IBM

Personal System/2
Hardware Interface Technical Reference
– Video Subsystems

42G2193
S42G-2193-00

Personal System/2
Hardware Interface Technical Reference
– Video Subsystems

Personal System/2
Hardware Interface Technical Reference
- Video Subsystems

Note

Before using this information and the product it supports, be sure to read the general information under "Notices" on page xiii.

First Edition (September 1992)

Contents

Figures v

Notices xiii
Trademarks and Service Marks xiii

Preface xv

Section 1. Introduction 1-1
Video Subsystem 1-3
Video BIOS 1-4
Type 2 Video Subsystems 1-6
Type 3 Video Subsystems 1-8

Section 2. VGA Function 2-1
VGA Function Introduction 2-5
Major Components 2-7
Hardware Considerations 2-11
Type 1 Subsystem Parameters 2-14
Video Memory Organization 2-23
Registers 2-39
VGA Programming Considerations 2-94
Video Digital-to-Analog Converter 2-101
VGA Video Extensions 2-104

Section 3. XGA Function 3-1
XGA Function Introduction 3-7
VGA Compatibility 3-14
132-Column Text Mode 3-14
Extended Graphics Mode 3-18
XGA Display Controller Registers 3-33
Coprocessor Description 3-90
Coprocessor Registers 3-128
XGA System Interface 3-163
Virtual Memory Description 3-170
XGA Subsystem Identification, Location, and XGA Mode Setting 3-185
VGA Modes 3-220
Programming the XGA Subsystem 3-227

Section 4. Display Connector 4-1
Display Connector Introduction 4-3

Section 5. XGA Sample Code 5-1

XGA Sample Code 5-3

Index X-1

Figures

1-1. BIOS Video Modes 1-4
1-2. Double Scanning and Border Support 1-5
2-1. Diagram of the VGA Function 2-6
2-2. Graphics Controller 2-9
2-3. Attribute Controller 2-10
2-4. Data Flow for Write Operations 2-12
2-5. Color Compare Operations 2-13
2-6. Character/Attribute Format 2-15
2-7. Attribute Byte Definitions 2-15
2-8. BIOS Color Set 2-16
2-9. Video Memory Format 2-17
2-10. Pel Format, Modes Hex 4 and 5 2-18
2-11. Color Selections, Modes Hex 4 and 5 2-18
2-12. Pel Format, Mode Hex 6 2-19
2-13. Bit Definitions C2,C0 2-20
2-14. Compatible Color Coding 2-22
2-15. 256KB Video Memory Map 2-23
2-16. Video Subsystem Register Overview 2-39
2-17. General Registers 2-40
2-18. Miscellaneous Output Register, Hex 03CC/03C2 2-40
2-19. Display Vertical Size 2-41
2-20. Clock Select Definitions 2-41
2-21. Input Status Register 0, Hex 03C2 2-42
2-22. Input Status Register 1, Hex 03DA/03BA 2-43
2-23. Feature Control Register, Hex 03DA/03BA and 03CA . 2-44
2-24. Video Subsystem Enable Register, Hex 03C3 2-44
2-25. Sequencer Registers 2-45
2-26. Sequencer Address Register 2-45
2-27. Reset Register, Index Hex 00 2-46
2-28. Clocking Mode Register, Index Hex 01 2-47
2-29. Map Mask Register, Index Hex 02 2-49
2-30. Character Map Select Register, Index Hex 03 2-50
2-31. Character Map Select A 2-51
2-32. Character Map Select B 2-51
2-33. Memory Mode Register, Index Hex 04 2-52
2-34. Map Selection, Chain 4 2-52
2-35. CRT Controller Registers 2-53
2-36. CRT Controller Address Register, Hex 03B4/03D4 . . . 2-54
2-37. Horizontal Total Register, Index Hex 00 2-54
2-38. Horizontal Display Enable-End Register, Index Hex 01 2-55
2-39. Start Horizontal Blanking Register, Index Hex 02 2-55
2-40. End Horizontal Blanking Register, Index Hex 03 2-56

2-41. Display Enable Skew 2-56
2-42. Start Horizontal Retrace Pulse Register, Index Hex 04 2-57
2-43. End Horizontal Retrace Register, Index Hex 05 2-58
2-44. Vertical Total Register, Index Hex 06 2-59
2-45. CRT Overflow Register, Index Hex 07 2-60
2-46. Preset Row Scan Register, Index Hex 08 2-61
2-47. Maximum Scan Line Register, Index Hex 09 2-62
2-48. Cursor Start Register, Index Hex 0A 2-63
2-49. Cursor End Register, Index Hex 0B 2-64
2-50. Start Address High Register, Index Hex 0C 2-65
2-51. Start Address Low Register, Index Hex 0D 2-65
2-52. Cursor Location High Register, Index Hex 0E 2-66
2-53. Cursor Location Low Register, Index Hex 0F 2-66
2-54. Vertical Retrace Start Register, Index Hex 10 2-67
2-55. Vertical Retrace End Register, Index Hex 11 2-67
2-56. Vertical Display-Enable End Register, Index Hex 12 . . 2-69
2-57. Offset Register, Index Hex 13 2-69
2-58. Underline Location Register, Index Hex 14 2-70
2-59. Start Vertical Blanking Register, Index Hex 15 2-71
2-60. End Vertical Blanking Register, Index Hex 16 2-71
2-61. CRT Mode Control Register, Index Hex 17 2-72
2-62. CRT Memory Address Mapping 2-73
2-63. Line Compare Register, Index Hex 18 2-75
2-64. Graphics Controller Register Overview 2-76
2-65. Graphics Controller Address Register, Hex 03CE . . . 2-76
2-66. Set/Reset Register, Index Hex 00 2-77
2-67. Enable Set/Reset Register, Index Hex 01 2-78
2-68. Color Compare Register, Index Hex 02 2-79
2-69. Data Rotate Register, Index Hex 03 2-80
2-70. Operation Select Bit Definitions 2-80
2-71. Read Map Select Register, Index Hex 04 2-81
2-72. Graphics Mode Register, Index Hex 05 2-82
2-73. Write Mode Definitions 2-83
2-74. Miscellaneous Register, Index Hex 06 2-84
2-75. Video Memory Assignments 2-84
2-76. Color Don't Care Register, Index Hex 07 2-85
2-77. Bit Mask Register, Index Hex 08 2-86
2-78. Attribute Controller Register Addresses 2-87
2-79. Address Register, Hex 03C0 2-87
2-80. Internal Palette Registers, Index Hex 00 - 0F 2-88
2-81. Attribute Mode Control Register, Index Hex 10 2-89
2-82. Overscan Color Register, Index Hex 11 2-91
2-83. Color Plane Enable Register, Index Hex 12 2-91
2-84. Horizontal Pel Panning Register, Index Hex 13 2-92
2-85. Image Shifting 2-92
2-86. Color Select Register, Index Hex 14 2-93

2-87. Character Table Structure 2-98
2-88. Character Pattern Example 2-99
2-89. Split Screen Definition 2-99
2-90. Screen Mapping within the Display Buffer Address Space 2-100
2-91. Video DAC Register 2-101
2-92. Auxiliary Video Connector Interface 2-105
2-93. Video Extension 2-106
2-94. Video Extension Signal Timing (DAC Signals) 2-108
3-1. XGA Video Subsystem 3-8
3-2. MFI Attribute Byte 3-16
3-3. MFI Blink Rates 3-16
3-4. Intel Order of the XGA Memory Map 3-19
3-5. Motorola Order of the XGA Memory Map 3-20
3-6. CRT Controller Register Definitions 3-23
3-7. Display Pel Map Offset and Width Definitions 3-24
3-8. Sprite Appearance Defined by 2-Bit Pel 3-25
3-9. Sprite Positioning 3-27
3-10. Direct Color Mode Data Word 3-30
3-11. XGA Direct Color Palette Load 3-31
3-12. Display Controller Register Addresses 3-33
3-13. Operating Mode Register, Address Hex 21x0 3-35
3-14. Display Mode Bit Assignments 3-36
3-15. Aperture Control Register, Address Hex 21x1 3-37
3-16. Aperture Size and Location Bit Assignments 3-37
3-17. Interrupt Enable Register, Address Hex 21x4 3-38
3-18. Interrupt Status Register, Address Hex 21x5 3-40
3-19. Aperture Index Register, Address Hex 21x8 3-42
3-20. Aperture Index Bit Assignments 3-42
3-21. Memory Access Mode Register, Address Hex 21x9 . . 3-43
3-22. Pel Size Bit Assignments 3-43
3-23. Index Register, Address Hex 21xA 3-44
3-24. XGA Index Register Assignments 3-45
3-25. Auto-Configuration Register, Index Hex 04 3-48
3-26. System Interface Bus Size 3-48
3-27. System Bus Type 3-49
3-28. Horizontal Total Registers, Indexes Hex 10 and 11 . . . 3-50
3-29. Horizontal Total Registers Value Assignments 3-50
3-30. Horizontal Display End Registers, Indexes Hex 12 and 13 3-51
3-31. Horizontal Display End Registers Value Assignments 3-51
3-32. Horizontal Blanking Start Registers, Indexes Hex 14 and 15 3-52
3-33. Horizontal Blanking Start Registers Value Assignments 3-52
3-34. Horizontal Blanking End Registers, Indexes Hex 16 and 17 3-53

3-35. Horizontal Blanking End Registers Value Assignments 3-53
3-36. Horizontal Sync Pulse Start Registers, Indexes Hex 18 and 19 3-54
3-37. Horizontal Sync Pulse Start Registers Value Assignments 3-54
3-38. Horizontal Sync Pulse End Registers, Indexes Hex 1A and 1B 3-55
3-39. Horizontal Sync Pulse End Registers Value Assignments 3-55
3-40. Horizontal Sync Pulse Position Registers (Index 1C and 1E) 3-56
3-41. Horizontal Sync Pulse Delay Bit Assignments 3-56
3-42. Vertical Total Registers, Indexes Hex 20 and 21 3-57
3-43. Vertical Total Registers Value Assignments 3-57
3-44. Vertical Display End Registers, Indexes Hex 22 and 23 3-58
3-45. Vertical Display End Registers Value Assignments . . 3-58
3-46. Vertical Blanking Start Registers, Indexes Hex 24 and 25 3-59
3-47. Vertical Blanking Start Registers Value Assignments . 3-59
3-48. Vertical Blanking End Registers, Indexes Hex 26 and 27 3-60
3-49. Vertical Blanking End Registers Value Assignments . 3-60
3-50. Vertical Sync Pulse Start Registers, Indexes Hex 28 and 29 3-61
3-51. Vertical Sync Pulse Start Registers Value Assignments 3-61
3-52. Vertical Sync Pulse End Register, Index Hex 2A 3-62
3-53. Vertical Line Compare Registers, Indexes Hex 2C and 2D 3-63
3-54. Vertical Line Compare Registers Value Assignments . 3-63
3-55. Sprite Horizontal Start Registers, Indexes Hex 30 and 31 3-64
3-56. Sprite Horizontal Start Registers Value Assignments . 3-64
3-57. Sprite Horizontal Preset, Index Hex 32 3-65
3-58. Sprite Horizontal Preset Value Assignments 3-65
3-59. Sprite Vertical Start Registers, Indexes Hex 33 and 34 3-66
3-60. Sprite Vertical Start Registers Value Assignments . . . 3-66
3-61. Sprite Vertical Preset, Index Hex 35 3-67
3-62. Sprite Vertical Preset Value Assignments 3-67
3-63. Sprite Control Register, Index Hex 36 3-68
3-64. Sprite Color Registers, Indexes Hex 38 – 3D 3-69
3-65. Display Pel Map Offset Registers, Indexes Hex 40 – 42 3-70
3-66. Display Pel Map Offset Registers Value Assignments . 3-70
3-67. Display Pel Map Width Registers, Indexes Hex 43 and 44 3-71
3-68. Display Pel Map Width Registers Value Assignments . 3-71
3-69. Display Control 1 Register, Index Hex 50 3-72
3-70. Sync Polarity Bit Assignments 3-72

3-71. Display Blanking Bit Assignments 3-73
3-72. Display Control 2 Register, Index Hex 51 3-74
3-73. Vertical Scale Factor Bit Assignments 3-74
3-74. Horizontal Scale Factor Bit Assignments 3-74
3-75. Display Control 2 Register Pel Size Bit Assignments . 3-75
3-76. Display ID and Comparator, Index Hex 52 3-76
3-77. Clock Frequency Selector Register, Index Hex 54 . . . 3-77
3-78. Border Color Register, Index Hex 55 3-77
3-79. Programmable Pel Clock register, Index Hex 58 3-78
3-80. Programmable Frequency Ranges 3-78
3-81. Direct Color Control register, Index Hex 59 3-79
3-82. Direct Color Modes 3-79
3-83. Sprite/Palette Index Registers, Indexes Hex 60 and 61 3-80
3-84. Sprite/Palette Prefetch Index Registers, Indexes 62 and 63 3-81
3-85. Palette Mask Register, Index Hex 64 3-82
3-86. Palette Data Register, Index Hex 65 3-82
3-87. Palette Sequence Register, Index Hex 66 3-83
3-88. Palette Sequence Register Color Order Bit Assignment 3-83
3-89. Palette Sequence Register Color Bit Assignments . . . 3-83
3-90. Palette Red Prefetch Register, Index Hex 67 3-84
3-91. Palette Green Prefetch Register, Index Hex 68 3-84
3-92. Palette Blue Prefetch Register, Index Hex 69 3-84
3-93. Sprite Data Register, Index Hex 6A 3-85
3-94. Sprite Prefetch Register, Index Hex 6B 3-85
3-95. Miscellaneous Control Register, Index 6C 3-86
3-96. MFI Control Register, Index 6D 3-86
3-97. Clock Frequency Select Registers 3-88
3-98. Clock Selected Bit Assignments 3-88
3-99. Video Clock Scale Factor Bit Assignments 3-89
3-100. Coprocessor Data Flow 3-92
3-101. XGA Pel Map Origin 3-96
3-102. Repeating Pattern (Tiling) 3-97
3-103. Destination Map Guardband 3-98
3-104. Mask Map Origin X and Y Offsets 3-99
3-105. Destination Boundary Scissor 3-101
3-106. Mask Map Boundary Scissor 3-102
3-107. Mask Map Enabled Scissor 3-103
3-108. Draw and Step Code 3-105
3-109. Draw and Step Example 3-106
3-110. Draw and Step Direction Codes 3-107
3-111. Programming Fewer Than Four Step Codes 3-108
3-112. Bresenham Line Draw Octant Encoding 3-109
3-113. Memory-to-Memory Line Draw Address Stepping . . 3-111
3-114. PxBlt Direction Codes 3-114
3-115. Inverting PxBlt 3-116

3-116. Pattern Filling 3-117
3-117. Foreground and Background Mixes 3-120
3-118. Carry Chain Mask for an 8-Bit Pel 3-121
3-119. Color Compare Conditions 3-123
3-120. XGA Coprocessor Register Space, Intel Format . . . 3-130
3-121. XGA Coprocessor Register Space, Motorola Format 3-131
3-122. Auxiliary Coprocessor Status Register, Offset Hex 09 3-133
3-123. Coprocessor Control Register, Offset Hex 11 3-134
3-124. Pel Map Index Register, Offset Hex 12 3-137
3-125. Pel Map Index 3-138
3-126. Pel Map n Base Pointer Register, Offset Hex 14 . . . 3-138
3-127. Pel Map n Width Register, Offset Hex 18 3-139
3-128. Pel Map n Height Register, Offset Hex 1A 3-140
3-129. Pel Map n Format Register, Offset Hex 1C 3-141
3-130. Pel Size Value Assignments 3-141
3-131. Bresenham Error Term E Register, Offset Hex 20 . . 3-142
3-132. Bresenham Constant K1 Register, Offset Hex 24 . . . 3-143
3-133. Bresenham Constant K2 Register, Offset Hex 28 . . . 3-143
3-134. Direction Steps Register, Offset Hex 2C 3-144
3-135. Foreground Mix Register, Offset Hex 48 3-145
3-136. Background Mix Register, Offset Hex 49 3-145
3-137. Destination Color Compare Condition Reg, Offset Hex 4A 3-146
3-138. Destination Color Compare Condition Bit Definition . 3-146
3-139. Destination Color Compare Value Register, Offset Hex 4C 3-147
3-140. Pel Bit Mask (Plane Mask) Register, Offset Hex 50 . . 3-148
3-141. Carry Chain Mask Field Register, Offset Hex 54 . . . 3-149
3-142. Foreground Color Register, Offset Hex 58 3-150
3-143. Background Color Register, Offset Hex 5C 3-150
3-144. Operation Dimension 1 Register, Offset Hex 60 3-151
3-145. Operation Dimension 2 Register, Offset Hex 62 3-151
3-146. Mask Map Origin X Offset Register, Offset Hex 6C . . 3-152
3-147. Mask Map Origin Y Offset Register, Offset Hex 6E . . 3-152
3-148. Source X Address Register, Offset Hex 70 3-153
3-149. Source Y Address Register, Offset Hex 72 3-153
3-150. Pattern C Address Register, Offset Hex 74 3-154
3-151. Pattern Y Address Register, Offset Hex 76 3-154
3-152. Destination X Address Register, Offset Hex 78 3-155
3-153. Destination Y Address Register, Offset Hex 7A 3-155
3-154. Pel Operations Register, Offset Hex 7C 3-156
3-155. Pel Operations Register, Byte 3 3-157
3-156. Pel Operations Register Background Source Value Assignments 3-157
3-157. Pel Operations Register Foreground Source Value Assignments 3-158

3-158. Pel Operations Register Step Function Value Assignments . . . 3-158
3-159. Pel Operations Register, Byte 2 . . . 3-159
3-160. Pel Operations Register Source Pel Map Value Assignments . . . 3-159
3-161. Pel Operations Register Destination Pel Map Value Assignments . . . 3-160
3-162. Pel Operations Register, Byte 1 . . . 3-160
3-163. Pel Operations Register Pattern Pel Map Value Assignments . . . 3-161
3-164. Pel Operations Register, Byte 0 . . . 3-161
3-165. Pel Operations Register Mask Pel Map Value Assignments . . . 3-161
3-166. Pel Operations Register Drawing Mode Value Assignments . . . 3-162
3-167. Pel Operations Register Direction Octant Values . . . 3-162
3-168. POS Register 2, Base Address + 2 . . . 3-165
3-169. XGA ROM, Memory-Mapped Register Assignments . 3-166
3-170. I/O Device Address Bit Assignment . . . 3-167
3-171. POS Register 4, Base Address + 4 . . . 3-168
3-172. XGA Video Memory Base Address . . . 3-168
3-173. POS Register 5, Base Address + 5 . . . 3-169
3-174. 1MB Aperture Base Address Value Assignments . . . 3-169
3-175. Linear Address Fields . . . 3-170
3-176. Linear to Physical Address Translation . . . 3-171
3-177. Page Directory and Page Table Entry . . . 3-172
3-178. Page Directory and Page Table Access Rights in User Mode . . . 3-173
3-179. Translate Look-Aside Buffer . . . 3-174
3-180. Page Directory Base Address Register, Offset Hex 0 3-179
3-181. Current Virtual Address Register, Offset Hex 4 . . . 3-180
3-182. Virtual Memory Control Register . . . 3-181
3-183. Virtual Memory Interrupt Status Register . . . 3-183
3-184. DMQS Display Information File Structure . . . 3-193
3-185. DMQS Display Information File Layout . . . 3-194
3-186. DMQS Display Information File Mode Data . . . 3-195
3-187. DMQS Extended Graphics Mode Register Settings . 3-197
3-188. DMQS Display Color Characteristics . . . 3-199
3-189. DMQS Display Pre-selected Colors . . . 3-200
3-190. DMQS Customized Tag Syntax . . . 3-203
3-191. XGA Video Memory Base Address . . . 3-208
3-192. The XGA Video Memory Base Address Diagram . . . 3-208
3-193. Reading the Display ID . . . 3-211
3-194. Video Memory Size Determination . . . 3-212
3-195. Availability of Extended Graphics Modes . . . 3-213
3-196. Capability of Graphic Displays . . . 3-214

3-197. Extended Graphics Mode Register Settings 3-215
3-198. VGA Mode Write Sequence 3-221
3-199. 132-Column Text Mode First Write Sequence 3-224
3-200. 132-Column Text Mode Second Write Sequence . . . 3-225
3-201. Coprocessor Register Write Values (Example A) . . . 3-237
3-202. Background and Foreground Mixes and Colors 3-238
3-203. Bit Layout Pel Operations Register 3-241
3-204. Operation Direction Diagram 3-244
3-205. Definition for Pel Operations Register (Example) . . . 3-245
3-206. Palette Color Line Draw Steps 3-245
3-207. Background and Foreground Mixes and Colors 3-246
3-208. Line Draw Example in Octant 0 3-247
3-209. Bit Layout Pel Operations Register 3-250
3-210. Direction Octant (Example) 3-253
3-211. Definition for Pel Operations Register (Example) . . . 3-254
4-1. Display Connector 4-3
4-2. Display Connector Signals 4-3
4-3. Vertical Size of Display 4-4
4-4. VGA Mode Display Timing—Set 1 4-6
4-5. VGA Mode Display Timing—Set 2 4-6
4-6. VGA Mode Display Timing—Set 3 4-6
4-7. Pel Frequencies (MHz) for Various Display Modes 4-7
4-8. Display Modes 1, 2 and 3. 4-7
4-9. Extended Graphics Mode 1024 x 768 Interlaced Display Timing 4-8
4-10. Pel Frequencies (MHz) for Various Display Modes 4-9

Notices

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program or service is not intended to state or imply that only IBM's product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any of IBM's intellectual property rights or other legally protectable rights may be used instead of the IBM product, program, or service. Evaluation and verification of operation in conjunction with other products, programs, or services, except those expressly designated by IBM, are the user's responsibility.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Commercial Relations, IBM Corporation, Purchase, NY 10577.

Trademarks and Service Marks

The following terms, denoted by an asterisk (*) in this publication, are trademarks of the IBM Corporation in the United States and/or other countries:

IBM
Micro Channel
OS/2
Personal System/2
PS/2
XGA
AT

The following terms, denoted by a double asterisk (**) in this publication, are trademarks of other companies as follows:

Intel	Intel Corporation
Lotus	Lotus Development Corporation
Microsoft	Microsoft Corporation
Motorola	Motorola, Incorporated
Windows	Microsoft Corporation

Notes:

Preface

This technical reference is for those who develop hardware and software products for IBM Personal Computers and IBM Personal System/2 products. Readers should understand computer architecture and programming concepts.

This technical reference should be used with the following publications, which contain additional information about many of the subjects discussed in this document.

IBM Personal System/2 Hardware Interface Technical Reference — AT-Bus Systems

IBM Personal System/2 Hardware Interface Technical Reference — Architectures

IBM Personal System/2 Hardware Interface Technical Reference — Common Interfaces

IBM Personal System/2 Hardware Interface Technical Reference — System-Specific Information

IBM Personal System/2 and Personal Computer BIOS Interface Technical Reference

Information about diskette drives, hard disk drives, adapters, and external options are in separate option technical references.

Warning: The term *reserved* describes certain signals, bits, and registers that should not be changed. Use of reserved areas can cause compatibility problems, loss of data, or permanent damage to the hardware. When the contents of a register are changed, the state of the reserved bits must be preserved. When possible, read the register first and change only the bits that must be changed.

Notes:

Section 1. Introduction

Video Subsystem 1-3
Type 1 Video 1-3
Type 2 Video 1-3
Type 3 Video 1-3
Video BIOS 1-4
Type 2 Video Subsystems 1-6
VGA Mode 1-6
132-Column Text Mode 1-6
Extended Graphics Mode 1-6
IBM PS/2 8514/A Adapter Interface Compatibility 1-6
High Resolution Support 1-6
Direct Color Mode 1-6
Packed Pel Format 1-7
Hardware Sprite 1-7
Display Identification 1-7
Coprocessor 1-7
Type 3 Video Subsystems 1-8

Notes:

Video Subsystem

This technical reference specifies the hardware interface to the Video Graphics Array (VGA), Extended Graphics Array (XGA*), and Extended Graphics Array-2 (XGA-2), video subsystems. These video subsystems are found in IBM* Micro Channel* - and Industry Standard Architecture (ISA)-based personal computer systems. The system video can be generated by a Type 1, Type 2, or Type 3 video subsystem:

- Type 1 video—Video Graphics Array (VGA)
- Type 2 video—Extended Graphics Array (XGA)
- Type 3 video—Extended Graphics Array-2 (XGA-2)

Type 1 Video

The Type 1 video provides VGA function. The capabilities and operation of the VGA function are described in Section 2, "VGA Function" on page 2-1.

Only one video subsystem can be enabled into VGA or 132-column text mode at any one time. Some Type 1 video subsystems contain the VGA function and 132-column text mode. The hardware interface for the VGA function is fully specified in this technical reference. BIOS should be used to determine the full capability of a Type 1 video subsystem.

Type 2 Video

The Type 2 video contains the XGA function, which supports the VGA mode, 132-column text mode, and extended graphics mode. The capabilities and operation of the XGA function are described in Section 3, "XGA Function" on page 3-1. One to eight Type 2 video subsystems are allowed in a system.

Type 3 Video

The Type 3 video contains all the functions of the Type 2 video, along with other enhancements. The capabilities and operation of the XGA-2 function are also described in Section 3, "XGA Function" on page 3-1.

---

* Trademark of the IBM Corporation

Video BIOS

The video function is classified by BIOS mode numbers, which in turn define the screen size, colors, and associated parameters of the specific mode.

The following figure describes the alphanumeric (A/N) and all points addressable (APA) graphics modes supported by BIOS. Each color is selected from 256K possibilities, and gray shades are selected from 64 possibilities. The variations within the basic BIOS modes are selected through BIOS calls that set the number of scan lines. The scan line count is set before the mode call is made.

BIOS should be used to determine the modes that are supported on a given subsystem.

Mode (hex)	Type	Colors	Alpha Format	Buffer Start	Box Size	Max. Pgs.	Vert. Pels
0,1	A/N	16	40x25	B8000	8x8	8	320x200
0†,1†	A/N	16	40x25	B8000	8x14	8	320x350
0‡,1‡	A/N	16	40x25	B8000	9x16	8	360x400
2,3	A/N	16	80x25	B8000	8x8	8	640x200
2†,3†	A/N	16	80x25	B8000	8x14	8	640x350
2‡,3‡	A/N	16	80x25	B8000	9x16	8	720x400
4,5	APA	4	40x25	B8000	8x8	1	320x200
6	APA	2	80x25	B8000	8x8	1	640x200
7	A/N	–	80x25	B0000	9x14	8	720x350
7‡	A/N	–	80x25	B0000	9x16	8	720x400
D	APA	16	40x25	A0000	8x8	8	320x200
E	APA	16	80x25	A0000	8x8	4	640x200
F	APA	–	80x25	A0000	8x14	2	640x350
10	APA	16	80x25	A0000	8x14	2	640x350
11	APA	2	80x30	A0000	8x16	1	640x480
12	APA	16	80x30	A0000	8x16	1	640x480
13	APA	256	40x25	A0000	8x8	1	320x200
14	A/N	16	132x25	B8000	8x16 or 9x16	4	1056x400 or 1188x400

Note: † or ‡ Enhanced modes
Mode 14 character box size is determined by hardware.

Figure 1-1. BIOS Video Modes

In the 200-scan-line modes, the data for each scan line is scanned twice. This double scanning allows the 200-scan-line image to be displayed in 400 scan lines.

Certain modes on previous IBM display adapters distinguished between monochrome and color displays. For example, mode 0 was the same as mode 1 with the color burst turned off. Because color burst is not supported by the PS/2* video, the mode pairs are exactly the same. The support logic for the VGA function recognizes the type of display, and adjusts the output accordingly. When a monochrome display is attached, the colors for the color modes appear as shades of gray.

Mode 3+ is the default mode with a color display attached and mode 7+ is the default mode with a monochrome display attached.

Border support and double scanning depend on the mode selected. The following table shows which modes use double scanning and which support a border.

Mode (Hex)	Double Scan	Border Support
0, 1	Yes	No
0*, 1*	No	No
0+, 1+	No	No
2, 3	Yes	Yes
2*, 3*	No	Yes
2+, 3+	No	Yes
4, 5	Yes	No
6	Yes	Yes
7	No	Yes
7+	No	Yes
D	Yes	No
E	Yes	Yes
F	No	Yes
10	No	Yes
11	No	Yes
12	No	Yes
13	Yes	Yes
14	No	Yes

Note: * or + Enhanced modes

Figure 1-2. Double Scanning and Border Support

* Trademark of the IBM Corporation

Type 2 Video Subsystems

The Base XGA function (including the VGA function) is generated by the Type 2 video subsystem.

The XGA function has three modes.

- VGA
- 132-column text
- Extended Graphics.

VGA Mode

In VGA mode, the XGA video subsystem is VGA register compatible, as defined in the VGA function description.

132-Column Text Mode

In this mode, text is displayed in 132 vertical columns, and is accessible through BIOS mode 14.

Extended Graphics Mode

Extended Graphics mode provides the following software and hardware support.

IBM PS/2 8514/A Adapter Interface Compatibility

Compatibility is provided through the XGA Adapter Interface, a device driver supplied with the subsystem as programming support for applications operating in the disk operating system (DOS) environment.

High Resolution Support

Depending on the display attached and the size of video memory installed, the image on a screen can be defined using 1024 pels and 768 scan lines with 256 colors.

Direct Color Mode

In this mode, each 16-bit pel in video memory specifies the color of the pel directly. This allows 65,536 colors to be displayed using 640 pels and 480 scan lines.

Packed Pel Format

In the packed pel format, reads and writes to the video memory access all the data that defines a pel (or pels) in a single operation.

Hardware Sprite

The sprite is a 64 x 64 pel image. When enabled, it overlays the picture that is being displayed. It can be positioned anywhere on the display without affecting the contents of video memory.

Display Identification

Signals from the attached display identify its characteristics. Applications use this information to determine the maximum resolution and whether the display is color or monochrome.

Coprocessor

The coprocessor provides hardware drawing-assist functions throughout real or virtual memory. The following functions can be used with the XGA Adapter Interface.

- Pel-block and bit-block transfers (PxBlt)
- Line drawing
- Area filling
- Logical and arithmetic mixing
- Map masking
- Scissoring
- X and Y axis addressing

Type 3 Video Subsystems

All functions of the Type 2 video subsystem are included in the Type 3 video subsystem. The Type 3 video subsystem is an enhanced version of the Type 2 Video subsystem. These enhancements include the following functions.

- Higher refresh rates, such as 75 Hz and 72 Hz, noninterlaced, for improved screen stability, even at higher resolution (1024 x 768)
- Coprocessor support for Direct Color mode operation
- The digital-to-analog converter (DAC) has been expanded from 18 bits to 24 bits, which allows up to 256 colors available from a palette of more than 16 million colors.
- Supports both VGA and mainframe interactive (MFI) character attributes in text mode.

Section 2. VGA Function

VGA Function Introduction 2-5
Major Components 2-7
BIOS 2-7
Support Logic 2-7
VGA Components 2-7
CRT Controller 2-7
Sequencer 2-8
Graphics Controller 2-8
Attribute Controller 2-10
Hardware Considerations 2-11
Differences in Type 1 and Other Video Subsystems 2-11
Memory Write Operations 2-12
Memory Read Operations 2-13
Type 1 Subsystem Parameters 2-14
Alphanumeric Modes 2-14
Graphics Modes 2-17
320 x 200 Four-Color Graphics (Modes Hex 4 and 5) 2-17
640 x 200 Two-Color Graphics (Mode Hex 6) 2-19
640 x 350 Graphics (Mode Hex F) 2-20
640 x 480 Two-Color Graphics (Mode Hex 11) 2-21
16-Color Graphics Modes (Modes Hex D, E, 10, and 12) . . 2-21
256-Color Graphics Mode (Mode Hex 13) 2-21
Video Memory Organization 2-23
Memory Modes 2-23
Modes Hex 0, 1 2-24
Modes Hex 2, 3 2-25
Modes Hex 4, 5 2-26
Mode Hex 6 2-27
Mode Hex 7 2-28
Mode Hex D 2-29
Mode Hex E 2-31
Mode Hex F 2-33
Mode Hex 10 2-34
Mode Hex 11 2-35
Mode Hex 12 2-36
Mode Hex 13 2-37
Mode Hex 14 2-38
Registers 2-39
General Registers 2-40
Miscellaneous Output Register 2-40
Input Status Register 0 2-42
Input Status Register 1 2-43

Feature Control Register 2-44
Video Subsystem Enable Register 2-44
Sequencer Registers 2-45
Sequencer Address Register 2-45
Reset Register 2-46
Clocking Mode Register 2-47
Map Mask Register 2-49
Character Map Select Register 2-50
Memory Mode Register 2-52
CRT Controller Registers 2-53
Address Register 2-54
Horizontal Total Register 2-54
Horizontal Display-Enable End Register 2-55
Start Horizontal Blanking Register 2-55
End Horizontal Blanking Register 2-56
Start Horizontal Retrace Pulse Register 2-57
End Horizontal Retrace Register 2-58
Vertical Total Register 2-59
Overflow Register 2-60
Preset Row Scan Register 2-61
Maximum Scan Line Register 2-62
Cursor Start Register 2-63
Cursor End Register 2-64
Start Address High Register 2-65
Start Address Low Register 2-65
Cursor Location High Register 2-66
Cursor Location Low Register 2-66
Vertical Retrace Start Register 2-67
Vertical Retrace End Register 2-67
Vertical Display-Enable End Register 2-69
Offset Register 2-69
Underline Location Register 2-70
Start Vertical Blanking Register 2-71
End Vertical Blanking Register 2-71
CRT Mode Control Register 2-72
Line Compare Register 2-75
Graphics Controller Registers 2-76
Address Register 2-76
Set/Reset Register 2-77
Enable Set/Reset Register 2-78
Color Compare Register 2-79
Data Rotate Register 2-80
Read Map Select Register 2-81
Graphics Mode Register 2-82
Miscellaneous Register 2-84
Color Don't Care Register 2-85

Bit Mask Register 2-86
Attribute Controller Registers 2-87
Address Register 2-87
Internal Palette Registers 0 through F 2-88
Attribute Mode Control Register 2-89
Overscan Color Register 2-91
Color Plane Enable Register 2-91
Horizontal Pel Panning Register 2-92
Color Select Register 2-93
VGA Programming Considerations 2-94
Programming the Registers 2-97
RAM Loadable Character Generator 2-98
Creating a Split Screen 2-99
Video Digital-to-Analog Converter 2-101
Device Operation 2-101
Video DAC to System Interface 2-101
Programming Considerations 2-103
VGA Video Extensions 2-104
Auxiliary Video Extension 2-106
Base Video Extension 2-106
Video Extension Signal Descriptions 2-107
Video Extension Signal Timing 2-108

Notes:

VGA Function Introduction

The circuitry that provides the VGA function includes a video buffer, a video digital-to-analog converter (DAC), and test circuitry. Video memory is mapped as four planes of 64KB by 8 bits (maps 0 through 3). The video DAC drives the analog output to the display connector. The test circuitry determines the type of display attached, color or monochrome.

