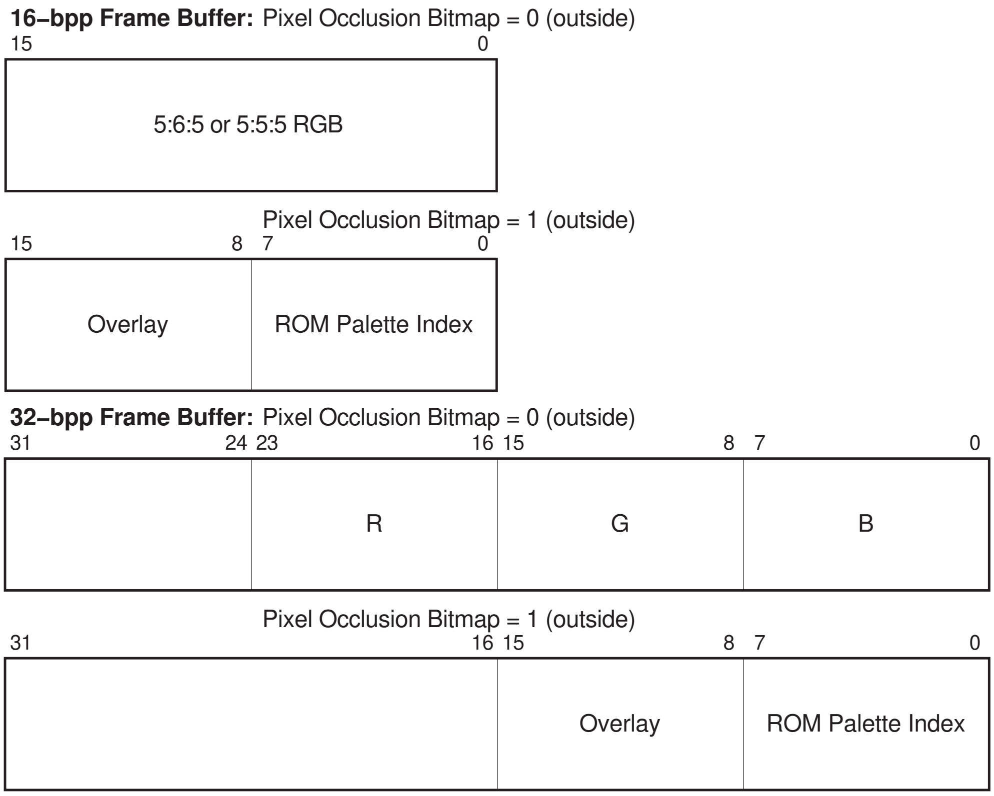
The 21130's ability to display high-quality video in 8 bits enables it to easily provide a superior 8-bpp overlay in 16-bpp and 32-bpp frame buffer depths. The 8-bpp overlay can be used to implement the DCI's chroma-keyed overlay surfaces.

For pixels with a pixel occlusion bitmap bit value of 1 (indicating that the pixel is part of a video window), the ROM palette index is displayed through the ROM LUT if the overlay value matches the 00_{16} chroma-key. If the overlay is not 00_{16}, the overlay value is displayed through the RAM LUT. Pixels with a pixel occlusion bitmap bit-value of 0 (pixels not part of the video window) are displayed as 16- or 24-bit direct color (the LUTs are not used).

The 21130 byte mask (GOPR <19:16>, Section 8.5.9) can be used to select the byte within the 16-bpp and 32-bpp visual to which the video pipeline writes the ROM LUT indices. The memory controller can detect the byte mask and, consequently, does not need to do read-modify-write operations to write the pixels.

Figure 11–2 shows the arrangement of overlay data in 16-bpp and 32-bpp frame buffers.

Figure 11–2 Overlay Data in 16-bpp and 32-bpp Frame Buffers



11.7 Fills

Sections 11.7.1 through 11.7.3 describe filling, stippling, and tiling functions.

11.7.1 Solid

A region can be solid-filled with X FillSpan or PolyFillRect calls, or under Win32 with a DrvPaint call and a solid brush. The best way to do a solid fill is to use the transparent-fill or opaque-fill mode. The specific mode used depends on the following conditions:

- Fill region size

11.7 Fills

- Required raster operation
- Destination bitmap

Because the transparent-fill mode only fills spans, software must do the following:

1. Break the fill region into spans no longer than 2K pixels.
2. Replicate the solid color as necessary across the foreground register (GFGR, Section 8.5.8).
3. Write the frame buffer as many times as necessary to fill the spans.

(The fill modes are described in Sections 10.2.4 and 10.2.5.)

11.7.2 Stippling or Filling with a Monochrome Brush

When stippling or filling with a monochrome brush, a 1-bpp bitmap is expanded into a foreground (and optionally, background) color to tile a solid or bitonal pattern across a region. The opaque-stipple or opaque-fill mode can be used, depending on the following conditions:

- Size of the fill region
- Number of pixels at which the pattern repeats
- Raster operation
- Destination bitmap
- Masking

Filling a region with a 4×4 or 8×8 monochrome brush is a common Windows operation. The pattern must repeat at intervals of 2^i and $i \leq 5$, and the foreground and background color must be specified.

If none of the fill modes can be used or the region is very small, the opaque-stipple (or transparent-stipple) mode can be used, in conjunction with the foreground and background registers.

11.7.3 Tiling or Filling with a Non-Monochrome Brush

When tiling or filling with a non-monochrome brush, a tile or brush pattern that is the same depth as the destination bitmap is repeated across the fill region. The width and number of colors in the pattern can be arbitrary.

The copy mode or DMA-read copy mode can be used to recopy the same pattern from off-screen memory or main memory, respectively, to the destination as many times as necessary.

In the DMA-read copy mode, the 21130 can read more than 100 MB/s from the PCI bus. It can write the frame buffer at approximately the same rate, depending on the length of the copy. Therefore, the DMA-read copy mode can theoretically tile (brush) at approximately 100 MB/s; however, the actual rate in a specific system implementation varies as a function of the PCI bus performance (that is, latency, burst lengths, use, and so on). By comparison, the standard copy-mode fill rate is more than 50 MB/s. The simple mode can also be used, and might be the best choice for small regions.

11.8 Lines

The X PolyLine or PolySegment calls or the Win32 DrvStrokePath call can request 2D lines. The 21130 can draw lines in either of the following ways:

- Standard mechanism — software initializes the Bresenham terms and then writes the frame buffer to initiate the drawing operation.
- Alternate mechanism — software writes a slope register (GSLR<7:0>, Section 8.4.1) and the 21130 automatically generates the Bresenham terms and initiates the drawing operation.

Drawing with the GSLRs is preferred because it is significantly faster than drawing lines using the standard mechanism. In either case, the continue register (GCTR) can be used to extend lines to an arbitrary length.

Typically, all X lines can be drawn using the GSLRs. Conversely, the GSLRs cannot always be used to draw Win32 lines.

11.8.1 Line Drawing Under X

The following sequences list the steps for drawing various types of 2D lines under X.

- **Solid or Bitonal Lines**
  1. Set the mode to opaque- or transparent-line mode, as desired.
  2. Set the foreground and background colors in the foreground register (GFGR) and background register (GBGR, Section 8.5.8).
  3. Write the starting address to the address register (GADR, Section 8.5.6).
  4. Initialize the data register (GDAR, Section 8.5.7) to XXXXFFFF to draw all pixels (X = unused).
  5. Write the appropriate GSLR.
  6. Use the GCTR to extend the line to the desired length.

11.8 Lines

- **Patterned or Styled Lines**

  Do the solid or bitonal lines sequence described previously, but write the desired pattern, rather than XXXXFFFF, to the GDAR.

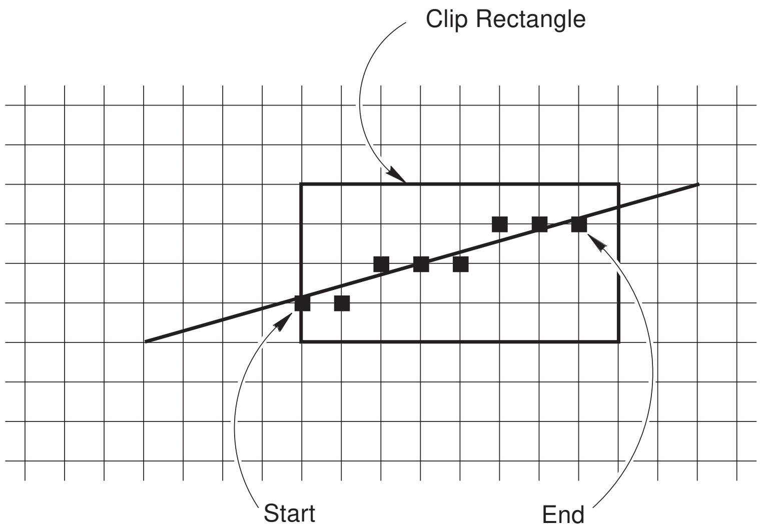
- **Connected Lines**

  Do the solid or bitonal lines sequence described previously, but do not write to the GADR. The 21130 will draw the new line starting 1 pixel beyond the end of the previous line.

- **Clipped Lines**

  Figure 11–3 shows a clipped line drawn through a clipping rectangle.

Figure 11–3 Drawing Clipped Lines



1. Write slope-no-go register 7 (GSNR7, Section 8.5.3) to draw in octant 7.
2. Write the starting pixel address to the GADR.
3. Write the initial error and line length to the Bresenham 3 register (GB3R).
4. Write the GCTR to draw the line (and repeat for rectangle wider than 16 pixels).

5. Repeat steps 2 through 4 for each rectangle in the clip list.

11.8.2 Line Drawing Under Win32

The coordinate constructs supported by Win32 do not allow the GSLRs to be used to draw all lines that the GDI might request from the display driver. To improve the appearance of rendered lines, Win32 supports subpixel coordinates. Each coordinate is in the 28.4 format (28 integer bits and 4 fraction bits). The coordinate system can be visualized as a grid in which an endpoint can reside at any grid intersection, but pixels reside at every sixteenth pixel in both X and Y. (For more information, see the Win32 device-driver kit documentation.) The following is a more detailed description of this ‘‘problem’’ and its ‘‘solution.’’

Problem

On a write to a GSLR, the hardware sets up the line; that is, it translates the absolute *dx* and absolute *dy* information into the Bresenham address and error increment values and initial error term. However, absolute *dx* and absolute *dy* are 16-bit quantities, assumed to be 16 bits of integer and 0 bits of fraction. This presents two problems:

- The setup hardware cannot properly correct a subpixel endpoint to the pixel centers.
- Lines passed by the GDI can be too long for the Bresenham engine to render without the risk of introducing error in the digital differential analyzer (DDA).

The first problem is complex. Determining the starting pixel can be critical, and depends on the location of the endpoint in the subpixel grid. For some endpoints, the first pixel drawn is the pixel closest to the intersection of the geometric line and the first major-axis grid, rather than the pixel nearest to the specified endpoint. That is, the first pixel drawn can be a function of the starting subpixel endpoint and the slope.

The 21130 setup hardware cannot handle such constructs. It cannot correctly choose the address of the proper endpoint and cannot calculate the proper Bresenham initial error term. The initial error generated would be relative to the first subpixel rather than the first real pixel, which can be up to 16 subpixels away.

Secondly, if the 21130 could properly correct the starting address error term, the 21130 DDA, with 16 bits of resolution, cannot draw lines greater than 64K pixels in major-axis length with guaranteed accuracy. Consequently, the GSLRs cannot be used to draw any line with endpoints having nonzero fractional components or any line longer than 64K pixels.

11.8 Lines

Solution

To support all possible lines requested by the Win32 GDI, the 21130 Win32 display driver must use a combination of GSLR accesses, direct manipulation of the Bresenham registers, and writes to the frame buffer. The following is a suggested strategy for dealing with an arbitrary GDI line drawing request:

1. If the line is less than 64K pixels in the major axis, go to step 2. Otherwise, do either of the following:
   - Default to the GDI.
   - Break long lines into smaller segments and go to step 2.
2. Screen for endpoints with nonzero fractional components.

   For integer endpoints, draw by writing the GSLR, passing a 16.0 format value for absolute *dx* and absolute *dy*.

   For noninteger endpoints, do the following (and refer to the Win32 device-driver kit documentation for more detail):

   a. Determine the starting pixel and calculate the address.

   b. Write a GSNR to calculate the address and error increment terms (that is, the parameters in the GB1R and GB2R) passing 12.4 format values for absolute *dx* and absolute *dy*. The setup process will calculate these terms correctly, regardless of the number of fractional bits.

   c. Adjust the initial error term relative to the starting pixel. This can be done by performing the DDA at subpixel increments until the first major-axis grid is reached (which might be necessary in any case) and scale the error term. Write the error term to the GB3R initial error field.

   d. Write the address of the starting pixel to the GADR and draw the first 16-pixel segment with a write to the GCTR. Repeat this step for lines longer than 16 pixels.

In effect, this operation appears to be a clipped line with the edge of the clipping rectangle set at the first integer major-axis grid crossed by the geometric line.

11.9 Text

The 21130 stipple modes can process a request for any of the X text or glyph calls or the Win32 DrvTextOut call. The opaque-stipple or transparent-stipple mode can be used, depending on the following:

- The destination bitmap
- Whether a nontrivial raster operation is required
- Whether the text foreground is filled with a solid, monochrome, or arbitrary patterned brush or tile

Transparent-stipple mode is used for a solid brush, with the glyph mask specified as the stipple mask. Opaque-stipple mode is used for a monochrome brush, with the glyph mask specified as the pixel mask. In either case, if a mix raster operation is specified for the foreground under Win32, each raster operation requires two passes using transparent-stipple mode. For an arbitrarily patterned brush (that is, other than simple monochrome) that does not repeat at appropriate intervals, simple mode can be used to write the glyph foreground through the glyph mask.

All stipple modes allow up to 32 pixels to be drawn per operation. Therefore, it is advantageous to try to group spans from multiple glyphs that are contiguous in display memory. For example, rather than draw four 8×16 glyphs one at a time, draw all four in parallel, one span at a time — one write can draw one span from each glyph at the same time.

11.10 Repeat Loop Examples

The following are several repeat-loop templates. Italics indicate parameters that should be adjusted to tailor the template for the particular Blt to be performed.

Monochrome or Bitonal Brush Fill

This sequence fills a rectangular area based on colors in the foreground and background registers and the pattern in the data register.

Alias*	Register	Value
(Miscellaneous setup)		
0	Address	*Byte address of top-left corner of destination rectangle*
0	Repeat begin	*Height* (in scanlines)

*Base address 0 register space alias (Section 7.5.1.2)

11.10 Repeat Loop Examples

Alias*	Register	Value
0	Continue	*Width* (in pixels)
1	Address	*Virtual screen width* (in bytes)
0	Repeat end	—

*Base address 0 register space alias (Section 7.5.1.2)

8 × 8 8-bpp Pattern Fill

This sequence fills a rectangular area with an 8 × 8 8-bpp pattern that was previously cached in off-screen memory.

Alias*	Register	Value
(Miscellaneous setup)		
0	Mode	Copy mode, 8-bpp
0	Copy-64 source	*Byte address of cached pattern*
0	Dither row	*(Top-left Y coordinate of destination rectangle) MOD 8*
0	Address	*Byte address of top-left corner of destination rectangle*
0	Mode	Extended-pattern fill mode, 8-bpp
0	Repeat begin	*Height* (in scanlines)
0	Continue	*Width* (in pixels)
1	Address	*Virtual screen width* (in bytes)
1	Dither row	01_{16}
0	Repeat end	—

*Base address 0 register space alias (Section 7.5.1.2)

Scaled Video DMA (Magnification Only)

This sequence transfers an image from host memory to a window within the frame buffer. Vertical Bresenham interpolation is used to vertically magnify the image.

Alias*	Register	Value
(Miscellaneous setup)		
0	DMA base address	*PCI byte address of top-left corner of source rectangle*
0	Data	*Initial error for vertical scaling*

*Base address 0 register space alias (Section 7.5.1.2)

Alias*	Register	Value
0	Address	*Byte address of top-left corner of destination rectangle*
0	Repeat begin	*Height of destination rectangle* (in scanlines)
0	Continue	*Source rectangle width* (in Dwords)
5	DMA base address	*Source rectangle stride* (in bytes)
5	Data	*Vertical error increment 1*
7	Data	*Vertical error increment 2*
0	Repeat end	—

*Base address 0 register space alias (Section 7.5.1.2)

8-bpp Screen-to-Screen Copy

This sequence performs a screen-to-screen copy of a rectangular region. The source and destination rectangle widths must be less than 320 pixels. Height is arbitrary.

Alias*	Register	Value
0	Address	*Byte address of top-left corner of source rectangle*
0	Repeat begin	*Height* (in scanlines)
0	Continue	FFFFFFFF
1	Address	*Byte offset = destination top-left − source top-left*
0	Continue	*Start mask*
1†	Address	*Byte offset = source top-left − destination top-left*
0†	Copy-64A source	00000000_{16}
1†	Address	*Byte offset = destination top-left − source top-left*
0†	Copy-64A destination	00000000_{16}
1	Address	*Byte offset = source top-left − destination top-left*
0	Continue	FFFFFFFF
1	Address	*Byte offset = destination top-left − source top-left*
0	Continue	*End mask*
1	Address	*Byte offset required to get to next source line*
0	Repeat end	—

*Base address 0 register space alias (Section 7.5.1.2)

†Repeat these register writes as necessary to reach the desired rectangle width.

11.11 Video Registers

Sections 11.11.1 through 11.11.3 describe programming considerations for the video control and video format registers described in Sections 8.7 through 8.8.5.

11.11.1 Modifying the Contents of the Video Registers

There are three categories of video registers (Table 11–4) in terms of when software can modify their contents:

- Fully shadowed and pseudo-shadowed registers

  Software can write these registers at any time; however, the programmed value is not effective until the beginning of the next frame. The fully shadowed and pseudo-shadowed registers are distinguished by the hardware shadowing mechanism — the difference is transparent to software.

- Video-disabled

  These registers should be written only when the video valid bit in the video valid register (VIVVR <0>, Section 8.7.2) is clear. Changing the registers during active video might create artifacts on the screen.

Table 11–4 lists video registers and their type according to programming class.

Table 11–4 Fully Shadowed, Pseudo-Shadowed, and Video-Disabled Registers

Register Name	Mnemonic	Type
Cursor mode	CMOR	Video disabled
Cursor base address	CCBR	Pseudo shadowed
Cursor XY	CXYR	Fully shadowed
Video base address	VIVBR	Pseudo shadowed
Video valid	VIVVR	Fully shadowed ❶
Video scanline increment	VISIR	Video disabled
Video line width	VILWR	Video disabled
Video pixel format	VFPFR	Video disabled ❷

(continued on next page)

Table 11–4 (Cont.) Fully Shadowed, Pseudo-Shadowed, and Video-Disabled Registers

Register Name	Mnemonic	Type
Video pixel occlusion bitmap base address	VFOBR	Pseudo shadowed
Video pixel occlusion bitmap current address	VFOAR	Read Only
Video current refresh address	VFCRR	Read Only
Alternate video control	VFAVR	Video disabled

Notes for Table 11–4

❶ Some VIVVR bits are in the video-disabled programming class. When such bits need to be changed, software should:

1. Write the VIVVR to turn off video.
2. Poll the VIVVR until the synchronized video valid bit (<8>) signals that video is inactive.
3. Modify the VIVVR bits that require video disabled.