The video subsystem controls the access to video memory from the system and the cathode-ray tube (CRT) controller. It also controls the system addresses assigned to video memory. Up to three starting addresses can be programmed for compatibility with previous video adapters.

In the graphics modes, the mode determines the way video information is formatted into memory, and the way memory is organized.

In alphanumeric modes, the system writes the ASCII character code and attribute data to video memory maps 0 and 1, respectively. Memory map 2 contains the character font loaded by BIOS during an alphanumeric mode set. The font is used by the character generator to create the character image on the display.

Three fonts are supplied with the system. These fonts are either 8 or 9 pels wide, and either 8, 14, or 16 pels high. Up to eight 256-character fonts can be loaded into the video memory map 2; two of these fonts can be active at one time, allowing a 512-character font.

The video subsystem formats the information in video memory and sends the output to the video DAC. For color displays, the video DAC sends three analog color signals (red, green, and blue) to the display connector. For monochrome displays, BIOS translates the color information in the DAC, and the DAC drives the summed signal onto the green output.

The auxiliary video connector allows video data to be passed between the video subsystem and an adapter plugged into the channel connector. See “VGA Video Extensions” on page 2-104.

When it is disabled, the video subsystem will not respond to video memory or I/O reads or writes; however, the video image continues to be displayed.

Note: Compatibility with other hardware is best achieved by using the BIOS interface or operating system interface whenever possible.

The following is a diagram of the VGA function.

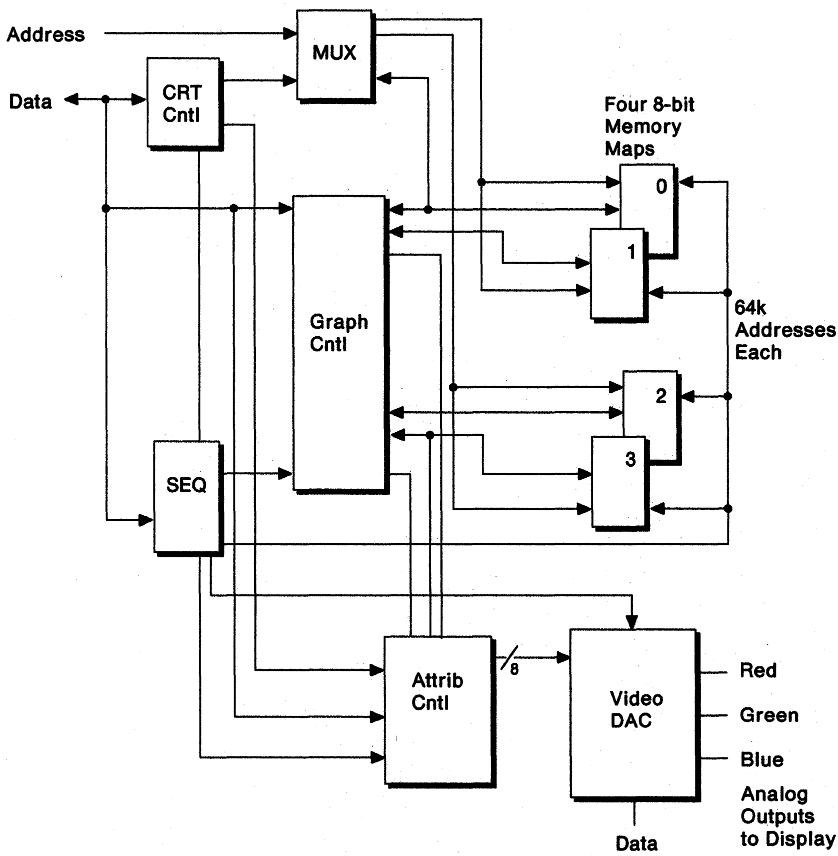


Figure 2-1. Diagram of the VGA Function

Major Components

The video subsystem contains all circuits necessary to generate the timing for the video memory and generates the video information going to the video DAC. The major components are: BIOS, support logic, and Video Graphics Array interface.

BIOS

BIOS provides software support and contains the character fonts and the system interface to run the video subsystem.

Support Logic

The support logic consists of the video memory, the clocks, and the video DAC. The video memory consists of at least 256KB; its use and mapping depend on the mode selected.

Two clock sources provide the dot rate. The clock source is selected in the Miscellaneous Output register.

The video DAC contains the color palette that is used to convert the video data into the video signal sent to the display. Three analog signals (red, green, and blue) are output from the DAC.

The maximum number of colors displayed is 256 out of 256K, and the maximum number of gray shades is 64 out of 64.

VGA Components

The VGA function has four major functional areas: the CRT controller, the sequencer, the graphics controller, and the attribute controller.

CRT Controller

The CRT controller generates horizontal and vertical synchronization signal timings, addressing for the regenerative buffer, cursor and underline timings, and refresh addressing for the video memory.

Sequencer

The sequencer generates basic memory timings for the video memory and the character clock for controlling regenerative buffer fetches. It allows the system to access memory during active display intervals by periodically inserting dedicated system microprocessor memory cycles between the display memory cycles. Map mask registers in the sequencer are available to protect entire memory maps from being changed.

Graphics Controller

The graphics controller is the interface between the video memory and the attribute controller during active display times, and between video memory and the system microprocessor during memory accesses.

During active display times, memory data is latched and sent to the attribute controller. In graphics modes, the memory data is converted from parallel to serial bit-plane data before being sent; in alphanumeric modes, the parallel attribute data is sent.

During system accesses of video memory, the graphics controller can perform logical operations on the memory data before it reaches video memory or the system data bus. These logical operations are composed of four logical write modes and two logical read modes. The logical operators allow enhanced operations, such as a color compare in the read mode, individual bit masking during write modes, internal 32-bit writes in a single memory cycle, and writing to the display buffer on nonbyte boundaries.

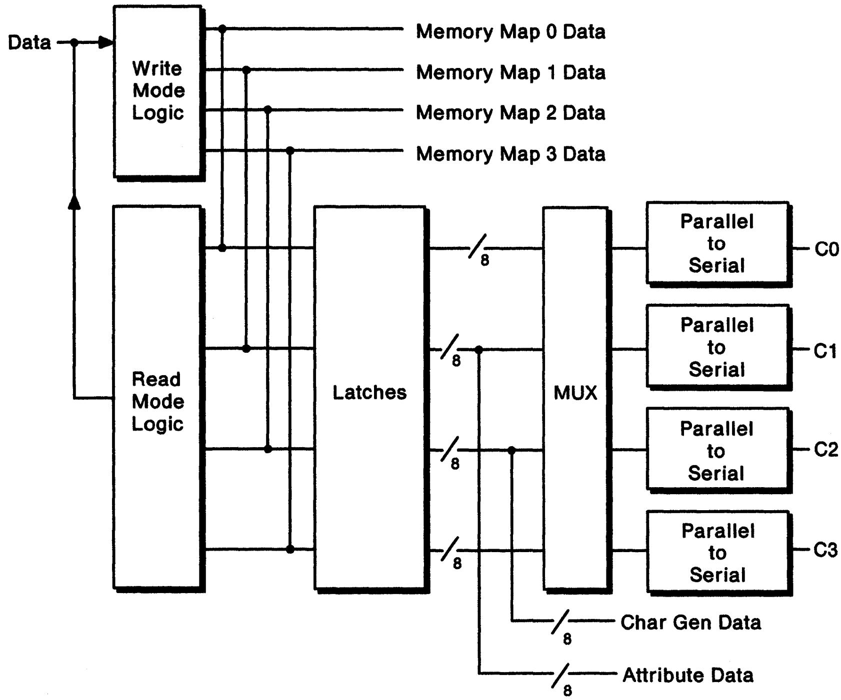


Figure 2-2. Graphics Controller

Attribute Controller

The attribute controller takes in data from video memory through the graphics controller and formats it for display. Attribute data in alphanumeric mode and serialized bit-plane data in graphics mode are converted to an 8-bit color value.

Each color value is selected from an internal color palette of 64 possible colors (except in 256-color mode). The color value is used as a pointer into the video DAC where it is converted to the analog signals that drive the display.

Blinking, underlining, cursor insertion, and pel panning are also controlled in the attribute controller.

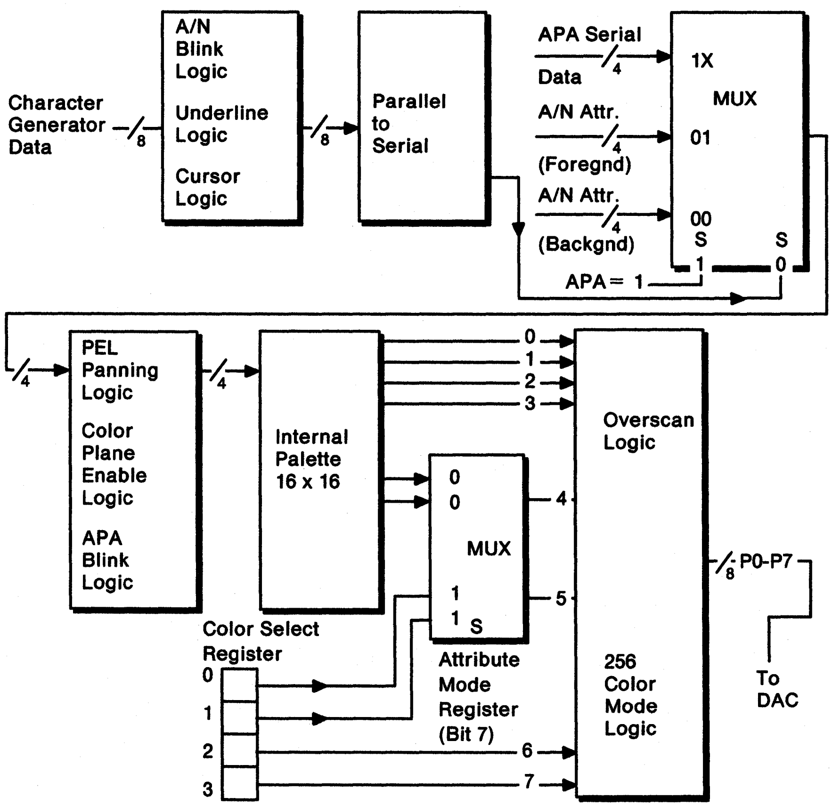


Figure 2-3. Attribute Controller

Hardware Considerations

The following are hardware characteristics of the Type 2 or Type 3 video subsystem that must be considered to ensure program compatibility with the Type 1 video subsystem.

Differences in Type 1 and Other Video Subsystems

To ensure program compatibility between Type 1 and Type 2 or between Type 1 and Type 3 video subsystems, the following hardware characteristics of the Type 2 and Type 3 video subsystem must be considered.

Performance: Type 2 and Type 3 video subsystems generally run faster than the Type 1 video subsystem. Programs that depend on execution time of the video subsystem will operate differently.

Video Buffer Compatibility: For each of the video modes, the Type 2 and Type 3 video subsystems maintain a memory mapping that is the same as the Type 1 video subsystem. To maintain this compatibility, the internal addresses to video memory are manipulated so that video memory looks the same. When switching video modes, video data may not be at the same address in video memory.

BIOS calls to set and change modes make allowances for changes in addresses, and should be used for all mode switches.

Character Generator: Differences in the character generator for the Type 2 and Type 3 video subsystems increase the time it takes to load a new font. Because of the additional load time, there is a chance of briefly observing spurious data on the display. BIOS compensates for this during video mode sets.

Register Differences: The following bits for the Type 2 and Type 3 video subsystems differ from the Type 1 video sybsystem.

- Bits 2 and 4 in the Clocking Mode register
- Bits 5 and 6 in the End Horizontal Blanking register
- Bits 2 and 4 in the Preset Row Scan register
- Bit 5 in the Address register of the attribute controller.

Memory Write Operations

When the system is writing to the display buffer, the maps are enabled by the logical decode of the memory address and the Map Mask register. The addresses used for video memory depend on the mode selected. The data flow for a system write operation is illustrated in the following figure.

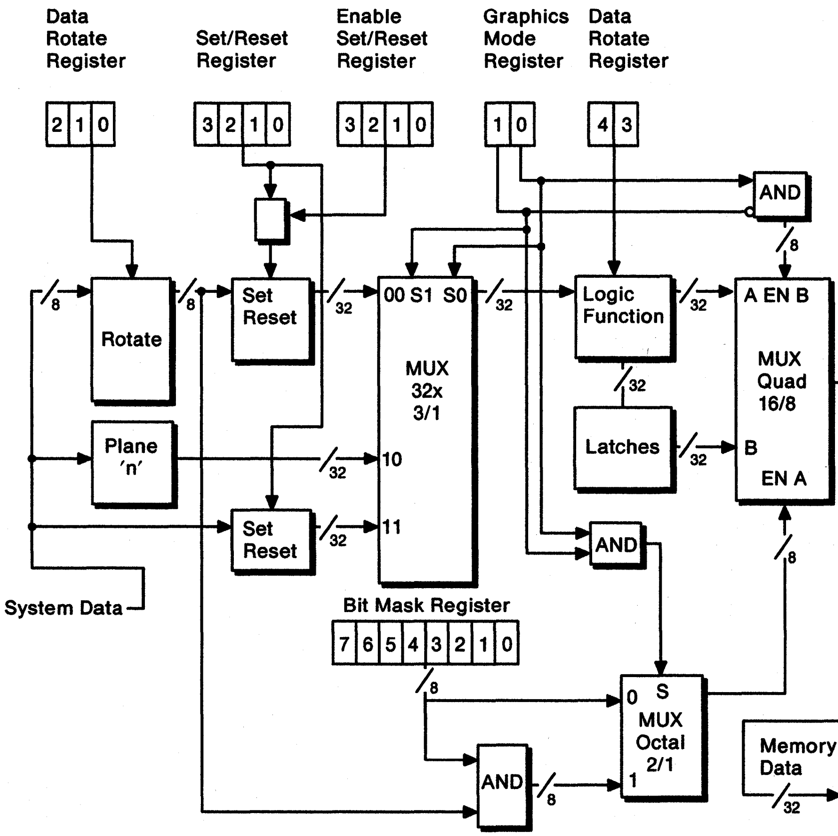


Figure 2-4. Data Flow for Write Operations

Memory Read Operations

The two ways to read the video buffer are selected through the Graphics Mode register in the graphics controller. The mode 0 read operation returns the 8-bit value determined by the logical decode of the memory address and, if applicable, the Read Map Select register. The mode 1 read operation returns the 8-bit value resulting from the color compare operation controlled by the Color Compare and Color Don't Care registers. The data flow for the color compare operation is shown in the following figure.

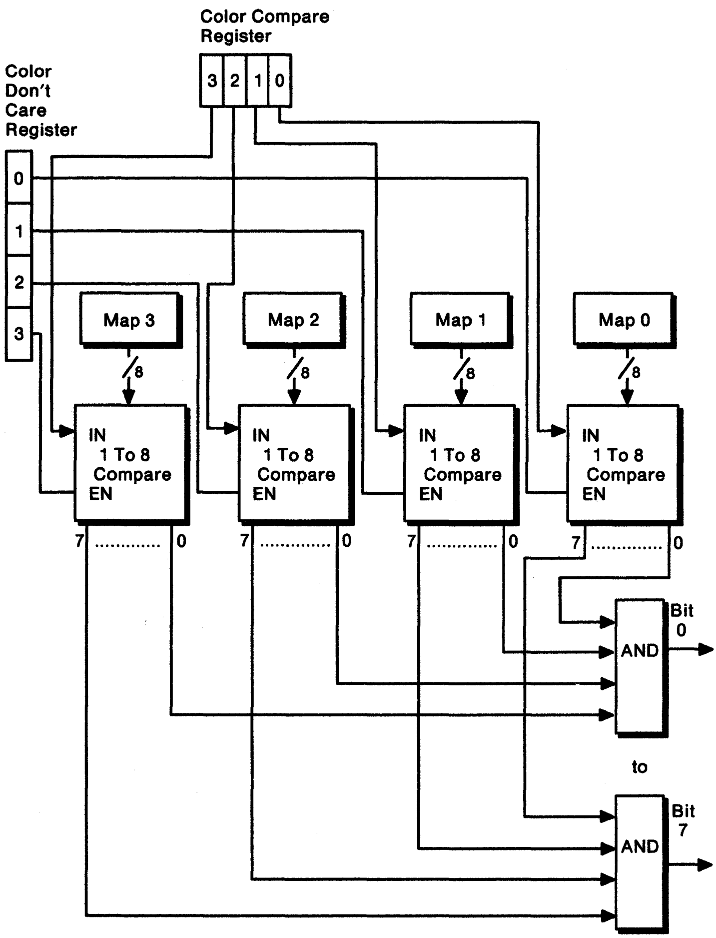


Figure 2-5. Color Compare Operations

Type 1 Subsystem Parameters

Alphanumeric Modes

The alphanumeric modes are modes hex 0 through 3 and 7. The mode chart lists the variations of these modes (see Figure 1-1 on page 1-4). The data format for alphanumeric modes is the same as the data format on the IBM Color/Graphics Monitor Adapter, the IBM Monochrome Display Adapter, and the IBM Enhanced Graphics Adapter.

BIOS initializes the video subsystem according to the selected mode and loads the color values into the video DAC. These color values can be changed to give a different color set from which to select. Bit 3 of the attribute byte can be redefined by the Character Map Select register to act as a switch between character sets, giving the programmer access to 512 characters at one time.

When an alphanumeric mode is selected, the BIOS transfers character font patterns from the ROM to map 2. The system stores the character data in map 0, and the attribute data in map 1. In the alphanumeric modes, the programmer views maps 0 and 1 as a single buffer. The CRT controller generates sequential addresses and fetches one character code byte and one attribute byte at a time. The character code and row scan count are combined to make up the address into map 2, which contains the character font. The appropriate dot patterns are then sent to the attribute controller, where color is assigned according to the attribute data.

Every display-character position in the alphanumeric mode is defined by two bytes in the display buffer. Both the color/graphics and the monochrome emulation modes use the following 2-byte character/attribute format.

Display Character Code Byte								Attribute Byte							
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Even Address								Odd Address							

Figure 2-6. Character/Attribute Format

See "Characters and Keystrokes" for characters loaded during a BIOS mode set.

The functions of the attribute bytes are defined in the following table. Bit 7 can be redefined in the Attribute Mode Control register to give 16 possible background colors; its default is to control character blinking. Bit 3 can be redefined in the Character Map Select register as a switch between two character fonts; its default is to control foreground color selection.

Bit	Color	Function
7	B/I	Blinking or Background Intensity
6	R	Background Color
5	G	Background Color
4	B	Background Color
3	I/CS	Foreground Intensity or Character Font Select
2	R	Foreground Color
1	G	Foreground Color
0	B	Foreground Color

Figure 2-7. Attribute Byte Definitions

For more information about the attribute bytes, see "Character Map Select Register" on page 2-50 and "Attribute Mode Control Register" on page 2-89.

The following are the color values loaded by BIOS for the 16-color modes.

Intensity	Red	Green	Blue	Color
0	0	0	0	Black
0	0	0	1	Blue
0	0	1	0	Green
0	0	1	1	Cyan
0	1	0	0	Red
0	1	0	1	Magenta
0	1	1	0	Brown
0	1	1	1	White
1	0	0	0	Gray
1	0	0	1	Light Blue
1	0	1	0	Light Green
1	0	1	1	Light Cyan
1	1	0	0	Light Red
1	1	0	1	Light Magenta
1	1	1	0	Yellow
1	1	1	1	White (High Intensity)

Figure 2-8. BIOS Color Set

Both 40-column and 80-column alphanumeric modes are supported. The features of the 40-column alphanumeric modes (all variations of modes hex 0 and 1) are:

- 25 rows of 40 characters
- 2KB of video memory per page
- One character byte and one attribute byte per character.

The features of the 80-column alphanumeric modes (all variations of modes hex 2, 3, and 7) are:

- 25 rows of 80 characters
- 4KB of video memory per page
- One character byte and one attribute byte per character.

Graphics Modes

The colors described in this section are generated when the BIOS is used to set the mode. BIOS initializes the video subsystem and the DAC palette to generate these colors. If the DAC palette is changed, different colors are generated.

320 x 200 Four-Color Graphics (Modes Hex 4 and 5)

Addressing, mapping, and data format are the same as the 320 x 200 pel mode of the IBM Color/Graphics Monitor Adapter. The display buffer is configured at hex B8000. Bit image data is stored in memory maps 0 and 1. The two bit planes (C0 and C1) are each formed from bits from both memory maps.

Features of this mode are:

- A maximum of 200 rows of 320 pels
- Double scanned to display as 400 rows
- Memory-mapped graphics
- Four colors for each pel
- Four pels per byte
- 16KB of read/write memory.

The video memory is organized into two banks of 8KB each, using the following format. Address hex B8000 contains the pel information for the upper-left corner of the display area.

Memory Address	Function
B8000	
	Even Scans (0,2,4,......,198)
B9F3F	
	Reserved
BA000	
	Odd Scans (1,3,5,......,199)
BBF3F	
	Reserved
BBFFF	

Figure 2-9. Video Memory Format

The following figure shows the format for each byte.

Bit	Function
7	C1 - First Display Pel
6	C0 - First Display Pel
5	C1 - Second Display Pel
4	C0 - Second Display Pel
3	C1 - Third Display Pel
2	C0 - Third Display Pel
1	C1 - Fourth Display Pel
0	C0 - Fourth Display Pel

Figure 2-10. Pel Format, Modes Hex 4 and 5

The color selected depends on the color set that is used. Color set 1 is the default. For information on changing the color set, see the *IBM Personal System/2 and Personal Computer BIOS Interface Technical Reference.*

Bits	Color Selected	
C1 C0	Color Set 1	Color Set 0
0 0	Black	Black
0 1	Light Cyan	Green
1 0	Light Magenta	Red
1 1	Intensified White	Brown

Figure 2-11. Color Selections, Modes Hex 4 and 5

640 x 200 Two-Color Graphics (Mode Hex 6)

Addressing, scan-line mapping, and data format are the same as the 640 x 200 pel black and white mode of the IBM Color/Graphics Monitor Adapter. The display buffer is configured at hex B8000. Bit image data is stored in memory map 0 and comprises a single bit plane (C0). Features of this mode are:

- A maximum of 200 rows of 640 pels
- Double scanned to display as 400 rows
- Same addressing and scan-line mapping as 320 x 200 graphics
- Two colors for each pel
- Eight pels per byte
- 16KB of read/write memory.

The following shows the format for each byte.

Bit	Function
7	First Display Pel
6	Second Display Pel
5	Third Display Pel
4	Fourth Display Pel
3	Fifth Display Pel
2	Sixth Display Pel
1	Seventh Display Pel
0	Eighth Display Pel

Figure 2-12. Pel Format, Mode Hex 6

The bit definition for each pel is 0 equals black and 1 equals intensified white.

640 x 350 Graphics (Mode Hex F)

This mode emulates the EGA graphics with the monochrome display and the following attributes: black, video, blinking video, and intensified video. A resolution of 640 x 350 uses 56KB of video memory to support the four attributes. This mode uses maps 0 and 2; map 0 is the video bit plane (C0), and map 2 is the intensity bit plane (C2). Both planes reside at address hex A0000.

The two bits, one from each bit plane, define one pel. The bit definitions are given in the following table.

C2	C0	Pel Color
0	0	Black
0	1	White
1	0	Blinking White
1	1	Intensified White

Figure 2-13. Bit Definitions C2,C0

Memory is organized with successive bytes defining successive pels. The first eight pels displayed are defined by the byte at hex A0000, the second eight pels by the byte at hex A0001, and so on. The most-significant bit in each byte defines the first pel for that byte.

Because both bit planes reside at address hex A0000, the user must select the plane to update through the Map Mask register of the sequence controller (see “Video Memory Organization” on page 2-23).

640 x 480 Two-Color Graphics (Mode Hex 11)

This mode provides two-color graphics with the same data format as mode 6. Addressing and mapping are shown under "Video Memory Organization" on page 2-23.

The bit image data is stored in map 0 and comprises a single bit plane (C0). The video buffer starts at hex A0000. The first byte contains the first eight pels; the second byte, at hex A0001, contains the second eight pels, and so on. The bit definition for each pel is 0 equals black and 1 equals intensified white.

16-Color Graphics Modes (Modes Hex D, E, 10, and 12)

These modes support 16 colors. For all modes, the bit image data is stored in all four memory maps. Each memory map contains the data for one bit plane. The bit planes are C0 through C3 and represent the following colors.

C0 = Blue
C1 = Green
C2 = Red
C3 = Intensified

The four bits define each pel on the screen by acting as an address (pointer) into the internal palette in the Extended Graphics mode.

The display buffer resides at address hex A0000. The Map Mask register selects any or all of the maps to be updated when the system writes to the display buffer.

256-Color Graphics Mode (Mode Hex 13)

This mode provides graphics with the capability of displaying 256 colors at one time.

The display buffer is sequential, starts at address hex A0000, and is 64,000 bytes long. The first byte contains the color information for the upper-left pel. The second byte contains the second pel, and so on, for 64,000 pels (320 x 200). The bit image data is stored in all four memory maps and comprises four bit planes. The four bit planes are sampled twice to produce eight bit-plane values that address the video DAC.

In this mode, the internal palette of the video subsystem is loaded by BIOS and should not be changed. The first 16 locations in the external palette, which is in the video DAC, contain the colors compatible with the alphanumeric modes. The second 16 locations contain 16 evenly spaced gray shades. The next 216 locations contain values based on a hue-saturation-intensity model tuned to provide a usable, generic color set that covers a wide range of color values.

The following figure shows the color information that is compatible with the colors in other modes.

Pel Bits 7 6 5 4 3 2 1 0	Color Output
0 0 0 0 0 0 0 0	Black
0 0 0 0 0 0 0 1	Blue
0 0 0 0 0 0 1 0	Green
0 0 0 0 0 0 1 1	Cyan
0 0 0 0 0 1 0 0	Red
0 0 0 0 0 1 0 1	Magenta
0 0 0 0 0 1 1 0	Brown
0 0 0 0 0 1 1 1	White
0 0 0 0 1 0 0 0	Dark Gray
0 0 0 0 1 0 0 1	Light Blue
0 0 0 0 1 0 1 0	Light Green
0 0 0 0 1 0 1 1	Light Cyan
0 0 0 0 1 1 0 0	Light Red
0 0 0 0 1 1 0 1	Light Magenta
0 0 0 0 1 1 1 0	Yellow
0 0 0 0 1 1 1 1	Intensified White

Figure 2-14. Compatible Color Coding

Each color in the palette can be programmed to one of 256K different colors.

The features of this mode are:

- A maximum of 200 rows with 320 pels
- Double scanned to display as 400 rows
- Memory-mapped graphics
- 256 of 256K colors for each pel
- One byte per pel
- 64KB of video memory

Video Memory Organization

The display buffer consists of 256KB of dynamic read/write memory configured as four 64KB memory maps.

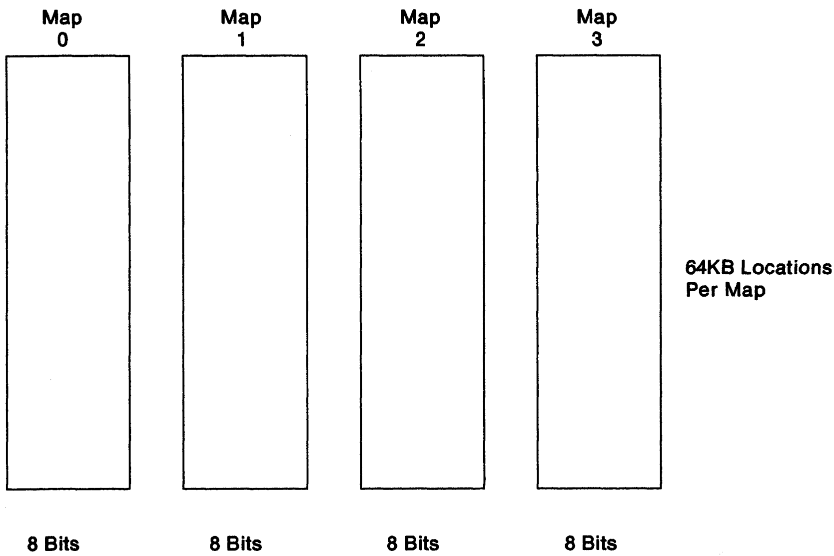


Figure 2-15. 256KB Video Memory Map

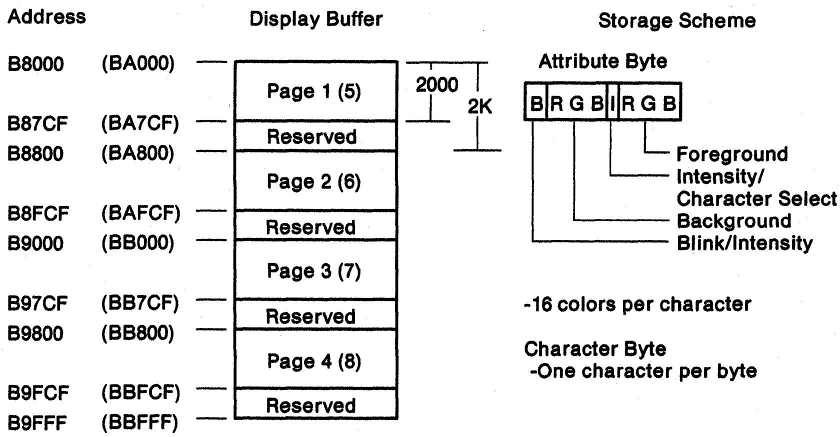
The starting address and size of the display buffer can be changed to maintain compatibility with other display adapters and application software. There are three configurations used by other adapters:

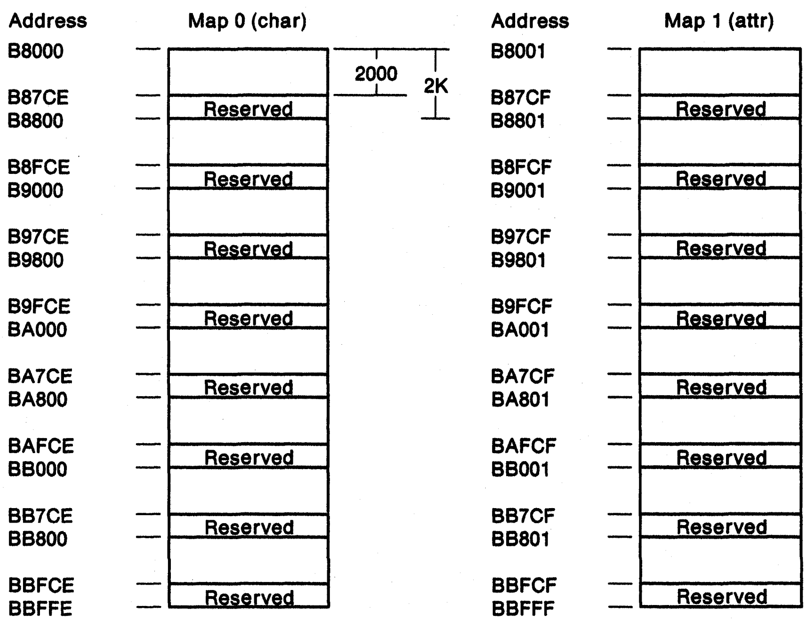
Address hex A0000 for a length of 64KB
Address hex B0000 for a length of 32KB
Address hex B8000 for a length of 32KB.

Memory Modes

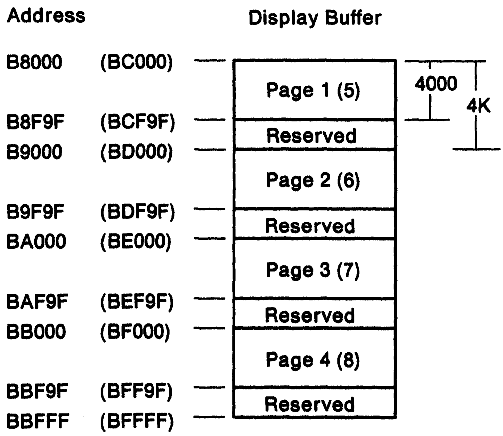
The following pages show the memory organization for each of the BIOS modes.

Modes Hex 0, 1

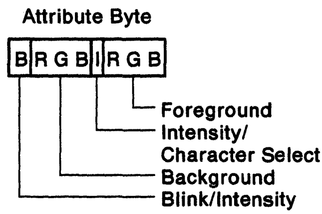

Address
Display Buffer
Storage Scheme
B8000 (BA000)
B87CF (BA7CF)
B8800 (BA800)
B8FCF (BAFCF)
B9000 (BB000)
B97CF (BB7CF)
B9800 (BB800)
B9FCF (BBFCF)
B9FFF (BBFFF)
Page 1 (5)
Reserved
Page 2 (6)
Reserved
Page 3 (7)
Reserved
Page 4 (8)
Reserved
2000
2K
Attribute Byte
B R G B I R G B
Foreground
Intensity/
Character Select
Background
Blink/Intensity
-16 colors per character
Character Byte
-One character per byte


Address
Map 0 (char)
B8000
B87CE
B8800
B8FCE
B9000
B97CE
B9800
B9FCE
BA000
BA7CE
BA800
BAFCE
BB000
BB7CE
BB800
BBFCE
BBFFE
Reserved
Reserved
Reserved
Reserved
Reserved
Reserved
Reserved
Reserved
2000
2K
Address
Map 1 (attr)
B8001
B87CF
B8801
B8FCF
B9001
B97CF
B9801
B9FCF
BA001
BA7CF
BA801
BAFCF
BB001
BB7CF
BB801
BBFCF
BBFFF
Reserved
Reserved
Reserved
Reserved
Reserved
Reserved
Reserved
Reserved

Modes Hex 2, 3

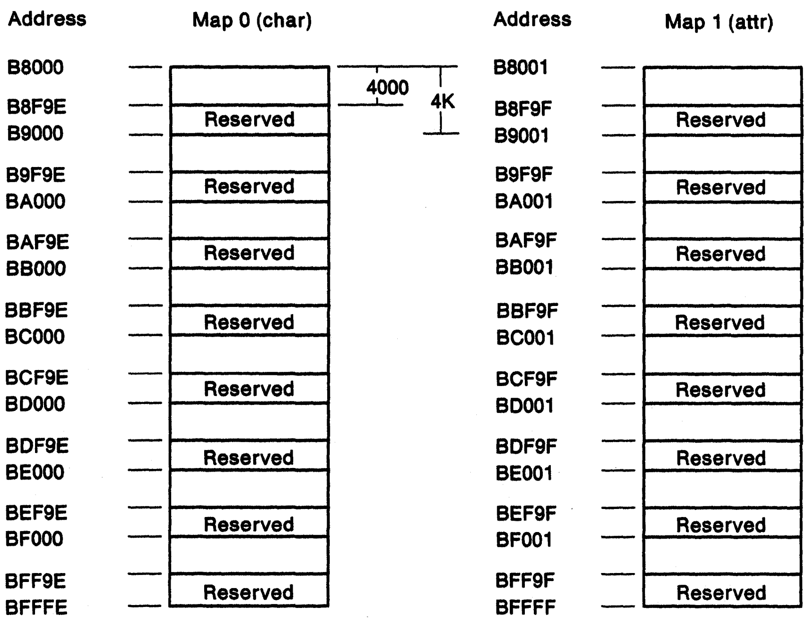


Storage Scheme

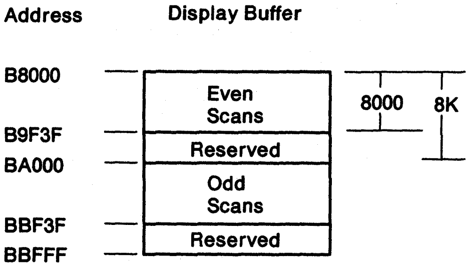


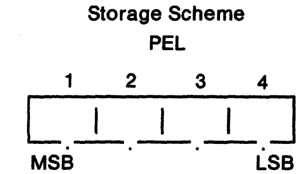
-16 colors per character

Character Byte
-One character per byte

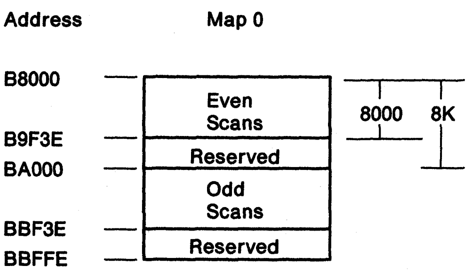


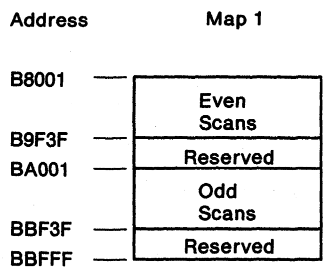
Modes Hex 4, 5



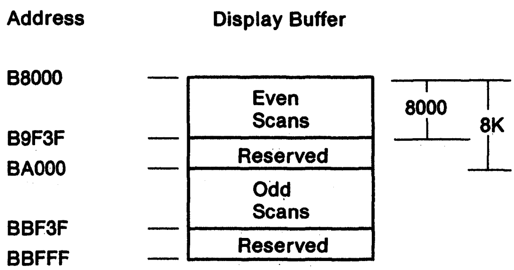


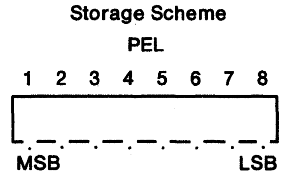
- 4 PELs per byte
- 4 colors per PEL
- First PEL is 2 most significant bits



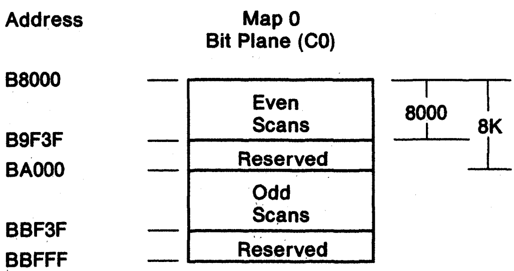


Mode Hex 6



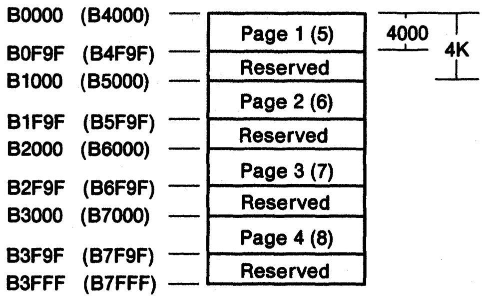


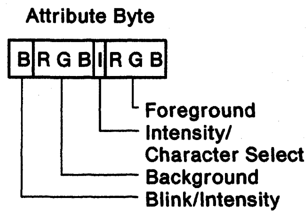
- 8 PELs per byte
- 2 colors per PEL
- First PEL is most significant bit



Mode Hex 7

Address · Display Buffer · Storage Scheme

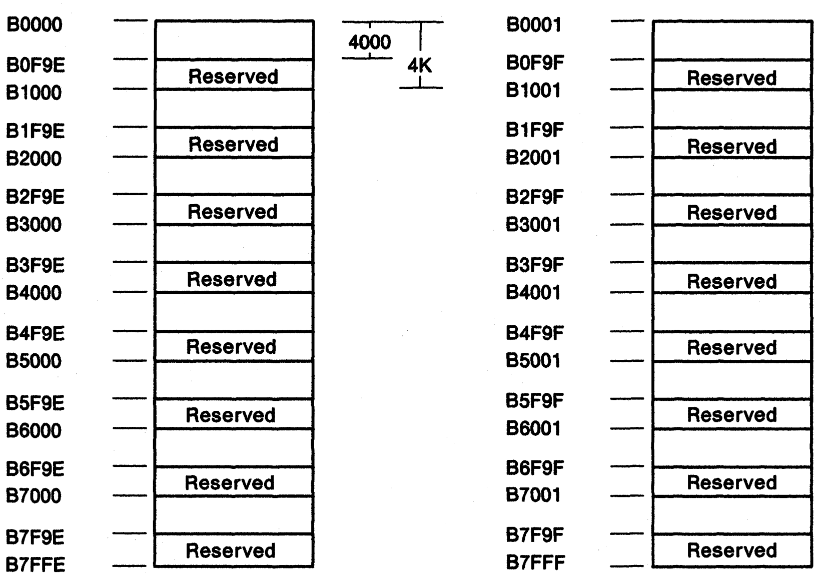




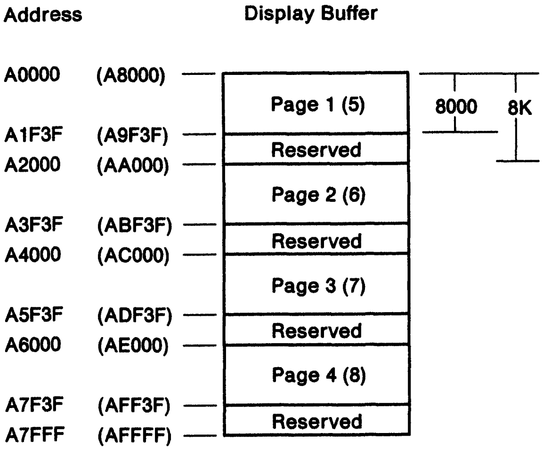
-Four attributes per character

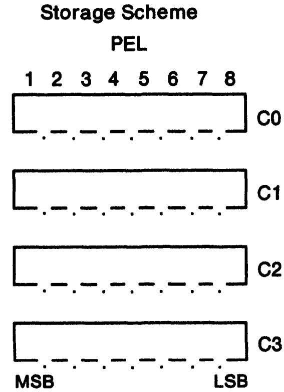
Character Byte
-One character per byte

Address · Map 0 (char) · Address · Map 1 (attr)

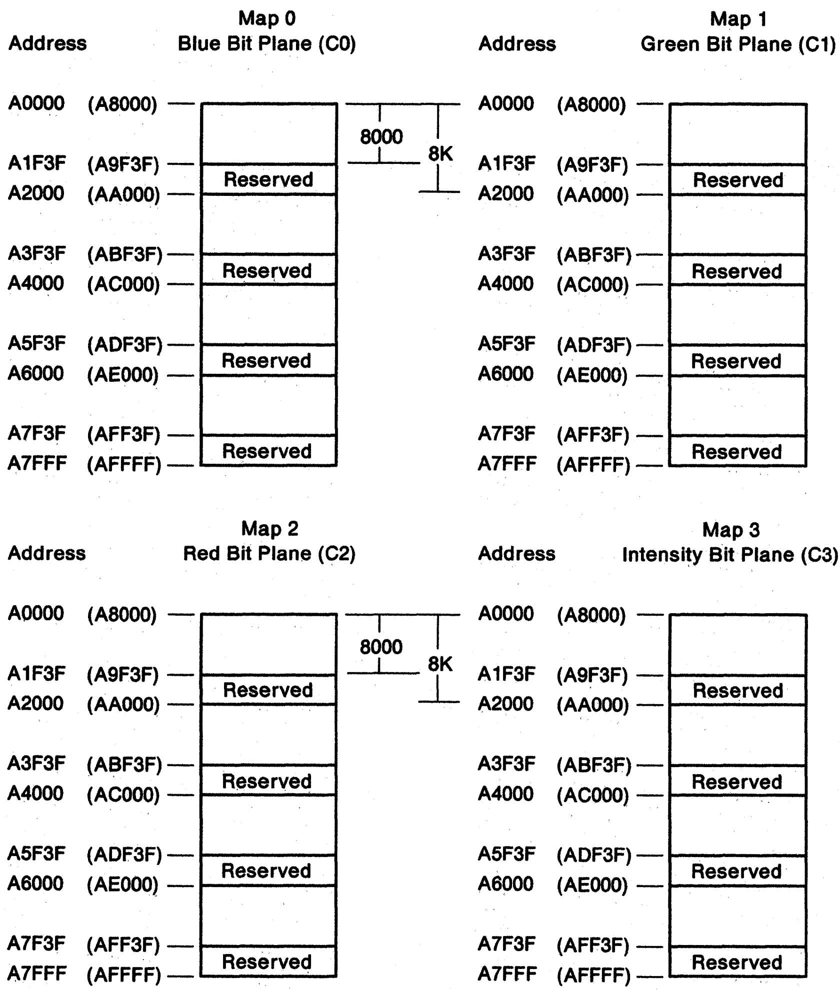


Mode Hex D

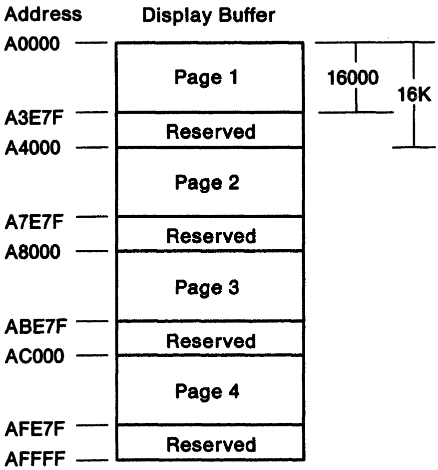


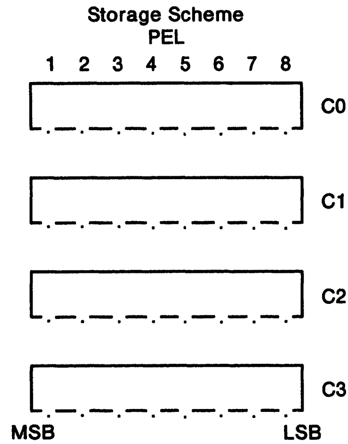


- 4 bits per PEL
- 16 colors per PEL
- 1 bit from each bit plane (C3,C2,C1,CO) per PEL
- First PEL is most significant bit of all 4 bit planes

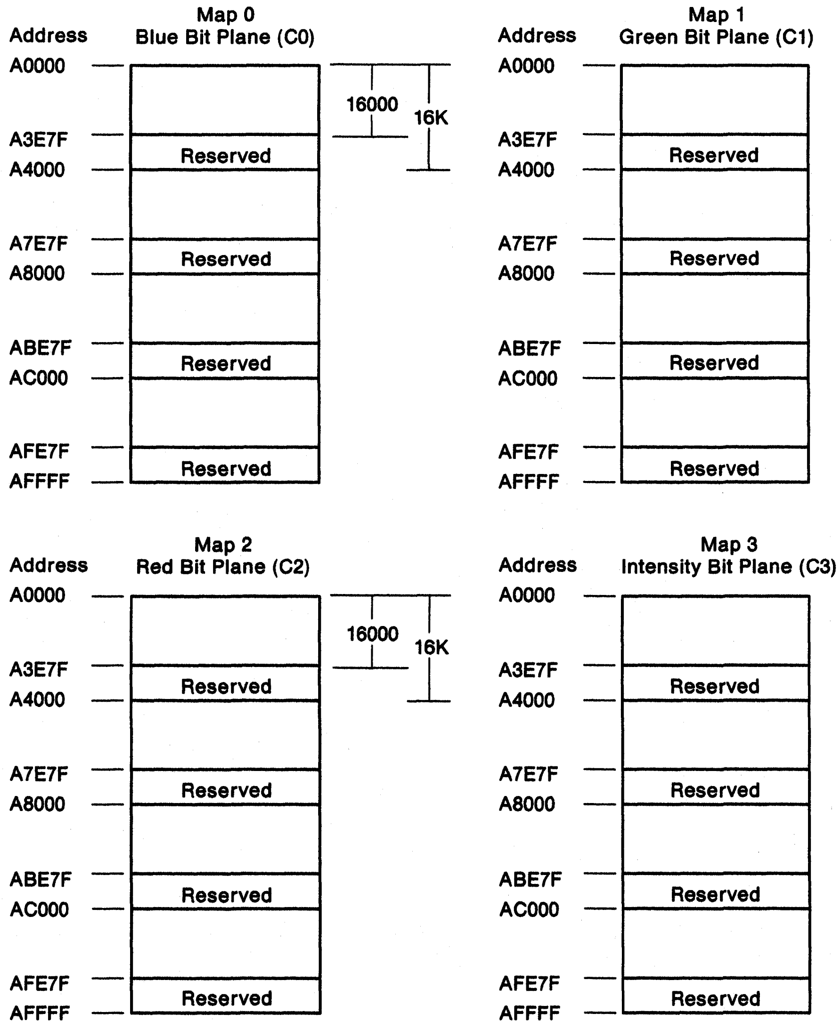

Address
Map 0
Blue Bit Plane (C0)
A0000 (A8000)
A1F3F (A9F3F)
A2000 (AA000)
A3F3F (ABF3F)
A4000 (AC000)
A5F3F (ADF3F)
A6000 (AE000)
A7F3F (AFF3F)
A7FFF (AFFFF)
Reserved
Reserved
Reserved
Reserved
8000
8K
Address
Map 1
Green Bit Plane (C1)
A0000 (A8000)
A1F3F (A9F3F)
A2000 (AA000)
A3F3F (ABF3F)
A4000 (AC000)
A5F3F (ADF3F)
A6000 (AE000)
A7F3F (AFF3F)
A7FFF (AFFFF)
Reserved
Reserved
Reserved
Reserved
Address
Map 2
Red Bit Plane (C2)
A0000 (A8000)
A1F3F (A9F3F)
A2000 (AA000)
A3F3F (ABF3F)
A4000 (AC000)
A5F3F (ADF3F)
A6000 (AE000)
A7F3F (AFF3F)
A7FFF (AFFFF)
Reserved
Reserved
Reserved
Reserved
8000
8K
Address
Map 3
Intensity Bit Plane (C3)
A0000 (A8000)
A1F3F (A9F3F)
A2000 (AA000)
A3F3F (ABF3F)
A4000 (AC000)
A5F3F (ADF3F)
A6000 (AE000)
A7F3F (AFF3F)
A7FFF (AFFFF)
Reserved
Reserved
Reserved
Reserved

Mode Hex E



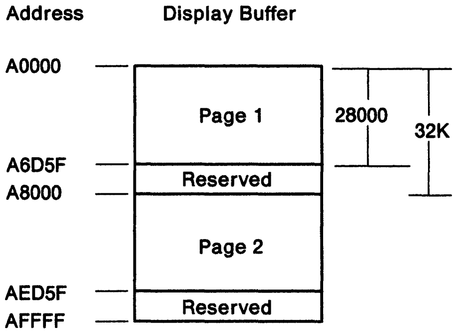


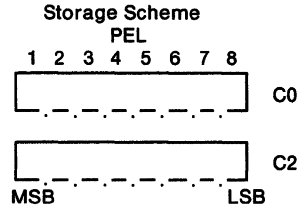
- 4 bits per PEL
- 16 colors per PEL
- 1 bit from each bit plane (C3,C2,C1,C0) per PEL
- First PEL is most significant bit of all 4 bit planes



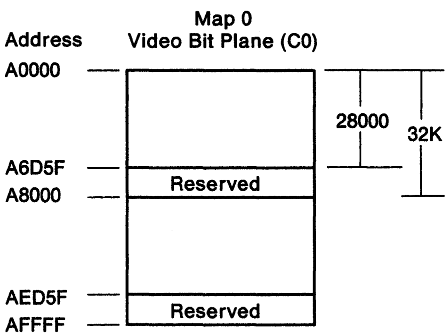
Map 0
Blue Bit Plane (C0)
Address
A0000
A3E7F
A4000
A7E7F
A8000
ABE7F
AC000
AFE7F
AFFFF
Reserved
Reserved
Reserved
Reserved
16000
16K
Map 1
Green Bit Plane (C1)
Address
A0000
A3E7F
A4000
A7E7F
A8000
ABE7F
AC000
AFE7F
AFFFF
Reserved
Reserved
Reserved
Reserved
Map 2
Red Bit Plane (C2)
Address
A0000
A3E7F
A4000
A7E7F
A8000
ABE7F
AC000
AFE7F
AFFFF
Reserved
Reserved
Reserved
Reserved
16000
16K
Map 3
Intensity Bit Plane (C3)
Address
A0000
A3E7F
A4000
A7E7F
A8000
ABE7F
AC000
AFE7F
AFFFF
Reserved
Reserved
Reserved
Reserved

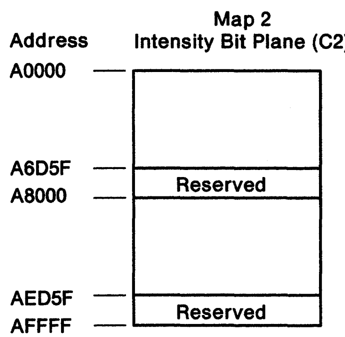
Mode Hex F



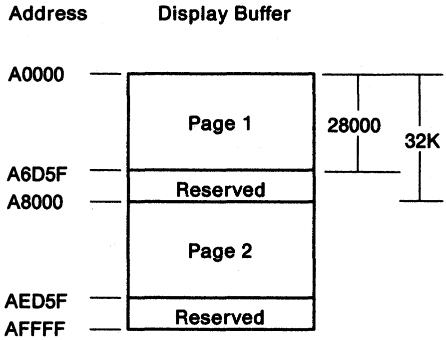


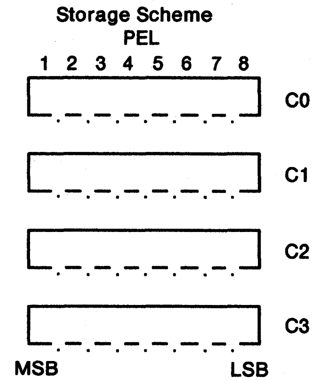
- 2 bits per PEL
- 4 attributes per PEL
- 1 bit from each bit plane (C2,C0)
- First PEL is most significant bit of video and intensity bit planes



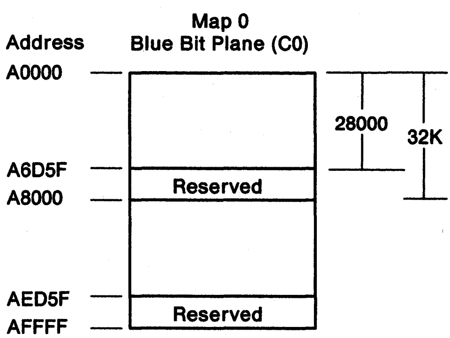


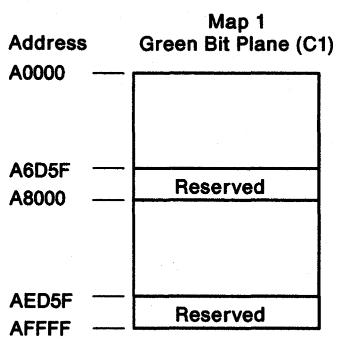
Mode Hex 10

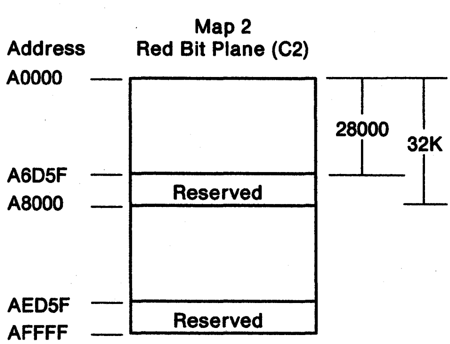


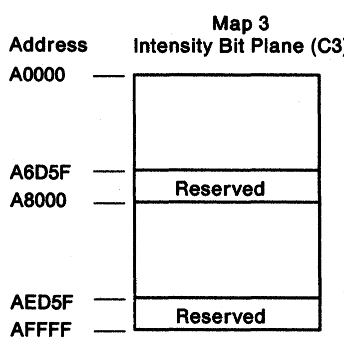


- 4 bits per PEL
- 16 colors per PEL
- 1 bit from each bit plane (C3,C2,C1,C0) per PEL
- First PEL is most significant bit of all 4 bit planes

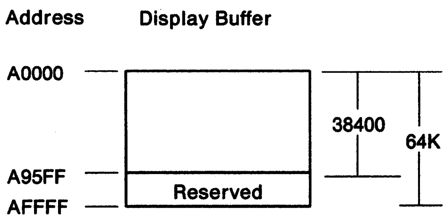


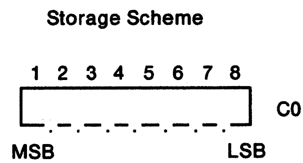




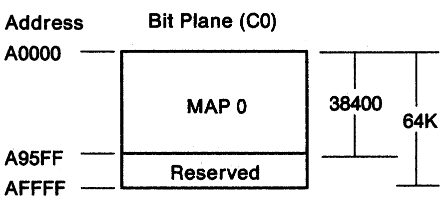


Mode Hex 11

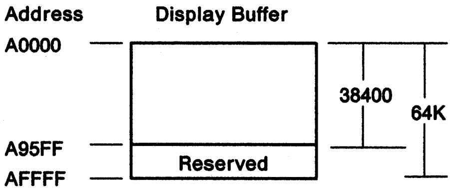


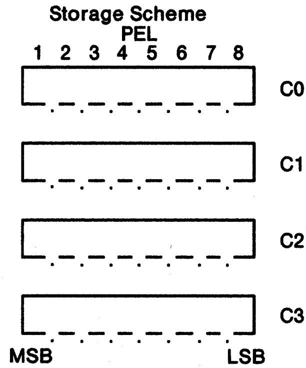


- One bit per PEL
- Two attributes per PEL
- First PEL is most significant bit

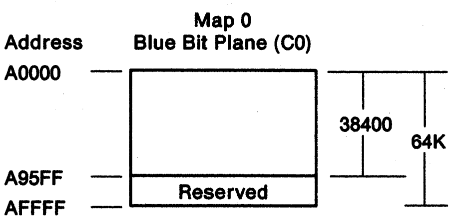


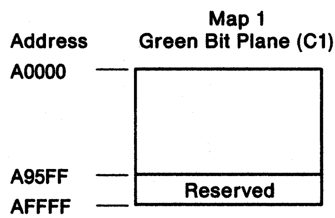
Mode Hex 12

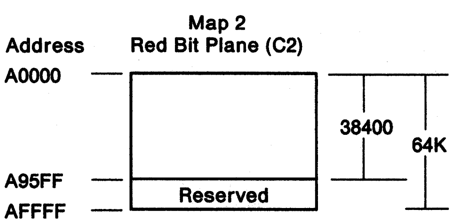


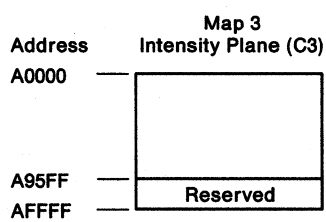


- 4 bits per PEL
- 16 colors per PEL
- 1 bit from each bit plane (C3,C2,C1,C0) per PEL
- First PEL is most significant bit of all 4 bit planes

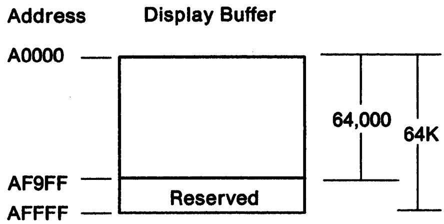






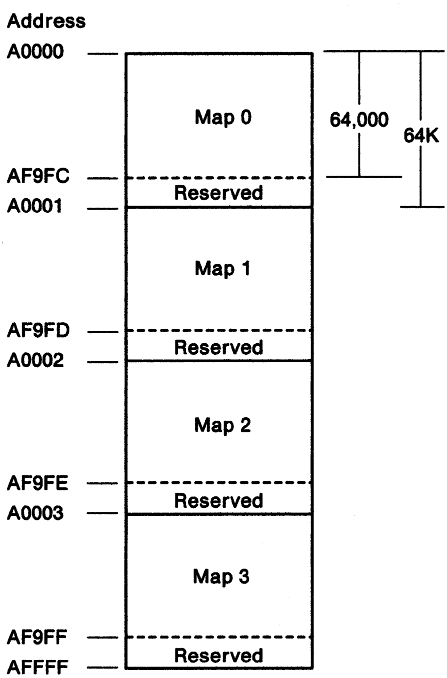


Mode Hex 13

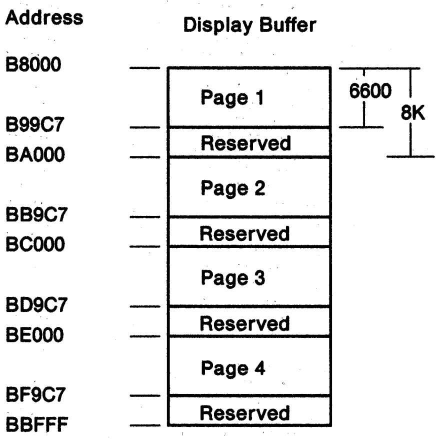


Storage Scheme

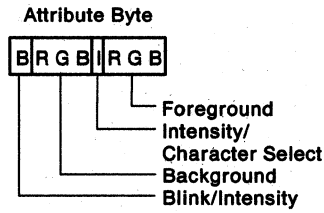
- 8 bits per PEL
- 256 colors per PEL
- 1 PEL per byte
- First PEL is at address A0000



Mode Hex 14

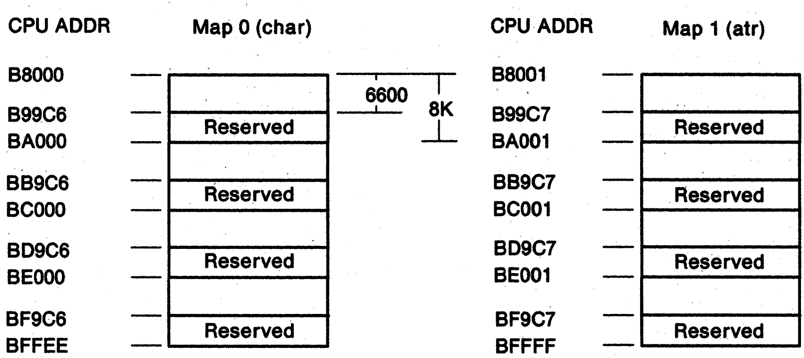


Storage Scheme



-16 colors per character

Character Byte
One character per byte



Registers

There are six groups of registers in the video subsystem. All video registers are readable except the system data latches and the attribute address flip-flop. The following figure lists the register groups, their I/O addresses with the type of access (read or write), and page reference numbers.

The question mark in the address can be a hex B or D depending on the setting of the I/O address bit in the Miscellaneous Output register, described in "General Registers" on page 2-40.

Note: All registers in the video subsystem are read/write. The value of reserved bits in these registers must be preserved. Read the register first and change only the bits required.

Registers	R/W	Port Address	Page Reference
General Registers			2-40
Sequencer Registers			2-45
Address Register	R/W	03C4	
Data Registers	R/W	03C5	
CRT Controller Registers			2-53
Address Register	R/W	03?4	
Data Registers	R/W	03?5	
Graphics Controller Registers			2-76
Address Register	R/W	03CE	
Data Registers	R/W	03CF	
Attribute Controller Registers			2-87
Address Register	R/W	03C0	
Data Registers	W	03C0	
	R	03C1	
Video DAC Palette Registers			2-101
Write Address	R/W	03C8	
Read Address	W	03C7	
Data	R/W	03C9	
Pel Mask	R/W	03C6	

Figure 2-16. Video Subsystem Register Overview

General Registers

Register	Read Address	Write Address
Miscellaneous Output Register	03CC	03C2
Input Status Register 0	03C2	–
Input Status Register 1	03?A	–
Feature Control Register	03CA	03?A
Video Subsystem Enable Register	03C3	03C3

Figure 2-17. General Registers

Miscellaneous Output Register

The read address for this register is hex 03CC and its write address is hex 03C2.

7	6	5	4	3	2	1	0
VSP	HSP	–	–	CS		ERAM	IOS

```
   - : Set to 0, Undefined on Read
 VSP : Vertical Sync Polarity
 HSP : Horizontal Sync Polarity
  CS : Clock Select
ERAM : Enable RAM
 IOS : I/O Address Select
```

Figure 2-18. Miscellaneous Output Register, Hex 03CC/03C2

The register fields are defined as follows:

VSP When set to 0, the Vertical Sync Polarity field (bit 7) selects a positive 'vertical retrace' signal. This bit works with bit 6 to determine the vertical size.

HSP When set to 0, the Horizontal Sync Polarity field (bit 6) selects a positive 'horizontal retrace' signal. Bits 7 and 6 select the vertical size as shown in the following figure.

Bits 7 6	Vertical Size
0 0	Reserved
0 1	400 lines
1 0	350 lines
1 1	480 lines

Figure 2-19. Display Vertical Size

CS The Clock Select field (bits 3, 2) selects the clock source according to the following figure. The external clock is driven through the auxiliary video extension. The input clock should be kept between 14.3 MHz and 28.4 MHz.

Bits 3 2	Function
0 0	Selects the clock for 640/320 Horizontal Pels
0 1	Selects the clock for 720/360 Horizontal Pels
1 0	Selects External Clock
1 1	Reserved

Figure 2-20. Clock Select Definitions

ERAM When set to 0, the Enable RAM field (bit 1) disables address decode for the display buffer from the system.

IOS The I/O Address Select field (bit 0) selects the CRT controller addresses. When set to 0, this bit sets the CRT controller addresses to hex 03Bx and the address for the Input Status Register 1 to hex 03BA for compatibility with the monochrome adapter. When set to 1, this bit sets CRT controller addresses to hex 03Dx and the Input Status Register 1 address to hex 03DA for compatibility with the color/graphics adapter. The write addresses to the Feature Control register are affected in the same manner.

Input Status Register 0

The address for this read-only register is address hex 03C2. *Do not write to* this register.

7	6	5	4	3	2	1	0
CI	–	–	SS	–	–	–	–

– : Undefined on Read
CI : CRT Interrupt
SS : Switch Sense

Figure 2-21. Input Status Register 0, Hex 03C2

The register fields are defined as follows:

CI When the CRT Interrupt field (bit 7) is 1, a vertical retrace interrupt is pending.

SS BIOS uses the Switch Sense field (bit 4) in determining the type of display attached.

Input Status Register 1

The address for this read-only register is address hex 03DA or 03BA. *Do not write to* this register.

7	6	5	4	3	2	1	0
–	–	–	–	VR	–	–	DE

– : Undefined on Read
VR : Vertical Retrace
DE : Display Enable

Figure 2-22. Input Status Register 1, Hex 03DA/03BA

The register fields are defined as follows:

VR When the Vertical Retrace field (bit 3) is 1, it indicates a vertical retrace interval. This bit can be programmed, through the Vertical Retrace End register, to generate an interrupt at the start of the vertical retrace.

DE When the Display Enable field (bit 0) is 1, it indicates a horizontal or vertical retrace interval. This bit is the real-time status of the inverted 'display enable' signal. In the past, programs have used this status bit to restrict screen updates to the inactive display intervals to reduce screen flicker. The video subsystem is designed to eliminate this software requirement; screen updates can be made at any time without screen degradation.

Feature Control Register

The write address of this register is hex 03DA or 03BA; its read address is hex 03CA. All bits are reserved.

7	6	5	4	3	2	1	0
Feature Control							

Figure 2-23. Feature Control Register, Hex 03DA/03BA and 03CA

Video Subsystem Enable Register

This register (hex 03C3) is reserved. To disable address decoding by the video subsystem, use BIOS INT 10 call, AH = hex 12, BL = hex 32.

7	6	5	4	3	2	1	0
Video Subsystem Enable							

Figure 2-24. Video Subsystem Enable Register, Hex 03C3

Sequencer Registers

The Address register is at address hex 03C4 and the data registers are at address hex 03C5. All registers within the sequencer are read/write.

Register	Index (Hex)
Sequencer Address	–
Reset	00
Clocking Mode	01
Map Mask	02
Character Map Select	03
Memory Mode	04

Figure 2-25. Sequencer Registers

Sequencer Address Register

The Address register is at address hex 03C4. This register is loaded with an index value that points to the desired sequencer data register.

7	6	5	4	3	2	1	0
–	–	–	–	–	SA		

– : Set to 0, Undefined on Read
SA : Sequencer Address

Figure 2-26. Sequencer Address Register

The register field is defined as follows:

SA The Sequencer Address field (bits 2–0) contains the index value that points to the data register to be accessed.

Reset Register

This read/write register has an index of hex 00; its address is hex 03C5.

7	6	5	4	3	2	1	0
–	–	–	–	–	–	SR	ASR

– : Set to 0, Undefined on Read
SR : Synchronous Reset
ASR : Asynchronous Reset

Figure 2-27. Reset Register, Index Hex 00

The register fields are defined as follows:

SR When set to 0, the Synchronous Reset field (bit 1) commands the sequencer to synchronously clear and halt. Bits 1 and 0 must be 1 to allow the sequencer to operate. To prevent the loss of data, bit 1 must be set to 0 during the active display interval before changing the clock selection. The clock is changed through the Clocking Mode register or the Miscellaneous Output register.

ASR When set to 0, the Asynchronous Reset field (bit 0) commands the sequencer to asynchronously clear and halt. Resetting the sequencer with this bit can cause loss of video data.

Clocking Mode Register

This read/write register has an index of hex 01; its address is hex 03C5.

7	6	5	4	3	2	1	0
–	–	SO	SH4	DC	SL	1	D89

```
  – : Set to 0, Undefined on Read
  1 : Set to 1, Undefined on Read
 SO : Screen Off
SH4 : Shift 4
 DC : Dot Clock
 SL : Shift Load
D89 : 8/9 Dot Clocks
```

Figure 2-28. Clocking Mode Register, Index Hex 01

The register fields are defined as follows:

SO When set to 1, the Screen Off field (bit 5) turns off the display and assigns maximum memory bandwidth to the system. Although the display is blanked, the synchronization pulses are maintained. This bit can be used for rapid full-screen updates.