❷ All of the VFPFR bits are in the video disabled class except the pixel occlusion bitmap enable bit (<12>) which is fully shadowed. When changing VFPFR <12> while video is enabled, the other VFPFR bits must be written with their current value to eliminate any unwanted side effects.

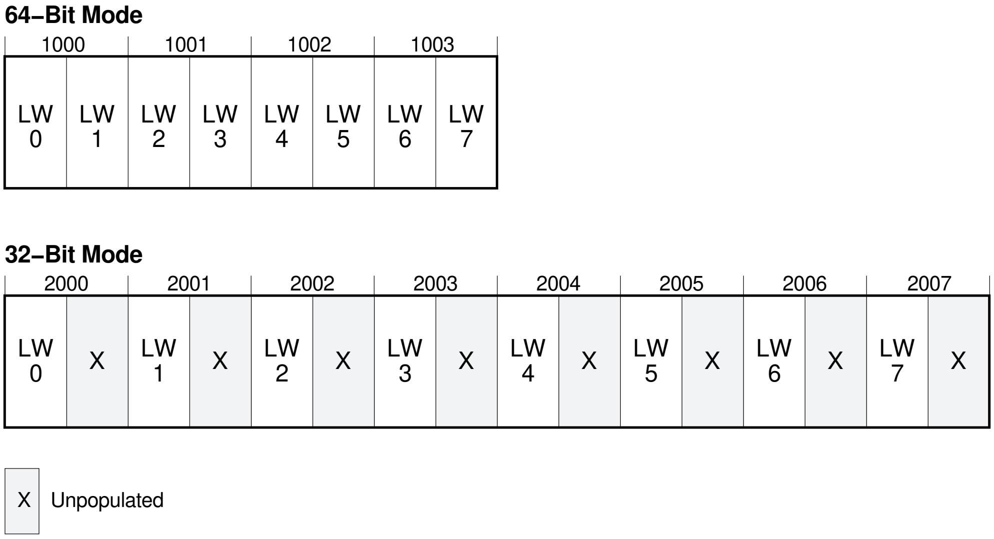
11.11.2 Video Registers in 64-Bit and 32-Bit Frame Buffer Modes

To maintain page locality when a 32-bit frame buffer is present, the memory controller splits each quadword operation from the graphics core into two longword operations (Figure 11–4). The quadword address is left-shifted 1 bit, and the address LSB is toggled to differentiate between the upper and lower longword (Figure 11–5). Consequently, 32-bit mode results in a sparsely populated quadword address space.

Figure 11–4 shows how the frame buffer is populated in 64-bit and 32-bit modes.

11.11 Video Registers

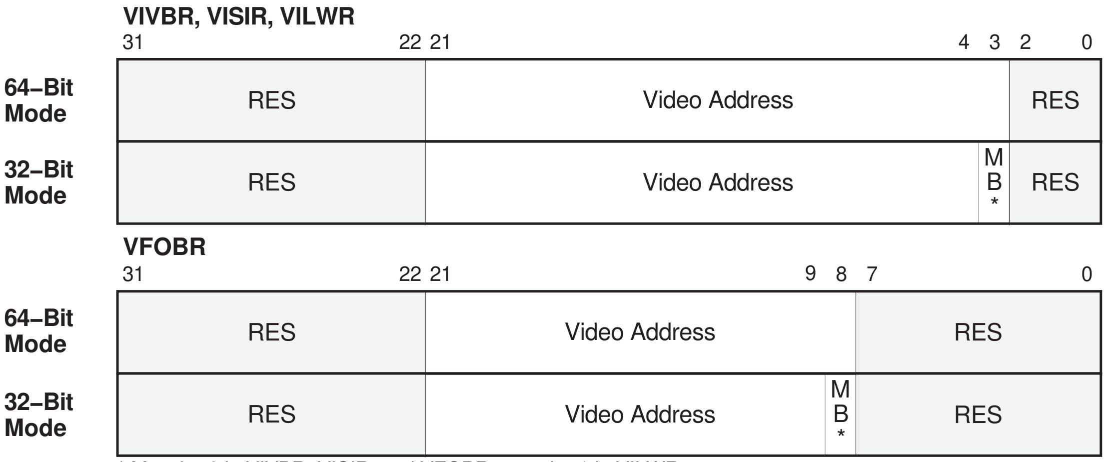
Figure 11–4 Frame Buffer Address Space in 64-Bit and 32-Bit Modes



To maintain correlation between drawing and video refresh, the values programmed in the video address configuration registers must be left-shifted 1 bit — the 64-bit MSB is discarded and the 32-bit LSB must be a zero or one (Figure 11–5).

Figure 11–5 shows 64-bit and 32-bit frame buffer video address format, and Table 11–5 lists the affected registers.

Figure 11–5 Video Address in 64-Bit and 32-Bit Modes



* Must be 0 in VIVBR, VISIR, and VFOBR; must be 1 in VILWR

Table 11–5 Video Address Configuration Registers

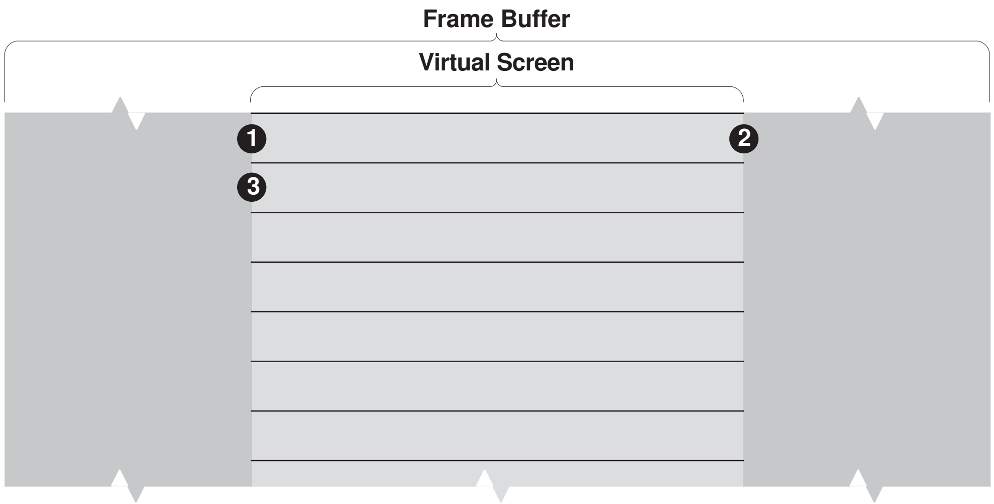
Register	Mnemonic	Section
Video base address register	VIVBR	8.7.1
Video line increment register	VISIR	8.7.1
Video line width register	VILWR	8.7.1
Video pixel occlusion bitmap base address register	VFOBR	8.8.2

11.11.3 Video Refresh Calculations

Figure 11–6 represents a virtual screen in the frame buffer, with the beginning and ending frame buffer address of several scanlines (numbered ❶ through ❸).

11.11 Video Registers

Figure 11–6 Video Scanline Addresses



Video Base Address:	➊
Scanline Width:	(➋ – ➊) – 1 quadword
Scanline Increment:	1 for normal screens
	➌ – ➋ for virtual screens

For a virtual screen, the video base address (➊) is loaded into the refresh address generator at top-of-frame. The refresh address is incremented until the end of the scanline is reached (specified by the scanline width: ➋ minus ➊ minus 1 quadword). At the end of the scanline, the scanline increment value (➌ minus ➋) is added to the refresh address to determine the starting address of the next scanline. This address is then used with the scanline width to determine the end of the next scanline, and so on.

11.12 Programming for Alpha CPUs

Sections 11.12.1 and 11.12.2 describe special programming considerations when using the 21130 with Alpha microprocessors.

11.12.1 Programmed I/O Through the CPU Write Buffer

Alpha microprocessors contain an internal 4-entry write buffer. To optimize the use of system bus bandwidth, the write buffer attempts to collapse and merge quadwords (64 bits) and Dwords before they are written externally. This mechanism has an unwanted side effect on write ordering. Specifically, an ordered packet of Dwords written by a simple string of STL instructions (as in writing a command packet to the 21130) is not necessarily written on the PCI bus in the same order or with all the Dwords intact.

To counter this unwanted side effect, a 21130 driver running on an Alpha microprocessor must:

- Avoid collapsing two separate writes to the same address.
- Enforce write ordering to order-critical 21130 registers.

To enforce write ordering, the Alpha instruction set includes the memory barrier (MB) instruction that allows software to flush the write buffer between stores. However, the MB instruction significantly degrades performance when it is used as frequently as is necessary with an order-dependent, bandwidth-consuming, programmed I/O device such as the 21130. Therefore, to selectively enforce ordering and eliminate collapsed writes, the 21130 software can:

- Access multiple aliased regions in the 21130 address space.
- Carefully order accesses within aligned hexawords (eight Dwords) as appropriate.

The 21130 memory space provides multiple aliases to access the 21130 registers as well as the frame buffer. In most cases, the multiple address-space aliases can be used to work around the CPU write buffer's lack of ordering, without using MB instructions.

For example, rather than writing to the same register twice and issuing an explicit MB instruction, software can write to two aliases of the same register. The different addresses will reside in different write-buffer entries, such that the writes will not merge and will maintain ordering.

Ordering within each CPU write-buffer entry must also be carefully monitored. Each hexaword (eight Dwords) write-buffer entry empties from least significant to most significant Dword (or so it appears on the PCI bus). Therefore, stores to the same hexaword are in low-to-high order regardless of when they were written.

However, strict ordering is not necessary for all writes to the 21130. A typical graphics drawing command packet (Section 10.1.2) written to the 21130 consists of several order-independent register writes, followed by an ordered write to another register or the frame buffer. The first several writes can be arbitrarily reordered among themselves, but they all must appear after the previous command packet and before the last write of the current packet.

The 21130 register-space core map is organized by hexaword to map cleanly to the CPU's write buffer. Within a typical command packet, order-independent register writes are mapped in the same hexaword, and the order-dependent register or frame buffer write is mapped either in the most-significant Dword location of the same hexaword or in another hexaword. If software needs to address another hexaword entry for the order-dependent write, it should choose a different alias for every fourth consecutive access. The order-dependent write then always appears after the order-independent writes.

11.12.2 Address and Continue Register Access

The alternate control space aliases of the address and continue registers (GADR and GCTR) is another mechanism for using the unenforcing write buffer in Alpha microprocessors. The GADR maps to all the even offsets in the first 512KB of alternate ROM space, and the GCTR maps to all the odd offsets (Section 7.5.1.3).

Any graphics operation invoked by a write to the frame buffer can also be invoked by a write to the GADR followed by a write to the GCTR. This allows the 21130 to be programmed by a continuous stream of alternating writes to the GADR and GCTR. By taking advantage of the odd and even aliases in alternate control space, software can effectively pack GADR and GCTR writes in the CPU write buffers. This also minimizes the translation-lookaside buffer (TLB) overhead in the CPU, because all the writes are local.

12

Hardware Interface

This chapter describes the DECchip 21130 external hardware interfaces, with the exception of the PCI interface. In addition to PCI, test, and power pins, the 21130 pins provide external connections to the following:

- 64-bit frame buffer memory
- VGA subsystem
- ROM and generic peripheral port (GPP)
- Video port including the VAFC and monitor
- Clocks

All of the 21130 pins and signals are described in Chapter 3. PCI operations are described in Chapter 9. Appendix A is a summary of the 21130 pinout.

12.1 Frame Buffer Interface

A total of 86 signals are used to move data between the 21130 and its frame buffer DRAMs — a 64-bit data path and 22 address and control signals. In normal operation these 86 signals represent data and control signals for frame buffer memory cycles. However, the physical pins are shared with other subsystems on the 21130 chip that access the graphics BIOS ROM, optional peripheral chips, and the VAFC.

12.1.1 Hardware Mode Restrictions

The use of shared pins restricts the functions available and imposes some limitations in a particular hardware mode. Tables 12–1 through 12–4 show how the shared pins and associated dedicated pins are used in each hardware mode.

12.1 Frame Buffer Interface

Note

The ordering of the VAFC and feature connector signals on shared pins is opposite to the pin number order (see Table 12–4).

Table 12–1 Pin Usage in VGA Mode

Pins	Memory Operations	ROM Operations
Shared Pins	Signals	
memdata<31:00>	**memdata<31:00>**	Not used
memdata<49:32>	Not used	**rom_adr<17:0>**
memdata<57:50>	Not used	**rom_d<7:0>**
memdata<58>	Not used	**rom_we#**
memdata<63:59>	Feature connector data <3:7>	Not used
vafc_data<2:0>	Feature connector data <2:0>	Not used
gp_stb#	Not used	**rom_oe#**
Dedicated Pins	Signals	
memaddr<8:0>	**memaddr<8:0>**	Not used
cas<7:0>#	**cas<7:0>#**	Not used
ras<2:0>#	**ras<2:0>#**	Not used
oeb#	**oeb#**	Not used
wrb#	**wrb#**	Not used
gp_cs#	Not used	Not used
rom_ce#	Not used	**rom_ce#**

Table 12–2 Pin Usage in 32-bit GPP and ROM Modes

Pins	Memory Operations	GPP Operations	ROM Operations
Shared Pins	Signals		
memdata<16:00>	**memdata<16:00>**	**gp_adr<16:00>**	Not used
memdata<17>	**memdata<17>**	Not used	Not used
memdata<25:18>	**memdata<25:18>**	**gp_data<7:0>**	Not used
memdata<26>	**memdata<26>**	**gp_rdsel#**	Not used
memdata<27>	**memdata<27>**	**gp_wrsel#**	Not used

(continued on next page)

Table 12–2 (Cont.) Pin Usage in 32-bit GPP and ROM Modes

Pins	Memory Operations	GPP Operations	ROM Operations
Shared Pins	Signals		
memdata<31:28>	**memdata<31:28>**	Not used	Not used
memdata<49:32>	Not used	Not used	**rom_adr<17:00>**
memdata<57:50>	Not used	Not used	**rom_d<7:0>**
memdata<58>	Not used	Not used	**rom_we#**
memdata<63:59>	Not used	Not used	Not used
vafc_data<2:0>	Not used	Not used	Not used
gp_stb#	Not used	**gp_stb#**	**rom_oe#**
Dedicated Pins	Signals		
memaddr<8:0>	**memaddr<8:0>**	Not used	Not used
cas<7:0>#	**cas<7:0>#**	Not used	Not used
ras<2:0>#	**ras<2:0>#**	Not used	Not used
oeb#	**oeb#**	Not used	Not used
wrb#	**wrb#**	Not used	Not used
gp_cs#	Not used	**gp_cs#**	Not used
rom_ce#	Not used	Not used	**rom_ce#**

Table 12–3 Pin Usage in 64-bit GPP and ROM Modes

Pins	Memory Operations	GPP Operations	ROM Operations
Shared Pins	Signals		
memdata<16:00>	**memdata<16:00>**	**gp_adr<16:00>**	Not used
memdata<17>	**memdata<17>**	Not used	Not used
memdata<25:18>	**memdata<25:18>**	**gp_data<7:0>**	Not used
memdata<26>	**memdata<26>**	**gp_rdsel#**	Not used
memdata<27>	**memdata<27>**	**gp_wrsel#**	Not used
memdata<31:28>	**memdata<31:28>**	Not used	Not used
memdata<49:32>	**memdata<49:32>**	Not used	**rom_adr<17:00>**
memdata<57:50>	**memdata<57:50>**	Not used	**rom_d<7:0>**
memdata<58>	**memdata<58>**	Not used	**rom_we#**
memdata<63:59>	**memdata<63:59>**	Not used	Not used
vafc_data<2:0>	Not used	Not used	Not used
gp_stb#	Not used	**gp_stb#**	**rom_oe#**

(continued on next page)

Table 12–3 (Cont.) Pin Usage in 64-bit GPP and ROM Modes

Pins	Memory Operations	GPP Operations	ROM Operations
Dedicated Pins	Signals		
memaddr<8:0>	**memaddr<8:0>**	Not used	Not used
cas<7:0>#	**cas<7:0>#**	Not used	Not used
ras<2:0>#	**ras<2:0>#**	Not used	Not used
oeb#	**oeb#**	Not used	Not used
wrb#	**wrb#**	Not used	Not used
gp_cs#	Not used	**gp_cs#**	Not used
rom_ce#	Not used	Not used	**rom_ce#**

Table 12–4 Pin Usage in 32-bit GPP and VAFC Modes

Pins	Memory Operations	GPP Operations	VAFC Operations
Shared Pins	Signals		
memdata<16:00>	**memdata<16:00>**	**gp_adr<16:00>**	Not used
memdata<17>	**memdata<17>**	Not used	Not used
memdata<25:18>	**memdata<25:18>**	**gp_data<7:0>**	Not used
memdata<26>	**memdata<26>**	**gp_rdsel#**	Not used
memdata<27>	**memdata<27>**	**gp_wrsel#**	Not used
memdata<31:28>	**memdata<31:28>**	Not used	Not used
memdata<50:32>	Not used	Not used	Not used
memdata<63:51>	Not used	Not used	**vafc_p<3:15>**
vafc_data<2:0>	Not used	Not used	**vafc_p<2:0>**
gp_stb#	Not used	**gp_stb#**	Not used

(continued on next page)

Table 12–4 (Cont.) Pin Usage in 32-bit GPP and VAFC Modes

Pins	Memory Operations	GPP Operations	VAFC Operations
	Signals		
Dedicated Pins			
memaddr<8:0>	**memaddr<8:0>**	Not used	Not used
cas<7:0>#	**cas<7:0>#**	Not used	Not used
ras<2:0>#	**ras<2:0>#**	Not used	Not used
oeb#	**oeb#**	Not used	Not used
wrb#	**wrb#**	Not used	Not used
gp_cs#	Not used	**gp_cs#**	Not used
vafc_dclk	Not used	Not used	**vafc_dclk**
vafc_vclk	Not used	Not used	**vafc_vclk**
blank#	Not used	Not used	**blank#**
hsync	Not used	Not used	**hsync**
vsync	Not used	Not used	**vsync**
grdy	Not used	Not used	**grdy**
evideo#	Not used	Not used	**evideo#**

After the PCI reset signal (**pci_rst#**) is asserted, the 21130 is operating with VGA enabled. This mode allows the VGA feature connector (not VAFC) output to be used, allows ROM accesses, and uses the lower-half of the 64-bit data bus for VGA frame buffer accesses. Sixteen-bit VAFC and GPP cycles are not available.

When the 2DA mode with the 64-bit data bus is selected, ROM and GPP cycles are available, and neither 8-bit (feature connector) nor 16-bit VAFC mode is available.

If a 32-bit data bus mode is selected while operating in 2DA mode, either GPP and VAFC modes or ROM and GPP modes are available. (VAFC and ROM are not available simultaneously because they use the same pins.) Table 2–1 in Section 2.12.2 summarizes these restrictions and limitations.