SH4 When the Shift 4 field (bit 4) and Shift Load field (bit 2) are set to 0, the video serializers are loaded every character clock. When the Shift 4 field is set to 1, the video serializers are loaded every fourth character clock, which is useful when 32 bits are fetched per cycle and chained together in the shift registers.

Extended Graphics mode behaves as if this bit is set to 0; therefore, programs should set it to 0.

DC When set to 0, the Dot Clock field (bit 3) selects the normal dot clocks derived from the sequencer master clock input. When set to 1, the master clock is divided by 2 to generate the dot clock. All other timings are affected because they are derived from the dot clock. The dot clock divided by 2 is used for 320 and 360 horizontal pel modes.

SL When the Shift Load field (bit 2) and Shift 4 field (bit 4) are set to 0, the video serializers are loaded every character clock. When the Shift Load field (bit 2) is set to 1, the video serializers are loaded every other character clock, which is useful when 16 bits are fetched per cycle and chained together in the shift registers.

Extended Graphics mode behaves as if this bit is set to 0; therefore, programs should set it to 0.

D89 When set to 0, the 8/9 Dot Clocks field (bit 0) directs the sequencer to generate character clocks 9 dots wide; when set to 1, it directs the sequencer to generate character clocks 8 dots wide. The 9-dot mode is for alphanumeric modes 0+, 1+, 2+, 3+, 7, and 7+ only; the 9th dot equals the 8th dot for ASCII codes hex C0 through DF. All other modes must use 8 dots per character clock. (See the Enable Line Graphics Character Code field in the Attribute Mode Control register on page 2-89.)

Map Mask Register

This read/write register has an index of hex 02; its address is hex 03C5.

7	6	5	4	3	2	1	0
–	–	–	–	M3E	M2E	M1E	M0E

```
  – : Set to 0, Undefined on Read
M3E : Map 3 Enable
M2E : Map 2 Enable
M1E : Map 1 Enable
M0E : Map 0 Enable
```

Figure 2-29. Map Mask Register, Index Hex 02

The register fields are defined as follows:

M3E, M2E, M1E, M0E

When set to 1, the map enable fields (bits 3, 2, 1, and 0) enable system access to the corresponding map. If all maps are enabled (chain 4 mode), the system can write its 8-bit value to all four maps in a single memory cycle. This substantially reduces the system overhead during display updates in graphics modes.

Data scrolling operations can be enhanced by enabling all maps and writing the display buffer address with the data stored in the system data latches. This is a read-modify-write operation.

When access to odd or even maps (odd/even modes) are selected, maps 0 and 1 and maps 2 and 3 should have the same map mask value.

When chain 4 mode is selected, all maps should be enabled.

Character Map Select Register

This register has an index of hex 03; its address is hex 03C5. In alphanumeric modes, bit 3 of the attribute byte normally defines the foreground intensity. This bit can be redefined as a switch between character sets, allowing 512 characters that can be displayed. To enable this feature:

1. Set the extended memory bit in the Memory Mode register (index hex 04) to 1.
2. Select different values for character map A and character map B.

This function is supported by BIOS and it is a function call within the character generator routines.

7	6	5	4	3	2	1	0
–	–	MAH	MBH	MAL		MBL	

- : Set to 0, Undefined on Read
MAH : Character Map A Select (MSB)
MBH : Character Map B Select (MSB)
MAL : Character Map A Select (LSB)
MBL : Character Map B Select (LSB)

Figure 2-30. Character Map Select Register, Index Hex 03

The register fields are defined as follows:

MAH The Character Map A Select field (bit 5) is the most-significant bit for selecting the location of character map A.

MBH The Character Map B Select field (bit 4) is the most-significant bit for selecting the location of character map B.

MAL The Character Map A Select field (bits 3, 2) and Character Map A Select field (bit 5) select the location of character map A. Map A is the area of map 2 containing the character font table used to generate characters when attribute bit 3 is set to 1. The selection is shown in the following figure.

Bits 5 3 2	Map Selected	Table Location
0 0 0	0	1st 8KB of Map 2
0 0 1	1	3rd 8KB of Map 2
0 1 0	2	5th 8KB of Map 2
0 1 1	3	7th 8KB of Map 2
1 0 0	4	2nd 8KB of Map 2
1 0 1	5	4th 8KB of Map 2
1 1 0	6	6th 8KB of Map 2
1 1 1	7	8th 8KB of Map 2

Figure 2-31. Character Map Select A

MBL The Character Map B Select field (bits 1, 0) and Character Map B Select field (bit 4) select the location of character map B. Map B is the area of map 2 containing the character font table used to generate characters when attribute bit 3 is set to 0. The selection is shown in the following figure.

Bits 4 1 0	Map Selected	Table Location
0 0 0	0	1st 8KB of Map 2
0 0 1	1	3rd 8KB of Map 2
0 1 0	2	5th 8KB of Map 2
0 1 1	3	7th 8KB of Map 2
1 0 0	4	2nd 8KB of Map 2
1 0 1	5	4th 8KB of Map 2
1 1 0	6	6th 8KB of Map 2
1 1 1	7	8th 8KB of Map 2

Figure 2-32. Character Map Select B

Memory Mode Register

This register has an index of hex 04; its address is hex 03C5.

7	6	5	4	3	2	1	0
–	–	–	–	CH4	OE	EM	–

```
  – : Set to 0, Undefined on Read
CH4 : Chain 4
 OE : Odd/Even
 EM : Extended Memory
```

Figure 2-33. Memory Mode Register, Index Hex 04

The register fields are defined as follows:

CH4 The Chain 4 field (bit 3) controls the map selected during system read operations. When set to 0, this bit enables system addresses to sequentially access data within a bit map by using the Map Mask register. When set to 1, this bit causes the 2 low-order bits to select the map accessed as shown in the following figure.

Address Bits A1 A0	Map Selected
0 0	0
0 1	1
1 0	2
1 1	3

Figure 2-34. Map Selection, Chain 4

OE When the Odd/Even field (bit 2) is set to 0, even system addresses access maps 0 and 2, while odd system addresses access maps 1 and 3. When set to 1, system addresses sequentially access data within a bit map, and the maps are accessed according to the value in the Map Mask register (hex 02).

EM When set to 1, the Extended Memory field (bit 1) enables the video memory from 64KB to 256KB. This bit must be set to 1 to enable the character map selection described for the previous register.

CRT Controller Registers

A data register is accessed by writing its index to the Address register at address hex 03D4 or 03B4, and then writing the data to the access port at address hex 03D5 or 03B5. The I/O address used depends on the setting of the I/O address select bit (bit 0) in the Miscellaneous Output register, which is described in "General Registers" on page 2-40. The following figure shows the variable part of the address as a question mark.

Note: When modifying a register, the setting of reserved bits must be preserved. Read the register first and change only the bits required.

Register Name	Address (Hex)	Index (Hex)
Address	03?4	–
Horizontal Total	03?5	00
Horizontal Display-Enable End	03?5	01
Start Horizontal Blanking	03?5	02
End Horizontal Blanking	03?5	03
Start Horizontal Retrace Pulse	03?5	04
End Horizontal Retrace	03?5	05
Vertical Total	03?5	06
Overflow	03?5	07
Preset Row Scan	03?5	08
Maximum Scan Line	03?5	09
Cursor Start	03?5	0A
Cursor End	03?5	0B
Start Address High	03?5	0C
Start Address Low	03?5	0D
Cursor Location High	03?5	0E
Cursor Location Low	03?5	0F
Vertical Retrace Start	03?5	10
Vertical Retrace End	03?5	11
Vertical Display-Enable End	03?5	12
Offset	03?5	13
Underline Location	03?5	14
Start Vertical Blanking	03?5	15
End Vertical Blanking	03?5	16
CRT Mode Control	03?5	17
Line Compare	03?5	18
Index values not listed are reserved.		

Figure 2-35. CRT Controller Registers

Address Register

This register is at address hex 03B4 or 03D4, and is loaded with an index value that points to the data registers within the CRT controller.

7	6	5	4	3	2	1	0
–	–	–	Index				

– : Set to 0, Undefined on Read

Figure 2-36. CRT Controller Address Register, Hex 03B4/03D4

The register field is defined as follows:

Index This field (bits 4–0) is the index that points to the data register accessed through address hex 03D5 or 03B5.

Horizontal Total Register

This register has an index of hex 00; its address is hex 03D5 or 03B5.

The Horizontal Total register (bits 7–0) defines the total number of characters in the horizontal scan interval including the retrace time. The value directly controls the period of the 'horizontal retrace' signal. A horizontal character counter in the CRT controller counts the character clock inputs; comparators are used to compare the register value with the horizontal width of the character to provide horizontal timings. All horizontal and vertical timings are based on this register.

7	6	5	4	3	2	1	0
Horizontal Total							

Figure 2-37. Horizontal Total Register, Index Hex 00

The value contained in the register is the total number of characters minus 5.

Horizontal Display-Enable End Register

This register has an index of hex 01; its address is hex 03D5 or 03B5.

7	6	5	4	3	2	1	0
Horizontal Display Enable End							

Figure 2-38. Horizontal Display Enable-End Register, Index Hex 01

The Horizontal Display-Enable End register (bits 7 – 0) defines the length of the 'horizontal display-enable' signal and determines the number of character positions per horizontal line. The value in this register is the total number of displayed characters minus 1.

Start Horizontal Blanking Register

This register has an index of hex 02; its address is hex 03D5 or 03B5.

7	6	5	4	3	2	1	0
Start Horizontal Blanking							

Figure 2-39. Start Horizontal Blanking Register, Index Hex 02

The value in the Start Horizontal Blanking register (bits 7 – 0) is the horizontal character count at which the 'horizontal blanking' signal becomes active.

End Horizontal Blanking Register

This register has an index of hex 03; its address is hex 03D5 or 03B5. It determines when the 'horizontal blanking' signal will become active.

7	6	5	4	3	2	1	0
1	DES		EB				

```
  1 : Set to 1, Undefined on Read
DES : Display Enable Skew Control
 EB : End Blanking
```

Figure 2-40. End Horizontal Blanking Register, Index Hex 03

The register fields are defined as follows:

DES The Display Enable Skew Control field (bits 6, 5) determines the amount of skew of the 'display enable' signal. This skew control is needed to provide sufficient time for the CRT controller to read a character and attribute code from the video buffer, gain access to the character generator, and go through the Horizontal Pel Panning register in the attribute controller. Each access requires the 'display enable' signal to be skewed one character clock so that the video output is synchronized with the horizontal and vertical retrace signals. The skew values are shown in the following figure.

DES Field (binary)	Amount of Skew
0 0	No character clock skew
0 1	One character clock skew
1 0	Two character clock skew
1 1	Three character clock skew

Figure 2-41. Display Enable Skew

Note: Character skew is not adjustable on the Type 2 video and the bits are ignored; however, programs should set these bits for the appropriate skew to maintain compatibility.

EB The End Blanking field (bits 4 – 0) contains the 5 low-order bits of a 6-bit value that is compared with the value in the Start Horizontal Blanking register to determine when the 'horizontal blanking' signal becomes inactive. The most-significant bit is bit 7 in the End Horizontal Retrace register (index hex 05).

To program these bits for a signal width of W, the following algorithm is used: the width W, in character clock units, is added to the value from the Start Horizontal Blanking register. The 6 low-order bits of the result are the 6-bit value programmed.

Start Horizontal Retrace Pulse Register

This register has an index of hex 04; its address is hex 03D5 or 03B5.

7	6	5	4	3	2	1	0
Start Horizontal Retrace Pulse							

Figure 2-42. Start Horizontal Retrace Pulse Register, Index Hex 04

The Start Horizontal Retrace Pulse register (bits 7 – 0) is used to center the screen horizontally by specifying the character position where the 'horizontal retrace' signal becomes active.

End Horizontal Retrace Register

This register has an index of hex 05; its address is hex 03D5 or 03B5.

7	6	5	4	3	2	1	0
EB5	HRD		EHR				

```
EB5 : End Horizontal Blanking, Bit 5
HRD : Horizontal Retrace Delay
EHR : End Horizontal Retrace
```

Figure 2-43. End Horizontal Retrace Register, Index Hex 05

The register fields are defined as follows:

EB5 The End Horizontal Blanking, Bit 5 field (bit 7) is the most-significant bit of the end horizontal blanking value in the End Horizontal Blanking register (index hex 03).

HRD The Horizontal Retrace Delay field (bits 6, 5) controls the skew of the 'horizontal retrace' signal. The value of this field is the amount of skew provided (from 0 to 3 character clock units). For certain modes, the 'horizontal retrace' signal takes up the entire blanking interval. Some internal timings are generated by the falling edge of the 'horizontal retrace' signal. To ensure that the signals are latched properly, the 'retrace' signal is started before the end of the 'display enable' signal and then skewed several character clock times to provide the proper screen centering.

EHR The End Horizontal Retrace field (bits 4–0) is compared with the Start Horizontal Retrace register to give a horizontal character count at which the 'horizontal retrace' signal becomes inactive.

To program these bits with a signal width of W, the following algorithm is used: the width W, in character clock units, is added to the value in the Start Retrace register. The 5 low-order bits of the result are the 5-bit value programmed.

Vertical Total Register

This register has an index of hex 06; its address is hex 03D5 or 03B5.

7	6	5	4	3	2	1	0
Vertical Total							

Figure 2-44. Vertical Total Register, Index Hex 06

The Vertical Total register (bits 7 – 0) contains the 8 low-order bits of a 10-bit vertical total. The value for the vertical total is the number of horizontal raster scans on the display, including vertical retrace, minus 2. This value determines the period of the 'vertical retrace' signal.

Bits 8 and 9 are in the Overflow register (index hex 07).

Overflow Register

This register has an index of hex 07; its address is hex 03D5 or 03B5.

7	6	5	4	3	2	1	0
VRS9	VDE9	VT9	LC8	VBS8	VRS8	VDE8	VT8

```
VRS9 : Vertical Retrace Start, Bit 9
VDE9 : Vertical Display Enable End, Bit 9
 VT9 : Vertical Total, Bit 9
 LC8 : Line Compare, Bit 8
VBS8 : Vertical Blanking Start, Bit 8
VRS8 : Vertical Retrace Start, Bit 8
VDE8 : Vertical Display Enable End, Bit 8
 VT8 : Vertical Total, Bit 8
```

Figure 2-45. CRT Overflow Register, Index Hex 07

The register fields are defined as follows:

VRS9 This is bit 9 of the Vertical Retrace Start register (index hex 10).

VDE9 This is bit 9 of the Vertical Display-Enable End register (index hex 12).

VT9 This is bit 9 of the Vertical Total register (index hex 06).

LC8 This is bit 8 of the Line Compare register (index hex 18).

VBS8 This is bit 8 of the Start Vertical Blanking register (index hex 15).

VRS8 This is bit 8 of the Vertical Retrace Start register (index hex 10).

VDE8 This is bit 8 of the Vertical Display-Enable End register (index hex 12).

VT8 This is bit 8 of the Vertical Total register (index hex 06).

Preset Row Scan Register

This register has an index of hex 08; its address is hex 03D5 or 03B5.

7	6	5	4	3	2	1	0
–	BP		SRS				

– : Set to 0, Undefined on Read
BP : Byte Panning
SRS : Starting Row Scan Count

Figure 2-46. Preset Row Scan Register, Index Hex 08

The register fields are defined as follows:

BP The Byte Panning field (bits 6, 5) controls byte panning in multiple shift modes. (BIOS modes do not use multiple shift operation.) These bits are used in pel-panning operations, and should normally be set to 0.

Extended Graphics mode behaves as if these bits are set to 0; therefore, programs should set it to 0.

SRS The Starting Row Scan Count field (bits 4–0) specifies the row scan count for the row starting after a vertical retrace. The row scan counter is incremented every horizontal retrace time until the maximum row scan occurs. When the maximum row scan is reached, the row scan counter is cleared (not preset).

Note: The CRT controller latches the start address at the start of the vertical retrace. These register values should be loaded during the active display time.

Maximum Scan Line Register

This register has an index of hex 09; its address is hex 03D5 or 03B5.

7	6	5	4	3	2	1	0
DSC	LC9	VBS9	MSL				

```
 DSC : 200 to 400 Line Conversion (Double Scanning)
 LC9 : Line Compare, Bit 9
VBS9 : Start Vertical Blanking, Bit 9
 MSL : Maximum Scan Line
```

Figure 2-47. Maximum Scan Line Register, Index Hex 09

The register fields are defined as follows:

DSC When the 200 to 400 Line Conversion field (bit 7) is set to 1, 200-scan-line video data is converted to 400-scan-line output. To do this, the clock in the row scan counter is divided by 2, which allows the 200-line modes to be displayed as 400 lines on the display (this is called double scanning; each line is displayed twice). When set to 0, the clock to the row scan counter is equal to the horizontal scan rate.

LC9 The Line Compare, Bit 9 field (bit 6) is bit 9 of the Line Compare register (index hex 18).

VBS9 The Start Vertical Blanking, Bit 9 field (bit 5) is bit 9 of the Start Vertical Blanking register (index hex 15).

MSL The Maximum Scan Line field (bits 4 – 0) specifies the number of scan lines per character row. The value of this field is the maximum row scan number minus 1.

Cursor Start Register

This register has an index of hex 0A; its address is hex 03D5 or 03B5.

7	6	5	4	3	2	1	0
–	–	CO	RSCB				

```
   – : Set to 0, Undefined on Read
  CO : Cursor Off
RSCB : Row Scan Cursor Begins
```

Figure 2-48. Cursor Start Register, Index Hex 0A

The register fields are defined as follows:

CO When the Cursor Off field (bit 5) is set to 1, the cursor is disabled.

RSCB The Row Scan Cursor Begins field (bits 4–0) specifies the row within the character box where the cursor begins. The value of this field is the first line of the cursor minus 1. When this value is greater than that in the Cursor End register, no cursor is displayed.

Cursor End Register

This register has an index of hex 0B; its address is hex 03D5 or 03B5.

7	6	5	4	3	2	1	0
–	CSK		RSCE				

```
   – : Set to 0, Undefined on Read
 CSK : Cursor Skew Control
RSCE : Row Scan Cursor Ends
```

Figure 2-49. Cursor End Register, Index Hex 0B

The register fields are defined as follows:

CSK The Cursor Skew Control field (bits 6, 5) controls the skew of the cursor. The skew value delays the cursor by the selected number of character clocks from 0 to 3. For example, a skew of 1 moves the cursor right one position on the screen.

RSCE The Row Scan Cursor Ends field (bits 4–0) specifies the row within the character box where the cursor ends. If this value is less than that in the Cursor Start register, no cursor is displayed.

Start Address High Register

This register has an index of hex 0C; its address is hex 03D5 or 03B5.

7	6	5	4	3	2	1	0
Start Address High							

Figure 2-50. Start Address High Register, Index Hex 0C

The Start Address High register (bits 7 – 0) contains the 8 high-order bits of a 16-bit value that specifies the starting address for the regenerative buffer. The start address points to the first address after the vertical retrace on each screen refresh.

Note: The CRT controller latches the start address at the start of the vertical retrace. These register values should be loaded during the active display time.

Start Address Low Register

This register has an index of hex 0D; its address is hex 03D5 or 03B5.

7	6	5	4	3	2	1	0
Start Address Low							

Figure 2-51. Start Address Low Register, Index Hex 0D

The Start Address Low register (bits 7 – 0) contains the 8 low-order bits of the starting address for the regenerative buffer.

Cursor Location High Register

This register has an index of hex 0E; its address is hex 03D5 or 03B5.

```
 7   6   5   4   3   2   1   0
+-----------------------------+
|    Cursor Location High     |
+-----------------------------+
```

Figure 2-52. Cursor Location High Register, Index Hex 0E

The Cursor Location High register (bits 7 – 0) contains the 8 high-order bits of the 16-bit cursor location.

Cursor Location Low Register

This register has an index of hex 0F; its address is hex 03D5 or 03B5.

```
 7   6   5   4   3   2   1   0
+-----------------------------+
|    Cursor Location Low      |
+-----------------------------+
```

Figure 2-53. Cursor Location Low Register, Index Hex 0F

The Cursor Location Low register (bits 7 – 0) contains the 8 low-order bits of the cursor location.

Vertical Retrace Start Register

This register has an index of hex 10; its address is hex 03D5 or 03B5.

7	6	5	4	3	2	1	0
Vertical Retrace Start							

Figure 2-54. Vertical Retrace Start Register, Index Hex 10

The Vertical Retrace Start register (bits 7 – 0) contains the 8 low-order bits of the 9-bit start position for the 'vertical retrace' signal; it is programmed in horizontal scan lines. Bit 8 is in the Overflow register (index hex 07).

Vertical Retrace End Register

This register has an index of hex 11; its address is hex 03D5 or 03B5.

7	6	5	4	3	2	1	0
PR	S5R	EVI	CVI	VRE			

```
 PR : Protect Registers 0-7
S5R : Select 5 Refresh Cycles
EVI : Enable Vertical Interrupt
CVI : Clear Vertical Interrupt
VRE : Vertical Retrace End
```

Figure 2-55. Vertical Retrace End Register, Index Hex 11

The register fields are defined as follows:

PR When the Protect Registers 0 – 7 field (bit 7) is set to 1, write access to the CRT controller registers at index 00 through 07 is disabled. The line compare bit in the Overflow register (index hex 07) is not protected.

S5R When the Select 5 Refresh Cycles field (bit 6) is set to 1, five memory refresh cycles per horizontal line are generated. When set to 0, three refresh cycles are selected. Selecting five refresh cycles allows use of the VGA chip with 15.75 kHz displays. This bit should be set to 0 for supported operations. It is set to 0 by a BIOS mode set, a reset, or a power on.

EVI When the Enable Vertical Interrupt field (bit 5) is set to 0, it enables a vertical retrace interrupt. The vertical retrace interrupt is IRQ2. This interrupt level can be shared; therefore, to determine whether the video generated the interrupt, check the CRT interrupt bit in Input Status Register 0.

CVI When the Clear Vertical Interrupt field (bit 4) is set to 0, it clears a vertical retrace interrupt. At the end of the active vertical display time, a flip-flop is set to indicate an interrupt. An interrupt handler resets this flip-flop by first setting this bit to 0, then resetting it to 1.

VRE The Vertical Retrace Start register is compared with the 4 bits in the Vertical Retrace End field (bits 3 – 0) to determine where the 'vertical retrace' signal becomes inactive. It is programmed in units of horizontal scan lines. To program these bits with a signal width of W, the following algorithm is used: the width W, in horizontal scan units, is added to the value in the Start Vertical Retrace register. The 4 low-order bits of the result are the 4-bit value programmed.

Vertical Display-Enable End Register

This register has an index of hex 12; its address is hex 03D5 or 03B5.

7	6	5	4	3	2	1	0
Vertical Display Enable End							

Figure 2-56. Vertical Display-Enable End Register, Index Hex 12

The Vertical Display-Enable End register (bits 7 – 0) contains the 8 low-order bits of a 10-bit value that defines the vertical-display-enable end position. The 2 high-order bits are contained in the Overflow register (index hex 07). The 10-bit value is equal to the total number of scan lines minus 1.

Offset Register

This register has an index of hex 13; its address is hex 03D5 or 03B5.

7	6	5	4	3	2	1	0
Offset							

Figure 2-57. Offset Register, Index Hex 13

The Offset register (bits 7 – 0) specifies the logical line width of the screen. The starting memory address for the next character row is larger than the current character row by 2 or 4 times the value of these bits. Depending on the method of clocking the CRT controller, this address is either a word or doubleword address.

Underline Location Register

This register has an index of hex 14; its address is hex 03D5 or 03B5.

7	6	5	4	3	2	1	0
–	DW	CB4	SUL				

```
  - : Set to 0, Undefined on Read
 DW : Doubleword Mode
CB4 : Count By 4
SUL : Start Underline
```

Figure 2-58. Underline Location Register, Index Hex 14

The register fields are defined as follows:

DW When the Doubleword Mode field (bit 6) is set to 1, memory addresses are doubleword addresses. See the description of the word/byte mode bit (bit 6) in the CRT Mode Control register on page 2-72.

CB4 When the Count By 4 field (bit 5) is set to 1, the memory-address counter is clocked with the character clock divided by 4, which is used when doubleword addresses are used.

SUL The Start Underline field (bits 4–0) specifies the horizontal scan line of a character row on which an underline occurs. The value programmed is the scan line desired minus 1.

Start Vertical Blanking Register

This register has an index of hex 15; its address is hex 03D5 or 03B5.

7	6	5	4	3	2	1	0
Start Vertical Blanking							

Figure 2-59. Start Vertical Blanking Register, Index Hex 15

The Start Vertical Blanking register (bits 7 – 0) contains the 8 low-order bits of a 10-bit value that specifies the starting location for the 'vertical blanking' signal. Bit 8 is in the Overflow register (index hex 07) and bit 9 is in the Maximum Scan Line register (index hex 09). The 10-bit value is the horizontal scan line count at which the 'vertical blanking' signal becomes active, minus 1.

End Vertical Blanking Register

This register has an index of hex 16; its address is hex 03D5 or 03B5.

7	6	5	4	3	2	1	0
End Vertical Blanking							

Figure 2-60. End Vertical Blanking Register, Index Hex 16

The End Vertical Blanking register (bits 7 – 0) specifies the horizontal scan count at which the 'vertical blanking' signal becomes inactive. The register is programmed in units of the horizontal scan line.

To program these bits with a 'vertical blanking' signal of width W, the following algorithm is used: the width W, in horizontal scan line units, is added to the value in the Start Vertical Blanking register minus 1. The 8 low-order bits of the result are the 8-bit value programmed.

CRT Mode Control Register

This register has an index of hex 17; its address is hex 03D5 or 03B5.

7	6	5	4	3	2	1	0
RST	WB	ADW	–	CB2	HRS	SRC	CMS0

```
   – : Set to 0, Undefined on Read
 RST : Hardware Reset
  WB : Word/Byte Mode
 ADW : Address Wrap
 CB2 : Count By Two
 HRS : Horizontal Retrace Select
 SRC : Select Row Scan Counter
CMS0 : CMS 0
```

Figure 2-61. CRT Mode Control Register, Index Hex 17

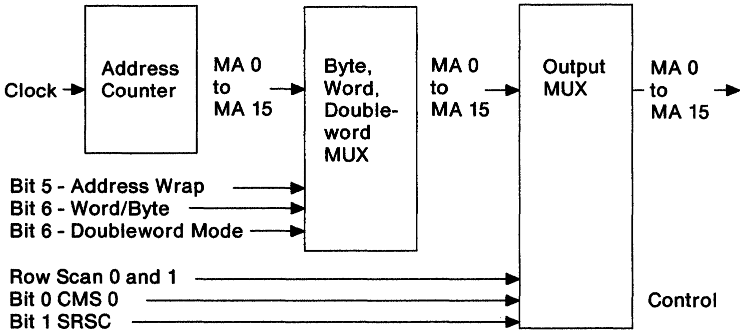
The register fields are defined as follows:

RST When the Hardware Reset field (bit 7) is set to 0, this bit disables the horizontal and vertical retrace signals and forces them to an inactive level. When set to 1, this bit enables the horizontal and vertical retrace signals. This bit does not reset any other registers or signal outputs.

WB When the Word/Byte Mode field (bit 6) is set to 0, the word mode is selected. The word mode shifts the memory-address counter bits down 1 bit; the most-significant bit of the counter appears on the least-significant bit of the memory address outputs.

The doubleword bit in the Underline Location register (index hex 14) also controls the addressing. When the doubleword bit is 0, the word/byte bit selects the mode. When the doubleword bit is set to 1, the addressing is shifted by 2 bits.

When the Word/Byte Mode field is set to 1, it selects the byte address mode. See the following figure for address output details.



Memory Address Outputs	Modes of Addressing		
	Byte	Word	Doubleword
MA 0	MA 0	MA 15 or 13	MA 12
MA 1	MA 1	MA 0	MA 13
MA 2	MA 2	MA 1	MA 0
MA 3	MA 3	MA 2	MA 1
MA 4	MA 4	MA 3	MA 2
MA 5	MA 5	MA 4	MA 3
MA 6	MA 6	MA 5	MA 4
MA 7	MA 7	MA 6	MA 5
MA 8	MA 8	MA 7	MA 6
MA 9	MA 9	MA 8	MA 7
MA 10	MA 10	MA 9	MA 8
MA 11	MA 11	MA 10	MA 9
MA 12	MA 12	MA 11	MA 10
MA 13	MA 13	MA 12	MA 11
MA 14	MA 14	MA 13	MA 12
MA 15	MA 15	MA 14	MA 13

Figure 2-62. CRT Memory Address Mapping

ADW The Address Wrap field (bit 5) selects the memory-address bit, bit MA 13 or MA 15, that appears on the output pin MA 0 in the word address mode. If VGA is not in the word address mode, bit 0 from the address counter appears on the output pin, MA 0.

When set to 1, the Address Wrap field bit selects MA 15. In odd/even mode, this bit should be set to 1 because 256KB of video memory is installed on the system board. (Bit MA 13 is selected in applications where only 64KB is present. This function maintains compatibility with the IBM Color/Graphics Monitor Adapter.)

CB2 When the Count By Two field (bit 3) is set to 0, the address counter uses the character clock. When set to 1, the address counter uses the character clock input divided by 2. This bit is used to create either a byte or word refresh address for the display buffer.

HRS The Horizontal Retrace Select field (bit 2) selects the clock that controls the vertical timing counter. The clocking is either the horizontal retrace clock or horizontal retrace clock divided by 2. When set to 1, the horizontal retrace clock is divided by 2.

Dividing the clock effectively doubles the vertical resolution of the CRT controller. The vertical counter has a maximum resolution of 1024 scan lines because the vertical total value is 10 bits wide. If the vertical counter is clocked with the horizontal retrace divided by 2, the vertical resolution is doubled to 2048 scan lines.

SRC The Select Row Scan Counter field (bit 1) selects the source of bit 14 of the output multiplexer. When set to 0, bit 1 of the row scan counter is the source. When set to 1, bit 14 of the address counter is the source.

CMS0 The CMS 0 field (bit 0) selects the source of bit 13 of the output multiplexer. When set to 0, bit 0 of the row scan counter is the source. When set to 1, bit 13 of the address counter is the source.

The CRT controller used on the IBM Color/Graphics Adapter was capable of using 128 horizontal scan-line addresses. For VGA to obtain 640-by-200 graphics resolution, the CRT controller is programmed for 100 horizontal scan lines with two scan-line addresses per character row. Row scan address bit 0 becomes the most-significant address bit to the display buffer. Successive scan lines of the display image are displaced in 8KB of memory. This bit allows compatibility with the graphics modes of earlier adapters.

Line Compare Register

This register has an index of hex 18; its address is hex 03D5 or 03B5.

7	6	5	4	3	2	1	0
Line Compare							

Figure 2-63. Line Compare Register, Index Hex 18

The Line Compare Register (bits 7 – 0) contains the 8 low-order bits of the 10-bit compare target. When the vertical counter reaches the target value, the internal start address of the line counter is cleared. This creates a split screen in which the lower screen is immune to scrolling. Bit 8 is in the Overflow register (index hex 07), and bit 9 is in the Maximum Scan Line register (index hex 09).

Graphics Controller Registers

The Address register for the graphics controller is at address hex 03CE. The data registers are at address hex 03CF. All registers are read/write.

Register Name	Address (Hex)	Index (Hex)
Address	03CE	
Set/Reset	03CF	00
Enable Set/Reset	03CF	01
Color Compare	03CF	02
Data Rotate	03CF	03
Read Map Select	03CF	04
Graphics Mode	03CF	05
Miscellaneous	03CF	06
Color Don't Care	03CF	07
Bit Mask	03CF	08

Figure 2-64. Graphics Controller Register Overview

Address Register

The Address register is at address hex 03CE. This register is loaded with the index value that points to the desired data register within the graphics controller.

7	6	5	4	3	2	1	0
–	–	–	–	Index			

– : Set to 0, Undefined on Read

Figure 2-65. Graphics Controller Address Register, Hex 03CE

The register field is defined as follows:

Index The Index field (bits 3 – 0) contains the index value that points to the data registers.

Set/Reset Register

This register has an index of hex 00; its address is hex 03CF.

7	6	5	4	3	2	1	0
–	–	–	–	SR3	SR2	SR1	SR0

```
  – : Set to 0, Undefined on Read
SR3 : Set/Reset Map 3
SR2 : Set/Reset Map 2
SR1 : Set/Reset Map 1
SR0 : Set/Reset Map 0
```

Figure 2-66. Set/Reset Register, Index Hex 00

The register fields are defined as follows:

SR3, SR2, SR1, SR0
When write mode 0 is selected, the system writes the value of each set/reset field (bits 3, 2, 1, or 0) to its respective memory map. For each write operation, the set/reset bit, if enabled, is written to all 8 bits within that map. Set/reset operation can be enabled on a map-by-map basis through the Enable Set/Reset register.

Enable Set/Reset Register

The index for this register is hex 01; its address is hex 03CF.

7	6	5	4	3	2	1	0
–	–	–	–	ESR3	ESR2	ESR1	ESR0

```
   - : Set to 0, Undefined on Read
ESR3 : Enable Set/Reset Map 3
ESR2 : Enable Set/Reset Map 2
ESR1 : Enable Set/Reset Map 1
ESR0 : Enable Set/Reset Map 0
```

Figure 2-67. Enable Set/Reset Register, Index Hex 01

The register fields are defined as follows:

ESR3, ESR2, ESR1, ESR0
These fields (bits 3, 2, 1, and 0) enable the set/reset function used when write mode 0 is selected in the Graphics Mode register (index hex 05). When set to 1, the respective memory map receives the value specified in the Set/Reset register. When Set/Reset is not enabled for a map, that map receives the value sent by the system.

Color Compare Register

This register has an index of hex 02; its address is hex 03CF.

7	6	5	4	3	2	1	0
–	–	–	–	CC3	CC2	CC1	CC0

```
  - : Set to 0, Undefined on Read
CC3 : Color Compare Map 3
CC2 : Color Compare Map 2
CC1 : Color Compare Map 1
CC0 : Color Compare Map 0
```

Figure 2-68. Color Compare Register, Index Hex 02

The register fields are defined as follows:

CC3, CC2, CC1, CC0

These color compare map fields (bits 3, 2, 1, and 0) make up the 4-bit color value to be compared when the read mode bit in the Graphics Mode register is set to 1. When the system does a memory read, the data returned from the memory cycle will be a 1 in each bit position where the four maps equal the Color Compare register. If the read mode bit is 0, the data is returned without comparison.

All bits of the corresponding map's byte are compared with the color compare bit. Each of the 8 bit positions in the selected byte are compared across the four maps, and a 1 is returned in each position where the bits of all four maps equal their respective color compare values.

Data Rotate Register

This register has an index of hex 03; its address is hex 03CF.