12.1.2 Frame Buffer Configuration Sensing

During reset, the state of the **gp_cs#** and **gp_stb#** pins are sampled and saved in an internal register. When a ROM read is done from sparse ROM space, the internal register contents are available on bits <9:8> of the returned ROM data (Section 7.5.2.5). **gp_cs#** is <9> and **gp_stb#** is <8>. The module designer can assign configuration data to these pins, to allow software to set the appropriate register bits. For example, to select 32- or 64-bit memory widths, an external 22 kΩ resistor can be connected to the pin and either **Vss** to read a zero or **Vdd** to read a one.

12.2 VGA Subsystem

The 21130 powers up with VGA active and the 2DA inactive. When the 21130 is operating in VGA mode, the PCI address decoders are disabled, and addresses propagate through to the PCI-to-VGA interface, which contains its own decoders.

Figure 12–1 shows the three primary interfaces between the VGA subsystem and the PCI interface, video back end, and frame buffer.

12.2.1 PCI–to–VGA Interface

Because the VGA controller has ISA characteristics on its system interface, the PCI-to-VGA interface translates PCI protocols, data formats, and addresses into their ISA-like equivalents. The PCI-to-VGA interface is a layer of logic and state machines between the back of the PCI interface and the ISA front end of the VGA controller.

12.2.2 VGA-to-Frame Buffer Memory Interface

In VGA mode, the **ras<1:0>** signals independently control a 16-bit-wide memory bank. In 2DA mode, the **ras<1:0>** signals are tied internally and have identical timing to drive 32 or 64 bits of frame buffer DRAMs. (The **ras2** signal is active in 2DA mode, if there is a second bank of frame buffer memory.) The VGA controller uses only 32 bits of frame buffer, regardless of the actual memory width.

The VGA memory control, address, and data signals are multiplexed with their equivalents from the 2DA memory controller immediately before the pins. In VGA mode, the VGA controller has complete control of the frame buffer, including display refresh and DRAM refresh functions.

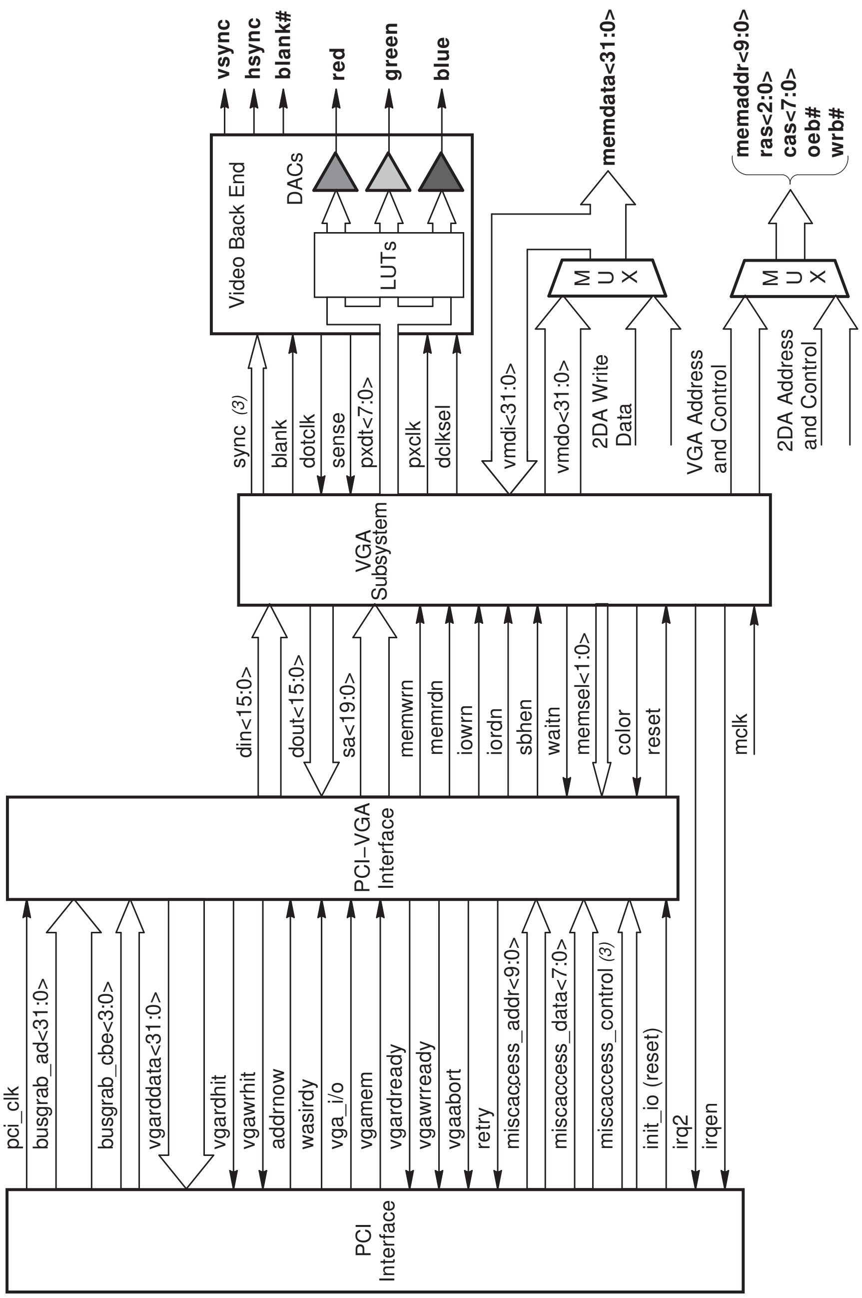
See Section 12.1.1 for more information about mode restrictions due to shared pins.

12.2.3 VGA-to-Video Back End Interface

The VGA controller outputs an 8-bit video pixel stream, sync, and blanking. The video stream passes through a multiplexer to the DACs for output to a monitor.

The 21130 uses the VGA CRT controller (CRTC) for the VGA and 2DA modes of operation. It generates timing for graphics resolutions up to 1280×1024. Because a common CRTC is used, the CRTC register addresses must be mapped in both modes of operation.

Figure 12–1 VGA Subsystem Interfaces



12.3 ROM and Generic Peripheral Port Interface

The BIOS ROM is accessed on system power-up, reconfiguration, or reset. Because these are infrequent and low-bandwidth events, most signals used to access the ROM are shared with frame buffer interface signals on common pins (see Section 12.1.1).

Note

When operating in 2DA mode, ROM accesses should be restricted to vertical blank time, or the time when video is disabled by the video valid bit in the video valid register (VIVVR <0>, Section 8.7.2).

The GPP also shares pins with the frame buffer data path for GPP data, address, and some of the control signals. Consequently, GPP accesses must arbitrate with other contenders for time on these pins. This arbitration adds up to 25 PCI clocks of latency on the PCI bus. The GPP bandwidth is appropriate for bidirectional traffic in the range of 1 MB/s and lower. This is suitable for the control needs of many multimedia devices, or for digitized audio streams; but inadequate for full-time video traffic. The VAFC is more appropriate for full-time video traffic, while control traffic or audio can simultaneously use the GPP. To simplify design, the GPP supports a single access speed of 300 ns.

Note

GPP accesses should be restricted to vertical blank time, or the time when video is disabled by the video valid bit in the video valid register (VIVVR <0>, Section 8.7.2).

See Section 7.5.2.2 for a description of GPP address space mapping.

12.3.1 GPP Read and Write Access

Some devices use a single read/write line and a strobe signal to indicate the start of a cycle (valid address, write data). For such devices, the **gp_wrsel#** signal functions as the read/write signal. In most cases, a single small external device can do any required signal translation to adapt to various interface requirements.

12.3.2 GPP Interrupts

The **gp_int#** interrupt signal can be asserted asynchronously with respect to the 21130 clocks. Internally, the 21130 samples the signal and passes it to the PCI interface, where it results in a PCI interrupt. If enabled, **gp_int#** sets bit <5> in the interrupt status register (MISR, Section 8.3.2). The interrupt service routine reads the MISR to determine if the interrupt is in the 21130 or on the GPP. If it is a GPP interrupt, the interrupt service routine then reads specific status registers in the peripheral device to determine the nature of the interrupt.

The **gp_int#** signal is also used to select the following:

- Internal or external clocks in the PCI clock control register (PCCR <1:0>, Section 8.2.8)
- Internal or external **vsync**, **hsync**, and **blank#** signals in the video valid register (VIVVR <7>, Section 8.7.2)
- With the **pci_rst#** pin, select clock frequencies in the clock control A and B registers (VXCKAR and VXCKBR, Section 8.14.10)

12.4 Video Port and Display Monitor Interface

Sections 12.4.1 through 12.4.5 describe the video port and monitor interfaces.

12.4.1 VESA Advanced Feature Connector

The VESA advanced feature connector (VAFC) provides a way to send or receive pixel data. VAFC base-level operation requires the following pins to be supported:

vafc_dclk	**evideo#**
vafc_vclk	**blank#**
vafc_p<0:15>	**hsync**
grdy	**vsync**

Additionally, the 21130 supplies the **vafc_en#** output signal to the external bus. The 21130 provides all these pins to support the VAFC base level. When operating in VGA mode, only 8-bit output is allowed (for backwards compatibility with the previous generation feature connector). The VAFC modes are described in Sections 12.4.1.3 and 12.4.1.4, and the signals are described in Chapter 3.

12.4 Video Port and Display Monitor Interface

Note

The *VESA Advanced Feature Connector (VAFC) Standard, Version 1.0* requires the 8-bit output mode at power up. This implies that the video source driving the VAFC must not assert the **evideo#** signal at power up, because the 21130 will not accept pixel input. If the video source does assert the **evideo#** signal, the 21130 deasserts the **vafc_en#** signal, to prevent possible bus contention.

12.4.1.1 VAFC Operation

For all modes, the pixel data appearing on the feature connector is taken from the internal pixel bus prior to being sent to the variable pixel formatting logic. In any mode, it must be ensured that the bits-per-pixel selected by the video pixel format register (VFPFR <11:10>, Section 8.8.1) matches the VAFC output format. (For example, the VAFC 16-bit pixel format can be used with the 15-bpp or 16-bpp formats.)

12.4.1.2 Relationship Between vafc_vclk and vafc_dclk

The output dot clock (**vafc_dclk**) is generated from either the 21130's internal pixel clock or pixel clock divided by 2. It is sent over the VAFC, to the external video card. In turn, the video card uses **vafc_dclk** to generate pixels. Because **vafc_dclk** cannot drive all of the logic on the video card, the card buffers **vafc_dclk**. It is this buffered version of **vafc_dclk** that is sent back to the 21130 as **vafc_vclk**. Input pixels are sampled by **vafc_vclk**, and **grdy** is a function of **vafc_vclk**. (Pixel clock generation is described in Section 12.5.3.)

12.4.1.3 VAFC Pixel Output Modes

The 21130 supports the following VAFC pixel output modes.

- 8-bit frame buffer or VGA data, 1 pixel per clock

  This mode supports standard VGA pass-through output. VGA pixels are 8-bpp indexed, and the 21130 in native mode can output 8-bpp indexed or 3:3:2 RGB pixels. The pixel data occupies bits **vafc_p<7:0>** on the VAFC bus (bits <15:8> are ignored).

- 8-bit frame buffer data, 2 pixels per clock

  In this mode, 2 pixels at 8-bpp indexed or 3:3:2 RGB are placed on the VAFC bus at the same time. The **vafc_dclk** is programmed to be one-half the frequency of the 21130's internal pixel clock. The left-most pixel displayed on the screen occupies VAFC bus bits **vafc_p<7:0>** and the right-most pixel occupies bits **vafc_p<15:8>**.

- 16-bit frame buffer data, 1 pixel per clock

This mode supports 5:5:5 RGB, 5:6:5 RGB, and 4:2:2 YUV pixel output.

12.4.1.4 VAFC Pixel Input Modes

The 21130 supports the following VAFC pixel input modes.

- 8-bit video system data, 1 pixel per clock

  The VAFC input pixels are treated as 8-bit frame buffer data. Pixel interpretation is controlled by the inside pixel format. The pixel data occupies bits **vafc_p<7:0>** on the VAFC bus (bits <15:8> are ignored).

- 8-bit video system data, 1 pixel per 2 clocks

  The VAFC dot clock (**vafc_dclk**) is programmed to be one-half the frequency of the 21130's internal pixel clock. Each pixel arriving over the VAFC bus is displayed during two successive pixels. Pixel interpretation is controlled by the inside pixel format. The pixel data occupies bits **vafc_p<7:0>** on the VAFC bus (bits <15:8> are ignored). The divided-down dot clock constrains pixels to be displayed on even pixel boundaries.

- 8-bit video system data, 2 pixels per clock

  The VAFC input pixels are treated as 8-bit frame buffer data. Pixel interpretation is controlled by the inside pixel format. The VAFC dot clock (**vafc_dclk**) is programmed to be one-half the frequency of the 21130's internal pixel clock. The left-most pixel displayed on the screen occupies VAFC bus bits **vafc_p<7:0>** and the right-most pixel occupies bits **vafc_p<15:8>**. The divided-down dot clock constrains pixels to be displayed on even pixel boundaries.

- 16-bit video system data, 1 pixel per clock

  The VAFC input pixels are treated as 16-bit frame buffer data. Pixel interpretation is controlled by the inside pixel format. The pixel data occupies bits **vafc_p<0:15>** on the VAFC bus.

- 16-bit video system data, 1 pixel per 2 clocks

  The VAFC dot clock (**vafc_dclk**) is programmed to be one-half the frequency of the 21130's internal pixel clock. Each pixel arriving over the VAFC bus is displayed during two successive pixels. Pixel interpretation is controlled by the inside pixel format. The pixel data occupies bits **vafc_p<0:15>** on the VAFC bus. The divided-down dot clock constrains pixels to be displayed on even pixel boundaries.

12.4.1.5 VAFC Input Windows

The external video system uses the **grdy** signal to enable VAFC bus transfers. The 21130 uses **grdy** to define a window in which VAFC input pixels can be displayed. VAFC input can be full screen, and the 21130 generates **grdy** from a blanking signal. VAFC input can also be a window within a full screen. To accomplish this, the 21130 uses the pixel occlusion bitmap (Section 8.8.1.3) to define the input window. When using the pixel occlusion bitmap with a divided-down **vafc_dclk**, every pair of bits within the pixel occlusion bitmap must be identical because **vafc_dclk** can sample the bitmap data only on every other clock cycle.

12.4.1.6 VAFC Blank Enable

The 21130's **blank#** pin signals valid VAFC output pixels. To blank the VAFC video system without also blanking the monitor connected directly to the 21130, the alternate video control register contains a VAFC-specific blank enable (VFAVR <1>, Section 8.8.5).

12.4.1.7 VAFC Output Screen Resolutions

The 21130 supports the following VAFC output screen resolutions.

- 1024 × 768 @ 75 Hz
  - 8-bit frame buffer or VGA data, 1 pixel per clock
  - 8-bit frame buffer data, 2 pixels per clock
- 800 × 600 @ 75 Hz
  - 8-bit frame buffer or VGA data, 1 pixel per clock
  - 8-bit frame buffer data, 2 pixels per clock
  - 16-bit frame buffer data, 1 pixel per clock
- 640 × 480 @ 75 Hz and below
  - 8-bit frame buffer or VGA data, 1 pixel per clock
  - 8-bit frame buffer data, 2 pixels per clock
  - 16-bit frame buffer data, 1 pixel per clock
- NTSC resolutions
  - 8-bit frame buffer or VGA data, 1 pixel per clock
  - 8-bit frame buffer data, 2 pixels per clock
  - 16-bit frame buffer data, 1 pixel per clock

12.4.1.8 VAFC Input Screen Resolutions

The 21130 supports the following VAFC input screen resolutions.

- 1024 × 768 @ 75 Hz, 8-bpp
  - 8-bit video system data, 1 pixel per 2 clocks
  - 8-bit video system data, 2 pixels per clock
  - 16-bit video system data, 1 pixel per 2 clocks
- 800 × 600 @ 75 Hz, 8-bpp
  - 8-bit video system data, 1 pixel per 2 clocks
  - 8-bit video system data, 2 pixels per clock
  - 16-bit video system data, 1 pixel per 2 clocks
- 640 × 480 @ 75 Hz and below, 8-bpp and 16-bpp
  - 8-bit video system data, 1 pixel per clock
  - 8-bit video system data, 1 pixel per 2 clocks
  - 8-bit video system data, 2 pixels per clock
  - 16-bit video system data, 1 pixel per clock
  - 16-bit video system data, 1 pixel per 2 clocks

See the *VESA Advanced Feature Connector (VAFC) Standard, Version 1.0* for more information about the VAFC.

12.4.2 Video Port Transceivers

When the 21130 is sourcing RGB video to an off-card destination, the **vafc_en#** signal is asserted. This enables transceivers in the 21130 to VAFC connector direction, which drive the card-top cable to its destination (typically, another video card which in turn drives a display monitor). The transceivers play an important role in buffering the critical frame buffer data path signals from the capacitive loading and reflections of the VAFC connectors and cables. Transceivers must be placed as close as possible to the 21130 data path pins to minimize stub length.

12.4.3 Monitor Connection

The 21130 drives three analog outputs to the monitor. The three color outputs can drive doubly-terminated 75-Ω coaxial signal lines to the display monitor. Sync can be combined with the green output, using the DAC command register 0 (DCOR0 <3>, Section 8.9.7).

12.4.4 Display Power Management Signaling

Display power management signaling (DPMS) is a VESA ‘‘green computer’’ standard that defines four levels of monitor operation for power management. The 21130 selects the level of monitor operation (DPMS state) with combinations of the presence and absence of horizontal sync (**hsync**) and vertical sync (**vsync**) pulses. The states are controlled by the DPMS and blank fields in the video valid register (VIVVR <5:4,1>, Section 8.7.2).

See the VESA *Display Power Management Signaling (DPMS) Proposal, Version 1.0p, Revision 0.7p* for more information.

Table 12–5 lists the DPMS states.

Table 12–5 DPMS States

State	hsync	vsync	Video	Power Savings	Recovery Time
On	Pulse	Pulse	Active	None	Not applicable
Standby	None	Pulse	Blanked	Minimal	Short
Suspend	Pulse	None	Blanked	Substantial	Longer
Off	None	None	Blanked	Maximum	System-dependent

12.4.5 Display Data Channel

The display data channel (DDC) is described in the VESA *Display Data Channel Standard, Version 1.0, Revision 0*. It specifies a data format that can be transmitted between a computer display and the host system. The 21130 provides low-level support for the two types of data channels (DDC1 and DDC2) that carry DDC data. DDC1 data is transferred in a single signal that is clocked by the **vsync** signal. DDC2 data is transferred over an ACCESS.bus channel.

The 21130 supports DDC1 with the DDCDI bit in the video valid register (VIVVR <6>, Section 8.7.2). The DDCDI bit, in conjunction with the VRI bit in the VGA input status 0 register (VEIS0R <7>, Section 8.11.3), allows software to deserialize the DDC data stream. Briefly, the VESA specification states that the DDC data will be valid when **vsync** is low, which corresponds to a set VRI bit. Consequently, software can poll the VRI bit, and accumulate DDC data when the bit is set.

To support the ACCESS.bus data channel, software must generate both the data and the clock signals, in accordance with the I^2C protocol. Software can use the DDCDO and TCLKO bits (VIVVR <11:12>) to generate, respectively, the I^2C data and clock signals SDA and SCL. The I^2C protocol states that, in the absence of collisions, data can be changed while the clock is low, and should be sampled when the clock is high. Additionally, software must detect and generate, start and stop conditions. For more information, see the I^2C specification in the Philips *Data Handbook for I^2C Peripherals for Microcontrollers.*

12.5 Clocks and Clock Control

In addition to the externally supplied PCI clock, the 21130 has two internally-generated primary clocks — the memory clock and the pixel clock. See Figure 12–2.

12.5.1 Memory Clock

The memory clock (**mem_clk**) is a 66-MHz (nominal) clock to the accelerator section, VGA controller, and memory controller. It is generated by a PLL-based clock generator circuit (**buffered_fastclk** in Figure 12–2). The memory clock frequency M term multiplier is programmable, and is selected in the PCI clock control register (PCCR, Section 8.2.8). Note that the N term divisor is a fixed value of 8 and the L term divisor is not used in the memory clock PLL.

12.5.2 Core Clock

The core clock (**core_clk**) is also used by the accelerator section. It is one-half the frequency of the memory clock (**buffered_slowclk** in Figure 12–2).

12.5.3 Pixel Clock

The pixel clock (**pix_clk**) is generated by a programmable source, based on a second PLL circuit. It can generate pixel clock rates between 8 MHz and 135 MHz. The frequency is selected in the clock control A and B registers (VXCKAR and VXCKBR, Section 8.14.10) L, M, and N terms. Both the memory clock and the pixel clock are derived from the same reference clock, provided by a low-cost 14.31818 MHz crystal on the **xtal1** and **xtal2** input pins. The **xtal2** pin also serves as the backup clock input for the memory clock, if an external source is selected (PCCR<0>).

The pixel clock for video generation can be sourced from an internal PLL circuit or on the **pixlck** pin from an external ICS2595 device (as a risk-reduction backup). Software uses the PCS bit (PCCR <1>) to control an internal multiplexer, which selects an internal or external source.

On power up or reset, the 21130 selects the internal source. Software must intervene to change to the external source. Use of an external pixel clock source is considered a backup scheme in the event the internal circuit does not meet requirements.

If the external pixel clock source is an ICS2595, its output frequency must be software-selected through the 21130. A 4-bit value is loaded into the ICS2595 to select one of 16 preprogrammed pixel clock frequencies. The 21130 uses the GPP to interface to the ICS2595. The lower four GPP data path bits connect to the four ICS2595 data input pins, and the **gp_cs#** signal connects to the ICS2595 strobe input.

The pixel clock also drives **vafc_dclk**. See Section 12.4.1.2 for more information.

12.5.4 VGA Dot Clock

The pixel clock (**buffered_pixclk** in Figure 12–2) is driven either by the PLL directly or the VGA controller. The PLL drives the VGA dot clock to the VGA controller where it is divided or not, depending on the specific VGA mode, and returned to the clock generation function as the VGA pixel clock. If VGA mode is enabled, the VGA pixel clock drives the buffered pixel clock; otherwise, in 2DA mode, the PLL pixel clock drives the buffered pixel clock directly.

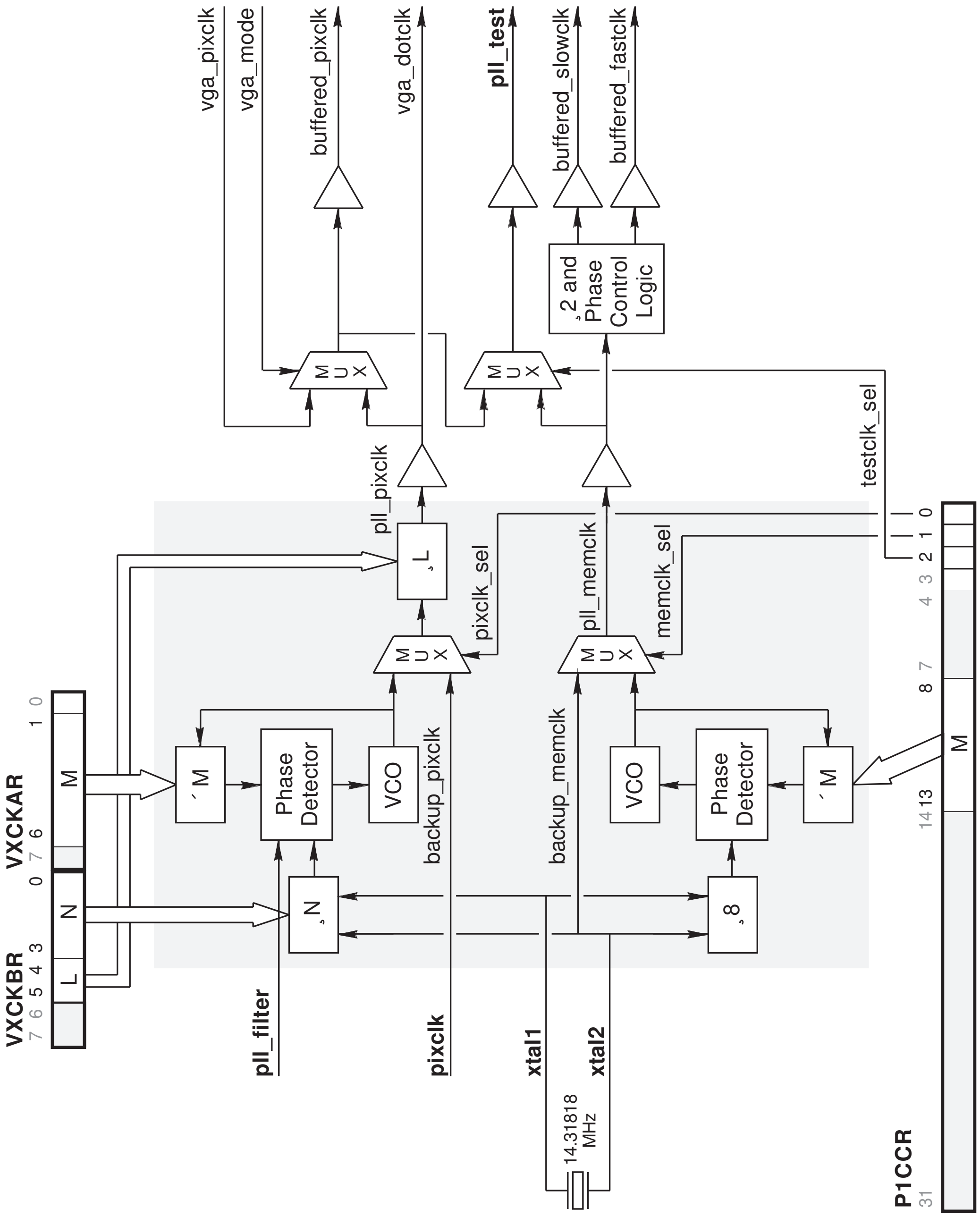
The VGA variable dot clock select bit (VXCKAR <0>) determines whether the VGA dot clock frequency is controlled by the VGA miscellaneous output register (VEMISR, Section 8.11.1) or directly by the L, M, and N fields in the VXCKAR and VXCKBR. If the VEMISR is selected, its clock source select bits control a multiplexer (not shown in Figure 12–2) that forces the PLL L, M, and N values to generate either 25.057 or 28.189 MHz.

12.5.5 Test Clock

In test mode, either of the two internally generated clocks can be selected as the **pll_test** test clock output. The TCS bit (PCCR <2>) selects the pixel clock or memory clock as the test clock source. The video valid register (VIVVR, Section 8.7.2) also contains test clock control bits.

Figure 12–2 is a simplified block diagram of the clock generation function.

Figure 12–2 Clock Generation



External signals are shown in **lowercase boldface** type.

A

Pin Summary

Table A–1 summarizes the DECchip 21130 signal pins. The following abbreviations are used in Table A–1:

#	Low-asserted
I	Input
I/O	Bidirectional
O	Output
P	Power
NA	Not applicable
TS	Tristate
OD	Open drain
DH	Driven, high
DL	Driven, low
DI	Driven, indeterminate
SH	Shared

Table A–1 Signals by Function

Signal	Qty	Type	Function	Value at Reset
PCI Interface				
pci_idsel	1	I	PCI initialization device select	NA
pci_gnt#	1	I	PCI DMA grant	NA
pci_rst#	1	I	PCI reset	NA
pci_clk	1	I	PCI clock	NA
pci_ad<31:0>	32	I/O	PCI address or data	TS
pci_cbe<3:0>#	4	I/O	PCI command and byte enable	TS
pci_frame#	1	I/O	PCI frame	TS

(continued on next page)

Table A–1 (Cont.) Signals by Function

Signal	Qty	Type	Function	Value at Reset
PCI Interface				
pci_irdy#	1	I/O	PCI initiator ready	TS
pci_trdy#	1	I/O	PCI target ready	TS
pci_devsel#	1	I/O	PCI device select	TS
pci_stop#	1	I/O	PCI stop transaction	TS
pci_perr#	1	I/O	PCI parity error	TS
pci_par	1	I/O	PCI parity	TS
pci_req#	1	O	PCI DMA request	TS
pci_inta#	1	O	PCI interrupt	OD
pci_serr#	1	O	PCI system error	OD
Vdd (pci<6:0>)	7	P	PCI I/O 5-V supply	NA
Vss (pci<7:0>)	8	P	PCI I/O ground	NA
Frame Buffer Interface				
memdata<63:0>	64	I/O	Memory data	DI
memaddr<8:0>	9	O	Memory address	DI
cas<7:0>#	8	O	Column address strobe	DH
ras<2:0>#	3	O	Row address strobe	DH
oeb#	1	O	Output enable	DH
wrb#	1	O	Write enable	DL
GPP and ROM Interface				
gp_int#	1	I	Generic peripheral interrupt	NA
gp_data<7:0>[1]	(8)	I/O	Generic peripheral data	SH
gp_adr<16:0>[2]	(17)	O	Generic peripheral address	SH
gp_rdsel#[3]	(1)	O	Generic peripheral read select	SH

[1]The **gp_data<7:0>** signals share the **memdata<25:18>** pins.

[2]The **gp_adr<16:0>** signals share the **memdata<16:0>** pins.

[3]The **gp_rdsel#** signal shares the **memdata<26>** pin.

(continued on next page)

Table A–1 (Cont.) Signals by Function

Signal	Qty	Type	Function	Value at Reset
GPP and ROM Interface				
gp_wrsel#[4]	(1)	O	Generic peripheral write select	SH
gp_cs#	1	O	Generic peripheral chip select	TS[5]
gp_reset#	1	O	Generic peripheral reset	DL
gp_stb#	1	O	Generic peripheral strobe	TS[5]
rom_d<7:0>[6]	(8)	I/O	ROM data path	SH
rom_adr<17:0>[7]	(18)	O	ROM address	SH
rom_ce#	1	O	ROM chip enable	DH
rom_oe#[8]	(1)	O	ROM output enable	SH
rom_we#[9]	(1)	O	ROM write enable	SH
VGA and VAFC Video Port Interface				
ddc_data	1	I/O	Display data channel	TS
evideo#	1	I	Enable external video data	NA
vafc_vclk	1	I	VAFC video clock	NA
vafc_p<0:15>[10]	(16)	I/O	Port	AH
vafc_en#	1	O	VAFC data enable	DL
vafc_dclk	1	O	VAFC dot clock	Pixel clock
grdy	1	O	Graphics device ready	DL
blank#	1	O	Composite video blank	DI
RGB-to-Monitor Interface				
hsync	1	O	Horizontal video sync	DI
vsync	1	O	Vertical video sync	DH

[4]The **gp_wrsel#** signal shares the **memdata<27>** pin.

[5]At reset, the **gp_cs#** and **gp_stb#** signals are inputs and are sampled.

[6]The **rom_d<7:0>** signals share the **memdata<57:50>** pins.

[7]The **rom_adr<17:0>** signals share the **memdata<49:32>** pins.

[8]The **rom_oe#** signal shares the **gp_stb#** pin.

[9]The **rom_we#** signal shares the **memdata<58>** pin.

[10]The **vafc_p<3:15>** signals share the **memdata<63:51>** pins.

(continued on next page)

Table A–1 (Cont.) Signals by Function

Signal	Qty	Type	Function	Value at Reset
RGB-to-Monitor Interface				
red	1	O	Red analog output	DI
green	1	O	Green analog output	DI
blue	1	O	Blue analog output	DI
DAC Interface				
comp	1	I	DAC external compensation	NA
fsadjust	1	I	DAC external resistor	NA
ref	1	I	DAC external voltage reference	NA
dac_Vdd	1	P	DAC 5-V supply	NA
dac_Vss	3	P	DAC ground	NA
opamp_Vdd	1	P	DAC op amp 5-V supply	NA
opamp_Vss	1	P	DAC op amp ground	NA
Clock Interface				
xtal1	1	I	Crystal input	Reference clock
xtal2	1	I	Crystal input/memory clock	NA
pixclk	1	I	Backup pixel clock	NA
pll_filter	1	I	External filter capacitors	NA
pll_test	1	O	Clock test output	DL
pll_Vdd	1	P	PLL 5-V supply	NA
pll_Vss	3	P	PLL ground	NA
Miscellaneous Test Pins				
test_in	1	I	Test input	NA
Miscellaneous Power Pins				
Vdd (video)	1	P	Video clock 5-V supply	NA

(continued on next page)

Table A–1 (Cont.) Signals by Function

Signal	Qty	Type	Function	Value at Reset
Miscellaneous Power Pins				
Vss (video)	1	P	Video clock ground	NA
Vdd (ac<2:0>)	3	P	I/O 5-V ac supply	NA
Vss (ac<4:0>)	5	P	I/O ac ground	NA
Vdd (dc<1:0>)	2	P	I/O 5-V dc supply	NA
Vss (dc<1:0>)	2	P	I/O dc ground	NA
Vdd (core<1:0>)	2	P	Core logic 5-V supply	NA
Vss (core<1:0>)	2	P	Core logic ground	NA

B
Register Summary

This appendix includes an alphabetical list (Table B–1) and a summary (Table B–2) of the DECchip 21130 registers.

Table B–1 is an alphabetical list of the 21130 registers, which references the sections in which they are described.

Table B–1 21130 Register Alphabetical List

Name	Mnemonic	Section
Address register	GADR	8.5.6
Alternate video control register	VFAVR	8.8.5
Background register	GBGR	8.5.8
Bresenham 1 register	GB1R	8.5.11
Bresenham 2 register	GB2R	8.5.12
Bresenham 3 register	GB3R	8.5.13
Bresenham width register	GBWR	8.5.14
Command status register	MCSR	8.3.1
Continue register	GCTR	8.4.3
Copy-64 destination register	GCDR	8.4.4
Copy-64 source register	GCSR	8.4.4
Copy-64A destination register	GCADR	8.4.5
Copy-64A source register	GCASR	8.4.5
Copy buffer register 7:0	GCBR<7:0>	8.5.4
Cursor base address register	CCBR	8.6.2
Cursor mode register	CMOR	8.6.1
Cursor XY register	CXYR	8.6.3
Data register	GDAR	8.5.7
Deep register	GDER	8.5.2
Dither column register	GDCR	8.5.17
Dither row register	GDRR	8.5.17
DMA base address register	GDBR	8.5.15

(continued on next page)

Table B–1 (Cont.) 21130 Register Alphabetical List

Name	Mnemonic	Section
Foreground register	GFGR	8.5.8
Interrupt status register	MISR	8.3.2
Mode register	GMOR	8.5.1
Palette and DAC blue signature register	DBSR	8.9.9
Palette and DAC command register 0	DCOR0	8.9.7
Palette and DAC command register 1	DCOR1	8.9.8
Palette and DAC cursor color register	DCCR	8.9.4
Palette and DAC cursor read address register	DCRR	8.9.3
Palette and DAC cursor write address register	DCWR	8.9.3
Palette and DAC green signature register	DGSR	8.9.9
Palette and DAC pixel mask register	DPMR	8.9.5
Palette and DAC RAM color register	DPCR	8.9.2
Palette and DAC RAM read address register	DPRR	8.9.1
Palette and DAC RAM write address register	DPWR	8.9.1
Palette and DAC red signature register	DRSR	8.9.9
Palette and DAC status register	DSTR	8.9.6
PCI class and revision register	PCRR	8.2.3
PCI clock control register	PCCR	8.2.8
PCI command and status register	PCSR	8.2.2
PCI device base address register 0	PDBR0	8.2.5
PCI device base address register 1	PDBR1	8.2.5
PCI expansion ROM base address register	PRBR	8.2.6
PCI identification register	PIDR	8.2.1
PCI interrupt line register	PILR	8.2.7
PCI latency timer and header type register	PLTR	8.2.4
Pixel mask register	GPXR	8.5.10
Pixel shift register	GPSR	8.5.5
Raster operation register	GOPR	8.5.9
Repeat begin register	GRBR	8.4.6
Repeat end register	GRER	8.4.6
Scaled-copy control register	GSCR	8.5.16
Slope registers 7:0	GSLR<7:0>	8.4.1
Slope-no-go registers 7:0	GSNR<7:0>	8.5.3
Span width register	GSWR	8.4.2
VGA attribute controller color plane enable register	VACPER	8.16.5
VGA attribute controller color select register	VACSLR	8.16.7
VGA attribute controller index/data register	VAIXDR	8.16.1
VGA attribute controller mode register	VAMODR	8.16.3
VGA attribute controller overscan register	VAOSCR	8.16.4

(continued on next page)

Table B–1 (Cont.) 21130 Register Alphabetical List

Name	Mnemonic	Section
VGA attribute controller palette register	VAPALR	8.16.2
VGA attribute controller pixel panning register	VAPXPR	8.16.6
VGA color DAC state register	VPDSTR	8.17.2
VGA color pixel address read mode register	VPPARR	8.17.1
VGA color pixel address write mode register	VPPAWR	8.17.1
VGA color pixel data register	VPPDAR	8.17.3
VGA color pixel mask register	VPPMAR	8.17.4
VGA CRTC cursor end register	VCCUER	8.13.11
VGA CRTC cursor location high register	VCCLHR	8.13.13
VGA CRTC cursor location low register	VCCLLR	8.13.13
VGA CRTC cursor start register	VCCUSR	8.13.11
VGA CRTC data register	VCDATR	8.13.2
VGA CRTC end horizontal blank register	VCHBER	8.13.5
VGA CRTC end horizontal sync register	VCHSER	8.13.6
VGA CRTC end vertical blanking register	VCVBER	8.13.18
VGA CRTC end vertical display register	VCVDER	8.13.15
VGA CRTC end vertical sync register	VCVSER	8.13.14
VGA CRTC horizontal display end register	VCHDER	8.13.4
VGA CRTC horizontal total register	VCHTOR	8.13.3
VGA CRTC index register	VCINXR	8.13.1
VGA CRTC line compare register	VCLCMR	8.13.20
VGA CRTC maximum scanline register	VCMSLR	8.13.10
VGA CRTC mode control register	VCMODR	8.13.19
VGA CRTC offset register	VCOFFR	8.13.16
VGA CRTC overflow register	VCOVRR	8.13.8
VGA CRTC preset row register	VCPROR	8.13.9
VGA CRTC start address high register	VCSAHR	8.13.12
VGA CRTC start address low register	VCSALR	8.13.12
VGA CRTC start horizontal blank register	VCHBSR	8.13.5
VGA CRTC start horizontal sync register	VCHSSR	8.13.6
VGA CRTC start vertical blanking register	VCVBSR	8.13.18
VGA CRTC start vertical sync register	VCVSSR	8.13.14
VGA CRTC underline row scan register	VCULRR	8.13.17
VGA CRTC vertical total register	VCVTOR	8.13.7
VGA extended clock control A register	VXCKAR	8.14.10
VGA extended clock control B register	VXCKBR	8.14.10
VGA extended equalization end register	VXEQER	8.14.5
VGA extended equalization start register	VXEQSR	8.14.5
VGA extended half-line register	VXHLNR	8.14.6