7	6	5	4	3	2	1	0
–	–	–	FS		RC		

```
 – : Set to 0, Undefined on Read
FS : Function Select
RC : Rotate Count
```

Figure 2-69. Data Rotate Register, Index Hex 03

The register fields are defined as follows:

FS Data written to the video buffer can be operated on logically by data already in the system latches. The Function Select field (bits 4, 3) determines whether and how this is done.

Data can be any of the choices selected by the write mode bits except system latches, which cannot be modified. If rotated data is selected also, the rotation is performed before the logical operation. The logical operations selected are shown in the following table.

FS Field (binary)	Function
0 0	Data Unmodified
0 1	Data ANDed with Latched Data
1 0	Data ORed with Latched Data
1 1	Data XORed with Latched Data

Figure 2-70. Operation Select Bit Definitions

RC In write mode 0, the Rotate Count field (bits 2–0) selects the number of positions the system data is rotated to the right during a system memory write operation. To write data that is not rotated in mode 0, all bits are set to 0.

Read Map Select Register

This register has an index of hex 04; its address is hex 03CF.

7	6	5	4	3	2	1	0
–	–	–	–	–	–	MS	

```
 – : Set to 0, Undefined on Read
MS : Map Select
```

Figure 2-71. Read Map Select Register, Index Hex 04

The register field is defined as follows:

MS The Map Select field (bits 1, 0) selects the memory map for system read operations. This register has no effect on the color compare read mode. In odd/even modes, the value can be a binary 00 or 01 to select the chained maps 0, 1 and the value can be a binary 10 or 11 to select the chained maps 2, 3.

Graphics Mode Register

This register has an index of hex 05; its address is hex 03CF.

7	6	5	4	3	2	1	0
–	C256	SR	OE	RM	–	WM	

```
   – : Set to 0, Undefined on Read
C256 : 256 - Color Mode
  SR : Shift Register Mode
  OE : Odd/Even
  RM : Read Mode
  WM : Write Mode
```

Figure 2-72. Graphics Mode Register, Index Hex 05

The register fields are defined as follows:

C256 When set to 0, the 256-Color Mode field (bit 6) allows bit 5 to control the loading of the shift registers. When set to 1, this field causes the shift registers to be loaded in a manner that supports the 256-color mode.

SR When set to 1, the Shift Register Mode field (bit 5) directs the shift registers in the graphics controller to format the serial data stream with even-numbered bits from both maps on even-numbered maps, and odd-numbered bits from both maps on the odd-numbered maps. This bit is used for modes 4 and 5.

OE When set to 1, the Odd/Even field (bit 4) selects the odd/even addressing mode used by the IBM Color/Graphics Monitor Adapter. Normally, the value here follows the value of Memory Mode register bit 2 in the sequencer.

RM When the Read Mode field (bit 3) is set to 1, the system reads the results of the comparison of the four memory maps and the Color Compare register.

When set to 0, the system reads data from the memory map selected by the Read Map Select register, or by the 2 low-order bits of the memory address (this selection depends on the chain-4 bit in the Memory Mode register of the sequencer).

WM The Write Mode field (bits 1, 0) determines the write mode selected. The write mode selected and its operation are defined in the following figure. The logic operation specified by the function select bits is performed on system data for modes 0, 2, and 3.

WM Field (binary)	Mode Description
0 0	Each memory map is written with the system data rotated by the count in the Data Rotate register. If the set/reset function is enabled for a specific map, that map receives the 8-bit value contained in the Set/Reset register.
0 1	Each memory map is written with the contents of the system latches. These latches are loaded by a system read operation.
1 0	Memory map *n* (0 through 3) is filled with 8 bits of the value of data bit *n*.
1 1	Each memory map is written with the 8-bit value contained in the Set/Reset register for that map (the Enable Set/Reset register has no effect). Rotated system data is ANDed with the Bit Mask register to form an 8-bit value that performs the same function as the Bit Mask register in write modes 0 and 2 (see also Bit Mask register on page 2-86).

Figure 2-73. Write Mode Definitions

Miscellaneous Register

This register has an index of hex 06; its address is hex 03CF.

7	6	5	4	3	2	1	0
–	–	–	–	MM		OE	GM

```
 - : Set to 0, Undefined on Read
MM : Memory Map
OE : Odd/Even
GM : Graphics Mode
```

Figure 2-74. Miscellaneous Register, Index Hex 06

The register fields are defined as follows:

MM The Memory Map field (bits 3, 2) controls the mapping of the regenerative buffer into the system address space. The bit functions are defined in the following figure.

MM Field (binary)	Addressing Assignment
0 0	A0000 for 128KB
0 1	A0000 for 64KB
1 0	B0000 for 32KB
1 1	B8000 for 32KB

Figure 2-75. Video Memory Assignments

OE When set to 1, the Odd/Even field (bit 1) directs the system address bit, A0, to be replaced by a higher-order bit. The odd map is then selected when A0 is 1, and the even map when A0 is 0.

GM The Graphics Mode field (bit 0) controls alphanumeric mode addressing. When set to 1, this bit selects graphics modes, which also disables the character generator latches.

Color Don't Care Register

This register has an index of hex 07; its address is hex 03CF.

7	6	5	4	3	2	1	0
–	–	–	–	M3	M2	M1	M0

```
 – : Set to 0, Undefined on Read
M3 : Compare Map 3
M2 : Compare Map 2
M1 : Compare Map 1
M0 : Compare Map 0
```

Figure 2-76. Color Don't Care Register, Index Hex 07

The register fields are defined as follows:

M3, M2, M1, M0
The compare map fields (bits 3, 2, 1, and 0) select whether a map is going to participate in the color compare cycle. When set to 1, the bits in that map are compared. When set to 0, the bits in that map are not compared.

Bit Mask Register

This register has an index of hex 08; its address is hex 03CF.

7	6	5	4	3	2	1	0
Bit Mask							

Figure 2-77. Bit Mask Register, Index Hex 08

When a bit in the Bit Mask register (bits 7 – 0) is set to 1, the corresponding bit position in each map can be changed. When a bit is set to 0, the bit position in the map is masked to prevent change, provided that the location being written was the last location read by the system microprocessor.

The bit mask applies to write modes 0 and 2. To preserve bits using the bit mask, data must be latched internally by reading the location. When data is written to preserve the bits, the most current data in the latches is written in those positions. The bit mask applies to all maps simultaneously.

Attribute Controller Registers

Each register for the attribute controller has two addresses. Address hex 03C0 is the write address and hex 03C1 is the read address. The individual data registers are selected by writing their index to the Address register (hex 03C0).

Register Name	Write Address	Read Address	Index
Address	03C0	03C0	–
Internal Palette	03C0	03C1	00 – 0F
Attribute Mode Control			10
Overscan Color			11
Color Plane Enable			12
Horizontal Pel Panning			13
Color Select			14

Figure 2-78. Attribute Controller Register Addresses

Address Register

This read/write register is at address hex 03C0.

The attribute controller registers do not have an input bit to control selection of the address and data registers. An internal address flip-flop controls this selection. Reading Input Status Register 1 clears the flip-flop and selects the Address register.

After the Address register has been loaded with the index, the next write operation to 03C0 loads the data register. The flip-flop toggles for each write operation to address hex 03C0. It does not toggle for Read operations to 03C0 or 03C1. (Also see "VGA Programming Considerations" on page 2-94.)

7	6	5	4	3	2	1	0
–	–	IPAS	Index				

– : Set to 0, Undefined on Read
IPAS : Internal Palette Address Source

Figure 2-79. Address Register, Hex 03C0

The register fields are defined as follows:

IPAS The Internal Palette Address Source field (bit 5) is set to 0 to load color values to the registers in the internal palette. It is set to 1 for normal operation of the attribute controller.

Note: Do not access the internal palette while this bit is set to 1. While this bit is 1, the Type 1 video subsystem disables accesses to the palette; however, the Type 2 does not, and the actual color value addressed cannot be ensured.

Index The Index field (bits 4 – 0) contains the index to the data registers in the attribute controller.

Internal Palette Registers 0 through F

These registers are at indexes hex 00 through 0F. Their write address is hex 03C0; their read address is hex 03C1.

7	6	5	4	3	2	1	0
–	–	P5	P4	P3	P2	P1	P0

– : Set to 0, Undefined on Read
P5 to P0 : Palette Data

Figure 2-80. Internal Palette Registers, Index Hex 00 - 0F

The register fields are defined as follows:

P5 – P0 These 6-bit registers (bits 5 – 0) allow a dynamic mapping between the text attribute or graphic color input value and the display color on the CRT screen. When set to 1, this bit selects the appropriate color. The Internal Palette registers should be modified only during the vertical retrace interval to avoid problems with the displayed image. These internal palette values are sent off-chip to the video DAC, where they serve as addresses into the DAC registers. (Also see the attribute controller block diagram on page 2-10.)

Note: These registers can be accessed only when bit 5 in the Address register is set to 0. When the bit is 1, writes are "don't care" and reads return undefined data.

Attribute Mode Control Register

This read/write register is at index hex 10. Its write address is hex 03C0; its read address is hex 03C1.

7	6	5	4	3	2	1	0
PS	PW	PP	–	EB	ELG	ME	G

```
  – : Set to 0, Undefined on Read
 PS : P5, P4 Select
 PW : Pel Width
 PP : Pel Panning Compatibility
 EB : Enable Blink/Select Background Intensity
ELG : Enable Line Graphics Character Code
 ME : Mono Emulation
  G : Graphics/Alphanumeric Mode
```

Figure 2-81. Attribute Mode Control Register, Index Hex 10

The register fields are defined as follows:

PS The P5, P4 Select field (bit 7) selects the source for the P5 and P4 video bits that act as inputs to the video DAC. When set to 0, P5 and P4 are the outputs of the Internal Palette registers. When set to 1, P5 and P4 are bits 1 and 0 of the Color Select register. For more information, see "VGA Programming Considerations" on page 2-94.

PW When the Pel Width field (bit 6) is set to 1, the video data is sampled so that 8 bits are available to select a color in the 256-color mode (hex 13). This bit is set to 0 in all other modes.

PP When the Pel Panning Compatibility field (bit 5) is set to 1, a successful line-compare in the CRT controller forces the output of the Pel Panning register to 0 until a vertical synchronization occurs, at which time the output returns to its programmed value. This bit allows a selected portion of a screen to be panned.

When set to 0, line compare has no effect on the output of the Pel Panning register.

EB When the Enable Blink/Select Background Intensity field (bit 3) is set to 0, the most-significant bit of the attribute selects the background intensity (allows 16 colors for background). When set to 1, this bit enables blinking.

ELG When the Enable Line Graphics Character Code field (bit 2) is set to 0, the ninth dot will be the same as the background. When set to 1, this bit enables the special line-graphics character codes for the monochrome emulation mode. This emulation mode forces the ninth dot of a line graphic character to be identical to the eighth dot of the character. The line-graphics character codes for the monochrome emulation mode are hex C0 through hex DF.

For character fonts that do not use these line-graphics character codes, bit 2 should be set to 0 to prevent unwanted video information from being displayed on the CRT screen.

BIOS will set this bit, the correct dot clock, and other registers when the 9-dot alphanumeric mode is selected.

ME When the Mono Emulation field (bit 1) is set to 1, monochrome emulation mode is selected. When set to 0, color emulation mode is selected.

G When the Graphics/Alphanumeric Mode field (bit 0) is set to 1, the graphics mode of operation is selected.

Overscan Color Register

This read/write register is at index hex 11. Its write address is hex 03C0; its read address is hex 03C1. This register determines the border (overscan) color.

7	6	5	4	3	2	1	0
Overscan Color							

Figure 2-82. Overscan Color Register, Index Hex 11

The Overscan Color register (bits 7 − 0) selects the border color used in the 80-column alphanumeric modes and in the graphics modes other than modes 4, 5, and D.

Color Plane Enable Register

This read/write register is at index hex 12. Its write address is hex 03C0; its read address is hex 03C1.

7	6	5	4	3	2	1	0
−	−	−	−	ECP			

```
  − : Set to 0, Undefined on Read
ECP : Enable Color Plane
```

Figure 2-83. Color Plane Enable Register, Index Hex 12

The register field is defined as follows:

ECP Setting a bit in the Enable Color Plane field (bits 3 − 0) to 1 enables the corresponding display-memory color plane.

Horizontal Pel Panning Register

This read/write register is at index hex 13. Its write address is hex 03C0; its read address is hex 03C1.

7	6	5	4	3	2	1	0
–	–	–	–	HPP			

```
  - : Set to 0, Undefined on Read
HPP : Horizontal Pel Panning
```

Figure 2-84. Horizontal Pel Panning Register, Index Hex 13

The register field is defined as follows:

HPP The Horizontal Pel Panning field (bits 3–0) selects the number of pels that the video data is shifted to the left. Pel panning is available in both alphanumeric (A/N) and graphics modes. The following figure shows the number of pels shifted for each mode.

Register Value	Number of Pels Shifted to the Left		
	Mode Hex 13	A/N Modes *	All Other Modes
0	0	1	0
1	–	2	1
2	1	3	2
3	–	4	3
4	2	5	4
5	–	6	5
6	3	7	6
7	–	8	7
8	–	0	–

* Only mode 7 and the A/N modes with 400 scan lines.

Figure 2-85. Image Shifting

Color Select Register

This read/write register is at index hex 14. Its write address is hex 03C0; its read address is hex 03C1.

7	6	5	4	3	2	1	0
–	–	–	–	SC7	SC6	SC5	SC4

```
  – : Set to 0, Undefined on Read
SC7 : S_color 7
SC6 : S_color 6
SC5 : S_color 5
SC4 : S_color 4
```

Figure 2-86. Color Select Register, Index Hex 14

The register fields are defined as follows:

SC7, SC6 In modes other than mode hex 13, the S_color 7 and S_color 6 fields (bits 3, 2) are the 2 most-significant bits of the 8-bit digital color value to the video DAC. In mode hex 13, the 8-bit attribute is the digital color value to the video DAC. These bits are used to switch rapidly between sets of colors in the video DAC. (For more information, see “VGA Programming Considerations” on page 2-94.)

SC5, SC4 The S_color 5 and S_color 4 fields (bits 1, 0) can be used in place of the P4 and P5 bits from the Internal Palette registers to form the 8-bit digital color value to the video DAC. Selecting these bits is done in the Attribute Mode Control register (index hex 10). These bits are used to switch rapidly between color sets within the video DAC.

VGA Programming Considerations

The following are some programming considerations for the VGA:

- The following rules must be followed to guarantee the critical timings necessary to ensure the proper operation of the CRT controller:
  - The value in the Horizontal Total register must be at least hex 19.
  - The minimum positive pulse width of the 'horizontal synchronization' signal must be 4 character clock units.
  - The End Horizontal Retrace register must be programmed such that the 'horizontal synchronization' signal goes to 0 at least 1 character clock time before the 'horizontal display enable' signal goes active.
  - The End Vertical Blanking register must be set to a value at least one horizontal scan line greater than the line-compare value.
- When pel panning compatibility is enabled in the Attribute Mode Control register, a successful line compare in the CRT controller forces the output of the Horizontal Pel Panning register to 0's until a vertical synchronization occurs. When the vertical synchronization occurs, the output returns to the programmed value. This allows the portion of the screen indicated by the Line Compare register to be operated on by the Horizontal Pel Panning register.
- A write to the Character Map Select register becomes valid on the next whole character line. This prevents deformed character images when changing character generators in the middle of a character scan line.
- For mode hex 13, the attribute controller is configured so that the 8-bit attribute in video memory becomes the 8-bit address (P0 through P7) into the video DAC. The user should not modify the contents of the Internal Palette registers when using this mode.
- To achieve smooth scrolling, see "Smooth Scrolling of VGA and 132-Column Text Modes" on page 3-222.

- The following is the sequence for accessing the attribute data registers:
  1. Disable interrupts.
  2. Reset the flip-flop for the Attribute Address register.
  3. Write the index.
  4. Access the data register.
  5. Enable interrupts.
- The Color Select register in the attribute controller section allows rapid switching between color sets in the video DAC. Bit 7 of the Attribute Mode Control register controls the number of bits in the Color Select register used to address the color information in the video DAC (either 2 or 4 bits are used). By changing the value in the Color Select register, an application can switch color sets in graphics and alphanumeric modes. (Mode hex 13 does not use this feature.)

  Note: For multiple color sets, the user must load the color values.
- An application that saves the video state must store the 4 bytes of information contained in the system microprocessor latches in the graphics controller subsection. These latches are loaded with 32 bits from video memory (8 bits per map) each time the system reads from video memory. The application must:
  1. Use write mode 1 to write the values in the latches to a location in video memory that is not part of the display buffer, such as the last location in the address range.
  2. Save the values of the latches by reading them back from video memory.

     Note: If memory addressing is in the chain-4 or odd/even mode, reconfigure the memory as four sequential maps prior to performing the sequence above.

BIOS provides support for completely saving and restoring the video state. Refer to the *IBM Personal System/2 and Personal Computer BIOS Interface Technical Reference* for more information.

- The Horizontal Pel Panning register allows programs to control the starting position of the display area on the screen. The display area can be shifted to the left up to eight pel positions. In single-byte shift modes, to pan to more than eight pel positions, the CRT controller start address is incremented and the Horizontal Pel Panning register is reset to 0.

  In multiple shift modes, the byte-panning bits (in the Preset Row Scan register) are used as extensions to the Horizontal Pel Panning register. This allows panning across the width of the video output. For example, in the 32-bit shift mode, the byte pan and pel-panning bits provide panning up to 31 bits. To pan from position 31 to 32, the CRT controller start address is incremented and the panning bits, both pel and byte, are reset to 0.

  Further panning can be accomplished by changing the start-address value in the CRT controller registers, Start Address High and Start Address Low. The sequence is:

  1. Use the Horizontal Pel Panning register to shift the maximum number of bits to the left.
  2. Increment the start address.
  3. Set the Horizontal Pel Panning register so that no bits are shifted.

     The screen is shifted one pel to the left of the position it was in at the end of Step 1. Repeat Step 1 through Step 3 as often as necessary.

- When operating in a mode with 200 scan lines, and using a split-screen application that scrolls a second screen on top of the first screen, the Line Compare register (CRT Controller register hex 19) must contain an even value. This is a requirement of the double scanning logic in the CRT controller.
- If the value in the Cursor Start register (CRT Controller register hex 0A) is greater than that in the Cursor End register (CRT Controller register hex 0B), the cursor is not displayed.
- In 8-dot character modes, the underline attribute produces a solid line across adjacent characters. In 9-dot character modes, the underline across adjacent characters is dashed. In 9-dot modes with the line-graphics characters (C0 through DF character codes), the underline is solid.

Programming the Registers

Each of the video components has an address register and a number of data registers. The data registers have addresses common to all registers for that component. The individual registers are selected by a pointer (index) in their Address register. To write to a data register, the Address register is loaded with the index of the desired data register, then the data register is loaded by writing to the common I/O address.

The general registers do not share a common address; they each have their own I/O address.

See "Video DAC to System Interface" on page 2-101 for details on programming the video DAC.

For compatibility with the IBM Enhanced Graphics Adapter (EGA), the internal video subsystem palette is programmed the same as the EGA. Using BIOS to program the palette produces a color compatible to that produced by the EGA. Mode hex 13 (256 colors) is programmed so that the first 16 locations in the DAC produce compatible colors.

When BIOS is used to load the color palette for a color mode and a monochrome display is attached, the color palette is changed. The colors are summed to produce shades of gray that allow color applications to produce a readable screen.

Modifying the following bits must be done while the sequencer is held in a synchronous reset through its Reset register. The bits are:

- Bits 3 and 0 of the Clocking Mode register
- Bits 3 and 2 of the Miscellaneous Output register.

RAM Loadable Character Generator

The character generator is RAM loadable and can support characters up to 32 scan lines high. Three character fonts are stored in BIOS, and one is automatically loaded when an alphanumeric mode is selected. The Character Map Select register can be programmed to redefine the function of bit 3 of the attribute byte to be a character-font switch. This allows the user to make a selection between any two character sets residing in map 2, and gives the user access to 512 characters instead of 256. Character fonts can be loaded off line, and up to eight fonts can be loaded at any one time.

The structure of the character fonts is described in the following figure. The character generator is in map 2 and must be protected using the map mask function.

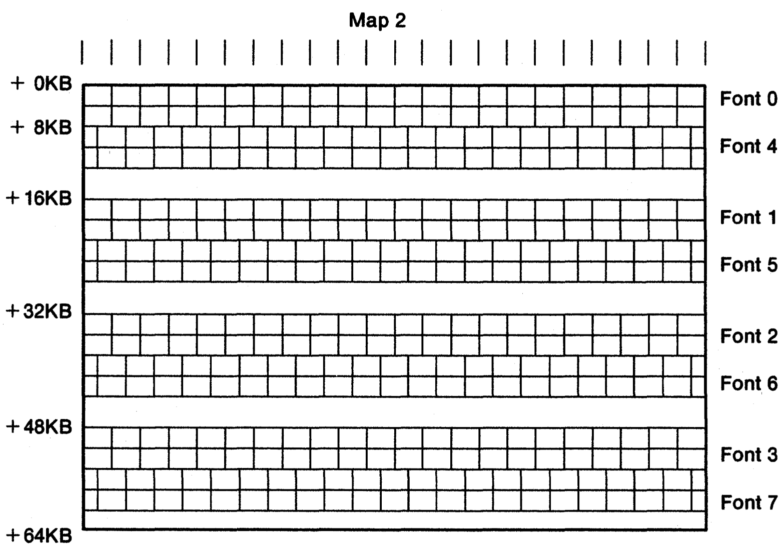


Figure 2-87. Character Table Structure

The following figure illustrates the structure of each character pattern. If the CRT controller is programmed to generate 16 row scans, then 16 bytes must be filled in for each character in the font. The example below assumes eight row scans per character.

Address	Byte Image								Data
CC * 32 + 0				X	X				18H
1			X	X	X	X			3CH
2		X	X			X	X		66H
3		X	X			X	X		66H
4		X	X	X	X	X	X		7EH
5		X	X			X	X		66H
6		X	X			X	X		66H
7		X	X			X	X		66H

*CC equals the value of the character code. For example, hex 41 equals and ASCII "A".

Figure 2-88. Character Pattern Example

Creating a Split Screen

The VGA hardware supports a split screen. The top portion of the screen is designated as screen A, and the bottom portion is designated as screen B, as shown in the following figure.

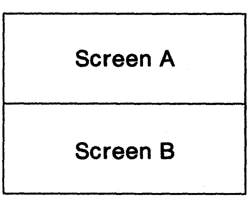


Figure 2-89. Split Screen Definition

The following figure shows the screen mapping for a system containing a 32KB alphanumeric storage buffer, such as the VGA. Information displayed on screen A is defined by the Start Address High and Low registers of the CRT controller. Information displayed on screen B always begins at video address hex 0000.

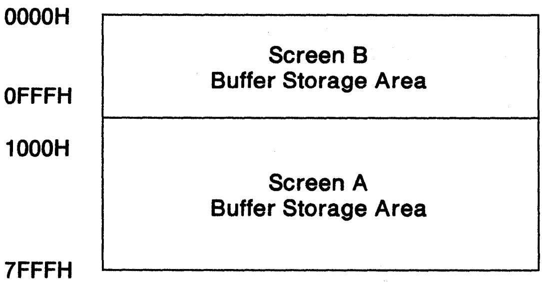


Figure 2-90. Screen Mapping within the Display Buffer Address Space

The Line Compare register of the CRT controller performs the split screen function. The CRT controller has an internal horizontal scan line counter and logic that compares the counter value to the value in the Line Compare register and clears the memory address generator when a comparison occurs. The linear address generator then sequentially addresses the display buffer starting at location 0. Each subsequent row address is determined by the 16-bit addition of the start-of-line latch and the Offset register.

Screen B can be smoothly scrolled onto the display by updating the Line Compare register in synchronization with the 'vertical retrace' signal. Screen B information is not affected by scrolling operations that use the Start Address registers to scroll through screen A information.

When pel-panning compatibility is enabled (Attribute Mode Control register), a successful line comparison forces the output of the Horizontal Pel Panning register to 0's until vertical synchronization occurs. This feature allows the information on screen B to remain unaffected by pel-panning operations on screen A.

Video Digital-to-Analog Converter

The video digital-to-analog converter (DAC) integrates the function of a color palette with three internal DACs for driving an analog display.

The DAC has 256 registers containing 18 bits each to allow the display of up to 256 colors from a possible 256K colors. Each output signal is driven by a 6-bit DAC.

Register Name	Read/ Write	Address (Hex)
Palette Address (Write Mode)	R/W	03C8
Palette Address (Read Mode)	W	03C7
DAC State	R	03C7
Palette Data	R/W	03C9
Pel Mask	R	03C6

Figure 2-91. Video DAC Register

Device Operation

The palette address (P7 through P0) and the blanking input are sampled on the rising edge of the pel clock. After three more pel clock cycles, the video reflects the state of these inputs.

During normal operation, the palette address is used as a pointer to one of the 256 data registers in the palette. The value in each data register is converted to an analog signal for each of the three outputs (red, green, blue). The blanking input is used to force the video output to 0 volts. The blanking operation is independent of the palette operation.

Each data register is 18 bits wide: 6 bits each for red, green, and blue. The data registers are accessible through the system interface.

Video DAC to System Interface

The Palette Address register holds an 8-bit value that is used to address a location within the video DAC. The Palette Address register responds to two addresses; the address depends on the type of palette access, read or write. Once the address is loaded, successive accesses to the data register automatically increment the address register.

For palette write operations, the address for the Palette Address register is hex 03C8. A write cycle consists of writing three successive bytes to the Data register at address hex 03C9. The 6 least-significant bits of each byte are concatenated to form the 18-bit palette data. The order is red value first, then green, then blue.

For palette read operations, the address for the Palette Address register is hex 03C7 (in the read mode, the Palette Address register is write only). A read cycle consists of reading three successive bytes from the Data register at address hex 03C9. The 6 least-significant bits of each byte contain the corresponding color value. The order is red value first, then green, then blue.

If the Palette Address register is written to during a read or write cycle, a new cycle is initialized and the unfinished cycle is terminated. The effects of writing to the Data register during a read cycle or reading from the Data register during a write cycle are undefined and can change the palette contents.

The DAC State register is a read-only register at address hex 03C7. Bits 1 and 0 return the last active operation to the DAC. If the last operation was a read operation, both bits are set to 1. If the last operation was a write, both bits are set to 0.

Reading the Read Palette Address register at hex 03C8 or the DAC State register at hex 03C7 does not interfere with read or write cycles.

Programming Considerations

- As explained in "Video DAC to System Interface" on page 2-101, the effects of writing to the Data register during a read cycle or reading from the Data register during a write cycle are undefined and can change the palette contents. Therefore, the following sequence must be followed to ensure the integrity of the color palette during accesses to it:

  1. Disable interrupts.
  2. Write the address to Pel Address register.
  3. Write or read three bytes of data.
  4. Go to Step 2, repeat for the desired number of locations.
  5. Enable interrupts.

  Note: All accesses to the DAC registers are byte-wide I/O operations.

- To prevent "snow" on the screen, an application reading data from or writing data to the DAC registers should ensure that the blank input to the DAC is asserted. This can be accomplished either by restricting data transfers to retrace intervals (use Input Status Register 1 to determine when retrace is occurring) or by using the screen off bit located in the Clocking Mode register in the sequencer.

  Note: BIOS provides read and write interfaces to the video DAC.

- Do not write to the Pel Mask register (hex 03C6). Palette information can be changed as a result. This register is correctly initialized to hex FF during a mode set.

VGA Video Extensions

The video extensions provide a means of transferring video information between the base video subsystem and an auxiliary video adapter.

The video extensions consist of:

- The auxiliary video extension
- The base video extension
- The auxiliary video signals

The base video is provided by the video subsystem integrated onto the system board, or, when not provided on the system board, by a suitable video adapter. Such an adapter can provide a Micro Channel connector with the base video extension. Video adapters supporting the base video extension must provide the VGA function as the default. For detailed connector dimensions, see "Micro Channel Adapter Design" in *Personal System/2 Hardware Interface Technical Reference - Architectures*.

The buffers for the base video can be turned off to allow video output from the auxiliary video to be sent through the base video DAC to the display. The video extension can be driven in only one direction at a time.

Note: The video extension is only available for use while the video subsystem is in the VGA mode and operating VGA Mode Display Timing Set 1. See "VGA Mode Display Timing" on page 4-5.

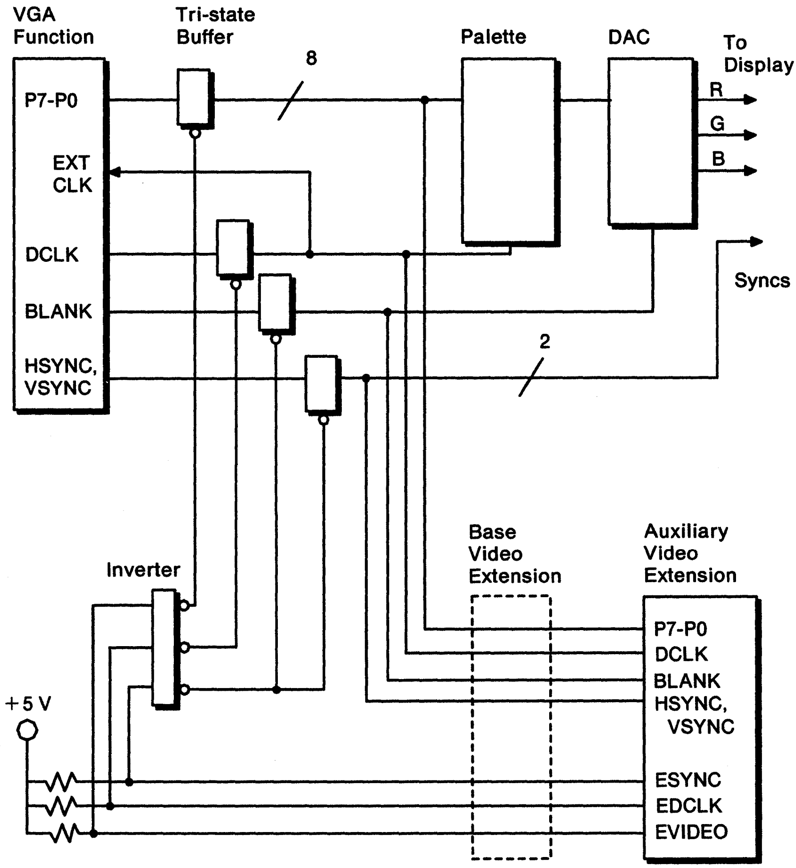


Figure 2-92. Auxiliary Video Connector Interface

Auxiliary Video Extension

This extension provides a video adapter with the ability to access the resources of the base video subsystem.

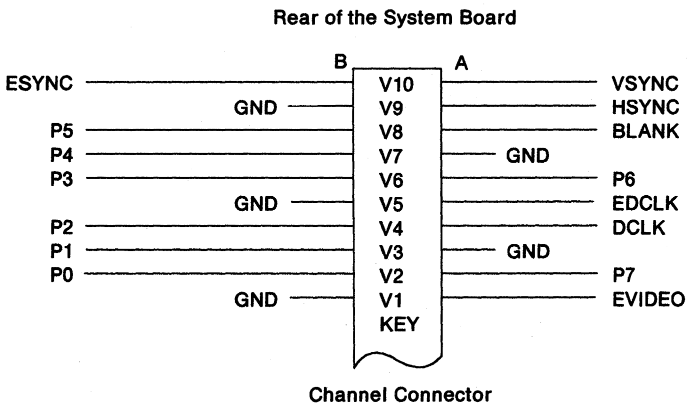


Figure 2-93. Video Extension

The auxiliary video extension is an optional part of the 16- or 32-bit Micro Channel connector.

Note: For more information on the auxiliary and base video connectors and extensions, see Micro Channel Architectures in *Personal System/2 Hardware Interface Technical Reference - Architectures.*

Base Video Extension

This extension is for adapters that provide the base video subsystem. Only systems without a base video subsystem on the system board have a connector with this extension. The base video extension signals and auxiliary video extension signals are identical.

Video adapters supporting the base video extension must provide the VGA function as the default.

The base video extension is an optional part of the 16- or 32-bit Micro Channel connector and is positioned at the end of the matched memory extension.

Video Extension Signal Descriptions

The following are signal descriptions for the auxiliary and base video extensions of the channel connector.

VSYNC: Vertical Synchronization: This signal is the vertical synchronization signal to the display. Also see the ESYNC description.

HSYNC: Horizontal Synchronization: This signal is the horizontal synchronization signal to the display. Also see the ESYNC description.

BLANK: Blanking Signal: This signal is connected to the BLANK input of the video DAC. When active (0 V dc), this signal tells the DAC to drive its analog color outputs to 0 V dc. Also see the ESYNC description.

P7−P0: Palette Bits: These eight signals contain video information and comprise the pel address inputs to the video DAC. See also the EVIDEO description.

DCLK: Dot Clock: This signal is the pel clock used by the DAC to latch the digital video signals, P7 through P0. The signals are latched into the DAC on the rising edge of DCLK.

This signal is driven through the EXTCLK input to the VGA when DCLK is driven by the adapter. If an adapter is providing the clock, it must also provide the video data to the DAC. Also see the EDCLK description.

ESYNC: External Synchronization: This signal is the output-enable signal for the buffer that drives BLANK, VSYNC, and HSYNC. ESYNC is tied to +5 V dc through a pull-up resistor. When ESYNC is high, the VGA drives BLANK, VSYNC, and HSYNC. When ESYNC is pulled low, the adapter drives BLANK, VSYNC, and HSYNC.

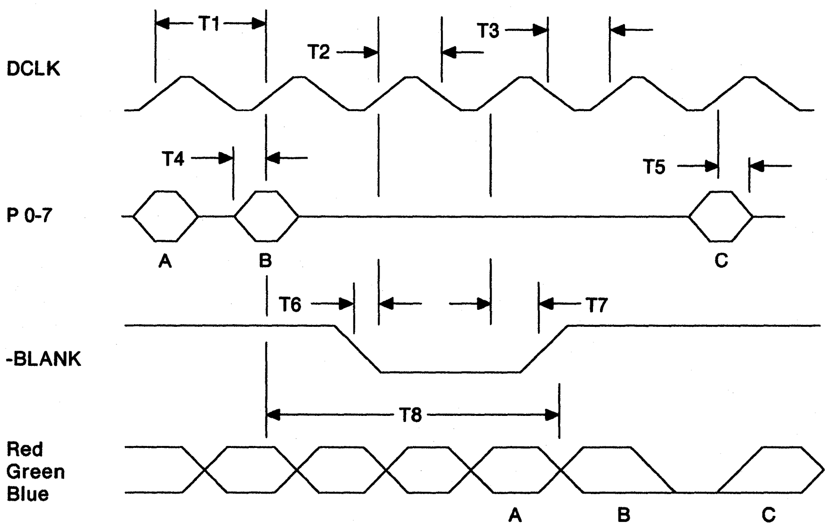
EVIDEO: External Video: This signal is the output-enable signal for the buffer that drives P7 through P0. EVIDEO is tied to +5 V dc through a pull-up resistor. When EVIDEO is high, the VGA drives P7 through P0. When it is pulled low, the adapter drives P7 through P0.