(continued on next page)

Table B–1 (Cont.) 21130 Register Alphabetical List

Name	Mnemonic	Section
VGA extended host page offset A register	VXHPAR	8.14.2
VGA extended host page offset B register	VXHPBR	8.14.2
VGA extended interface control register	VXEICR	8.14.11
VGA extended interlace control register	VXICOR	8.14.4
VGA extended paging control register	VXPCOR	8.14.1
VGA extended split-screen start address high byte register	VXSAHR	8.14.3
VGA extended split-screen start address low byte register	VXSALR	8.14.3
VGA extended timing control A register	VXTCAR	8.14.7
VGA extended timing control B register	VXTCBR	8.14.8
VGA extended video FIFO control register	VXFCOR	8.14.9
VGA feature control register	VEFCOR	8.11.2
VGA graphics controller bit mask register	VGBMKR	8.15.11
VGA graphics controller color compare register	VGCCMR	8.15.5
VGA graphics controller color don’t care register	VGCDCR	8.15.10
VGA graphics controller data register	VGDATR	8.15.2
VGA graphics controller data rotate register	VGDROR	8.15.6
VGA graphics controller enable set/reset register	VGESRR	8.15.4
VGA graphics controller miscellaneous register	VGMISR	8.15.9
VGA graphics controller mode register	VGMODR	8.15.8
VGA graphics controller index register	VGINXR	8.15.1
VGA graphics controller read map select register	VGRMSR	8.15.7
VGA graphics controller set/reset register	VGSRER	8.15.3
VGA input status 0 register	VEIS0R	8.11.3
VGA input status 1 register	VEIS1R	8.11.4
VGA miscellaneous output register	VEMISR	8.11.1
VGA sequencer character map select register	VSCMSR	8.12.6
VGA sequencer clocking mode register	VSCMOR	8.12.4
VGA sequencer data register	VSDATR	8.12.2
VGA sequencer index register	VSINXR	8.12.1
VGA sequencer memory mode register	VSMMOR	8.12.7
VGA sequencer plane mask register	VSPLMR	8.12.5
VGA sequencer reset register	VSRESR	8.12.3
Video base address register	VIVBR	8.7.1
Video current refresh address register	VFCRR	8.8.4
Video line width register	VILWR	8.7.1
Video pixel format register	VFPFR	8.8.1
Video pixel occlusion bitmap base address register	VFOBR	8.8.2

(continued on next page)

Table B–1 (Cont.) 21130 Register Alphabetical List

Name	Mnemonic	Section
Video pixel occlusion bitmap current address register	VFOAR	8.8.3
Video scanline increment register	VISIR	8.7.1
Video valid register	VIVVR	8.7.2

Table B–2 is a summary of the 21130 registers grouped according to function. It includes the type of access and reset state.

Table B–2 21130 Register Summary

Name	Mnemonic	Access	Address	Reset State
Configuration Space Header Block	**PxxR**	**—**	**Range[1]**	**—**
PCI identification register	PIDR	RW	03..00	000C1011
PCI command and status register	PCSR	RW	07..04	02800000
PCI class and revision register	PCRR	RW	0B..08	03000002
PCI latency timer and header type register	PLTR	RW	0F..0C	00800000
PCI device base address register 0	PDBR0	RW	13..10	00000008
PCI device base address register 1	PDBR1	RW	17..14	00000000
Reserved	—	—	2F..18	—
PCI expansion ROM base address register	PRBR	RW	33..30	00000000
Reserved	—	—	3B..34	—
PCI interrupt line register	PILR	RW	3F..3C	00040100
Device-Dependent Configuration Space	**PxxR**	**—**	**Range[1]**	**—**
PCI clock control register	PCCR	RW	43..40	0000280X
Reserved	—	—	FF..40	—
Miscellaneous Registers	**MxxR**	**—**	**Offset**	**—**
Command status register	MCSR	RO	1F8[2]	Cleared
Interrupt status register	MISR	RW	07FFFF.. 040000[3]	Cleared

[1] Address = hexadecimal byte address range for PCI registers.

[2] Address = hexadecimal offset into PDBR0 register space.

[3] Address = hexadecimal offset into PDBR1 memory space.

(continued on next page)

Table B–2 (Cont.) 21130 Register Summary

Name	Mnemonic	Access	Address	Reset State
Graphics Command Registers	**GxxR**	**—**	**Offset[2]**	**—**
Slope register 7	GSLR7	WO	13C	Undefined
Slope register 6	GSLR6	WO	138	Undefined
Slope register 5	GSLR5	WO	134	Undefined
Slope register 4	GSLR4	WO	130	Undefined
Slope register 3	GSLR3	WO	12C	Undefined
Slope register 2	GSLR2	WO	128	Undefined
Slope register 1	GSLR1	WO	124	Undefined
Slope register 0	GSLR0	WO	120	Undefined
Span width register	GSWR	RW	0BC	Cleared
Continue register	GCTR	WO	04C	Cleared
Copy-64 source register	GCSR	WO	160	Cleared
Copy-64 destination register	GCDR	WO	164	Cleared
Copy-64A source register	GCASR	WO	360	Cleared
Copy-64A destination register	GCADR	WO	364	Cleared
Repeat begin register	GRBR	WO	340	Cleared
Repeat end register	GRER	WO	350	Cleared
Graphics Control Registers	**GxxR**	**—**	**Offset[2]**	**—**
Mode register	GMOR	RW	030	00100000
Deep register	GDER	RW	050	0050001C
Slope-no-go register 7	GSNR7	WO	11C	Undefined
Slope-no-go register 6	GSNR6	WO	118	Undefined
Slope-no-go register 5	GSNR5	WO	114	Undefined
Slope-no-go register 4	GSNR4	WO	110	Undefined
Slope-no-go register 3	GSNR3	WO	10C	Undefined
Slope-no-go register 2	GSNR2	WO	108	Undefined
Slope-no-go register 1	GSNR1	WO	104	Undefined
Slope-no-go register 0	GSNR0	WO	100	Undefined
Copy buffer register 7	GCBR7	RW	01C	Cleared
Copy buffer register 6	GCBR6	RW	018	Cleared
Copy buffer register 5	GCBR5	RW	014	Cleared
Copy buffer register 4	GCBR4	RW	010	Cleared
Copy buffer register 3	GCBR3	RW	00C	Cleared
Copy buffer register 2	GCBR2	RW	008	Cleared
Copy buffer register 1	GCBR1	RW	004	Cleared

[2] Address = hexadecimal offset into PDBR0 register space.

(continued on next page)

Table B–2 (Cont.) 21130 Register Summary

Name	Mnemonic	Access	Address	Reset State
Graphics Control Registers	**GxxR**	**—**	**Offset[2]**	**—**
Copy buffer register 0	GCBR0	RW	000	Cleared
Pixel shift register	GPSR	RW	038	Cleared
Address register	GADR	WO	03C	Cleared
Data register	GDAR	RW	080	Cleared
Foreground register	GFGR	RW	020	Cleared
Background register	GBGR	RW	024	Cleared
Raster operation register	GOPR	RW	034	00000003
Pixel mask register (one shot)	GPXR	RW	02C	Cleared
Pixel mask register (persistent)	GPXR	WO	05C	Cleared
Bresenham 1 register	GB1R	RW	040	Cleared
Bresenham 2 register	GB2R	RW	044	Cleared
Bresenham 3 register	GB3R	RW	048	Cleared
Bresenham width register	GBWR	WO	09C	Cleared
DMA base address register	GDBR	RW	098	Cleared
Dither row register	GDRR	RW	0B0	Cleared
Dither column register	GDCR	RW	0B4	Cleared
Scaled-copy control register	GSCR	RW	0C4	Cleared
Hardware Cursor Registers	**CxxR**	**—**	**Offset[2]**	**—**
Cursor mode register	CMOR	RW	0EC	Cleared
Cursor base address register	CCBR	RW	060	Cleared
Cursor XY register	CXYR	RW	074	Cleared
Video Control Registers	**VIxxR**	**—**	**Offset[2]**	**—**
Video base address register	VIVBR	RW	06C	Cleared
Video valid register	VIVVR	RW	070	00001400
Video scanline increment register	VISIR	RW	0CC	Cleared
Video line width register	VILWR	RW	0D0	Cleared
Video Format Registers	**VFxxR**	**—**	**Offset[2]**	**—**
Video pixel format register	VFPFR	RW	0D4	Cleared
Video pixel occlusion bitmap base address register	VFOBR	RW	0E0	Cleared

[2] Address = hexadecimal offset into PDBR0 register space.

(continued on next page)

Table B–2 (Cont.) 21130 Register Summary

Name	Mnemonic	Access	Address	Reset State
Video Format Registers	**VFxxR**	**—**	**Offset[2]**	**—**
Video pixel occlusion bitmap current address register	VFOAR	RW	1F4	Cleared
Video current refresh address register	VFCRR	RO	1FC	Cleared
Alternate video control register	VFAVR	RW	0E8	Cleared
Palette and DAC Registers	**DxxR**	**—**	**Offset[3]**	**—**
Palette and DAC RAM write address register	DPWR	RW	1000	Undefined
Palette and DAC RAM color register	DPCR	RW	1004	Undefined
Palette and DAC pixel mask register	DPMR	RW	1008	Undefined
Palette and DAC RAM read address register	DPRR	RW	100C	Undefined
Palette and DAC cursor write address register	DCWR	RW	1010	Undefined
Palette and DAC cursor color register	DCCR	RW	1014	Undefined
Palette and DAC command register 0	DCOR0	RW	1018	Cleared
Palette and DAC cursor read address register	DCRR	RW	101C	Undefined
Reserved	—	—	1020	—
Reserved	—	—	1024	—
Palette and DAC status register	DSTR	RO	1028	Undefined
Reserved	—	—	102C	—
Palette and DAC command register 1	DCOR1	RW	1030	Cleared
Palette and DAC red signature register	DRSR	RW	1034	Undefined
Palette and DAC green signature register	DGSR	RW	1038	Undefined
Palette and DAC blue signature register	DBSR	RW	103C	Undefined
VGA External and General Registers	**VExxxR**	**—**	**Index[4]**	**—**
VGA miscellaneous output register	VEMISR	WO RO	3C2 3CC	Undefined

[2]Address = hexadecimal offset into PDBR0 register space.

[3]Address = hexadecimal offset into PDBR1 memory space.

[4]Address = hexadecimal address (3xx) or index for VGA registers.

(continued on next page)

Table B–2 (Cont.) 21130 Register Summary

Name	Mnemonic	Access	Address	Reset State
VGA External and General Registers	**VExxxR**	**—**	**Index[4]**	**—**
VGA feature control register	VEFCOR	WO	3BA[5]	Undefined
		WO	3DA[6]	
		RO	3CA	
VGA input status 0 register	VEIS0R	RO	3C2	Undefined
VGA input status 1 register	VEIS1R	RO	3BA[5]	Undefined
		RO	3DA[6]	
VGA Sequencer Registers	**VSxxxR**	**—**	**Index[4]**	**—**
VGA sequencer index register	VSINXR	RW	3C4	Undefined
VGA sequencer data register	VSDATR	RW	3C5	Undefined
VGA sequencer reset register	VSRESR	RW	0	Undefined
VGA sequencer clocking mode register	VSCMOR	RW	1	Undefined
VGA sequencer plane mask register	VSPLMR	RW	2	Undefined
VGA sequencer character map select register	VSCMSR	RW	3	Undefined
VGA sequencer memory mode register	VSMMOR	RW	4	Undefined
VGA CRT Controller Registers	**VCxxxR**	**—**	**Index[4]**	**—**
VGA CRTC index register	VCINXR	RW	3B4[5]	Undefined
			3D4[6]	
VGA CRTC data register	VCDATR	RW	3B5[5]	Undefined
			3D5[6]	
VGA CRTC horizontal total register	VCHTOR	RW	0	Undefined
VGA CRTC horizontal display end register	VCHDER	RW	1	Undefined
VGA CRTC start horizontal blank register	VCHBSR	RW	2	Undefined
VGA CRTC end horizontal blank register	VCHBER	RW	3	Undefined
VGA CRTC start horizontal sync register	VCHSSR	RW	4	Undefined
VGA CRTC end horizontal sync register	VCHSER	RW	5	Undefined
VGA CRTC vertical total register	VCVTOR	RW	6	Undefined
VGA CRTC overflow register	VCOVRR	RW	7	Undefined
VGA CRTC preset row register	VCPROR	RW	8	Undefined
VGA CRTC maximum scanline register	VCMSLR	RW	9	Undefined
VGA CRTC cursor start register	VCCUSR	RW	0A	Undefined
VGA CRTC cursor end register	VCCUER	RW	0B	Undefined

[4]Address = hexadecimal address (3xx) or index for VGA registers.

[5]Monochrome

[6]Color

(continued on next page)

Table B–2 (Cont.) 21130 Register Summary

Name	Mnemonic	Access	Address	Reset State
VGA CRT Controller Registers	**VCxxxR**	**—**	**Index[4]**	**—**
VGA CRTC start address high register	VCSAHR	RW	0C	Undefined
VGA CRTC start address low register	VCSALR	RW	0D	Undefined
VGA CRTC cursor location high register	VCCLHR	RW	0E	Undefined
VGA CRTC cursor location low register	VCCLLR	RW	0F	Undefined
VGA CRTC start vertical sync register	VCVSSR	RW	10	Undefined
VGA CRTC end vertical sync register	VCVSER	RW	11	Undefined
VGA CRTC end vertical display register	VCVDER	RW	12	Undefined
VGA CRTC offset register	VCOFFR	RW	13	Undefined
VGA CRTC underline row scan register	VCULRR	RW	14	Undefined
VGA CRTC start vertical blanking register	VCVBSR	RW	15	Undefined
VGA CRTC end vertical blanking register	VCVBER	RW	16	Undefined
VGA CRTC mode control register	VCMODR	RW	17	Undefined
VGA CRTC line compare register	VCLCMR	RW	18	Undefined
VGA Extended Registers	**VXxxxR**	**—**	**Index[4,7]**	**—**
VGA extended paging control register	VXPCOR	RW	8D	Undefined
VGA extended host page offset A register	VXHPAR	RW	90	Undefined
VGA extended host page offset B register	VXHPBR	RW	91	Undefined
VGA extended split-screen start address low byte register	VXSALR	RW	93	Undefined
VGA extended split-screen start address high byte register	VXSAHR	RW	94	Undefined
VGA extended interlace control register	VXICOR	RW	97	Undefined
VGA extended equalization start register	VXEQSR	RW	9A	Undefined
VGA extended equalization end register	VXEQER	RW	9B	Undefined
VGA extended half-line register	VXHLNR	RW	9C	Undefined
VGA extended timing control A register	VXTCAR	RW	9D	Undefined
VGA extended timing control B register	VXTCBR	RW	9E	Undefined
VGA extended video FIFO control register	VXFCOR	RW	A0	Undefined
VGA extended clock control A register	VXCKAR	RW	A1	<7:1> = undefined, <0> = 0
VGA extended clock control B register	VXCKBR	RW	A2	Undefined
VGA extended interface control register	VXEICR	RW	A3	Undefined

[4] Address = hexadecimal address (3xx) or index for VGA registers.

[7] Indexed by VGA CRTC index register (VCINXR)

(continued on next page)

Table B–2 (Cont.) 21130 Register Summary

Name	Mnemonic	Access	Address	Reset State
VGA Graphics Controller Registers	**VGxxxR**	**—**	**Index[4]**	**—**
VGA graphics controller index register	VGINXR	RW	3CE	Undefined
VGA graphics controller data register	VGDATR	RW	3CF	Undefined
VGA graphics controller set/reset register	VGSRER	RW	0	Undefined
VGA graphics controller enable set/reset register	VGESRR	RW	1	Undefined
VGA graphics controller color compare register	VGCCMR	RW	2	Undefined
VGA graphics controller data rotate register	VGDROR	RW	3	Undefined
VGA graphics controller read map select register	VGRMSR	RW	4	Undefined
VGA graphics controller mode register	VGMODR	RW	5	Undefined
VGA graphics controller miscellaneous register	VGMISR	RW	6	Undefined
VGA graphics controller color don't care register	VGCDCR	RW	7	Undefined
VGA graphics controller bit mask register	VGBMKR	RW	8	Undefined
VGA Attribute Controller Registers	**VAxxxR**	**—**	**Index[4]**	**—**
VGA attribute controller index/data register	VAIXDR	WO RO	3C0 3C1	Undefined
VGA attribute controller palette register	VAPALR	RW	00:0F	Undefined
VGA attribute controller mode register	VAMODR	RW	10	Undefined
VGA attribute controller overscan register	VAOSCR	RW	11	Undefined
VGA attribute controller color plane enable register	VACPER	RW	12	Undefined
VGA attribute controller pixel panning register	VAPXPR	RW	13	Undefined
VGA attribute controller color select register	VACSLR	RW	14	Undefined
VGA Color Registers	**VPxxxR**	**—**	**Index[4]**	**—**
VGA color pixel address write mode register	VPPAWR	RW	3C8	Undefined
VGA color pixel address read mode register	VPPARR	WO	3C7	Undefined
VGA color DAC state register	VPDSTR	RO	3C7	Undefined
VGA color pixel data register	VPPDAR	RW	3C9	Undefined
VGA color pixel mask register	VPPMAR	RW	3C6	Undefined

[4]Address = hexadecimal address (3xx) or index for VGA registers.

C

Technical Support, Ordering Information, and Associated Literature

Technical Support

If you need technical support or help deciding which literature best meets your needs, call the Semiconductor Information Line:

United States and Canada	**1–800–332–2717**
Outside North America	**+1–508–628–4760**

Ordering Digital Semiconductor Products

To order the DECchip 21130 PCI Integrated Graphics and Video Accelerator and the associated evaluation board, contact your local distributor.

You can order the following semiconductor products from Digital:

Product	Order Number
DECchip 21130 PCI Integrated Graphics and Video Accelerator	21130–AA
DECchip 21130 PCI Integrated Graphics and Video Accelerator Evaluation Board	21A30–OA

Ordering Associated Semiconductor Literature

The following table lists some of the available Digital Semiconductor literature. For a complete list, contact the Digital Semiconductor Information Line.

Title	Order Number
Alpha Architecture Reference Manual[1]	EY–L520E–DP–YCH
DECchip 21130 PCI Integrated Graphics and Video Accelerator Evaluation Board User's Guide	EC–QHV0B–TE
DECchip 21130 PCI Integrated Graphics and Video Accelerator Product Brief	EC–QD2NA–TE

[1]To order and purchase the *Alpha Architecture Reference Manual*, call **1–800–DIGITAL** from the U.S. or Canada, or contact your local Digital office, or technical or reference bookstore where Digital Press books are distributed by Prentice Hall.

Ordering Third-Party Literature

You can order the following third-party literature directly from the vendor:

Title	Vendor
PCI Local Bus Specification, Revision 2.0 PCI Multimedia Design Guide, Revision 1.0 PCI System Design Guide	PCI Special Interest Group 1–800–433–5177 (U.S.) 1–503–797–4207 (International) 1–503–234–6762 (FAX)
VESA standards: • Display Data Channel Standard, Version 1.0, Revision 0 • Display Power Management Signaling (DPMS) Proposal, Version 1.0p, Revision 0.7p • VESA Monitor Timing Proposed Standard for 640X480, 800X600, and 1280X1024 at 75 Hz, VDMT 75HZ Rev 1.2P • VESA Advanced Feature Connector (VAFC) Standard, Version 1.0 • VESA Advanced Feature Connector (VAFC) Proposal, Version 1.0p, Revision 0.4	VESA 2150 N. First Street, Suite 440 San Jose, CA 95131–2029 FAX: 1–408–435–8225

Title	Vendor
Data Handbook for I^2C Peripherals for Microcontrollers	Philips Semiconductors Contact your nearest Philips Semiconductors national organization.

Index

1-shot GPXR, 8–67
2D accelerator (2DA), 7–1
  expected operation during VGA mode, 11–5
  memory space, 7–2
    base address 0, 7–2
    base address 1, 7–10
2DA-to-VGA mode switching, 11–4
4:2:2 output (422OUT) bit, 8–77
4:2:2 UYVY format, 10–45
4:2:2 VYUY format, 10–46
4:2:2 YVYU
  destination pixel format, 10–45
  format, 10–45
4:4:4 αVYU format, 10–44
8-bpp screen-to-screen copy repeat loop, 11–21
8 $\times$ 8 8-bpp pattern fill repeat loop, 11–20
16-bpp and 32-bpp RGB formats, 10–46
256-color mode (256CM) bit, 8–179

A

Abbreviations, xx
Aborted DMA transaction termination, 9–11
Absolute dx field, 8–30, 8–50
Absolute dy field, 8–30, 8–50
Absolute maximum ratings, 4–1
ac specifications, 4–6
Access abbreviations defined, xx
Access granularity, PCI, 9–5
Address attributes field, 8–184
Address convention, xxi
Address increment
  1 field, 8–68
  2 field, 8–70
Address mapping, device, 11–1
Address mask field, 8–47
Address register (GADR), 8–56
  access, 11–28
  status (ARS) bit, 8–43
Address space, 7–1
Address state (AS) field, 8–110, 8–193
Address stepping
  enable (ASE) bit, 8–13
  PCI, not supported, 9–12
Address wrap (AW) bit, 8–155
Addresses
  color register address field, 8–192
  device base address field, 11–1
  frame buffer address space in 64-bit and 32-bit modes, 11–23
  frame buffer destination address field, 8–38
  frame buffer source address field, 8–38
  RAM LUT, 8–96
  video address in 64-bit and 32-bit modes, 11–24
Alias space
  See Base address 0 register space
Aligned convention, xxi
Alpha CPU programming, 11–26
Alpha documentation, C–2
Alphabetical list of registers, B–1
Alphanumeric or graphics mode (GAM) bit, 8–187

Alternate control space, 7–9
GCTR writes, 8–37
write targets, 7–9
Alternate drawing mechanism, 8–29
Alternate register space, 7–10 to 7–11
Alternate video control register (VFAVR), 8–103
Application specific data (ASD) field, 7–15
Architecture, internal, 2–1
Assigning pixel shift values, 10–25
Associated literature, C–2
Asynchronous reset (AR) bit, 8–124
Attribute controller registers
See VGA attribute controller registers
Attribute data field, 8–185

B

Back-to-back capable (BBC) bit, 8–12
Back-to-back enable (BBE) bit, 8–12
Background as a function of bitmap depth, 8–61
Background field, 8–60
Background register (GBGR), 8–60
Backward copies, 10–25
Bandwidth (BW) bit, 8–125
Base address
device base address field, 11–1
high field, 8–75
low field, 8–75
register, DMA (GDBR), 8–74
Base address 0
alternate control space, 7–9
core space, 7–3
memory space, 7–2
register alias space, 7–5 to 7–6
register space, 7–5 to 7–7
Base address 1
alternate register space, 7–10
GPP space, 7–11
memory space, 7–10
MISR space, 7–12
palette and DAC register space, 7–12
ROM sparse space, 7–13
VGA register map, 7–11
Base class field, 8–16
Basic programming model, 1–5
extensions, 1–6
BIOS ROM
See Expansion ROM, Flash ROM, ROM
Bit descriptions
8/9, 8–126
AR, 8–124
ARS, 8–43
ASE, 8–13
AW, 8–155
BBC, 8–12
BBE, 8–12
BIA, 8–187
BLANK, 8–90
BM, 8–13, 11–2
BR3S, 8–43
Burst, 8–166
Busy, 8–25
BW, 8–125
C4, 8–129
CAS precharge period, 8–165
CB2, 8–155
CB4, 8–153
CBS, 8–44
CE, 8–44
CEN, 8–145
256CM, 8–179
CME, 8–187
CMS, 8–156
COE, 8–181
Color don’t care, 8–182
CR, 8–137
CS, 8–117
CSEN, 8–162
CVSI, 8–150
DC, 8–125
DDCDI, 8–89
DDCDO, 8–88
DDCSB, 8–88
DE, 8–21, 8–121
DEV, 8–12
DITHEN, 8–77
DPD, 8–12
DPE, 8–12

Bit descriptions (cont’d)
DRS, 8–111
DW, 8–153
EM, 8–129
EOFEN, 8–27
EOFST, 8–27
ER, 8–118
EVSI, 8–150
FC0, 8–119
FC1, 8–119
FRWE, 8–46
GAM, 8–187
GCC, 8–187
GE, 8–44
GIB, 8–46
GIEN, 8–27
GIST, 8–27
GM, 8–181
GSE, 8–111
HPAGE, 8–158
HR, 8–155
HRS, 8–155
HSP, 8–117
INLACE, 8–162
INTR, 8–27
IO, 8–13
IOA, 8–118
IRQEN, 8–166
LADMD, 8–158
MA, 8–12
MCD, 8–166
MCS, 8–23
MD32, 8–46
MS, 8–13
MSEL, 8–112
O/E, 8–129, 8–179
422OUT, 8–77
PAN, 8–187
PAS, 8–184
PB, 8–117
PCS, 8–23
PDE, 8–111
PER, 8–13
PF, 8–18
PLD, 8–171

Bit descriptions (cont’d)
PMS, 8–43
POBE, 8–91
POBM, 8–91
PSEL, 8–187
PW, 8–187
RM, 8–179
RMD, 8–166
RWS, 8–110, 8–193
S4, 8–125
SA, 8–128
SAEN, 8–112
SAH, 8–128
SB, 8–128
SBH, 8–128
SBLNK, 8–89
SBS, 8–89
SC<5:4>, 8–191
SD, 8–144
SE, 8–111
SEN, 8–13
SL, 8–125
SO, 8–125
Space, 8–18
SPOBE, 8–91
SR, 8–124, 8–179
SRC, 8–149
SRES, 8–23
SRS, 8–155
SS, 8–110, 8–120, 8–193
SSE, 8–12
SVV, 8–89
TAM, 8–12
TAT, 8–12
TCLKD, 8–89
TCLKI, 8–88
TCLKO, 8–88
TCS, 8–23
TVS, 8–171
VAFC BC, 8–104
VAFC D, 8–103
VAFC E, 8–104
VAFC FS, 8–103
VAFC IW, 8–103
VAFC M, 8–103

Bit descriptions (cont'd)
VAFC OS, 8–103
VGAAD, 8–158
VGAE, 8–46
VPS, 8–13
VR, 8–121
VRI, 8–120
VSEG, 8–159
VSP, 8–117
VSS, 8–119
VV, 8–90
VVDS, 8–168
WB, 8–155
WP, 8–149
WPG, 8–46
YUVCEN, 8–77
Bit mask field, 8–183
Bit notation conventions, xxi
Bit-block transfer (BitBlt), 1–2, 11–5
copy mode example, 11–6
stretchBlt, 1–2
Bitmap destination (DB) field, 8–64
Bitmap source (SB) field, 8–44
Bitmap width field, 8–73
Blank
synchronized (SBLNK) bit, 8–89
VAFC blank control (VAFC BC) bit, 8–104
VAFC enable, 12–12
Blank and sync source (SBS) bit, 8–89
BLANK bit, 8–90
blank# signal description, 3–4
Blink or intensity attribute (BIA) bit, 8–187
Block diagram, 2–1
blue signal description, 3–4
Blue signature field, 8–114
Boolean raster operation table, 8–64
bpp abbreviation, xx
Bresenham 1 register (GB1R), 8–68
line mode, 8–68
scaled-copy mode, 8–69
Bresenham 2 register (GB2R), 8–70
Bresenham 3 register (GB3R), 8–71
line mode, 8–71
scaled-copy mode, 8–72
Bresenham 3 register (GB3R) (cont'd)
status (BR3S) bit, 8–43
Bresenham engine, 1–2, 2–5
Bresenham setup hardware, 2–4
Bresenham width register (GBWR), 8–73
Burst read cycles, unsupported, 9–3
Bus master enable (BM) bit, 8–13, 11–2
Bus mastering, 11–2
Bus parking, 9–11
Busy bit, 8–25
Byte mask field, 8–63
Byte or word mode select (WB) bit, 8–155
Byte panning (BPAN) field, 8–143
Byte shifter, 10–22

C

Cap ends
(CE) bit, 8–44
specifying, 10–60
CAS precharge period bit, 8–165
CAS width field, 8–165
cas<7:0># signal description, 3–4
Caution convention, xxi
Chain 4 (C4) bit, 8–129
Chain odd/even (COE) bit, 8–181
Clear vertical sync interrupt (CVSI) bit, 8–150
Clock
See also Core clock, Dot clock, Memory clock, Pixel clock, Test clock
Clock control, 2–13, 12–15
register (PCCR), 8–23
Clock domains, 4–3
PCI clock, 4–7
pixel clock, 4–23
VAFC clock, 4–23
Clock frequency tables
memory clock, 8–24
pixel clock, 8–170
Clock generation, 12–16
Clock source select (CS) bit, 8–117
Color compare field, 8–176

Color don't care bit, 8–182
Color expansion, 1–2
Color or monochrome emulation (CME) bit, 8–187
Color plane enable field, 8–189
Color register
  See also VGA color registers
  address field, 8–192
  data field, 8–194
Column address strobe
  See CAS
Command buffer
  See Command FIFO
Command FIFO, 2–1
  determining entry availability, 10–48
Command parser, 2–3
Command status register (MCSR), 8–25
comp signal description, 3–4
Compatibility mode support (CMS) bit, 8–156
Compatible read (CR) bit, 8–137
Composite sync enable (CSEN) bit, 8–162
Configuration
  See also PCI configuration
  frame buffer, 2–10, 12–5
  operations, 9–1
  space, 7–2
Continue register (GCTR), 8–35
  access, 11–28
  indirect frame buffer addressing, 8–36
  line or span continuation, 8–36
  read, 8–37
  write, 8–35
    alternate control space, 8–37
    line mode, 8–36
Conventions, xx to xxiii
Copy
  backward, 10–25
  forward span, 10–23
    primed, 10–26
  host-to-screen, 11–8
  screen-to-screen, 11–5
    8-bpp repeat loop example, 11–21
Copy buffer, 1–2, 2–8
  layout, 8–51, 10–30
  operation, 10–29
    programmed I/O, 10–31
  registers (GCBR<7:0>), 8–53
    fast frame buffer access, 10–32
    write requirement, 8–53
  status (CBS) bit, 8–44
  write pointer, 8–53
Copy direction flag, 8–44, 8–55, 10–28
Copy mode, 1–6, 1–7, 10–19
  64-byte unmasked span, 10–29
  BitBlt example, 11–6
  span limits, 10–21
Copy-64 destination register (GCDR), 8–38
  write requirement, 8–39
Copy-64 source register (GCSR), 8–38
  write requirement, 8–39
Copy-64A destination register (GCADR), 8–40
Copy-64A source register (GCASR), 8–40
Core clock, 12–15
Core registers, 2–7
  See also Registers
Core space, 7–3
core_clk signal description, 12–15
Count by 2 (CB2) bit, 8–155
Count by 4 (CB4) bit, 8–153
CRT controller (CRTC), 1–3
  registers, *see* VGA CRTC registers
Current refresh address field, 8–102
Cursor, 1–3
  color displayed with monochrome overlay, 11–11
  generation, 2–12
  pixel value bit description, 8–83
  value bits to pixels mapping, 8–82
Cursor address bit 16 (CURA16), 8–158
Cursor base address
  field, 8–84
  register (CCBR), 8–84
Cursor color data field, 8–108
Cursor enable (CEN) bit, 8–145

Cursor end field, 8–145
Cursor location
  high field, 8–148
  low field, 8–148
Cursor mode
  (CM) field, 8–82
  register (CMOR), 8–82
Cursor read address field, 8–107
Cursor registers, 8–82
  See also Palette and DAC registers, VGA CRTC registers
Cursor skew field, 8–145
Cursor start field, 8–145
Cursor write address field, 8–107
Cursor X position field, 8–85
Cursor XY register (CXYR), 8–85
Cursor Y position field, 8–85

D

dac_Vdd signal description, 3–4
dac_Vss signal description, 3–4
Data parity error detected (DPD) bit, 8–12
Data register (GDAR), 8–58
  fill modes, 8–58
  line mode, 8–59
  write requirement, 8–58, 8–59
Data stepping, PCI, not supported, 9–12
Data units defined, xxii
dc
  operating specifications, 4–5
  parameters, 4–6
  specifications, 4–3
DDC, 12–14
  data input (DDCDI) bit, 8–89
  data output (DDCDO) bit, 8–88
  sync bypass (DDCSB) bit, 8–88
ddc_data signal description, 3–4
Decode enable (DE) bit, 8–21
Deep register (GDER), 8–46
  software scheduling requirement, 2–3
Destination alignment, 10–22
Destination bitmap (DB) field, 8–64
Destination operands, 10–5
  according to mode, 10–5
Detected parity error (DPE) bit, 8–12
Device address mapping, 11–1
Device base address
  field, 11–1
  LSBs field, 8–18
Device ID field, 8–11
Device select timing (DEV) bit, 8–12
Diagnostic (DIA) field, 8–121
Digital differential analyzer (DDA), 11–17
Digital-to-analog converter (DAC), 1–3
  See also Palette and DAC
  resolution select (DRS) bit, 8–111
Direct memory access
  See DMA
Display data channel
  See DDC
Display enable
  (DE) bit, 8–121
  skew field, 8–137
Display modes, VESA, 1–4
Display power management signaling
  See DPMS
Dither
  logic, 2–5
  mathematics, 11–8
  phases, 11–9
Dither column
  field, 8–81
  register (GDCR), 8–81
Dither enable (DITHEN) bit, 8–77
Dither row
  field, 8–81
  register (GDRR), 8–81
Dithering, 1–2
DMA address field, 8–74
DMA base-address register (GDBR), 8–74
  DMA-read copy mode, 8–74
  scaled-copy mode, 8–75
DMA engine, 1–2
DMA read
  FIFO, 2–7
  transfers, 9–10

DMA-read copy mode, 10–33
edge mask for short spans, 10–37
edge mask settings, 10–36
GB1R, 8–74
operation, 10–37
Documentation, C–2
Dot clock, 12–16
divide by 2 (DC) bit, 8–125
divide by 8 or 9 (8/9) bit, 8–126
divisor field, 8–162
variable dot clock select (VVDS) bit, 8–168
Double-word mode (DW) bit, 8–153
DPMS, 1–4, 12–14
field, 8–89
states, 12–14
DRAM burst mode (Burst) bit, 8–166
DRAW# field, 8–77
Drawing
clipped lines, 11–16
lines with slope registers, 10–52, 10–54
octants, 8–31
Drawing mechanism
alternate, 8–29
standard, 8–29
Dword, defined, xxii
dx field, 8–30, 8–50
dxGE0 field, 8–33
dxGEdy field, 8–33
dy field, 8–30, 8–50
dyGE0 field, 8–33

E

Edge mask
for short spans in DMA-read copy mode, 10–37
settings in DMA-read copy mode, 10–36
EEPROM
See Expansion ROM, Flash ROM, ROM
Electrical specifications, 4–1
ac, 4–6
dc, 4–3
Enable RAM (ER) bit, 8–118
Enable set/reset plane field, 8–175
Enable vertical sync interrupt (EVSI) bit, 8–150
End horizontal blank
(EHB) bit <5>, 8–139
LSBs field, 8–138
End horizontal sync field, 8–139
End pixel (EPIX) field, 8–77
End vertical blanking field, 8–154
End vertical display
(EVD<8>) bit 8, 8–142
(EVD<9>) bit 9, 8–142
LSBs field, 8–151
End vertical sync field, 8–150
End-of-frame
enable (EOFEN) bit, 8–27
status (EOFST) bit, 8–27
Equalization end field, 8–163
Equalization start
<7:0> field, 8–163
<9:8> field, 8–162
Error increment
1 field, 8–68
2 field, 8–70
evideo# signal description, 3–4
Exclusive access, PCI, not supported, 9–12
Expansion ROM, 11–3
See also Flash ROM, ROM
Extended memory (EM) bit, 8–129
Extended registers
See VGA extended registers
Extending a single line, 10–59
Extending and linking 2D lines, 10–57
Extensions to the basic programming model, 1–6
Extents convention, xxii
External and general registers
See VGA external and general registers
External to DECchip 21130, xxii