EDCLK: External Dot Clock: This signal is the output-enable signal for the buffer that drives DCLK. EDCLK is tied to +5 V dc through a pull-up resistor.

When EDCLK is high, the VGA is the source of DCLK to the DAC and the adapter. The Miscellaneous Output register should not select clock source 2 (010 binary) when EDCLK is high.

When EDCLK is pulled low, the adapter drives DCLK. If the adapter is driving the clock, it must also provide the video data to the DAC, and the Miscellaneous Output register must select clock source 2 (010 binary).

Video Extension Signal Timing



Symbol	Description	Minimum (ns)	Maximum (ns)
T1	Pel Clock Period	28	10,000
T2	Clock Pulse Width High	7	10,000
T3	Clock Pulse Width Low	9	10,000
T4	Pel Set-Up Time	4	–
T5	Pel Hold Time	4	–
T6	Blank Set-Up Time	4	–
T7	Blank Hold Time	4	–
T8	Analog Output Delay	3(T1) + 5	7.5(T1) + 6

Figure 2-94. Video Extension Signal Timing (DAC Signals)

Section 3. XGA Function

XGA Function Introduction 3-7
  XGA Components 3-7
    System Bus Interface 3-9
    Memory and CRT Controller 3-9
    Coprocessor 3-9
    Video Memory 3-10
    Attribute Controller 3-10
    Sprite Controller 3-10
    The Serializer, Palette, and Video DAC 3-10
    Alphanumeric (A/N) Font and Sprite Buffer 3-10
  Compatibility 3-11
    8514/A Adapter Interface 3-11
    LIM EMS Drivers 3-13
    XGA Applications (Written to the Hardware Interface) . . . 3-13
VGA Compatibility 3-14
132-Column Text Mode 3-14
  Mainframe Interactive (MFI) Support 3-15
Extended Graphics Mode 3-18
  Display Controller 3-18
    Video Memory Format 3-18
    Pel Color Mapping 3-21
    Border Color Mapping 3-21
    Direct Access to Video Memory 3-21
  CRT Controller 3-22
    CRT Controller Register Interpretations 3-23
    Scrolling 3-24
  Sprite 3-25
    Sprite Color Mapping 3-25
    Sprite Buffer Accesses 3-26
    Sprite Positioning 3-27
  Palette 3-28
    Palette Accesses 3-28
  Direct Color Mode 3-30
    Coprocessor Functions 3-32
XGA Display Controller Registers 3-33
  Register Usage Guidelines 3-34
  Direct Access I/O Registers 3-35
    Operating Mode Register (Address 21x0) 3-35
    Aperture Control Register (Address 21x1) 3-37
    Interrupt Enable Register (Address 21x4) 3-38
    Interrupt Status Register (Address 21x5) 3-40
    Virtual Memory Control Register (Address 21x6) 3-41

Virtual Memory Interrupt Status Register (Address 21x7) . 3-41
Aperture Index Register (Address 21x8) 3-42
Memory Access Mode Register (Address 21x9) 3-43
Index Register (Address 21xA) 3-44
Data Registers (Addresses 21xB to 21xF) 3-47
Indexed Access I/O Registers 3-48
Auto-Configuration Register (Index 04) 3-48
Coprocessor Save/Restore Data Registers (Index 0C and 0D) 3-49
Horizontal Total Registers (Index 10 and 11) 3-50
Horizontal Display End Registers (Index 12 and 13) 3-51
Horizontal Blanking Start Registers (Index 14 and 15) . . . 3-52
Horizontal Blanking End Registers (Index 16 and 17) 3-53
Horizontal Sync Pulse Start Registers (Index 18 and 19) . 3-54
Horizontal Sync Pulse End Registers (Index 1A and 1B) . . 3-55
Horizontal Sync Pulse Position Registers (Index 1C and 1E) 3-56
Vertical Total Registers (Index 20 and 21) 3-57
Vertical Display End Registers (Index 22 and 23) 3-58
Vertical Blanking Start Registers (Index 24 and 25) 3-59
Vertical Blanking End Registers (Index 26 and 27) 3-60
Vertical Sync Pulse Start Registers (Index 28 and 29) . . . 3-61
Vertical Sync Pulse End Register (Index 2A) 3-62
Vertical Line Compare Registers (Index 2C and 2D) 3-63
Sprite Horizontal Start Registers (Index 30 and 31) 3-64
Sprite Horizontal Preset Register (Index 32) 3-65
Sprite Vertical Start Registers (Index 33 and 34) 3-66
Sprite Vertical Preset Register (Index 35) 3-67
Sprite Control Register (Index 36) 3-68
Sprite Color Registers (Index 38 – 3D) 3-69
Display Pel Map Offset Registers (Index 40 – 42) 3-70
Display Pel Map Width Registers (Index 43 and 44) 3-71
Display Control 1 Register (Index 50) 3-72
Display Control 2 Register (Index 51) 3-74
Display ID and Comparator Register (Index 52) 3-76
Clock Frequency Select 1 Register (Index 54) 3-77
Border Color Register (Index 55) 3-77
Programmable Pel Clock Register (Index 58) 3-78
Direct Color Control Register (Index 59) 3-79
Sprite/Palette Index Registers (Index 60 and 61) 3-80
Sprite/Palette Prefetch Index Registers (Index 62 and 63) 3-81
Palette Mask Register (Index 64) 3-82
Palette Data Register (Index 65) 3-82
Palette Sequence Register (Bits 2 – 0 only) (Index 66) . . . 3-83
Palette Red Prefetch Register (Index 67) 3-84
Palette Green Prefetch Register (Index 68) 3-84
Palette Blue Prefetch Register (Index 69) 3-84

Sprite Data Register (Index 6A) 3-85
Sprite Prefetch Register (Index 6B) 3-85
Miscellaneous Control Register (Index 6C) 3-85
MFI Control Register (Index 6D) 3-86
Clock Frequency Select 2 Register (Index 70) 3-88
Coprocessor Description 3-90
Programmer's View 3-93
Pel Formats 3-93
Pel Fixed and Variable Data 3-94
The Coprocessor View of Memory 3-94
Pel Maps 3-95
Pel Maps A, B, and C (General Maps) 3-95
Pel Map M (Mask Map) 3-96
Map Origin 3-96
X and Y Pointers 3-97
Scissoring with the Mask Map 3-100
Drawing Operations 3-105
Draw and Step 3-105
Line Draw 3-109
Pel Block Transfer (PxBlt) 3-113
Area Fill 3-117
Logical and Arithmetic Functions 3-119
Mixes 3-120
Breaking the Coprocessor Carry Chain 3-121
Generating the Pattern from the Source 3-122
Color Expansion 3-122
Pel Bit Masking 3-123
Color Compare 3-123
Controlling Coprocessor Operations 3-124
Starting a Coprocessor Operation 3-124
Suspending a Coprocessor Operation 3-124
Terminating a Coprocessor Operation 3-124
Coprocessor Operation Completion 3-125
Coprocessor-Operation-Complete Interrupt 3-125
Coprocessor Busy Bit 3-125
Accesses to the Coprocessor During an Operation 3-126
Coprocessor State Save/Restore 3-126
Suspending Coprocessor Operations 3-126
Save/Restore Mechanism 3-127
Coprocessor Registers 3-128
Register Usage Guidelines 3-132
Virtual Memory Registers 3-132
State Save/Restore Registers 3-133
Auxiliary Coprocessor Status Register (Offset 09) 3-133
Coprocessor Control Register (Offset 11) 3-134
State Length Registers (Offset C and D) 3-136

Save/Restore Data Ports Register (I/O Index C and D) . . 3-136
Pel Interface Registers 3-137
Pel Map Index Register (Offset 12) 3-137
Pel Map n Base Pointer Register (Offset 14) 3-138
Pel Map n Width Register (Offset 18) 3-139
Pel Map n Height Register (Offset 1A) 3-140
Pel Map n Format Register (Offset 1C) 3-141
Pel Maps A, B, and C 3-142
Mask Map 3-142
Bresenham Error Term E Register (Offset 20) 3-142
Bresenham Constant K1 Register (Offset 24) 3-143
Bresenham Constant K2 Register (Offset 28) 3-143
Direction Steps Register (Offset 2C) 3-144
Foreground Mix Register (Offset 48) 3-145
Background Mix Register (Offset 49) 3-145
Destination Color Compare Condition Register (Offset 4A) 3-146
Destination Color Compare Value Register (Offset 4C) . 3-147
Pel Bit Mask (Plane Mask) Register (Offset 50) 3-148
Carry Chain Mask Register (Offset 54) 3-149
Foreground Color Register (Offset 58) 3-150
Background Color Register (Offset 5C) 3-150
Operation Dimension 1 Register (Offset 60) 3-151
Operation Dimension 2 Register (Offset 62) 3-151
Mask Map Origin X Offset Register (Offset 6C) 3-152
Mask Map Origin Y Offset Register (Offset 6E) 3-152
Source X Address Register (Offset 70) 3-153
Source Y Address Register (Offset 72) 3-153
Pattern X Address Register (Offset 74) 3-154
Pattern Y Address Register (Offset 76) 3-154
Destination X Address Register (Offset 78) 3-155
Destination Y Address Register (Offset 7A) 3-155
Pel Operations Register (Offset 7C) 3-156
XGA System Interface 3-163
Multiple Instances 3-163
Multiple XGA Subsystems in VGA Mode 3-163
Multiple XGA Subsystems in 132-Column Text Mode . . . 3-163
Multiple XGA Subsystems in Extended Graphics Mode . 3-163
XGA POS Registers 3-164
Register Usage Guidelines 3-164
Subsystem Identification Low Byte Register (Base + 0) . 3-164
Subsystem Identification High Byte Register (Base + 1) 3-165
POS Register 2 (Base + 2) 3-165
POS Register 4 (Base + 4) 3-168
POS Register 5 (Base + 5) 3-169
Virtual Memory Description 3-170
Address Translation 3-170

Page Directory and Page Table Entries 3-172
The XGA Implementation of Virtual Memory 3-174
The Translate Look-aside Buffer 3-174
TLB Misses 3-175
System Coherency 3-176
VM Page-Not-Present Interrupts 3-177
VM Protection-Violation Interrupts 3-177
The XGA in Segmented Systems 3-178
Virtual Memory Registers 3-179
Page Directory Base Address Register (Coprocessor Registers, Offset 0) 3-179
Current Virtual Address Register (Coprocessor Registers, Offset 4) 3-180
Virtual Memory Control Register (I/O Address 21x6) . . . 3-181
Virtual Memory Interrupt Status Register (I/O Address 21x7) 3-183
XGA Subsystem Identification, Location, and XGA Mode Setting 3-185
XGA Display Mode Query and Set (DMQS) 3-185
DMQS Architecture Overview 3-185
DMQS BIOS Interface 3-188
DMQS Display Information Files 3-191
DMQS Display Information File Structure 3-192
Mode setting from the DMQS Display Information File . . 3-196
DMQS Information File Optional Extensions 3-197
Composite Display ID 3-200
XGA Level Identifier 3-201
DMQS Customized File 3-201
Locating and Initializing the XGA Subsystem Without Using DMQS 3-205
XGA Subsystem Identification 3-206
Location of XGA Subsystem I/O Spaces 3-207
Display Type Detection 3-209
Video Memory Size Determination 3-212
Extended Graphics Modes Available 3-213
Extended Graphics Mode Setting Procedure 3-215
VGA Primary Subsystem Considerations 3-216
Multiple XGA Subsystems 3-220
VGA Modes 3-220
XGA Subsystem Coexistence with VGA 3-220
Switching the XGA Subsystem from XGA to VGA Mode . . . 3-221
Smooth Scrolling of VGA and 132-Column Text Modes . . . 3-222
132-Column Text Mode 3-223
Effects of VGA and XGA Mode Setting on Video Memory 3-226
Programming the XGA Subsystem 3-227
General Systems Considerations 3-227
Coexisting with LIM Expanded Memory Managers 3-227

INT 2Fh, Screen Switch Notification 3-227
PS/2 System Video Memory Apertures 3-228
64KB System Video Memory Aperture 3-229
1MB System Video Memory Aperture 3-230
4MB System Video Memory Aperture 3-231
Video Memory Address Range 3-232
Programming the XGA Subsystem in Extended Graphics Mode 3-233
General Register Usage 3-233
XGA Coprocessor Pel Interface Registers 3-233
Using the Coprocessor to Perform a Pel Blit (PxBlt) . . . 3-237
Using the Coprocessor to Perform a Bresenham Line Draw 3-245
Memory Access Modes 3-255
Motorola and Intel Formats 3-255
Other Programming Considerations 3-256
Waiting for Hardware Not Busy 3-256
Overlapping PxBlts 3-257
Inverting PxBlt 3-257
Area Fill 3-258
Restrictions 3-258
Common Problems 3-259
Performance Tips 3-261
Sprite Handling 3-262
Palette formats 3-263
XGA Subsystem Save, Restore, Suspend, and Resume . 3-263
Alternative XGA Coprocessor Register Set 3-266
System Register Usage 3-266
Direct Color Mode 3-267
Use of DMA Bus Master Functions 3-268
Physical Addressability to System Memory 3-268
Bus Master Memory Address Limitation 3-268
Real-Mode DOS Environments 3-269

XGA Function Introduction

The XGA function is generated by Type 2 and Type 3 video subsystems. There are two types of XGA functions available:

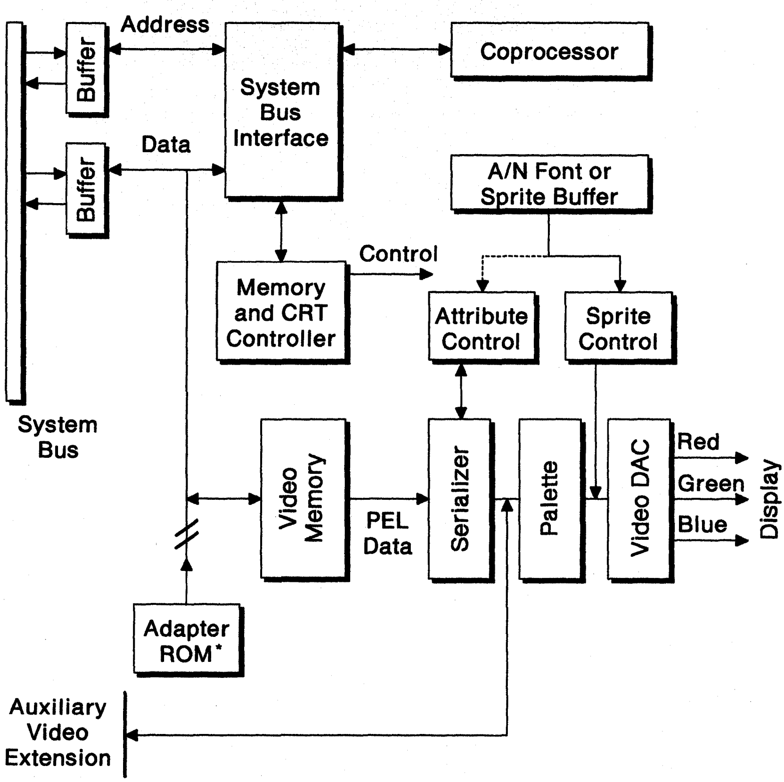
XGA	Type 2 video
XGA-2	Type 3 video

This chapter describes the capabilities and operation of the XGA and the XGA-2 functions.

XGA Components

The XGA video subsystem components include:

- System bus interface
- Memory and CRT controller
- Coprocessor
- Video memory
- Attribute controller
- Sprite controller
- Alphanumeric (A/N) font and sprite buffer
- Serializer
- Palette
- Video digital-to-analog convertor (DAC)



* ROM is only present on adapters.

Figure 3-1. XGA Video Subsystem

System Bus Interface

The system bus interface controls the interface between the video subsystem and the system microprocessor. It decodes the addresses for VGA and XGA I/O registers, the memory addresses for the coprocessor memory-mapped registers, and video memory.

It also provides the bus master function, and determines whether the system data bus is 16 or 32 bits wide.

Memory and CRT Controller

The memory and CRT controller controls access to video memory by the system microprocessor, displays the contents of video memory on the display, and provides support for the VGA and 132-column text modes.

Coprocessor

The coprocessor provides hardware drawing-assist functions. These functions can be performed on graphics data in video memory and system memory.

The coprocessor updates the video memory independently of the system microprocessor. Instructions are written to a set of memory-mapped registers; the coprocessor then executes the drawing function.

The coprocessor functions are:

Pel-Block or Bit-Block Transfers
: Transfers a bit map, or part of a bit map, from one location to another:

- Within video memory
- Within system memory
- Between system and video memory

Line Drawing
: Draws lines, with a programmable style, into a bit map in video memory or system memory.

Area Fill
: Fills an outlined area in video memory or system memory with a programmable pattern.

Logical and Arithmetic Mixing
Provides logical and arithmetic operations for use with data in video memory or system memory.

Map Masking
Controls updates to each pel for all drawing functions.

Scissoring
Provides a rectangular-mask function for use instead of the mask map.

X and Y Axis Addressing
Allows a pel to be specified by its X and Y coordinates within a pel map, instead of by its linear address in memory.

Video Memory

The video subsystem uses a dual-port video memory to store on-screen data, so that video memory can be read serially to display its contents as the data is being updated.

Attribute Controller

The attribute controller works with the memory and CRT controller to control the color selection and character generation in the 132-column text mode and VGA text modes.

Sprite Controller

The sprite controller is used to display and control the position and image of the sprite (cursor). The sprite is not available in 132-column text mode or VGA modes.

The Serializer, Palette, and Video DAC

The serializer takes data from the serial port of video memory in 16- or 32-bit widths (depending on the size of video memory) and converts it to a serial stream of pel data. The pel data addresses a palette location, which contains the color value. The color value is passed to the DAC, which converts the digital information into red, green, and blue analog signals for the display.

Alphanumeric (A/N) Font and Sprite Buffer

This buffer holds the character fonts in 132-column text mode and VGA modes. It also stores the sprite image in Extended Graphics mode.

Compatibility

8514/A Adapter Interface

The XGA function is *not* hardware register compatible with the 8514/A Adapter Interface. Applications written directly to the register-level interface of the 8514/A Adapter Interface do not run.

The XGA function is 8514/A Adapter Interface compatible in the DOS environment through a DOS Adapter Interface driver supplied with the XGA video subsystem.

Applications written to the 8514/A DOS Adapter Interface should run unchanged with the XGA Adapter Interface. The following differences, however, should be noted:

OS/2* protect mode adapter interface
An XGA Adapter Interface driver is not available for the OS/2 protect mode.

640 x 480, 4 + 4 mode with 512KB display buffer
This is not an Extended Graphics mode, but applications using this mode and written to the rules for the 8514/A Adapter Interface will run.

Dual-display buffer applications
8514/A applications using VGA or other advanced function modes that rely on two separate video display buffers do not run on a single-display configuration. These applications run correctly with two video subsystems (when one is an XGA), and each has a display attached.

Nondisplay memory
The XGA and 8514/A nondisplay (off-screen) memory are mapped differently. Applications using areas of the off-screen memory for storage might not run.

Adapter interface code size
The XGA Adapter Interface code size is larger than that for the 8514/A. This reduces the amount of system memory available to applications.

* Trademark of the IBM Corporation

Adapter interface enhancements

The XGA Adapter Interface is a superset of that provided with the 8514/A. Any 8514/A applications using invalid specifications of parameter blocks might trigger some of the additional functions provided by the XGA Adapter Interface.

Use of LIM EMS drivers

Applications written to the 8514/A Adapter Interface that locate resources, such as bit maps or font definitions, in LIM EMS memory, and pass addresses of these resources to the adapter interface, require a LIM driver that has implemented the Physical Address Services Interface for bus masters.

Time-dependent applications

Some XGA and 8514/A functions run at different speeds. Applications that rely on a fixed performance might be affected by these differences.

XGA Adapter Interface directory and module name

The directory and module name of the XGA Adapter Interface \XGAPCDOS\XGAAIDOS.SYS is different from that of the 8514/A \HDIPCDOS\HDILOAD.EXE. Applications written to rely on the existence of either the specific 8514/A module name or directory do not run on the XGA Adapter Interface.

8514/A and XGA Adapter Interface code type
The XGA Adapter Interface is implemented as a .SYS device driver. The 8514/A Adapter Interface is implemented as a terminate and stay resident program. Applications written to rely on the adapter interface as a terminate and reside program do not run on the XGA Adapter Interface.

LIM EMS Drivers

The XGA coprocessor memory-mapped registers are located in system memory address space. They reside in the top 1KB of an 8KB block of memory assigned to the XGA subsystem. The lower 7KB of this block is used to address the ROM of an XGA subsystem on an adapter card.

Although an XGA subsystem integrated on the system board does not have a subsystem ROM, an 8KB block of memory is allocated to it to support the coprocessor memory-mapped registers. While the lower 7KB of this 8KB block does not contain any memory, the memory-mapped registers are accessed in the top 1KB of the block.

Applications or drivers, such as LIM EMS drivers that scan memory addresses looking for RAM or ROM signatures, might assume incorrectly that all 8KB of memory is available for use.

The location of the 8KB block of memory assigned to the XGA subsystem can be determined by using the System Unit Reference diskette. See the LIM driver installation instructions for details on how to avoid address conflicts.

XGA Applications (Written to the Hardware Interface)

If an XGA application is dependent on specific monitor IDs or characteristics, it might not function on an XGA-2 subsystem. Applications written using Display Mode Query and Set (DMQS) will be insulated from the differences in the displays. Applications not written using DMQS should use only the displays shown in Figure 3-195 on page 3-213.

VGA Compatibility

The XGA subsystem is register compatible with the VGA, as defined in the VGA function description (see "Effects of VGA and XGA Mode Setting on Video Memory" on page 3-226 for switching between the different XGA subsystem modes).

In addition to normal VGA text mode character attributes, the XGA-2 subsystem has hardware support for mainframe interactive (MFI) character attributes. See "Mainframe Interactive (MFI) Support" on page 3-15.

132-Column Text Mode

In this mode, the XGA subsystem is capable of displaying 132 alphanumeric characters on the display. See "132-Column Text Mode" on page 3-223 for setting 132-column text mode. See "Smooth Scrolling of VGA and 132-Column Text Modes" on page 3-222 for details on achieving smooth scrolling. The capabilities of the 132-column text mode are:

XGA Each character is 8 pels wide. VGA character attributes are available.

XGA-2 Each character can be either 8 or 9 pels wide. VGA or mainframe interactive (MFI) character attributes are available. See "Mainframe Interactive (MFI) Support" on page 3-15.

The 132-column text mode is register compatible with the VGA, except for the following VGA CRT controller registers:

Horizontal Total
VGA requires that this register holds a value that is five less than the number of characters on a scan line. In 132-column text mode, this register requires a value that is one less than the number of characters on a scan line.

The End Horizontal Retrace

In 132-column text mode the End Horizontal Retrace field has no effect. Instead, Extended Graphics mode Horizontal Sync Pulse End register (index hex 1A) is used to give a larger horizontal count.

The Horizontal Retrace Delay field has no effect. Instead, Extended Graphics mode Horizontal Sync Pulse Position registers (index hex 1C and 1E) are used.

The End Horizontal Blanking, Bit 5 field continues to be effective.

Mainframe Interactive (MFI) Support

MFI character attribute support is available on the XGA-2 subsystem. It is available when operating in VGA or 132-Column text mode.

To ensure compatibility, video BIOS should be used to select or deselect the MFI character attribute support.

This function can also be enabled/disabled directly using the following registers:

- "Operating Mode Register (Address 21x0)" on page 3-35
- "MFI Control Register (Index 6D)" on page 3-86

However, this is not recommended.

When the XGA-2 subsystem is in text mode with the MFI attribute function enabled, the attribute byte of a character in the display buffer is redefined from the normal VGA format. This emulates the IBM 3270 and 5250 attributes.

7	6	5	4	3	2	1	0
BL	RV	UL	CS	I	R	G	B

```
     BL : Blinking Character / Background Intensity
     RV : Reverse Video
     UL : Underline On/Off
     CS : 5250 Column Separator On/Off
      I : Foreground Intensity / Character Select
  R,G,B : Foreground Color
```

Figure 3-2. MFI Attribute Byte

The fields of the byte are defined as follows:

BL Operates in the same manner as it does in the VGA attribute definition. When Character Blink is enabled (VGA Register: hex 3C0, Index: hex 10), BL will cause the character to blink at the MFI rate. When Background Intensity is selected, BL will select color 8 as the background color instead of the normal color 0. This is used for the trim border feature of some 3270 emulators, for marking areas of text, for printing, and for other purposes. MFI Blink rate for cursor and characters is defined below:

Mode	Blink Rate	Duty Cycle
VGA Cursor	VSYNC/16	50% On, 50% Off
VGA Characters	VSYNC/32	50% On, 50% Off
MFI Cursor	VSYNC/32	50% On, 50% Off
MFI Characters	VSYNC/64	75% On, 25% Off

Figure 3-3. MFI Blink Rates

RV Reverse Video swaps the foreground and background colors. When set to 1, Reverse Video is selected. When set to 0, Normal Video is selected. When Normal Video is selected, the background color is color 0 (or color 8 if Background Intensity is set). When Reverse Video is selected, the foreground color is color 0 (or color 8 if Background Intensity is set).

UL When set to 1, the Underline Bar of the character cell is set to the foreground color (visible) as opposed to a decode of the foreground and background colors with VGA attributes. When set to 0, the Underline Bar of the character cell is disabled.

CS When CS is set to 1, visible column-separator pels are displayed in the first and last pel positions of the character cell on the same character scan line as the underline bar. If UL is set to 1, the column-separator pels are set to the background color. If UL is set to 0, the column-separator pels are set to the foreground color. The column-separator pels are always visible, even if Underline is selected.

When CS is set to 0, the column-separator pels are not displayed.

I This bit selects low intensity or high intensity for the specified foreground color. When this bit is set to 0, the low intensity value is used. When this bit is set to 1, the high intensity value is used. This bit is also used to select between character sets. It operates in the same manner as the VGA equivalent bit. See "Alphanumeric Modes" on page 2-14.

R,G,B These bits define the foreground color. They operate in the same manner as the VGA foreground color bits. See "Alphanumeric Modes" on page 2-14.

When MFI function is enabled, control of the cursor type, blinking, and color are controlled using "MFI Control Register (Index 6D)" on page 3-86.

Extended Graphics Mode

Extended Graphics mode provides applications with high resolution, a wide range of colors, and high performance. The XGA coprocessor provides hardware assistance in drawing and moving data in video memory and in system memory. Extended Graphics mode is controlled using a bank of 16 I/O registers, and the coprocessor is controlled by a bank of 128 memory-mapped registers.

See "XGA Subsystem Identification, Location, and XGA Mode Setting" on page 3-185 to locate the subsystem in I/O and memory space.

Display Controller

Video Memory Format

The XGA video memory appears to the system as a byte-addressable, packed array of pels. The pels can be 1, 2, 4, 8, or 16 bits long. The first pel in memory is displayed at the top left corner of the screen. The next pel is immediately to its right and so on. Addressing is not necessarily contiguous, going from one horizontal line to the next. Addressing depends on the values in the Display Pel Map Width registers. See "CRT Controller" on page 3-22.

Two orders of pels are supported, Intel** and Motorola**.

To allow the XGA subsystem to function in either environment, the Memory Access Mode register (for display controller accesses) and the Pel Map n Format register (for coprocessor accesses) are used to make the pels appear in the required order.

The two formats are described in the following paragraphs.

** Intel is a trademark of the Intel Corporation and Motorola is a trademark of Motorola, Incorporated.

Intel Order: This table represents the first 3 bytes of the memory map in Intel order and shows the layout of the pels within those bytes for all pel sizes (bpp = bits-per-pel).

PEL	Byte = n + 2								Byte = n + 1								Byte = n + 0							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Size = 1 bpp																								
Number	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit significance	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Size = 2 bpp																								
Number	11	11	10	10	9	9	8	8	7	7	6	6	5	5	4	4	3	3	2	2	1	1	0	0
Bit significance	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
Size = 4 bpp																								
Number	5	5	5	5	4	4	4	4	3	3	3	3	2	2	2	2	1	1	1	1	0	0	0	0
Bit significance	3	2	1	0	3	2	1	0	3	2	1	0	3	2	1	0	3	2	1	0	3	2	1	0
Size = 8 bpp																								
Number	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
Bit significance	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Size = 16 bpp																								
Number	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit significance	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Figure 3-4. Intel Order of the XGA Memory Map

Motorola Order: This table represents the first 3 bytes of the memory map in Motorola order and shows the layout of the pels within those bytes for all pel sizes (bpp = bits-per-pel).

PEL	Byte = n + 0								Byte = n + 1								Byte = n + 2							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Size = 1 bpp																								
Number	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
Bit significance	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Size = 2 bpp																								
Number	0	0	1	1	2	2	3	3	4	4	5	5	6	6	7	7	8	8	9	9	10	10	11	11
Bit significance	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
Size = 4 bpp																								
Number	0	0	0	0	1	1	1	1	2	2	2	2	3	3	3	3	4	4	4	4	5	5	5	5
Bit significance	3	2	1	0	3	2	1	0	3	2	1	0	3	2	1	0	3	2	1	0	3	2	1	0
Size = 8 bpp																								
Number	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	2	2	2	2	2	2	2	2
Bit significance	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Size = 16 bpp																								
Number	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
Bit significance	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8

Figure 3-5. Motorola Order of the XGA Memory Map

Pel Color Mapping

In 1, 2, 4, or 8 bits-per-pel modes, the palette address is the numerical value of the pel.

In 16 bits-per-pel mode (direct color), the color mapping is 5 bits red, 6 bits green, and 5 bits blue. See "Direct Color Mode" on page 3-30.

Border Color Mapping

In the border area of the display, the palette is addressed by the Border Color register (index hex 55). The border area is defined in "CRT Controller" on page 3-22.

Direct Access to Video Memory

An application can use normal memory accesses to read or write pels in video memory. All bits of one or more pels can be accessed in a single memory cycle.

System Apertures into Video Memory: The XGA subsystem video memory is accessed in system memory address space through three possible apertures:

4MB Aperture
: This allows up to 4MB of video memory to be addressed consecutively. If an access is made at an offset higher than the size of memory installed, no memory is written and undefined values are returned when read.

1MB Aperture
: This allows up to 1MB of video memory to be addressed consecutively. If an access is made at an offset higher than the size of memory installed, no memory is written and undefined values are returned when read.

  Note: To use the 1MB aperture, the Aperture Index register must be set to 0.

64KB Aperture
: This allows up to 64KB of video memory to be addressed consecutively.

  The aperture can be located at any 64KB section of the video memory using the Aperture Index register.

See "PS/2 System Video Memory Apertures" on page 3-228 for details on locating and using these apertures.

CRT Controller

The CRT controller generates all timing signals required to drive the serializer and the display. It consists of two counters, one for horizontal parameters and one for vertical parameters, and a series of registers. The counters run continuously, and when the count-value reaches that specified in one of the associated registers, the event controlled by that register occurs.

See “Effects of VGA and XGA Mode Setting on Video Memory” on page 3-226 for mode tables, including CRT controller register values.

CRT Controller Register Interpretations

A representation of the function of each of the CRT controller registers is given in the following figure.

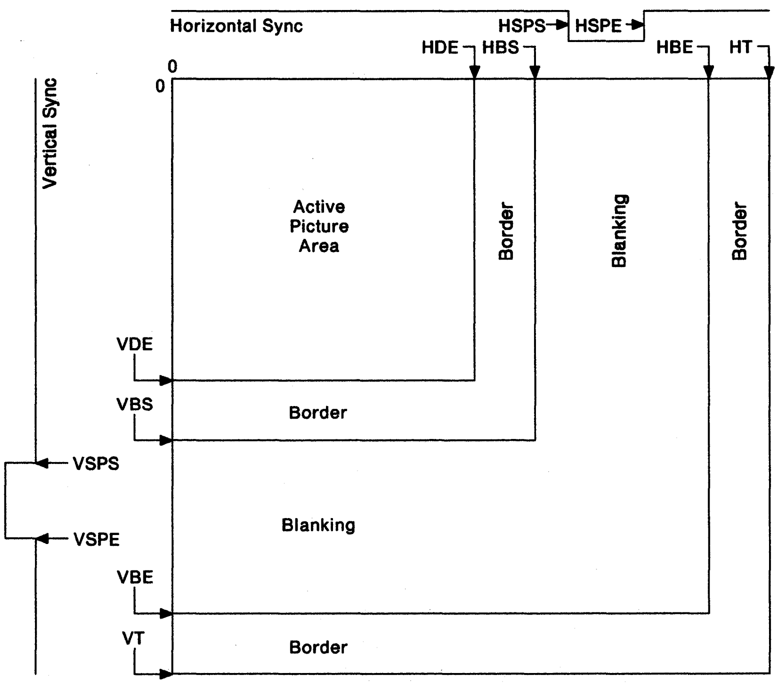


Figure 3-6. CRT Controller Register Definitions

The registers that control a horizontal scan of the display are:

HT	Horizontal Total register
HDE	Horizontal Display End register
HBS	Horizontal Blanking Start register
HBE	Horizontal Blanking End register
HSPS	Horizontal Sync Pulse Start register
HSPE	Horizontal Sync Pulse End register

The registers that control a vertical scan of the display are:

VT	Vertical Total register
VDE	Vertical Display End register
VBS	Vertical Blanking Start register
VBE	Vertical Blanking End register
VSPS	Vertical Sync Pulse Start register
VSPE	Vertical Sync Pulse End register

By using a system interrupt, the XGA subsystem can be programmed to inform the system microprocessor of the start and the end of the active picture area. An enable bit exists for each interrupt in the "Interrupt Enable Register (Address 21x4)" on page 3-38, and the "Interrupt Status Register (Address 21x5)" on page 3-40 contains a status bit for each interrupt.

Scrolling

Some or all of the displayed picture can be made to scroll. The first pel displayed on the screen is controlled by the Display Pel Map Offset registers. These can be altered to a granularity of 8 bytes, giving coarse horizontal scrolling. Vertical scrolling is achieved by altering the Display Pel Map Offset registers in units of one line length. The line length is stored in the Display Pel Map Width registers. The value stored in the width registers is the amount of memory allocated to each line, not necessarily the physical length of the line being displayed.