F

Feature connector, 2–10, 12–5
  See also VAFC
Feature control
  <0> (FC0) bit, 8–119
  <1> (FC1) bit, 8–119
Features, 1–1
FIFOs
  command, 2–1
    read pointer field, 8–25
    write pointer field, 8–25
  DMA read, 2–7
  video
    depth field, 8–167
    enable bit, 8–167
    reset bit, 8–167
    wrap reset (FWSTN) bit, 8–171
    wrapped (FWRAP) bit, 8–171
Fill mask field, 8–58
Fill mode, 2–4
  opaque, 10–14
  opaque extended-pattern, 10–16
  stipple, 1–6
  transparent, 10–17
  transparent extended-pattern, 10–18
Filling
  8 × 8 8-bpp pattern repeat loop example, 11–20
  monochrome brush, 11–14
  monochrome or bitonal brush repeat loop example, 11–19
  non-monochrome brush, 11–14
Fills, 11–13
  solid, 11–13
Flags
  copy direction, 8–44, 8–55, 10–28
  write memory barrier, 8–26
Flash ROM
  See also Expansion ROM, ROM
  write enable (FRWE) bit, 8–46
Flicker-free monochrome overlay, 11–10
Flushing the residue register
  copy mode, 10–26
  DMA-read copy mode, 10–36
Foreground as a function of bitmap depth, 8–61
Foreground field, 8–60
Foreground register (GFGR), 8–60
Forward span copy, 10–23
  primed, 10–26
Frame buffer
  access with copy buffer registers, 10–32
  configuration sensing, 12–5
  configurations, 2–10
  core space, 7–3
  interface, 12–1
  memory, 2–9
  mode-dependent write operations, 10–1
  VGA interface, 12–6
  writes, 2–4, 10–1
Frame buffer address
  destination field, 8–38
  field, 8–56
  source field, 8–38
Frame buffer address space in 64-bit and 32-bit modes, 11–23
Frame buffer and device access (FBDA), 2–8, 9–3
  requests, 2–6
Frame buffer color depth (FBCD) field, 8–91
fsadjust signal description, 3–5
Fully shadowed registers, 11–22
Function select field, 8–177
Functions not supported, 1–4
  PCI, 9–12

G

Generic peripheral port
  See GPP
Gib-endian
  (GIB) bit, 8–46
  support, 8–47
  transfer formats, 8–47

GPP, 1–3, 2–8
access restriction, 2–8, 7–12, 12–8
interface, 12–8
interrupts, 12–9
read and write access, 12–8
space, 7–11
GPP interrupt
enable (GIEN) bit, 8–27
status (GIST) bit, 8–27
gp_adr<16:0> signal description, 3–5
gp_cs# signal description, 3–5
gp_data<7:0> signal description, 3–5
gp_int# signal description, 3–5
gp_rdsel# signal description, 3–5
gp_reset# signal description, 3–5
gp_stb# signal description, 3–5
gp_wrsel# signal description, 3–5
Grant time, minimum field, 8–22
Graphics
device interface (GDI), 1–5
modes, table, 8–45
operations, 10–1
operations, invoking, 10–3
pipeline, 1–2
Graphics address field, 8–172
Graphics character codes (GCC) bit, 8–187
Graphics command
register writes, 10–2
registers, 8–29
Graphics control registers, 8–42
Graphics controller data field, 8–173
Graphics controller registers
See VGA graphics controller registers
Graphics environment(GE) bit, 8–44
Graphics mode (GM) bit, 8–181
Graphics or alphanumeric mode (GAM) bit, 8–187
grdy signal description, 3–5
green signal description, 3–5
Green signature field, 8–114
Green sync enable (GSE) bit, 8–111
Grid intersect quantization (GIQ) specification, 10–56

H

Half-line location field, 8–164
Hardware
interface, 12–1
mode restrictions, 2–10, 12–1
Hardware reset (HR) bit, 8–155
Header type field, 8–17
Height field, 8–144
Hexaword, defined, xxii
Horizontal blank
end
(EHB) bit <5>, 8–139
LSBs field, 8–138
start field, 8–137
Horizontal display end field, 8–136
Horizontal retrace select (HRS) bit, 8–155
Horizontal sync
delay field, 8–139
end field, 8–139
polarity (HSP) bit, 8–117
start field, 8–139
width field, 8–165
Horizontal total field, 8–135
Host page offset
A field, 8–160
B field, 8–160
Host page select (HPAGE) bit, 8–158
Host-to-screen
copy, 11–8
scaled-copy and video rendering pixel flow, 10–42
hsync signal description, 3–5

I

I^2C, 1–3, 2–9, 12–14
I/O address select (IOA) bit, 8–118
I/O space enable (IO) bit, 8–13
Ignore (IGN) convention, xx
Indirect frame buffer addressing, GCTR, 8–36

Initial error field, 8–71
Input/output
  See I/O
Inside pixel format
  field, 8–92
  table, 8–92
Interlaced enabled (INLACE) bit, 8–162
Internal architecture, 2–1
Interrupt acknowledge, ignored, 9–12
Interrupt enable (IRQEN) bit, 8–166
Interrupt line field, 8–22
Interrupt pin field, 8–22
Interrupt status
  (INTR) bit, 8–27
  register (MISR), 8–27
    space, 7–12
Interrupts
  clear vertical sync interrupt (CVSI) bit, 8–150
  enable vertical sync interrupt (EVSI) bit, 8–150
  end-of-frame
    enable (EOFEN) bit, 8–27
    status (EOFST) bit, 8–27
  GPP, 12–9
    interrupt enable (GIEN) bit, 8–27
    interrupt status (GIST) bit, 8–27
  routing, 11–3
  vertical retrace interrupt (VRI) bit, 8–120

L

L term field, VXCKBR, 8–169
Latency timer field, 8–17, 11–2
Latency, maximum field, 8–22
Length field, 8–71
Line compare
  (LC<8>) bit 8, 8–142
  (LC<9>) bit 9, 8–144
  LSBs field, 8–157
Line drawing, 1–2
  engine, 1–2
  under Win32, 11–17
  under X, 11–15
  with frame buffer writes, 10–52
Line drawing (cont'd)
  with slope registers, 10–54
Line mask field, 8–36, 8–59
Line mode, 1–6, 1–7
  GB1R, 8–68
  GB3R, 8–71
  GDAR, 8–59
  opaque, 1–6, 10–52
  transparent, 1–6, 10–62
Line or span continuation, GCTR, 8–36
Line width field, 8–87
Linear address mode (LADMD) bit, 8–158
Lines, 11–15
  extending a single line, 10–59
  extending and linking 2D lines, 10–57
  linking multiple lines, 10–60
  opaque drawing, 10–58
    sequence, 10–60
Linking and extending 2D lines, 10–57
Linking multiple lines, 10–60
Literature, C–2
LOCK cycle, PCI, not supported, 9–12
Longword, defined, xxii
Look-up table (LUT)
  See RAM LUT, ROM LUT

M

M term field
  PCCR, 8–23
  VXCKAR, 8–168
Magnification, 10–49
Mask data field, DPMR, 8–109
Mask field, VPPMAR, 8–195
Mask GPXR field, 8–67
Mask memory plane field, VSPLMR, 8–127
Master abort
  issued by 21130, 8–13
  (MA) bit, 8–12
Master operation, PCI, 9–9
Maximum latency field, 8–22
Mechanical specifications, 5–1
memaddr<8:0> signal description, 3–6

memdata<63:0> signal description, 3–6
Memory
  frame buffer, 2–9
  interface, 12–1
Memory barrier, write, 8–26
Memory clock, 12–15
  frequency table, 8–24
  source (MCS) bit, 8–23
Memory controller, 1–3, 2–5
Memory map <1:0> field, 8–181
Memory read, 9–2
  core space, 9–3
  interlock, 9–3
Memory space
  See also Core space, Register space
  2DA, 7–2
    base address 0, 7–2
    base address 1, 7–10
  enable
    field, 11–1
    (MS) bit, 8–13
  map, base address 1, 7–10
  organization, 7–3
  VGA, 7–2
Memory write, 9–2
  core space, 9–2
Memory, extended (EM) bit, 8–129
mem_clk signal description, 12–15
Minimum grant time field, 8–22
Minimum system, 1–5
Miscellaneous registers, 8–25
Mode 32 (MD32) bit, 8–46
Mode field
  graphics, 8–44
  scaled-copy, 8–76
MODE field, scaled-copy, 8–76
Mode register (GMOR), 8–43
Mode select (MSEL) bit, 8–112
Mode-dependent frame buffer write operations, 10–1
Mode-specific data field, 8–35
Modes
  copy, 1–6, 1–7, 10–19, 10–21
  DMA-read copy, 10–33
    operation, 10–37
Modes (cont'd)
  fill, 2–4
  graphics, 10–6
    table, 8–45
  graphics or alphanumeric mode (GAM) bit, 8–187
  line, 1–6, 1–7
    opaque, 1–6
    transparent, 1–6
  opaque bit-reversed stipple, 10–11
  opaque extended-pattern fill, 10–16
  opaque-fill, 10–14
    operation, 10–15
  opaque-line, 10–52
  opaque-stipple, 10–9
    operation, 10–10
  primary, 1–5
  restrictions, 2–10
  scaled-copy, 10–39
  simple, 1–6, 10–7
  stipple, 1–6
    opaque, 1–6
    transparent, 1–6
  stipple-fill, 1–6
  switching, 11–3
    2DA to VGA, 11–4
    VGA to 2DA, 11–3
  transparent extended-pattern fill, 10–18
  transparent-fill, 10–17
  transparent-line, 10–62
  transparent-stipple, 10–12
    operation, 10–13
    with pixel mask, 10–13
  VGA graphics controller write modes, 8–179
Monitor
  connection, 12–13
  timing generation, 2–11
Monochrome or bitonal brush fill repeat loop, 11–19
Multimedia, 1–1
  pipeline, 1–2
Multiplexer-to-CAS delay (MCD) bit, 8–166

Must be zero (MBZ) convention, xx

N

N term field, VXCKBR, 8–169
Nontrivially occluded windows, 10–48
Normal operating conditions, 4–2
Note convention, xxii
Numbering convention, xxii

O

Occluded
    See Unoccluded, Trivially occluded
Odd or even (O/E) bit, 8–129, 8–179
oeb# signal description, 3–6
Offset field, 8–152
opamp_Vdd signal description, 3–6
opamp_Vss signal description, 3–6
Opaque bit-reversed stipple mode, 10–11
Opaque fill mode, 10–14
    extended-pattern, 10–16
    GDAR, 8–58
    operation, 10–15
Opaque line drawing, 10–58
    sequence, 10–60
Opaque line mode, 1–6, 10–52
Opaque stipple mode, 1–6, 10–9
    operation, 10–10
Operands
    source and destination, 10–5
        according to mode, 10–5
Operating specifications, dc, 4–5
Ordering products, C–1
Outside pixel format
    field, 8–92
    table, 8–92
Overlays, 11–10
    data in 16-bpp and 32-bpp frame buffers, 11–13
    flicker-free monochrome, 11–10
    true 8-bpp, 11–12
    true monochrome, 11–11
Overscan color field, 8–188

P

P<5:4> select (PSEL) bit, 8–187
Page bit (PB), 8–117
Palette address source (PAS) bit, 8–184
Palette and DAC, 1–3, 2–13
    register space, 7–12
Palette and DAC registers, 8–104
    blue signature analysis register (DBSR), 8–114
    command register 0 (DCOR0), 8–111
    command register 1 (DCOR1), 8–112
    cursor color register (DCCR), 8–108
    cursor read address register (DCRR), 8–107
    cursor write address register (DCWR), 8–107
    green signature analysis register (DGSR), 8–114
    pixel mask register (DPMR), 8–109
    RAM color register (DPCR), 8–106
    RAM read address register (DPRR), 8–105
    RAM write address register (DPWR), 8–105
    red signature analysis register (DRSR), 8–114
    status register (DSTR), 8–110
Palette data field, 8–106, 8–186
Palette mask data field, 8–109
Palette read address field, 8–105
Palette snoop
    response table, 8–14
    (VPS) bit, 8–13
Palette write address field, 8–105
Parallel load strobe (PLD) bit, 8–171
Parameters
    copy mode, 10–19
    DMA-read copy mode, 10–33
    opaque-fill mode, 10–14
    opaque-line mode, 10–52, 10–54
    opaque-stipple mode, 10–9
    scaled-copy mode, 10–39

Parameters (cont'd)
simple mode, 10–7
transparent-fill mode, 10–17
transparent-line mode, 10–62
transparent-stipple mode, 10–12
Parity error response (PER) bit, 8–13
Parity, PCI, 9–11
Parser, command, 2–3
Parts ordering, C–1
PCI
aborted DMA transaction termination, 9–11
access granularity, 9–5
address stepping, not supported, 9–12
bus master enable bit, 11–2
bus mastering, 11–2
bus parking, 9–11
data stepping, not supported, 9–12
electrical specification conformance, 4–1
exclusive access, not supported, 9–12
functions not supported, 9–12
interrupt routing, 11–3
operations, 9–1
parity, 9–11
read data format, ROM sparse space, 7–14
PCI clock signal parameters, 4–7
PCI configuration
registers, *see* PCI registers
firmware, 11–1
operations, 9–1
PCI expansion ROM
See Expansion ROM, Flash ROM, ROM
PCI interface, 1–1, 2–1
PCI interrupt acknowledge, ignored, 9–12
PCI latency timer field, 11–2
PCI LOCK cycle, not supported, 9–12
PCI master operation, 9–9
PCI master transaction termination, 9–10
PCI registers, 2–1
class and revision register (PCRR), 8–16
clock control register (PCCR), 8–23
command and status register (PCSR), 8–12
PCI registers (cont'd)
device base address registers (PDBR0, PDBR1), 8–18
expansion ROM base address register (PRBR), 8–21
identification register (PIDR), 8–11
interrupt line register (PLIR), 8–22
latency timer and header type register (PLTR), 8–17
PCI special cycle, ignored, 9–12
PCI target operations, 9–4
PCI target transaction termination, 9–5
PCI transactions
to 2DA memory space, 9–5
to configuration space and expansion ROM space, 9–7
to VGA memory and I/O space, 9–8
PCI write-data format
copy mode, 10–19
DMA-read copy mode, 10–33
opaque-fill mode, 10–14
opaque-line mode, 10–53
opaque-stipple mode, 10–9
simple mode, 10–7
transparent-stipple mode, 10–12
PCI-to-VGA interface, 12–6
pci_ad<31:0> signal description, 3–6
pci_cbe<3:0># signal description, 3–6
pci_clk signal description, 3–6
pci_devsel# signal description, 3–6
pci_frame# signal description, 3–6
pci_gnt# signal description, 3–6
pci_idsel signal description, 3–7
pci_inta# signal description, 3–7
pci_irdy# signal description, 3–7
pci_par signal description, 3–7
pci_perr# signal description, 3–7
pci_req# signal description, 3–8
pci_rst# signal description, 3–8
pci_serr# enable (SEN) bit, 8–13
pci_serr# signal description, 3–8
pci_stop# signal description, 3–8
pci_trdy# signal description, 3–8

Persistent GPXR, 8–67
Pin characteristics, 4–4
Pin interfaces, 12–1
Pin summary, A–1
Pinout, 3–1
Pins, shared
 32-bit GPP and ROM modes, 12–2
 64-bit GPP and ROM modes, 12–3
 32-bit GPP and VAFC modes, 12–4
 VGA mode, 12–2
Pipeline
 graphics and multimedia video, 1–2
 pixel processing, 2–3
pixclk signal description, 3–8
Pixel clock, 12–15
 frequency table, 8–170
 signal parameters, 4–23
 source (PCS) bit, 8–23
Pixel engine, 2–4
Pixel format (PIXFOR) field, 8–78
Pixel mask
 field, 8–66
 register (GPXR), 8–66
  any mode, 8–67
  mask field, 8–67
  simple mode, 8–67
  stipple modes, 8–66
 status (PMS) bit, 8–43
Pixel merge function, 2–5
Pixel occlusion bitmap, 2–11, 8–98
 base address field, 8–100
 base address register (VFOBR), 8–100
 current address field, 8–101
 current address register (VFCRR), 8–102
 current address register (VFOAR), 8–101
 enable (POBE) bit, 8–91
 field, 8–98
 hardware restriction, 11–11
 mode (POBM) bit, 8–91
 synchronized enable (SPOBE) bit, 8–91
Pixel order (PIXORD) field, 8–78
Pixel panning
 field, 8–190
 (PAN) bit, 8–187
Pixel phase select (PPS) field, 8–112
Pixel processing pipeline, 2–3
Pixel shift
 field, 8–55
 register (GPSR), 8–55
Pixel width (PW) bit, 8–187
Pixels
 assigning shift values, 10–25
 formatting, 8–95
 inside and outside pixel formats table, 8–92
 inside format field, 8–92
 outside format field, 8–92
 VAFC input modes, 12–11
 VAFC output modes, 12–10
 variable pixel formats, 8–92 to 8–95
 YUV formats, 10–44
Pixels per longword (PIXLW) field, 8–78
pix_clk signal description, 12–15
pll_filter signal description, 3–8
pll_test signal description, 3–8
pll_Vdd signal description, 3–8
pll_Vss signal description, 3–8
Pointer, copy-buffer write pointer, 8–53
Postscaling filter restriction, 10–42
Power dissipation, 4–2
Power-down enable (PDE) bit, 8–111
Power-on self-test (POST) code, 8–16
Prefetchable (PF) bit, 8–18
Prescaling filter restriction, 10–42
Preset row (PROW) field, 8–143
Primary operating modes, 1–5
Priming the residue register
 copy mode, 10–26
 DMA-read copy mode, 10–36
Programmed I/O
 copy buffer operation, 10–31
 through CPU write buffer, 11–27
Programming, 11–1
 Alpha CPUs, 11–26
 basic programming model, 1–5
  extensions, 1–6
 the Bresenham scaler for unoccluded spans, 10–49

Programming interface field, 8–16
Pseudo-shadowed registers, 11–22

Q

Quadword, defined, xxii

R

RAM LUT, 1–3
  addressing, 8–96
RAM registers
  See Palette and DAC registers
RAMDAC
  See Palette and DAC
Ranges convention, xxii
RAS precharge period field, 8–166
RAS setup period field, 8–166
RAS to multiplexer delay (RMD) bit, 8–166
ras<2:0># signal description, 3–8
Raster operation
  field, 8–64
  register (GOPR), 8–63
  table, 8–64
Read as zero (RAZ) convention, xx
Read clears (RC) convention, xx
Read FIFO, DMA, 2–7
Read map select (RMS) field, 8–178
Read mode (RM) bit, 8–179
Read only (RO) convention, xxi
Reads
  DMA transfers, 9–10
  GCTR, 8–37
  GSNR<7:0>, 8–51
  GSWR, 8–33
  interlock, 9–3
  memory core space, 9–3
  memory interlock, 9–3
Read/write (RW) convention, xxi
Read/write one to clear (R/W1C) convention, xxi
Read/write state (RWS) bit, 8–110, 8–193
red signal description, 3–9
Red signature field, 8–114
Reduction, 10–49
ref signal description, 3–9
Refresh address, current field, 8–102
Refresh calculations, 11–25
Refresh cycles select (SRC) bit, 8–149
Register
  access abbreviations defined, xx
  load synchronization, 10–4
  space
    alternate register space map, 7–11
    base address 0, 7–5
    base address 0 map, 7–7
    MISR, base address 1, 7–12
    palette and DAC, base address 1, 7–12
    VGA alternate, base address 1, 7–10
    VGA register map, 7–11
Register alias space
  See Base address 0 register space
Registers
  address register (GADR), 8–56
  alphabetical list, B–1
  attribute controller registers
    See VGA attribute controller registers
  background register (GBGR), 8–60
  Bresenham
    1 register (GB1R), 8–68
    2 register (GB2R), 8–70
    3 register (GB3R), 8–71
    width register (GBWR), 8–73
  clock control register (PCCR), 8–23
  color registers
    See VGA color registers
  command status register (MCSR), 8–25
  continue register (GCTR), 8–35
  copy buffer registers (GCBR<7:0>), 8–53
  copy-64 destination register (GCDR), 8–38
  copy-64 source register (GCSR), 8–38
  copy-64A destination register (GCADR), 8–40
  copy-64A source register (GCASR), 8–40
  core registers, 2–7