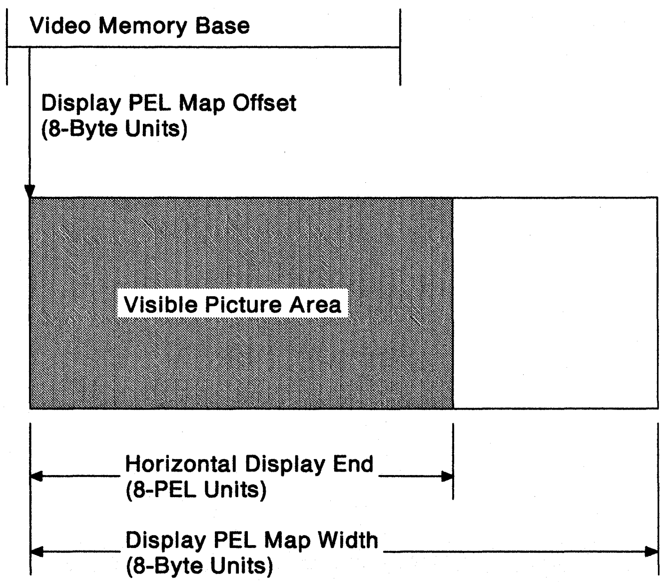


Figure 3-7. Display Pel Map Offset and Width Definitions

The Display Pel Map Width registers must be loaded with a value greater than or equal to the length of line being displayed. The most efficient use of video memory is achieved when the width value is made equal to the length of the line being displayed. However, it is often more convenient to load a width value that specifies the start of each line on a suitable address boundary.

An area at the bottom of the display can be prevented from scrolling by using the Vertical Line Compare registers (index hex 2C and 2D).

Sprite

The sprite is a 64 x 64-pel image stored in the XGA subsystem alpha/sprite buffer. When active, the sprite overlays the picture that is displayed. Each pel in the sprite can take on four values that can be used to achieve the effect of a colored marker of arbitrary shape.

Sprite Color Mapping

The sprite is stored as 2-bit packed pels, using Intel format, in the sprite buffer. Address zero is at the top left corner of the sprite.

These 2-bit pels determine the sprite appearance as shown in the following figure:

Bits 1 0	Sprite Effect
0 0	Sprite color 0
0 1	Sprite color 1
1 0	Transparent
1 1	Complement

Figure 3-8. Sprite Appearance Defined by 2-Bit Pel

The sprite effect definitions are as follows:

Sprite Colors 0 and 1
These colors are set by writing to the Sprite Color registers (index hex 38 through 3D).

Transparent
The underlying pel color is displayed.

Complement
The ones complement of the underlying pel color is displayed.

Sprite Buffer Accesses

The sprite buffer is written to by loading a number into the Sprite Index High and Sprite/Palette Index Low registers. These registers indicate the location of the first group of four sprite pels to be updated (2 bits-per-pel implies 4 pels-per-byte). Then the first four pels are written to the Sprite Data register. This stores the sprite pels in the sprite buffer and automatically increments the index registers. Subsequent writes to the Sprite Data register load the remaining sprite pels, four at a time.

The prefetch function is used to read from the sprite buffer. The index or address of the first sprite buffer location to be read is loaded into the index registers. Writing to either the Sprite Prefetch Index High or the Sprite/Palette Prefetch Index Low registers increments both registers as a single value. The first byte of the index must be written to a non-prefetch index register, and the second byte to the other prefetch index register. For example, write to Sprite Index High, then Sprite/Palette Prefetch Index Low.

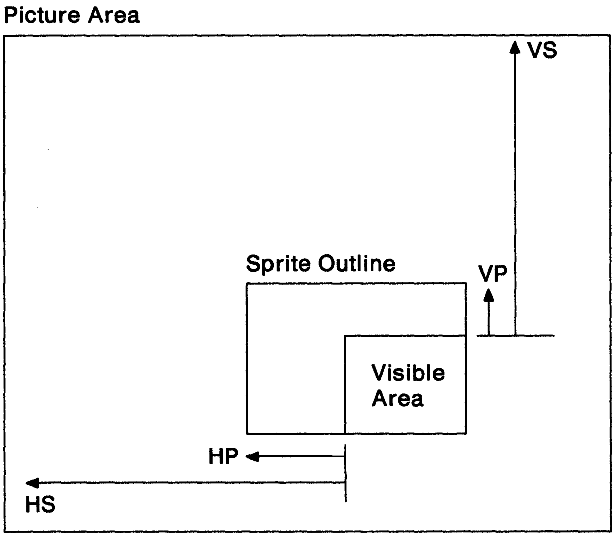
Writing to a prefetch index register loads the sprite data that is stored at the location specified in the index registers into a holding register, then increments the index registers as a single value. Reading the Sprite Data register returns the four sprite pels that were prefetched, and loads the next four sprite pels into the holding register. Subsequent reads from the Sprite Data register return the remaining sprite pels, four at a time.

The sprite and the palette use the same hardware registers during reading and writing, so any task that is updating either the sprite or palette when an interrupt occurs must save and restore the following registers:

- Sprite/Palette Index Low register (index hex 60)
- Sprite Index High register (index hex 61)
- Palette Sequence register (index hex 66)
- Palette Red Prefetch register (index hex 67)
- Palette Green Prefetch register (index hex 68)
- Palette Blue Prefetch register (index hex 69)
- Sprite Prefetch register (index hex 6B).

Note: The Sprite/Palette Prefetch Index Low register (index hex 62) and Sprite Prefetch Index High register (index hex 63) must not be saved and restored.

Sprite Positioning



HS - Horizontal Sprite Start
HP - Horizontal Sprite Preset
VS - Vertical Sprite Start
VP - Vertical Sprite Preset

Figure 3-9. Sprite Positioning

The sprite position is controlled by Start and Preset registers. The Start registers control where the first displayed sprite pel appears on the screen, and the Preset registers control which sprite pel is first displayed within the 64 x 64 sprite definition. Using these registers, the sprite can be made to appear at any point in the picture area. If the sprite overlaps any edge, the part of the sprite outside the picture area is not visible (does not wrap). See “Sprite Handling” on page 3-262.

The XGA subsystem can be programmed to inform the system microprocessor when the last line of the sprite has been displayed on each frame using a sprite-display-complete system interrupt. An enable bit exists for each interrupt in the Interrupt Enable register, and the Interrupt Status register contains a status bit for each interrupt. See “Interrupt Enable Register (Address 21x4)” on page 3-38 and “Interrupt Status Register (Address 21x5)” on page 3-40 for the location of the bits.

Palette

The palette has 256 locations and each location contains three fields, one each for red, green, and blue. The palette is used to translate the pel value into a displayed color.

Before the pel value is used to address the palette, it is masked by the Palette Mask register. All bits in the pel corresponding to 0's in the Palette Mask register are forced to 0 before reaching the palette.

Palette Accesses

The Palette Data register is 1 byte wide. Because each palette location is made up of three fields (red, green, and blue) three writes to the Palette Data register are required for each palette location. Palette data is held in a three-field holding register, and the contents are loaded into the palette RAM when all three fields have been filled. The Palette Sequence register controls the Holding Register field (red, green, or blue) selected for access with each write to the Palette Data register.

Two update sequences are possible:

1. Red, green, blue
2. Red, blue, green, no access

Data is written to the palette by first loading the index, or address, of the first group of three palette-color locations into the non-prefetched Sprite/Palette Index Low register. Because the palette has only 256 locations, the Sprite Index High register is not used. The first color byte is then written to the Palette Data register. This stores the color byte in the Holding Register field indicated by the Palette Sequence register. The Palette Sequence register then increments to point to the next field as determined by the update order.

A second write to the Palette Data register loads the next Holding Register field, and the Palette Sequence register increments again. A third write to the Palette Data register loads the remaining Holding Register field. If update sequence 1 is selected, the palette location is loaded from the holding register and the Palette Sequence register increments again, returning to its starting value. If update sequence 2 is selected, a fourth write to the Palette Data register is necessary before the palette location is loaded. The no-access data is ignored. Update sequence 2 allows the application to take advantage of the word or doubleword access possible with the XGA subsystem. See

"Data Registers (Addresses 21xB to 21xF)" on page 3-47 and "XGA Display Controller Registers" on page 3-33 for more details.

The prefetch function is used to read from the palette. The index or address of the first palette location to be read is loaded into the Sprite/Palette Prefetch Index Low register.

Writing to this register loads the three color fields stored at the location specified in the index register into the palette holding register, then increments the index register.

A subsequent read from the Palette Data register returns the data from the holding register color field, indicated by the Palette Sequence register, and increments the sequence register to point to the next color field. When the last color field, indicated by the Palette Sequence register, is read, the holding register is loaded with the next palette location data, and the index is incremented.

Note: If the subsystem has a monochrome display attached, all of the palette red and blue locations must be loaded with 0's. Alternatively, on the XGA-2 subsystem, the red and blue DAC outputs can be blanked using the BRB field of the "Miscellaneous Control Register (Index 6C)" on page 3-85.

The sprite and the palette use the same hardware registers during reading and writing, so any task that is updating either the sprite or palette when an interrupt occurs must save and restore the following registers:

- Sprite/Palette Index Low register (index hex 60)
- Sprite Index High register (index hex 61)
- Palette Sequence register (index hex 66)
- Palette Red Prefetch register (index hex 67)
- Palette Green Prefetch register (index hex 68)
- Palette Blue Prefetch register (index hex 69)
- Sprite Prefetch register (index hex 6B)

Note: The Sprite/Palette Prefetch Index Low register (index hex 62) and Sprite Prefetch Index High register (index hex 63) must not be saved and restored.

Direct Color Mode

In direct color mode the pel values in the video memory directly specify the displayed color.

The XGA subsystem can display direct color as a 16-bit pel. The color fields provide the most significant bits of the inputs to the video DACs with the color value.

The bits in the 16-bit direct color data word are allocated to the DAC bits as follows:

Word bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
5R, 6G, 5B	msb		RED		lsb	msb		GREEN			lsb	msb		BLUE		lsb

Figure 3-10. Direct Color Mode Data Word

When selecting this mode on the XGA subsystem, the palette must be loaded with data shown in Figure 3-11 on page 3-31. Only half of the palette should be loaded. Bit 7 of the Border Color register (index hex 55) specifies which half to load. If the Border Color register bit 7 = 0, load the upper half of the palette (locations hex 80 to FF). If the Border Color register bit 7 = 1, load the lower half (locations hex 00 to 7F).

The values shown in the figure are written to the Palette Data register as a byte value.

The XGA-2 subsystem can be initialized in this manner, but there are algorithmic alternatives to loading the palette:

Zero Intensity Black Undefined DAC bits held at 0. This is equivalent to the palette loading method specified above.

Full Intensity White Undefined DAC bits held at 1.

Linear Color Undefined DAC bits made equal to most significant defined bits.

Nonzero Color Undefined DAC bits set to 1 if color is nonzero.

See "Direct Color Control Register (Index 59)" on page 3-79 for control of undefined bits.

Note: This algorithmic method is only available on the XGA-2 subsystem. For applications that function on both levels of the

XGA subsystem, the palette loading method of initializing the mode must be used. Otherwise, the subsystem level can be determined and the appropriate method selected. See "XGA Level Identifier" on page 3-201.

Location (Hex) When Border Color (Bit 7) = 0	1	Data Written Red (Hex)	Green (Hex)	Blue (Hex)
80	0	0	0	0
81	1	0	0	8
82	2	0	0	10
83	3	0	0	18
84	4	0	0	20
•	•	•	•	•
•	•	•	•	•
9E	1E	0	0	F0
9F	1F	0	0	F8
A0	20	0	0	0
A1	21	0	0	8
•	•	•	•	•
•	•	•	•	•
BE	3E	0	0	F0
BF	3F	0	0	F8
C0	40	0	0	0
C1	41	0	0	8
•	•	•	•	•
•	•	•	•	•
DE	5E	0	0	F0
DF	5F	0	0	F8
E0	60	0	0	0
E1	61	0	0	8
•	•	•	•	•
•	•	•	•	•
FE	7E	0	0	F0
FF	7F	0	0	F8

Figure 3-11. XGA Direct Color Palette Load

The values shown in Figure 3-11 were chosen to ensure future compatibility.

See "XGA Subsystem Identification, Location, and XGA Mode Setting" on page 3-185 and "Direct Color Mode" on page 3-267 for more details on this mode.

Coprocessor Functions

Full coprocessor support in 16-bits-per-pel mode is available on the XGA-2 subsystem only. The XGA subsystem coprocessor functions do not work in 16-bits-per-pel mode. However, the coprocessor can function in 8-bits-per-pel mode while data is being displayed in 16 bits-per-pel. As a result, the coprocessor can be used to move data (in PxBlts) from one area of memory to another. See "XGA Level Identifier" on page 3-201 to identify different XGA levels.

When displaying in 16 bits-per-pel and using the coprocessor in 8-bits-per-pel mode, care should be taken when using any of the logical or arithmetic functions because each operation is performed on only 1 byte of data at a time, not the full 16-bits-per-pel.

If the coprocessor is used to move data into the video display buffer in 8-bits-per-pel format while displaying in 16-bits-per-pel mode, the width of the destination map must be doubled.

See "Direct Color Mode" on page 3-267 for more information.

XGA Display Controller Registers

The display controller registers occupy 16 I/O addresses. The addresses are hex 21x0 through 21xF. The x is the Instance as defined in Figure 3-170 on page 3-167. "XGA Subsystem Identification, Location, and XGA Mode Setting" on page 3-185 provides details of locating and using these registers.

An indexed addressing scheme is used to select additional registers. The index of the registers is written to hex 21xA; the data can then be accessed using hex 21xB through 21xF. Because there are multiple addresses for the data port, writes to a single register are achieved in a single 16-bit instruction; the low byte contains the address, and the high byte contains the data. Registers that need to be accessed repeatedly (sprite data, palette data, and coprocessor save/restore data) are accessed by setting the index correctly, then performing string I/O instructions, either 2 or 4 bytes at a time. See "Data Registers (Addresses 21xB to 21xF)" on page 3-47.

The 16 I/O addresses are assigned as shown in the following figure.

Address (hex)	Function	Page Reference
21x0	Operating Mode register	3-35
21x1	Aperture Control register	3-37
21x2	Reserved	
21x3	Reserved	
21x4	Interrupt Enable register	3-38
21x5	Interrupt Status register	3-40
21x6	Virtual Memory Control register	3-41
21x7	Virtual Memory Interrupt Status register	3-41
21x8	Aperture Index register	3-42
21x9	Memory Access mode	3-43
21xA	Index	3-44
21xB	Data	3-47
21xC	Data	3-47
21xD	Data	3-47
21xE	Data	3-47
21xF	Data	3-47

Figure 3-12. Display Controller Register Addresses

Register Usage Guidelines

Unless specified otherwise, the following are guidelines when using the display controller registers:

- All registers are 8 bits wide.
- Registers can be read and written at the same address or index.
- When registers are read, they return the last written data for all implemented bits.
- Registers are *not* initialized by reset.
- Special reserved register bits must be used as follows:
  - Register bits marked with '–' must be set to 0. These bits are undefined when read and should be masked off if the contents of the register is to be tested.
  - Register bits marked with '#' are reserved and the state of these bits must be preserved. When writing the register, read the register first and change only the bits that must be changed.
- Unspecified registers or registers marked as reserved in the XGA I/O address space are reserved. They must not be written to or read from.
- During a read, the values returned from write-only registers are reserved and unspecified.
- The contents of read-only registers must not be modified.
- Counters must not be relied upon to wrap from the high value to the low value.
- Register fields defined with valid ranges must not be loaded with a value outside the specified range.
- Register field values defined as reserved must not be written.
- The function that all XGA subsystem registers imply is only operative in XGA subsystem modes, even though the registers themselves are still readable and writable in VGA modes.

  Writing to the XGA subsystem registers when in VGA mode may cause the VGA registers to be damaged.

Direct Access I/O Registers

The following registers are directly addressable in the I/O space hex 21x0 through 21xF.

Operating Mode Register (Address 21x0)

This read/write register has an address of hex 21x0.

7	6	5	4	3	2	1	0
–	–	–	MFI	RF	DM		

```
  – : Set to 0, Undefined on Read
MFI : Enable MFI Function
 RF : Coprocessor Register Interface Format
 DM : Display Mode
```

Figure 3-13. Operating Mode Register, Address Hex 21x0

The register fields are defined as follows:

MFI This bit is only available on the XGA-2 subsystem. On the XGA subsystem, it should only be Set to 0 and assumed to be undefined on a read. When the Enable MFI Function field (bit 4) is set to 1 the display of MFI or VGA character attributes is controlled using "MFI Control Register (Index 6D)." When set to 0 this bit forces VGA character attributes to be displayed. See "Mainframe Interactive (MFI) Support" on page 3-15 for details on MFI attribute byte.

This bit must be set before "MFI Control Register (Index 6D)" can be accessed.

RF The Coprocessor Register Interface Format field (bit 3) selects whether the coprocessor registers are arranged in Intel or Motorola format. When set to 0, Intel format is selected. When set to 1, Motorola format is selected. See "Coprocessor Registers" on page 3-128.

DM The Display Mode field (bits 2 – 0) selects between the display modes available. Both VGA and 132-column text modes respond to VGA I/O and memory addresses. When the XGA subsystem is in either of these modes, the addressing of the I/O registers and the video memory can be inhibited.

DM Field (binary)	Display Mode
0 0 0	VGA Mode (address decode disabled)
0 0 1	VGA Mode (address decode enabled)
0 1 0	132-Column Text Mode (address decode disabled)
0 1 1	132-Column Text Mode (address decode enabled)
1 0 0	Extended Graphics Mode
1 0 1	Reserved
1 1 0	Reserved
1 1 1	Reserved

Figure 3-14. Display Mode Bit Assignments

Aperture Control Register (Address 21x1)

This read/write register has address of hex 21x1.

```
  7   6   5   4   3   2   1   0
┌───┬───┬───┬───┬───┬───┬───────┐
│ - │ - │ - │ - │ - │ - │  ASL  │
└───┴───┴───┴───┴───┴───┴───────┘

      - : Set to 0, Undefined on Read
    ASL : Aperture Size And Location
```

Figure 3-15. Aperture Control Register, Address Hex 21x1

The register fields are defined as follows:

ASL The Aperture Size and Location field (bits 1, 0) controls a 64KB aperture through which XGA memory can be accessed in system address space. This aperture gives real mode applications and operating systems a means of accessing the XGA video memory. The 64KB area of the XGA video memory accessed by this aperture is selected using the Aperture Index register. By varying the value of the index register, the 64KB aperture is used to access the entire memory contents of the subsystem.

The aperture is controlled as follows:

ASL Field (binary)	Aperture Size and Location
0 0	No 64KB Aperture
0 1	64KB at Address Hex 000A0000
1 0	64KB at Address Hex 000B0000
1 1	Reserved

Figure 3-16. Aperture Size and Location Bit Assignments

The 64KB aperture and a 1MB aperture cannot be used together because they are both paged using the Aperture Index register. See "System Apertures into Video Memory" on page 3-21.

Interrupt Enable Register (Address 21x4)

This read/write register has an address of hex 21x4.

7	6	5	4	3	2	1	0
CC	CR	–	–	–	SC	SP	SB

```
 – : Set to 0, Undefined on Read
CC : Coprocessor Operation Complete Enable
CR : Coprocessor Access Rejected Enable
SC : Sprite Display Complete Enable
SP : Start Of Picture (End Of Blanking) Enable
SB : Start Of Blanking (End Of Picture) Enable
```

Figure 3-17. Interrupt Enable Register, Address Hex 21x4

The register fields are defined as follows:

CC The Coprocessor Operation Complete Enable field (bit 7) enables and disables the Coprocessor Operation Complete interrupt condition that can be generated by the subsystem. When set to 1, the interrupt is enabled. When set to 0, the interrupt is disabled. The status of the bit in this field has no effect on the interrupt status bits as defined in the Interrupt Status register, but prevents the interrupt condition from causing a system interrupt.

CR The Coprocessor Access Rejected Enable field (bit 6) enables and disables the Coprocessor Access Rejected interrupt condition that can be generated by the subsystem. When set to 1, the interrupt is enabled. When set to 0, the interrupt is disabled. The status of the bit in this field has no effect on the interrupt status bits as defined in the Interrupt Status register, but prevents the interrupt condition from causing a system interrupt.

SC The Sprite Display Complete Enable field (bit 2) enables and disables the Sprite Display Complete interrupt condition that can be generated by the subsystem. When set to 1, the interrupt is enabled. When set to 0, the interrupt is disabled. The status of the bit in this field has no effect on the interrupt status bits as defined in the Interrupt Status register, but prevents the interrupt condition from causing a system interrupt.

SP The Start of Picture (End of Blanking) Enable field (bit 1) enables and disables the Start of Picture interrupt condition that can be generated by the subsystem. When set to 1, the interrupt is enabled. When set to 0, the interrupt is disabled. The status of the bit in this field has no effect on the interrupt status bits as defined in the Interrupt Status register, but prevents the interrupt condition from causing a system interrupt.

SB The Start of Blanking (End of Picture) Enable field (bit 0) enables and disables the Start of Blanking interrupt condition that can be generated by the subsystem. When set to 1, the interrupt is enabled. When set to 0, the interrupt is disabled. The status of the bit in this field has no effect on the interrupt status bits as defined in the Interrupt Status register, but prevents the interrupt condition from causing a system interrupt.

Interrupt Status Register (Address 21x5)

This read/write register has an address of hex 21x5.

7	6	5	4	3	2	1	0
CC	CR	–	–	–	SC	SP	SB

```
 – : Set to 0, Undefined on Read
CC : Coprocessor Operation Complete Status
CR : Coprocessor Access Rejected Status
SC : Sprite Display Complete Status
SP : Start Of Picture (End Of Blanking) Status
SB : Start Of Blanking (End Of Picture) Status
```

Figure 3-18. Interrupt Status Register, Address Hex 21x5

The register fields are defined as follows:

CC The Coprocessor Operation Complete Status field (bit 7) contains the interrupt status bit that can be generated by the subsystem to reset the Coprocessor Operation Complete interrupt. When read, 1 indicates that the interrupt condition has occurred, and 0 that it has not. Writing a 1 to the bit clears the interrupt condition, while writing a 0 has no effect. See "Programmer's View" on page 3-93 for more information.

CR The Coprocessor Access Rejected Status field (bit 6) contains the interrupt status bit that can be generated by the subsystem to reset the Coprocessor Access Rejected interrupt. When read, 1 indicates that the interrupt condition has occurred, and 0 that it has not. Writing a 1 to the bit clears the interrupt condition, while writing a 0 has no effect. See "Accesses to the Coprocessor During an Operation" on page 3-126 for more information.

SC The Sprite Display Complete Status field (bit 2) contains the interrupt status bit that can be generated by the subsystem to reset the Sprite Display Complete interrupt. When read, 1 indicates that the interrupt condition has occurred, and 0 that it has not. Writing a 1 to the bit clears the interrupt condition, while writing a 0 has no effect. See "Sprite" on page 3-25 for more information.

SP The Start of Picture (End of Blanking) Status field (bit 1) contains the interrupt status bit that can be generated by the subsystem to reset the Start of Picture interrupt. When read, 1 indicates that the interrupt condition has occurred, and 0 that it has not. Writing a 1 to the bit clears the interrupt condition, while writing a 0 has no effect. See "CRT Controller" on page 3-22 (End of blanking) for more information.

SB The Start of Blanking (End of Picture) Status field (bit 0) contains the interrupt status bit that can be generated by the subsystem to reset the Start of Blanking interrupt. When read, 1 indicates that the interrupt condition has occurred, and 0 that it has not. Writing a 1 to the bit clears the interrupt condition, while writing a 0 has no effect. See "CRT Controller" on page 3-22 (Start of blanking) for more information.

Virtual Memory Control Register (Address 21x6)

This read/write register has an address of hex 21x6. Full details of this register are in "Virtual Memory Control Register (I/O Address 21x6)" on page 3-181.

Virtual Memory Interrupt Status Register (Address 21x7)

This read/write register has an address of hex 21x7. Full details of this register are in "Virtual Memory Interrupt Status Register (I/O Address 21x7)" on page 3-183.

Aperture Index Register (Address 21x8)

This read/write register has an address of hex 21x8.

7	6	5	4	3	2	1	0
–	–	Aperture Index					

– : Set to 0, Undefined on Read

Figure 3-19. Aperture Index Register, Address Hex 21x8

The register field is defined as follows:

Aperture Index

The Aperture Index field (bits 5–0) provides address bits to video memory when the aperture in the system address space being used is smaller than the size of video memory installed. They are used to move both the 64KB aperture and the 1MB aperture. All 6 bits are used to move the 64KB aperture in the video memory, with a granularity of 64KB. When moving the 1MB aperture, the granularity is restricted to 1MB and only bits 5 and 4 are used. In this case, the lower order bits must be written with 0's.

See "System Apertures into Video Memory" on page 3-21 for details on the use of video memory apertures.

The bits are used as follows:

Aperture Size	Index Bits Used
64KB	5–0
1MB	5–4

Figure 3-20. Aperture Index Bit Assignments

Memory Access Mode Register (Address 21x9)

This read/write register has an address of hex 21x9.

7	6	5	4	3	2	1	0
–	–	–	–	PO	PS		

```
 – : Set to 0, Undefined on Read
PO : Pel Order
PS : Pel Size
```

Figure 3-21. Memory Access Mode Register, Address Hex 21x9

The register fields are defined as follows:

PO The Pel Order field (bit 3) controls pel ordering when the video memory is being accessed by the system (not the coprocessor). Intel or Motorola order can be selected. When set to 0, Intel format is selected. When set to 1, Motorola format is selected.

PS The Pel Size field (bits 2–0) selects the pel size. The pel size must be selected because this register is controlling a pel swapper that converts from the external format specified to the internal format used by the adapter when the pels are written, and converts back when they are read.

It is important to set this register correctly when accessing video memory with the system processor.

Pel size values are assigned as follows:

PS Field (binary)	Pel Size
0 0 0	1 Bit
0 0 1	2 Bits
0 1 0	4 Bits
0 1 1	8 Bits
1 0 0	16 Bits
1 0 1	Reserved
1 1 0	Reserved
1 1 1	Reserved

Figure 3-22. Pel Size Bit Assignments

Index Register (Address 21xA)

This read/write register has an address of hex 21xA.

7	6	5	4	3	2	1	0
Register Index							

Figure 3-23. Index Register, Address Hex 21xA

The Register Index register selects the indexed Extended Graphics mode register accessed when any address with index hex B through F is read or written. Index values are assigned as shown in the following figure.

Register Index Field (hex)	Register
04	Auto-Configuration
0C	Coprocessor Save/Restore Data A
0D	Coprocessor Save/Restore Data B
10	Horizontal Total Low
11	Horizontal Total High
12	Horizontal Display End Low
13	Horizontal Display End High
14	Horizontal Blanking Start Low
15	Horizontal Blanking Start High
16	Horizontal Blanking End Low
17	Horizontal Blanking End High
18	Horizontal Sync Pulse Start Low
19	Horizontal Sync Pulse Start High
1A	Horizontal Sync Pulse End Low
1B	Horizontal Sync Pulse End High
1C	Horizontal Sync Position
1E	Horizontal Sync Position
20	Vertical Total Low
21	Vertical Total High
22	Vertical Display End Low
23	Vertical Display End High
24	Vertical Blanking Start Low
25	Vertical Blanking Start High
26	Vertical Blanking End Low
27	Vertical Blanking End High
28	Vertical Sync Pulse Start Low
29	Vertical Sync Pulse Start High
2A	Vertical Sync Pulse End
2C	Vertical Line Compare Low
2D	Vertical Line Compare High
30	Sprite Horizontal Start Low
31	Sprite Horizontal Start High
32	Sprite Horizontal Preset
33	Sprite Vertical Start Low
34	Sprite Vertical Start High

Note: Undefined index values are reserved.

Figure 3-24 (Part 1 of 2). XGA Index Register Assignments

Register Index Field (hex)	Register
35	Sprite Vertical Preset
36	Sprite Control
38	Sprite Color 0 Red
39	Sprite Color 0 Green
3A	Sprite Color 0 Blue
3B	Sprite Color 1 Red
3C	Sprite Color 1 Green
3D	Sprite Color 1 Blue
40	Display Pel Map Offset Low
41	Display Pel Map Offset Middle
42	Display Pel Map Offset High
43	Display Pel Map Width Low
44	Display Pel Map Width High
50	Display Control 1
51	Display Control 2
52	Display ID and Comparator
54	Clock Frequency Select 1
55	Border Color
58	Programmable Pel Clock Frequency
59	Direct Color Control
60	Sprite/Palette Index Low
61	Sprite Index High
62	Sprite/Palette Prefetch Index Low
63	Sprite Prefetch Index High
64	Palette Mask
65	Palette Data
66	Palette Sequence
67	Palette Red Prefetch
68	Palette Green Prefetch
69	Palette Blue Prefetch
6A	Sprite Data
6B	Sprite Prefetch
6C	Miscellaneous Control
6D	MFI Control
70	Clock Frequency Select 2

Note: Undefined index values are reserved.

Figure 3-24 (Part 2 of 2). XGA Index Register Assignments

Data Registers (Addresses 21xB to 21xF)

These read/write data registers have addresses of hex 21xB to 21xF. The data registers are used when reading and writing to the register indexed by the Index register (Address 21xA). The read/write operation can be of byte, word, or doubleword size.

To perform a byte write to an indexed register, a single 16-bit cycle to address hex 21xA can be used with the index in the lower byte and the data to be written in the upper byte. For indexed registers requiring successive writes, the index can be loaded using a byte write to address hex 21xA, followed by either a word or a doubleword access to address hex 21xC. Only the byte-wide register selected by the index is updated. Word or doubleword accesses result in two or four byte-wide accesses to the same indexed register.

Indexed Access I/O Registers

See “Index Register (Address 21xA)” on page 3-44 for a figure of the indexed registers.

Auto-Configuration Register (Index 04)

This read-only register has an index of hex 04. *Do not write to* this register.

7	6	5	4	3	2	1	0
–	–	BT		BS1	–	–	BS0

```
  –  : Undefined on Read
 BS0 : Bus Size 0
 BS1 : Bus Size 1
  BT : System Bus Type
```

Figure 3-25. Auto-Configuration Register, Index Hex 04

The register field is defined as follows:

BS0 and BS1 The Bus Size field (bits 0 and 3) indicate whether the subsystem is interfaced to an 8-bit, a 16-bit or a 32-bit system, as defined in the following table:

BS1 Field	BS0 Field	System Interface Size
0	0	16 Bits
0	1	32 Bits
1	0	8 Bits
1	1	Reserved

Figure 3-26. System Interface Bus Size

BT The System Bus Type field (bits 5 and 4) indicates the bus type to which the subsystem is attached.

BT Field (binary)	System Bus Type
0 0	Micro Channel
0 1	ISA (AT* Bus)
1 0	Reserved
1 1	Reserved

Figure 3-27. System Bus Type

Coprocessor Save/Restore Data Registers (Index 0C and 0D)

These read/write registers have indexes of hex 0C and 0D. The registers are an image of a port in the coprocessor. See "Coprocessor State Save/Restore" on page 3-126 for a description of their use.

* Trademark of the IBM Corporation

Horizontal Total Registers (Index 10 and 11)

These read/write registers have indexes of hex 10 and 11.

```
 7    6    5    4    3    2    1    0
┌───────────────────────────────────────┐
│         Horizontal Total Low          │ (Index 10H)
└───────────────────────────────────────┘

 7    6    5    4    3    2    1    0
┌───────────────────────────────────────┐
│         Horizontal Total High         │ (Index 11H)
└───────────────────────────────────────┘
```

Figure 3-28. Horizontal Total Registers, Indexes Hex 10 and 11

The Horizontal Total Low and Horizontal Total High registers (bits 7 – 0) define the total length of a scan line in units of eight pels. They *must* be loaded as a 16-bit value in the range hex 0000 to 00FF. Values are assigned as shown in the following figure.

Value (hex)	Horizontal Total (Pels)
0000	8
0001	16
0002	24
•	•
•	•
00FF	2048

Figure 3-29. Horizontal Total Registers Value Assignments

Horizontal Display End Registers (Index 12 and 13)

These read/write registers have indexes of hex 12 and 13.

7	6	5	4	3	2	1	0	
Horizontal Display End Low								(Index 12H)

7	6	5	4	3	2	1	0	
Horizontal Display End High								(Index 13H)

Figure 3-30. Horizontal Display End Registers, Indexes Hex 12 and 13

The Horizontal Display End Low and Horizontal Display End High registers (bits 7 – 0) define the position of the end of the active picture area relative to the start of the active picture area in units of eight pels. They *must* be loaded as a 16-bit value in the range hex 0000 to 00FF. Values are assigned as shown in the following figure.

Value (hex)	Display End (Pels)
0000	8
0001	16
0002	24
•	•
•	•
00FF	2048

Figure 3-31. Horizontal Display End Registers Value Assignments

Horizontal Blanking Start Registers (Index 14 and 15)

These read/write registers have indexes of hex 14 and 15.

7	6	5	4	3	2	1	0	
Horizontal Blanking Start Low								(Index 14H)

7	6	5	4	3	2	1	0	
Horizontal Blanking Start High								(Index 15H)

Figure 3-32. Horizontal Blanking Start Registers, Indexes Hex 14 and 15

The Horizontal Blanking Start Low and Horizontal Blanking Start High registers (bits 7 – 0) define the position of the end of the picture border area relative to the start of the active picture area in units of eight pels. They *must* be loaded as a 16-bit value in the range hex 0000 to 00FF. Values are assigned as shown in the following figure.

Value (hex)	Blanking Start (Pels)
0000	8
0001	16
0002	24
•	•
•	•
00FF	2048

Figure 3-33. Horizontal Blanking Start Registers Value Assignments

Horizontal Blanking End Registers (Index 16 and 17)

These read/write registers have indexes of hex 16 and 17.

```
 7   6   5   4   3   2   1   0
+-------------------------------+
|  Horizontal Blanking End Low  |  (Index 16H)
+-------------------------------+

 7   6   5   4   3   2   1   0
+-------------------------------+
|  Horizontal Blanking End High |  (Index 17H)
+-------------------------------+
```

Figure 3-34. Horizontal Blanking End Registers, Indexes Hex 16 and 17

The Horizontal Blanking End Low and Horizontal Blanking End High registers (bits 7 – 0) define the position of the start of the picture border area relative to (after) the start of the active picture area in units of eight pels. They *must* be loaded as a 16-bit value in the range hex 0000 to 00FF. Values are assigned as shown in the following figure.

Value (hex)	Blanking End (Pels)
0000	8
0001	16
0002	24
•	•
•	•
00FF	2048

Figure 3-35. Horizontal Blanking End Registers Value Assignments

Horizontal Sync Pulse Start Registers (Index 18 and 19)

These read/write registers have indexes of hex 18 and 19.

7	6	5	4	3	2	1	0	
HSYNC Pulse Start Low								(Index 18H)

7	6	5	4	3	2	1	0	
HSYNC Pulse Start High								(Index 19H)

Figure 3-36. Horizontal Sync Pulse Start Registers, Indexes Hex 18 and 19

The Horizontal Sync Pulse Start Low and Horizontal Sync Pulse Start High registers (bits 7 – 0) define the position of the start of horizontal sync pulse relative to the start of the active picture area in units of eight pels. They *must* be loaded as a 16 bit-value in the range hex 0000 to 00FF. Values are assigned as shown in the following figure.