Registers (cont'd)
CRTC registers
See VGA CRTC registers
cursor registers
See Cursor registers, Palette and DAC registers, VGA CRTC registers
data register (GDAR), 8–58
deep register (GDER), 8–46
dither column register (GDCR), 8–81
dither row register (GDRR), 8–81
DMA base-address register (GDBR), 8–74
extended registers
See VGA extended registers
external and general registers
See VGA external and general registers
figures conventions, xxiii
foreground register (GFGR), 8–60
graphics command registers, 8–29
graphics control registers, 8–42
graphics controller registers
See VGA graphics controller registers
interrupt status register (MISR), 8–27
miscellaneous registers, 8–25
mode register (GMOR), 8–43
palette and DAC registers
See Palette and DAC registers
PCI registers
See PCI registers
pixel mask register (GPXR), 8–66
pixel shift register (GPSR), 8–55
raster operation register (GOPR), 8–63
repeat begin register (GRBR), 8–41
repeat end register (GRER), 8–41
reset state, B–5
residue register
description, 10–22
priming and flushing, 10–26, 10–36
scaled-copy control register (GSCR), 8–76
sequencer registers
See VGA sequencer registers
shadowed registers, 11–22
slope registers (GSLR<7:0>), 8–30

Registers (cont'd)
slope-no-go registers (GSNR<7:0>), 8–50
span width register (GSWR), 8–33
summary, B–1
video control registers
See Video registers
video format registers
See Video registers
Related documentation, C–2
Rendering full frames, 10–46
Repeat begin register (GRBR), 8–41
Repeat count field, 8–99
Repeat end register (GRER), 8–41
Repeat loop examples, 11–19 to 11–21
Repeat number field, 8–41
Required software interlock, 10–51
Reserved (RES) convention, xxi
Reset state, registers, B–5
Residue register
description, 10–22
priming and flushing in copy mode, 10–26
priming and flushing in DMA-read copy mode, 10–36
Revision ID field, 8–16
RGB 16-bpp and 32-bpp formats, 10–46
ROM, 2–6, 7–2
See also Expansion ROM, Flash ROM
access restriction, 7–14, 12–8
ROM interface, 12–8
ROM LUT, 1–3
ROM read byte field, 7–15
ROM space, 7–2
ROM sparse space, 7–13
PCI read data field description, 7–15
PCI read data format, 7–14
ROM write enable (FRWE) bit, 8–46
rom_adr<17:0> signal description, 3–9
rom_ce# signal description, 3–9
rom_d<7:0> signal description, 3–9
rom_oe# signal description, 3–9
rom_we# signal description, 3–9

Rotate count field, 8–177
Row address strobe
  See RAS

S

Scaled video DMA repeat loop example, 11–20
Scaled-copy, 11–8
  PCI DMA start address, 10–40
Scaled-copy and video rendering pixel flow, 10–42
Scaled-copy control register (GSCR), 8–76
Scaled-copy mode, 10–39
  GB1R, 8–69
  GB3R, 8–72
  GDBR, 8–75
  operations table, 8–78
  PCI write data field description, 10–40
  PCI write data format, 10–40
Scaling, 10–48
  filters, 11–9
  occluded spans, 10–50
  unity scaling, 10–50
Scan double (SD) bit, 8–144
Scanline addresses, 11–25
Scanline increment field, 8–87
Screen off (SO) bit, 8–125
Screen parameters, 8–130
Screen-to-screen copy, 11–5
  8-bpp repeat loop example, 11–21
Select character generator A
  high order (SAH) bit, 8–128
  (SA) bit, 8–128
Select character generator B
  high order (SBH) bit, 8–128
  (SB) bit, 8–128
Select color
  <5:4> (SC<5:4>) bit, 8–191
  <7:6> (SC<7:6>) field, 8–191
Select horizontal retrace (HRS) bit, 8–155
Select host page (HPAGE) bit, 8–158
Select refresh cycles (SRC) bit, 8–149
Select row scan counter (SRS) bit, 8–155
Select vertical sync (VSS) bit, 8–119
Select word or byte mode (WB) bit, 8–155
Semiconductor
  documentation, C–2
  information line, C–1
Sense status (SS) bit, 8–110, 8–120, 8–193
Sequencer data field, 8–123
Sequencer index (SI) field, 8–122
Sequencer registers
  See VGA sequencer registers
Set/Reset plane field, 8–174
Setup-enable (SE) bit, 8–111
Shadowed registers, 11–22
Shared pins
  32-bit GPP and ROM modes, 12–2
  64-bit GPP and ROM modes, 12–3
  32-bit GPP and VAFC modes, 12–4
  VGA mode, 12–2
Sharpening filter (FSH) field, 8–77
Shift four (S4) bit, 8–125
Shift load (SL) bit, 8–125
Shift register (SR) bit, 8–179
Signaled system error (SSE) bit, 8–12
Signals
  active level, 3–15
  by direction, 3–11, 3–15
  by function, 3–11, A–1
  descriptions, 3–4, 12–15
  list, 3–1
  naming convention, xxiii
  notation convention, 3–1
  summary, A–1
Signature analysis enable (SAEN) bit, 8–112
Signature analysis registers
  See Palette and DAC registers
Simple mode, 1–6, 10–7
  GPXR, 8–67
Slope registers (GSLR<7:0>), 8–30
  drawing lines with, 10–54
  drawing octants, 8–31
  write requirement, 8–30

Slope-no-go registers (GSNR<7:0>), 8–50
read, 8–51
write, 8–50
write requirement, 8–50
Smoothing filter (FSM) field, 8–76
Snoop response table, 8–14
Snooped DAC write PCI transactions to VGA space, 9–9
Soft reset (SRES) bit, 8–23
Software interlock required, 10–51
Solid fills, 11–13
Source alignment, 10–22
Source bitmap (SB) field, 8–44
Source operands, 10–5
according to mode, 10–5
Space bit, 8–18
Span limits, copy mode, 10–21
Span or line continuation, GCTR, 8–36
Span starting and trailing edges, 10–51
Span width register (GSWR), 8–33
read, 8–33
write, 8–34
Sparse space
base address 1 memory space, 7–10
GPP space, 7–11
palette and DAC register space, 7–12
ROM, 7–13
ROM read field description, 7–15
VGA alternate register space, 7–10
Special cycle, ignored, 9–12
Specifications
electrical, 4–1
mechanical, 5–1
thermal, 6–1
Specifying cap ends, 10–60
Split-screen address bit 16 (SAA16), 8–158
Split-screen start address
<7:0> field, 8–161
<15:8> field, 8–161
Standard drawing mechanism, 8–29
Start address
high field, 8–147
low field, 8–147
Start horizontal
blank field, 8–137
sync field, 8–139
Start pixel (SPIX) field, 8–77
Start vertical blanking
LSBs field, 8–154
(SVB <9>) bit 9, 8–144
(SVB<8>) bit 8, 8–142
Start vertical sync
LSBs field, 8–149
(SVS<8>) bit 8, 8–142
(SVS<9>) bit 9, 8–142
Stencil
See Pixel occlusion bitmap
Stipple logic, 2–4
Stipple mode, 1–6
Stipple modes
GPXR, 8–66
opaque, 1–6, 10–9
opaque bit-reversed, 10–11
transparent, 1–6, 10–12, 10–13
Stipple-fill mode, 1–6
Stippling, monochrome brush, 11–14
StretchBlt, 1–2
Subclass field, 8–16
Supply current, 4–2
Sync
composite sync enable (CSEN) bit, 8–162
green sync enable (GSE) bit, 8–111
horizontal sync
delay field, 8–139
end field, 8–139
polarity (HSP) bit, 8–117
start field, 8–139
width field, 8–165
hsync signal description, 3–5
vertical sync
end field, 8–150
interrupt clear (CVSI) bit, 8–150
interrupt enable (EVSI) bit, 8–150
polarity (VSP) bit, 8–117
select (VSS) bit, 8–119
start (SVS<8>) bit 8, 8–142
start (SVS<9>) bit 9, 8–142
start LSBs field, 8–149

Sync
vertical sync (cont'd)
strobe (TVS) bit, 8–171
vsync signal description, 3–10
Sync and blank source (SBS) bit, 8–89
Synchronized blank (SBLNK) bit, 8–89
Synchronized pixel occlusion bitmap enable (SPOBE) bit, 8–91
Synchronized video valid (SVV) bit, 8–89
Synchronous reset (SR) bit, 8–124

T

Target abort
issued by 21130, 8–13
master (TAM) bit, 8–12
target (TAT) bit, 8–12
Target operations, PCI, 9–4
Technical support, C–1
Test clock, 12–16
input (TCLKI) bit, 8–88
output control (TCLKO) bit, 8–88
output disable (TCLKD) bit, 8–89
source (TCS) bit, 8–23
Test conditions, 4–2
test_in signal description, 3–9
Text, 11–19
Thermal specifications, 6–1
Third-party documentation, C–2
Tiling, non-monochrome brush, 11–14
Transaction termination
aborted DMA, 9–11
PCI master, 9–10
PCI target, 9–5
Transparent fill mode, 10–17
extended-pattern, 10–18
GDAR, 8–58
Transparent line mode, 1–6, 10–62
Transparent stipple mode, 1–6, 10–12
operation, 10–13
with pixel mask, 10–13
Trivially occluded or unoccluded target windows, 10–46
Trivially occluded target window, 10–47
Trivially occluded, defined, 10–47
True 8-bpp overlay in 16-bpp or 32-bpp frame buffers, 11–12
True monochrome overlay, 11–11
Type field, 8–18

U

Unaligned convention, xxi
Underline location field, 8–153
Unity scaling, 10–50
Unmasked span copies, 10–29
Unoccluded or trivially occluded target windows, 10–46
Unoccluded target window, 10–47
Unsupported functions, 1–4
PCI, 9–12

V

VAFC, 12–9
See also Feature connector
blank enable, 12–12
input screen resolutions, 12–13
input windows, 12–12
operation, 12–10
output screen resolutions, 12–12
pixel input modes, 12–11
pixel output modes, 12–10
port, 1–3, 2–12
signal ordering, 12–2
VAFC blank control (VAFC BC) bit, 8–104
VAFC clock domain signal parameters, 4–23
VAFC direction (VAFC D) bit, 8–103
VAFC enable (VAFC E) bit, 8–104
VAFC frequency select (VAFC FS) bit, 8–103
VAFC input window (VAFC IW) bit, 8–103
VAFC mode (VAFC M) bit, 8–103
VAFC output source (VAFC OS) bit, 8–103
vafc_dclk signal description, 3–9
vafc_en# signal description, 3–9
vafc_p<0:15> signal description, 3–9

vafc_vclk signal description, 3–9
Variable dot clock select (VVDS) bit, 8–168
Variable pixel formats, 8–95
Vdd signal description, 3–10
Vendor ID field, 8–11
Vertical
  size as function of HSP and VSP, 8–118
  vsync signal description, 3–10
Vertical blanking end field, 8–154
Vertical blanking start
  LSBs field, 8–154
  (SVB <9>) bit 9, 8–144
  (SVB<8>) bit 8, 8–142
Vertical display end
  (EVD<8>) bit 8, 8–142
  (EVD<9>) bit 9, 8–142
  LSBs field, 8–151
Vertical retrace
  interrupt (VRI) bit, 8–120
  (VR) bit, 8–121
Vertical sync end field, 8–150
Vertical sync interrupt
  clear (CVSI) bit, 8–150
  enable (EVSI) bit, 8–150
Vertical sync polarity (VSP) bit, 8–117
Vertical sync select (VSS) bit, 8–119
Vertical sync start
  LSBs field, 8–149
  (SVS<8>) bit 8, 8–142
  (SVS<9>) bit 9, 8–142
Vertical sync strobe (TVS) bit, 8–171
Vertical total
  LSBs field, 8–141
  (VT<8>) bit 8, 8–142
  (VT<9>) bit 9, 8–142
VESA advanced feature connector
  See VAFC
VESA display modes, 1–4
VGA alternate register space, 7–10 to 7–11
VGA attribute controller registers, 8–183
  color plane enable register (VACPER), 8–189
  color select register (VACSLR), 8–191
  index/data register (VAIXDR), 8–184
  mode register (VAMODR), 8–187
VGA attribute controller registers (cont'd)
  overscan register (VAOSCR), 8–188
  palette registers (VAPALR), 8–186
  pixel panning register (VAPXPR), 8–190
VGA color registers, 8–191
  DAC state register (VPDSTR), 8–193
  pixel address
    read mode register (VPPARR), 8–192
    write mode register (VPPAWR), 8–192
  pixel data register (VPPDAR), 8–194
  pixel mask register (VPPMAR), 8–195
VGA compatibility address (VGAAD) bit, 8–158
VGA controller, 1–3
VGA CRTC registers, 8–130
  cursor end register (VCCUER), 8–145
  cursor location
    high register (VCCLHR), 8–148
    low register (VCCLLR), 8–148
  cursor start register (VCCUSR), 8–145
  data register (VCDATR), 8–134
    data field, 8–134
  end horizontal blank register (VCHBER), 8–137
  end horizontal sync register (VCHSER), 8–139
  end vertical blanking register (VCVBER), 8–154
  end vertical display register (VCVDER), 8–151
  end vertical sync register (VCVSER), 8–149
  horizontal display end register (VCHDER), 8–136
  horizontal total register (VCHTOR), 8–135
  index register (VCINXR), 8–132
    index field, 8–132
  line compare register (VCLCMR), 8–157
  maximum scanline register (VCMSLR), 8–144
  mode control register (VCMODR), 8–155
  offset register (VCOFFR), 8–152
  overflow register (VCOVRR), 8–142

VGA CRTC registers (cont'd)
preset row register (VCPROR), 8–143
start address
high register (VCSAHR), 8–147
low register (VCSALR), 8–147
start horizontal blank register (VCHBSR), 8–137
start horizontal sync register (VCHSSR), 8–139
start vertical blanking register (VCVBSR), 8–154
start vertical sync register (VCVSSR), 8–149
underline row scan register (VCULRR), 8–153
vertical total register (VCVTOR), 8–141
VGA dot clock
See Dot clock
VGA enable (VGAE) bit, 8–46
VGA extended registers, 8–157
clock control
A register (VXCKAR), 8–168
B register (VXCKBR), 8–168
data register (VCDATR)
data field, 8–134
equalization
end register (VXEQER), 8–163
start register (VXEQSR), 8–163
half-line register (VXHLNR), 8–164
host page offset
A register (VXHPAR), 8–160
B register (VXHPBR), 8–160
index register (VCINXR)
index field, 8–132
interface control register (VXEICR), 8–171
interlace control register (VXICOR), 8–162
paging control register (VXPCOR), 8–158
split-screen start address
high byte register (VXSAHR), 8–161
low byte register (VXSALR), 8–161
timing control
A register (VXTCAR), 8–165
B register (VXTCBR), 8–166
VGA extended registers (cont'd)
video FIFO control register (VXFCOR), 8–167
VGA external and general registers, 8–116
feature control register (VEFCOR), 8–119
input status
0 register (VEIS0R), 8–120
1 register (VEIS1R), 8–121
miscellaneous output register (VEMISR), 8–117
port map, 8–116
VGA graphics controller registers, 8–171
bit mask register (VGBMKR), 8–183
color compare register (VGCCMR), 8–176
color don't care register (VGCDCR), 8–182
data register (VGDATR), 8–173
data rotate register (VGDROR), 8–177
enable set/reset register (VGESRR), 8–175
index register (VGINXR), 8–172
miscellaneous register (VGMISR), 8–181
mode register (VGMODR), 8–179
read map select register (VGRMSR), 8–178
set/reset register (VGSRER), 8–174
VGA graphics controller write modes, 8–179
VGA memory space, 7–2
VGA palette snoop (VPS) bit, 8–13
VGA registers, 8–115
port map, 8–115
VGA sequencer registers, 8–122
character map select register (VSCMSR), 8–128
clocking mode register (VSCMOR), 8–125
data register (VSDATR), 8–123
index register (VSINXR), 8–122
memory mode register (VSMMOR), 8–129
plane mask register (VSPLMR), 8–127
reset register (VSRESR), 8–124
VGA subsystem, 2–9, 12–6
interfaces, 12–6

VGA variable dot clock select (VVDS) bit, 8–168
VGA-to-2DA mode switching, 11–3
VGA-to-frame buffer memory interface, 12–6
VGA-to-PCI interface, 12–6
VGA-to-video back end interface, 12–6
Video address
configuration registers, 11–25
in 64-bit and 32-bit modes, 11–24
segment (VSEG) bit, 8–159
Video back end, 2–11
VGA interface, 12–6
Video base address
field, 8–86
register (VIVBR), 8–86
Video line width register (VILWR), 8–86
Video pipeline, 1–2
Video pixel format
register (VFPFR), 8–91
table, 8–92
Video pixel occlusion bitmap registers
See Pixel occlusion bitmap registers
Video port and display monitor interface, 12–9
Video port transceivers, 12–13
Video refresh, 2–11
calculations, 11–25
Video registers, 11–22
alternate video control register (VFAVR), 8–103
base address register (VIVBR), 8–86
control registers, 8–85
format registers, 8–90
in 64-bit and 32-bit frame buffer modes, 11–23
line width register (VILWR), 8–86
modifying the contents, 11–22
pixel format register (VFPFR), 8–91
pixel occlusion bitmap registers
See Pixel occlusion bitmap registers
scanline increment register (VISIR), 8–86
video valid register (VIVVR), 8–88
Video rendering pixel flow, 10–42
Video scanline addresses, 11–25
Video scanline increment register (VISIR), 8–86
Video valid
register (VIVVR), 8–88
synchronized (SVV) bit, 8–89
(VV) bit, 8–90
Video-disabled registers, 11–22
Vss signal description, 3–10
vsync signal description, 3–10

W

Word or byte mode select (WB) bit, 8–155
wrb# signal description, 3–10
Write buffer, 2–6
Write memory barrier, 8–26
Write mode
VGA graphics controller, 8–179
(WM) field, 8–179
Write only (WO) convention, xxi
Write pointer, copy-buffer, 8–53
Write protect (WP) bit, 8–149
Writes
frame buffer, 2–4, 10–1
mode-dependent operations, 10–1
GCTR, 8–35
alternate control space, 8–37
line mode, 8–36
graphics command register, 10–2
operations, 10–3
GSNR<7:0>, 8–50
GSWR, 8–34
memory, 9–2
Wrong parity generate (WPG) bit, 8–46

X

xtal1 signal description, 3–10
xtal2 signal description, 3–10

Y

YUV convert enable (YUVCEN) bit, 8–77
YUV pixel formats, 10–44
- 4:2:2 UYVY, 10–45
- 4:4:4 αVYU, 10–44
- 4:2:2 VYUY, 10–46
- 4:2:2 YVYU, 10–45
- 4:2:2 YVYU destination pixel, 10–45