Value (hex)	Horizontal Pulse Start (Pels)
0000	8
0001	16
0002	24
•	•
•	•
00FF	2048

Figure 3-37. Horizontal Sync Pulse Start Registers Value Assignments

Horizontal Sync Pulse End Registers (Index 1A and 1B)

These read/write registers have indexes of hex 1A and 1B.

7	6	5	4	3	2	1	0	
HSYNC Pulse End Low								(Index 1AH)

7	6	5	4	3	2	1	0	
HSYNC Pulse End High								(Index 1BH)

Figure 3-38. Horizontal Sync Pulse End Registers, Indexes Hex 1A and 1B

The Horizontal Sync Pulse End Low and Horizontal Sync Pulse End High registers (bits 7 – 0) define the position of the end of the horizontal sync pulse relative to the start of the active picture area in units of eight pels. They *must* be loaded as a 16-bit value in the range hex 0000 to 00FF.

This XGA subsystem register is also used in 132-column text mode in place of the VGA End Horizontal Retrace register. In 132-column text mode, each eight-pel unit is equivalent to one eight-pel character. Values are assigned as shown in the following figure.

Value (hex)	Horizontal Sync Pulse End (Pels)
0000	8
0001	16
0002	24
•	•
•	•
00FF	2048

Figure 3-39. Horizontal Sync Pulse End Registers Value Assignments

Horizontal Sync Pulse Position Registers (Index 1C and 1E)

These write-only registers have indexes of hex 1C and 1E.

7	6	5	4	3	2	1	0	
–	PD		–	–	–	–	–	(Index 1CH)

7	6	5	4	3	2	1	0	
–	–	–	–	–	PD		–	(Index 1EH)

– : Set to 0
PD : Sync Pulse Delay

Figure 3-40. Horizontal Sync Pulse Position Registers (Index 1C and 1E)

The register field is defined as follows:

PD The Sync Pulse Delay field (bits 6, 5 or bits 2, 1) allows the 'horizontal sync' (HSYNC) signal to be delayed by up to four pels. The same value *must* be written to both registers, as shown in the following figure.

PD Field (binary)	Sync Pulse Delay in Pels
0 0	0
0 1	Reserved
1 0	4
1 1	Reserved

Figure 3-41. Horizontal Sync Pulse Delay Bit Assignments

These XGA subsystem registers are also used in 132-column text mode in place of the HRD field in the VGA End Horizontal Retrace register.

Vertical Total Registers (Index 20 and 21)

These read/write registers have indexes of hex 20 and 21.

7	6	5	4	3	2	1	0	
Vertical Total Low								(Index 20H)

7	6	5	4	3	2	1	0	
Vertical Total High								(Index 21H)

Figure 3-42. Vertical Total Registers, Indexes Hex 20 and 21

The Vertical Total Low and Vertical Total High registers (bits 7 – 0) define the total length of a frame in units of one scan line. They *must* be written as a 16-bit value in the range hex 0000 to 07FF. Values are assigned as shown in the following figure.

Value (hex)	Total Length (Scan Lines)
0000	1
0001	2
0002	3
•	•
•	•
07FF	2048

Figure 3-43. Vertical Total Registers Value Assignments

Vertical Display End Registers (Index 22 and 23)

These read/write registers have indexes of hex 22 and 23.

7	6	5	4	3	2	1	0	
Vertical Display End Low								(Index 22H)

7	6	5	4	3	2	1	0	
Vertical Display End High								(Index 23H)

Figure 3-44. Vertical Display End Registers, Indexes Hex 22 and 23

The Vertical Display End Low and Vertical Display End High registers (bits 7 – 0) define the position of the end of the active picture area relative to the start of the active picture area in units of one scan line. They *must* be written as a 16-bit value in the range hex 0000 to 07FF. Values are assigned as shown in the following figure.

Value (hex)	Display End (Scan Lines)
0000	1
0001	2
0002	3
•	•
•	•
07FF	2048

Figure 3-45. Vertical Display End Registers Value Assignments

Vertical Blanking Start Registers (Index 24 and 25)

These read/write registers have indexes of hex 24 and 25.

7	6	5	4	3	2	1	0	
Vertical Blanking Start Low								(Index 24H)

7	6	5	4	3	2	1	0	
Vertical Blanking Start High								(Index 25H)

Figure 3-46. Vertical Blanking Start Registers, Indexes Hex 24 and 25

The Vertical Blanking Start Low and Vertical Blanking Start High registers (bits 7 – 0) define the position of the end of the picture border area relative to the start of the active picture area in units of one scan line. They *must* be loaded as a 16-bit value in the range hex 0000 to 07FF. Values are assigned as shown in the following figure.

Value (hex)	Border End (Scan Lines) Blanking Start
0000	1
0001	2
0002	3
•	•
•	•
07FF	2048

Figure 3-47. Vertical Blanking Start Registers Value Assignments

Vertical Blanking End Registers (Index 26 and 27)

These read/write registers have indexes of hex 26 and 27.

7 6 5 4 3 2 1 0	
Vertical Blanking End Low	(Index 26H)

7 6 5 4 3 2 1 0	
Vertical Blanking End High	(Index 27H)

Figure 3-48. Vertical Blanking End Registers, Indexes Hex 26 and 27

The Vertical Blanking End Low and Vertical Blanking End High registers (bits 7 – 0) define the position of the start of the picture border area relative to the start of the active picture area in units of one scan line. They *must* be loaded as a 16-bit value in the range hex 0000 to 07FF. Values are assigned as shown in the following figure.

Value (hex)	Border Start (Scan Lines) Blanking End
0000	1
0001	2
0002	3
•	•
•	•
07FF	2048

Figure 3-49. Vertical Blanking End Registers Value Assignments

Vertical Sync Pulse Start Registers (Index 28 and 29)

These read/write registers have indexes of hex 28 and 29.

7	6	5	4	3	2	1	0	
VSYNC Pulse Start Low								(Index 28H)

7	6	5	4	3	2	1	0	
VSYNC Pulse Start High								(Index 29H)

Figure 3-50. Vertical Sync Pulse Start Registers, Indexes Hex 28 and 29

The Vertical Sync Pulse Start Low and Vertical Sync Pulse Start High registers (bits 7 – 0) define the position of the start of the vertical sync pulse relative to the start of the active picture area in units of one scan line. They *must* be loaded as a 16-bit value in the range hex 0000 to 07FF. Values are assigned as shown in the following figure.

Value (hex)	Sync Pulse Start (Scan Lines)
0000	1
0001	2
0002	3
•	•
•	•
07FF	2048

Figure 3-51. Vertical Sync Pulse Start Registers Value Assignments

Vertical Sync Pulse End Register (Index 2A)

This read/write register has an index of hex 2A.

7	6	5	4	3	2	1	0
VSYNC Pulse End							

Figure 3-52. Vertical Sync Pulse End Register, Index Hex 2A

The Vertical Sync Pulse End register (bits 7 – 0) defines the position of the end of the vertical sync pulse. The value loaded is the least significant byte of a 16-bit value that defines the end of the vertical sync pulse relative to the start of the active picture area in units of one scan line. The vertical sync end position *must* be within 31 scan lines of the vertical sync start position.

Note: Before setting the Operating Mode register (address 21x0) into VGA or 132-column text mode, bit 5 of this register must be set to 1.

This register might not return the value written, but the returned value is valid for save/restore operations.

Vertical Line Compare Registers (Index 2C and 2D)

These read/write registers have indexes of hex 2C and 2D.

7	6	5	4	3	2	1	0	
Vertical Line Compare Low								(Index 2CH)

7	6	5	4	3	2	1	0	
Vertical Line Compare High								(Index 2DH)

Figure 3-53. Vertical Line Compare Registers, Indexes Hex 2C and 2D

The Vertical Line Compare Low and Vertical Line Compare High registers (bits 7 – 0) define the position of the end of the scrollable picture area relative to the start of the active picture area in units of one scan line. They *must* be loaded as a 16-bit value in the range hex 0000 to 07FF. Values are assigned as shown in the following figure.

Value (hex)	Scrollable End (scan lines)
0000	1
0001	2
0002	3
•	•
•	•
07FF	2048

Figure 3-54. Vertical Line Compare Registers Value Assignments

Sprite Horizontal Start Registers (Index 30 and 31)

These read/write registers have indexes of hex 30 and 31.

7	6	5	4	3	2	1	0	
Sprite Horizontal Start Low								(Index 30H)

7	6	5	4	3	2	1	0	
Sprite Horizontal Start High								(Index 31H)

Figure 3-55. Sprite Horizontal Start Registers, Indexes Hex 30 and 31

The Sprite Horizontal Start Low and Sprite Horizontal Start High registers (bits 7 – 0) define the position of the start of the sprite relative to the start of the active picture area in pels. They *must* be loaded with a 16-bit value in the range hex 0000 to 07FF. See "Sprite Positioning" on page 3-27. Values are assigned as shown in the following figure.

Value (hex)	Sprite Start (Pels)
0000	0
0001	1
0002	2
•	•
•	•
07FF	2047

Figure 3-56. Sprite Horizontal Start Registers Value Assignments

Sprite Horizontal Preset Register (Index 32)

This read/write register has an index of hex 32.

7	6	5	4	3	2	1	0
–	–	Sprite Horizontal Preset					

– : Set to 0, Undefined on Read

Figure 3-57. Sprite Horizontal Preset, Index Hex 32

The register fields are defined as follows:

Sprite Horizontal Preset

The Sprite Horizontal Preset field (bits 5–0) defines the horizontal position within the 64 x 64-pel sprite area where the sprite starts. The sprite always ends at position 63 (it does not wrap). See "Sprite Positioning" on page 3-27. Values are assigned as shown in the following figure.

Value (hex)	Sprite Start (Pels)
00	0
01	1
02	2
•	•
•	•
3F	63

Figure 3-58. Sprite Horizontal Preset Value Assignments

Sprite Vertical Start Registers (Index 33 and 34)

These read/write registers have indexes of hex 33 and 34.

7	6	5	4	3	2	1	0	
Sprite Vertical Start Low								(Index 33H)

7	6	5	4	3	2	1	0	
Sprite Vertical Start High								(Index 34H)

Figure 3-59. Sprite Vertical Start Registers, Indexes Hex 33 and 34

The Sprite Vertical Start Low and Sprite Vertical Start High registers (bits 7 – 0) define the position of the start of the sprite relative to the start of the active picture area in units of one scan line. They *must* be loaded with a 16-bit value in the range hex 0000 to 07FF. See "Sprite Positioning" on page 3-27. Values are assigned as shown in the following figure.

Value (hex)	Sprite Start (Scan Lines)
0000	0
0001	1
0002	2
•	•
•	•
07FF	2047

Figure 3-60. Sprite Vertical Start Registers Value Assignments

Sprite Vertical Preset Register (Index 35)

This read/write register has an index of hex 35.

7	6	5	4	3	2	1	0
–	–	Sprite Vertical Preset					

– : Set to 0, Undefined on Read

Figure 3-61. Sprite Vertical Preset, Index Hex 35

The register fields are defined as follows:

Sprite Vertical Preset
The Sprite Vertical Preset field (bits 5 – 0) defines the vertical position within the 64 x 64-pel sprite area where the sprite starts. The sprite always ends at position 63 (it does not wrap). See "Sprite Positioning" on page 3-27. Values are assigned as shown in the following figure.

Value (hex)	Sprite Start (Pels)
00	0
01	1
02	2
•	•
•	•
3F	63

Figure 3-62. Sprite Vertical Preset Value Assignments

Sprite Control Register (Index 36)

This read/write register has an index of hex 36.

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	SC

– : Set to 0, Undefined on Read
SC : Sprite Control

Figure 3-63. Sprite Control Register, Index Hex 36

The register field is defined as follows:

SC The Sprite Control field (bit 0) controls the visibility of the sprite. When set to 1, the sprite appears on the screen at the location controlled by the sprite position registers. When set to 0, a sprite is not displayed. This bit must be set to 0 before any attempt is made to access the sprite image in the sprite buffer, otherwise the sprite buffer contents are damaged.

Sprite Color Registers (Index 38 – 3D)

These read/write registers have indexes of hex 38 through 3D.

7	6	5	4	3	2	1	0	
Sprite Color 0 Red								(Index 38H)
Sprite Color 0 Green								(Index 39H)
Sprite Color 0 Blue								(Index 3AH)
Sprite Color 1 Red								(Index 3BH)
Sprite Color 1 Green								(Index 3CH)
Sprite Color 1 Blue								(Index 3DH)

Figure 3-64. Sprite Color Registers, Indexes Hex 38 – 3D

The Sprite Color registers (bits 7 – 0) define the red, green, and blue components of the pels displayed when the sprite data for those pels selects color 0 or color 1. These colors are passed directly to the DACs, not through the palette, and must be programmed to give the actual color required.

Note: The XGA-2 subsystem uses all 8 bits of these registers. The XGA subsystem only uses the 6 most-significant bits.

Display Pel Map Offset Registers (Index 40 – 42)

These read/write registers have indexes of hex 40 through 42.

7	6	5	4	3	2	1	0	
Display Pel Map Offset Low								(Index 40H)

7	6	5	4	3	2	1	0	
Display Pel Map Offset Middle								(Index 41H)

7	6	5	4	3	2	1	0	
Display Pel Map Offset High								(Index 42H)

Figure 3-65. Display Pel Map Offset Registers, Indexes Hex 40 – 42

The Display Pel Map Offset registers (bits 7 – 0) define the address of the start of the visible portion of the video buffer in units of 8 bytes. They *must* be loaded as a single value in the range hex 00000 to 1FFFF. See "Scrolling" on page 3-24. Values are assigned as shown in the following figure.

Value (hex)	Display Pel Map Offset (bytes)
00000	0
00001	8
00002	16
•	•
•	•
1FFFF	1048568

Figure 3-66. Display Pel Map Offset Registers Value Assignments

Display Pel Map Width Registers (Index 43 and 44)

These read/write registers have indexes of hex 43 and 44.

7	6	5	4	3	2	1	0	
Display Pel Map Width Low								(Index 43H)

7	6	5	4	3	2	1	0	
Display Pel Map Width High								(Index 44H)

Figure 3-67. Display Pel Map Width Registers, Indexes Hex 43 and 44

The Display Pel Map Width Low and Display Pel Map Width High registers (bits 7 – 0) define the width of the display pel map in units of 8 bytes. They *must* be loaded as a single value in the range hex 000 to 7FF. See "Scrolling" on page 3-24. Values are assigned as shown in the following figure.

Value (hex)	Display Pel Map Width (bytes)
000	0
001	8
002	16
•	•
•	•
7FF	16376

Figure 3-68. Display Pel Map Width Registers Value Assignments

Display Control 1 Register (Index 50)

This read/write register has an index of hex 50.

7	6	5	4	3	2	1	0
SP		#	VE	SO	1	DB	

```
 # : Preserve Value Read When Writing
 1 : Set to 1, Undefined on Read
SP : SYNC Polarity
VE : Video Extension
SO : Display Scan Order
DB : Display Blanking
```

Figure 3-69. Display Control 1 Register, Index Hex 50

The register fields are defined as follows:

SP The SYNC Polarity field (bits 7, 6) value is assigned as shown in the following figure.

SP Field (binary)	Vertical	Horizontal	Lines
0 0	+	+	768
0 1	+	–	400
1 0	–	+	350
1 1	–	–	480

Figure 3-70. Sync Polarity Bit Assignments

VE The Video Extension field (bit 4) determines whether the video extension is enabled. When set to 0, the video extension is disabled. When set to 1, the video extension is enabled. See "VGA Video Extensions" on page 2-104.

SO The Display Scan Order field (bit 3) determines whether the display scan order is interlaced. When set to 0, the display scan order is not interlaced. When set to 1, the display scan order is interlaced.

DB The Display Blanking field (bits 1, 0) value is assigned as shown in the following figure.

DB Field (binary)	Display Blanking
0 0	Display Blanked, CRT Controller Reset
0 1	Display Blanked, Prepare for Reset
1 0	Reserved
1 1	Normal Operation

Figure 3-71. Display Blanking Bit Assignments

When resetting the CRT controller, the display blanking bits must be set to 01 (prepare for reset) first, followed by 00 (CRT controller reset).

Display Control 2 Register (Index 51)

This read/write register has an index of hex 51.

7	6	5	4	3	2	1	0
VSF		HSF		-	PS		

```
  - : Set to 0, Undefined on Read
VSF : Vertical Scale Factor
HSF : Horizontal Scale Factor
 PS : Pel Size
```

Figure 3-72. Display Control 2 Register, Index Hex 51

The register fields are defined as follows:

VSF The Vertical Scale Factor field (bits 7, 6) controls how many times each line is replicated. Values are assigned as shown in the following figure.

VSF Field (binary)	Vertical Scale Factor
0 0	1
0 1	2
1 0	4
1 1	Reserved

Figure 3-73. Vertical Scale Factor Bit Assignments

HSF The Horizontal Scale Factor field (bits 5, 4) controls how many times each pel is replicated horizontally. Values are assigned as shown in the following figure.

HSF Field (binary)	Horizontal Scale Factor
0 0	1
0 1	2
1 0	4
1 1	Reserved

Figure 3-74. Horizontal Scale Factor Bit Assignments

PS The Pel Size field (bits 2 − 0) defines the pel size (for the serializer, palette, and DAC), and the display scale factors (horizontal and vertical). Values are assigned as shown in the following figure.

PS Field (binary)	Pel Size
0 0 0	1 Bit
0 0 1	2 Bits
0 1 0	4 Bits
0 1 1	8 Bits
1 0 0	16 Bits (Direct Color Mode)
1 0 1	Reserved
1 1 0	Reserved
1 1 1	Reserved

Figure 3-75. Display Control 2 Register Pel Size Bit Assignments

Display ID and Comparator Register (Index 52)

This read-only register has an index of hex 52. *Do not write to* this register.

7	6	5	4	3	2	1	0
BD	GD	RD	–	DID			

```
  - : Undefined on Read
 BD : Blue DAC Comparator Status
 GD : Green DAC Comparator Status
 RD : Red DAC Comparator Status
DID : Display Identifier
```

Figure 3-76. Display ID and Comparator, Index Hex 52

The register fields are defined as follows:

BD The Blue DAC Comparator Status field (bit 7) indicates the state of the blue DAC output. When read as 1, the blue DAC output is low; when read as 0, the blue DAC output is high.

GD The Green DAC Comparator Status field (bit 6) indicates the state of the green DAC output. When read as 1, the green DAC output is low; when read as 0, the green DAC output is high.

RD The Red DAC Comparator Status field (bit 5) indicates the state of the red DAC output. When read as 1, the red DAC output is low; when read as 0, the red DAC output is high.

DID The Display Identifier field (bits 3 – 0) indicates the type of display attached. Bit values are defined by displays. Display identifier field bits are returned from the DMQS. See "XGA Subsystem Identification, Location, and XGA Mode Setting" on page 3-185. If required to read the display identifier bits directly, see "Display Type Detection" on page 3-209.

Clock Frequency Select 1 Register (Index 54)

This read/write register has an index of hex 54.

7	6	5	4	3	2	1	0
PCS	–	–	–	CS1		VCS	

```
  - : Set to 0, Undefined on Read
PCS : Programmable Clock Select
CS1 : Clock Select 1
VCS : Video Clock Scale Factor
```

Figure 3-77. Clock Frequency Selector Register, Index Hex 54

The Clock Frequency Select 1 register must be used in conjunction with the Clock Frequency Select 2 register. It is defined under "Clock Frequency Select 2 Register (Index 70)" on page 3-88.

Border Color Register (Index 55)

This read/write register has an index of hex 55.

7	6	5	4	3	2	1	0
Border Color							

Figure 3-78. Border Color Register, Index Hex 55

The Border Color register (bits 7–0) holds the border color palette index, which is the index of the palette location selected to be displayed in the picture border area of the display.

The inverse of bit 7 is used for palette address bit 7 in direct color mode. See "Direct Color Mode" on page 3-30.

Programmable Pel Clock Register (Index 58)

This read/write register has an index of hex 58. It is only available on the XGA-2 subsystem.

7	6	5	4	3	2	1	0
FR		Frequency Index					

```
FR : Frequency Range
```

Figure 3-79. Programmable Pel Clock register, Index Hex 58

The register is defined as follows:

FR The FR field (bits 7–6) defines the range of frequencies that can be programmed by the Frequency Index field. The value in this field defines the Division Factor used in the formula below. Values are assigned as shown in the following figure.

FR Field (binary)	Division Factor	Frequency Range (MHz)
0 0	4	16.25 to 32.00 in 0.25 MHz increments
0 1	2	32.50 to 64.00 in 0.50 MHz increments
1 0	1	65.00 to 128.00 in 1.00 MHz increments
1 1	–	Reserved

Figure 3-80. Programmable Frequency Ranges

Frequency Index The Frequency Index field (bits 5–0) in conjunction with the Division Factor, defines the Pel frequency. The value loaded in the Frequency Index field is derived from the following formula:

Frequency Index = (f × Division Factor) – 65

where f is the Pel Frequency required.

Note: The maximum Pel Frequency that should be programmed for the XGA-2 subsystem is 90 MHz.

Direct Color Control Register (Index 59)

This read/write register has an index of hex 59. It is only available on the XGA-2 subsystem.

7	6	5	4	3	2	1	0
-	-	-	-	-	DCM		

```
  - : Set to 0, Undefined on Read
DCM : Direct Color Mode
```

Figure 3-81. Direct Color Control register, Index Hex 59

The register is defined as follows:

DCM The DCM field (bits 2—0) selects which algorithm is used to define the low order DAC bits which are not defined in the 16 bit Color Value. This field only takes effect when the XGA subsystem is displaying in Direct Color mode.

DCM Field (binary)	Algorithm
0 0 0	Palette Defined (see "Direct Color Mode" on page 3-30)
0 0 1	Missing bits set to 1 if Color is Non Zero
0 1 0	Missing bits set to 0
0 1 1	Missing bits set to 1
1 0 0	Missing bits equal to the most significant bits of same color field
1 0 1	Reserved
1 1 0	Reserved
1 1 1	Reserved

Figure 3-82. Direct Color Modes

Sprite/Palette Index Registers (Index 60 and 61)

These read/write registers have indexes of hex 60 and 61.

7	6	5	4	3	2	1	0	
Sprite/Palette Index Low								(Index 60H)

7	6	5	4	3	2	1	0	
Sprite Index High								(Index 61H)

Figure 3-83. Sprite/Palette Index Registers, Indexes Hex 60 and 61

The Sprite/Palette Index Low and Sprite Index High registers (bits 7 – 0) specify the index when writing to the sprite or the palette. See "Sprite Buffer Accesses" on page 3-26 and "Palette Accesses" on page 3-28 for details of these registers.

The Sprite/Palette Index Low register is used for the 256 locations of the palette that are available. It can be loaded with any palette index value in the range hex 00 to FF.

The Sprite/Palette Index Low and the Sprite Index High registers are both used to access the sprite. The registers can be loaded with any sprite index value in the range hex 0000 to 3FFF.

Accessing these registers does not cause any action other than loading or returning the value of the next index.

The registers must be saved, and subsequently restored, by any interrupting task that uses the palette or sprite registers.

Sprite/Palette Prefetch Index Registers (Index 62 and 63)

These read/write registers have indexes of hex 62 and 63.

```
  7    6    5    4    3    2    1    0
┌────────────────────────────────────────┐
│    Sprite/Palette Prefetch Index Low   │  (Index 62H)
└────────────────────────────────────────┘

  7    6    5    4    3    2    1    0
┌────────────────────────────────────────┐
│       Sprite Prefetch Index High       │  (Index 63H)
└────────────────────────────────────────┘
```

Figure 3-84. Sprite/Palette Prefetch Index Registers, Indexes 62 and 63

The Sprite/Palette Prefetch Index Low and Sprite Prefetch Index High registers (bits 7 – 0) specify the index when reading from the sprite or the palette. See "Sprite Buffer Accesses" on page 3-26 and "Palette Accesses" on page 3-28 for details of these registers.

When reading from the palette, the Sprite/Palette Prefetch Index Low register must be used. Writing the Sprite/Palette Prefetch Index Low register also causes the palette prefetch registers to be loaded, and the index value to be incremented.

When reading from the sprite, use either the Sprite/Palette Prefetch Index Low register or the Sprite Prefetch Index High register. Writing to either register also causes the sprite prefetch registers to be loaded, and the index value to be incremented as a single value.

These registers must *not* be saved and subsequently restored in hardware task switches.

Palette Mask Register (Index 64)

This read/write register has an index of hex 64.

7	6	5	4	3	2	1	0
Palette Mask							

Figure 3-85. Palette Mask Register, Index Hex 64

The contents of the Palette Mask register (bits 7–0) are ANDed with each display memory Pel value, and the result is used to index the palette.

Palette Data Register (Index 65)

This read/write register has an index of hex 65.

7	6	5	4	3	2	1	0
Palette Data							

Figure 3-86. Palette Data Register, Index Hex 65

The Palette Data register (bits 7–0) is an image of the currently selected palette RAM location. The data returned on read may not be that last written because of the selection mechanism described in "Palette Accesses" on page 3-28.

For monochrome displays, all of the palette red and blue locations *must* be loaded with 0's. Alternatively on the XGA-2 subsystem only, the Red and Blue DAC outputs can be blanked using the BRB field of the "Miscellaneous Control Register (Index 6C)" on page 3-85.

Note: The XGA-2 subsystem uses all 8 bits of these registers. The XGA subsystem only uses the 6 most-significant bits.

Palette Sequence Register (Bits 2 – 0 only) (Index 66)

This read/write register has an index of hex 66.

7	6	5	4	3	2	1	0
–	–	–	–	–	CO	PC	

```
 – : Set to 0, Undefined on Read
CO : Color Order
PC : Palette Color
```

Figure 3-87. Palette Sequence Register, Index Hex 66

The register fields are defined as follows:

CO The Color Order field (bit 2) defines the sequence to be followed for selecting the red, green, and blue elements during successive Palette Data register accesses. The color order is shown in the following figure.

CO Field (binary)	Color Order
0	R, G, B, R, G, B, ...
1	R, B, G, x, R, B, G, x, ...

Note: x = discarded data.

Figure 3-88. Palette Sequence Register Color Order Bit Assignment

PC The Palette Color field (bits 1, 0) defines which of the red, green, or blue elements of the currently selected palette location is the current one for the Palette Data register. See "Palette Accesses" on page 3-28 for more information. The palette color selection is shown in the following figure.

PC Field (binary)	Color
0 0	R
0 1	G
1 0	B
1 1	x

Note: x = discarded data.

Figure 3-89. Palette Sequence Register Color Bit Assignments

Palette Red Prefetch Register (Index 67)

This read/write register has an index of hex 67.

7	6	5	4	3	2	1	0
Palette Red Prefetch							

Figure 3-90. Palette Red Prefetch Register, Index Hex 67

The Palette Red Prefetch register (bits 7–0) is not used for any normal function but must be saved, and subsequently restored, by any interrupting code that uses the sprite or palette registers.

Palette Green Prefetch Register (Index 68)

This read/write register has an index of hex 68.

7	6	5	4	3	2	1	0
Palette Green Prefetch							

Figure 3-91. Palette Green Prefetch Register, Index Hex 68

The Palette Green Prefetch register (bits 7–0) is not used for any normal function but must be saved, and subsequently restored, by any interrupting code that uses the sprite or palette registers.

Palette Blue Prefetch Register (Index 69)

This read/write register has an index of hex 69.

7	6	5	4	3	2	1	0
Palette Blue Prefetch							

Figure 3-92. Palette Blue Prefetch Register, Index Hex 69

The Palette Blue Prefetch register (bits 7–0) is not used for any normal function but must be saved, and subsequently restored, by any interrupting code that uses the sprite or palette registers.

Sprite Data Register (Index 6A)

This read/write register has an index of hex 6A.

7	6	5	4	3	2	1	0
Sprite Data							

Figure 3-93. Sprite Data Register, Index Hex 6A

The Sprite Data register (bits 7 – 0) is an image of the currently selected sprite buffer location. The data returned on read may not be that last written because of the selection mechanism described in "Sprite Buffer Accesses" on page 3-26.

When used for writing sprite data, the sprite pels are Intel format packed pels.

Sprite Prefetch Register (Index 6B)

This read/write register has an index of hex 6B.

7	6	5	4	3	2	1	0
Sprite Prefetch							

Figure 3-94. Sprite Prefetch Register, Index Hex 6B

The Sprite Prefetch register (bits 7 – 0) is not used for any normal function but must be saved, and subsequently restored, by any interrupting code that uses the sprite or palette registers.

Miscellaneous Control Register (Index 6C)

This read/write register has an index of hex 6C. It is only available on the XGA-2 subsystem.

7	6	5	4	3	2	1	0
-	–	–	–	–	–	–	BRB

```
  – : Set to 0, Undefined on Read
BRB : Blank Red and Blue
```

Figure 3-95. Miscellaneous Control Register, Index 6C

The register is defined as follows:

BRB When the BRB field (bit 0) is set to 1, the outputs of the Red and Blue DACs are forced to 0, regardless of the DAC inputs. This should be used when a monochrome display is connected to the subsystem.

When the BRB field is set to 0, the DACs function as normal.

MFI Control Register (Index 6D)

This read/write register has an index of hex 6D. It is only available on the XGA-2 subsystem. This register can only be accessed (written or read) when the MFI function has been enabled. See “Mainframe Interactive (MFI) Support” on page 3-15 and “Operating Mode Register (Address 21x0)” on page 3-35.

7	6	5	4	3	2	1	0
Cursor Color				CCC	CBD	CT	MAE

```
CCC : Constant Cursor Color
CBD : Cursor Blink Disable
 CT : Cursor Type
MAE : MFI Attribute Enable
```

Figure 3-96. MFI Control Register, Index 6D

The register is defined as follows:

Cursor Color The Cursor Color field (bits 7 – 4) defines the cursor to be one of 16 colors when the CCC field is set to 1. It is defined in the same manner as the I, R, G, and B fields in the attribute byte. See "Mainframe Interactive (MFI) Support" on page 3-15

CCC When the CCC field (bit 3) is set to 1, a constant color cursor is displayed as defined by the Cursor Color field. When the CCC field is set to 0, the cursor color adopts the foreground color of the character upon which is it placed.

CBD When the CBD field (bit 2) is set to 1, a non-blinking cursor is displayed. When the CBD field is set to 0, the cursor will blink at the MFI rate. See "Mainframe Interactive (MFI) Support" on page 3-15.

CT When the CT field (bit 1) is set to 0, the cursor forces the foreground color of the character upon which it is placed. When the CT field is set to 1, the cursor forces the reverse video of the character upon which it is placed.

MAE When the MAE field (bit 0) is set to 0, normal VGA characters attributes are displayed and the other fields in this register have no effect. When set to 1 the MFI character attributes are displayed and the other fields of this register become active.

Clock Frequency Select 2 Register (Index 70)

This read/write register has an index of hex 70. It is used with the Clock Frequency Select 1 Register (index hex 54) for clock selection.

7	6	5	4	3	2	1	0	
PCS	–	–	–	CS1		VCS		(Index 54H)

7	6	5	4	3	2	1	0	
CS2	–	–	–	–	–	–	–	(Index 70H)

```
  – : Set to 0, Undefined on Read
PCS : Programmable Clock Select (XGA-2 only)
CS1 : Clock Select 1
CS2 : Clock Select 2
VCS : Video Clock Scale Factor
```

Figure 3-97. Clock Frequency Select Registers

The clock frequency select registers fields are defined as follows:

PCS The Programmable Clock Select field (bit 7 of the Clock Frequency Select 1 register) is used in conjunction with the Clock Select 1 and Clock Select 2 fields. See the description of the Clock Select 1 field to learn how it is used. Programmable Clock Select is supported by XGA-2 only.

CS2 The Clock Select 2 field (bit 7 of the Clock Frequency Select 2 register) is used in conjunction with the Clock Select 1 and Programmable Clock Select fields. See the description of the Clock Select 1 field to learn how it is used.

CS1 The Clock Select 1 field (bits 3, 2 of the Clock Frequency Select 1 register) must be used in conjunction with the Clock Select 2 and Programmable Clock Select fields Clock selection is shown in the following figure.

PCS Field (binary)	CS2 Field (binary)	CS1 Field (binary)	Selected Clock
0	0	0 0	VGA 8-Pel Character Mode and 640x480 Graphics Mode Clock
0	0	0 1	VGA 9-Pel Character Mode Clock
0	0	1 0	Clock Sourced from Video Extension Interface
0	0	1 1	1024x768 Graphics Mode Clock
0	1	0 0	132-Column Text Mode Clock (8 Pel Characters)
0	1	0 1	Reserved
0	1	1 0	Reserved
0	1	1 1	Reserved
1	0	0 0	Programmed Pel Clock (XGA-2 only)
1	0	0 1	Reserved
1	0	1 0	Reserved
1	0	1 1	Reserved
1	1	0 0	Reserved
1	1	0 1	Reserved
1	1	1 0	Reserved
1	1	1 1	Reserved

Figure 3-98. Clock Selected Bit Assignments

Note: See "Effects of VGA and XGA Mode Setting on Video Memory" on page 3-226 for details of mode setting.

VCS The Video Clock Scale Factor field (bits 1, 0 of the Clock Frequency Select 1 register) controls the divide ratio of the selected video clock before it is used by the CRT controller. The operation of the video clock scale factor is invisible to the programmer, but it must be set as shown for correct operation of the hardware.

VCS Field (binary)	Video Clock Scale Factor	Mode
0 0	1	VGA and 640x480 Graphics
0 1	2	1024x768 Graphics and 132-Column Text
1 0	Reserved	
1 1	Reserved	

Figure 3-99. Video Clock Scale Factor Bit Assignments

Coprocessor Description

The XGA coprocessor provides autonomous drawing functions for the video subsystem. Autonomous drawing functions means that the coprocessor draws into memory (either video memory or system memory) independently of the system microprocessor, while the system microprocessor is performing some other operation.

The coprocessor supports 1, 2, 4, 8, or 16 bits-per-pel on the XGA-2 subsystem. Support is limited to 1, 2, 4, or 8 bits-per-pel on the XGA subsystem. See "Direct Color Mode" on page 3-267 for details of using the coprocessor when displaying in 16-bits-per-pel (direct color) mode.

The execution of an operation using the coprocessor involves the following steps:

1. The system microprocessor sets up the coprocessor registers to perform a particular operation.
2. The system microprocessor writes to the pel Operations register to start the coprocessor operation.
3. The coprocessor performs the drawing operation. The system microprocessor can be performing some other function at this time.
4. The coprocessor completes the drawing operation, informs the system microprocessor, and becomes idle.
5. The process is repeated.

The coprocessor operates on pels within pel maps. A pel map is an area of memory at a given address with a defined height, width, and pel format (see "Pel Maps" on page 3-95).

Pels from a source are combined with pels from a destination under the control of a pattern and mask, and the result is written back to the destination.

After each access, the source, destination, pattern, and mask addresses are updated according to the function being performed, and the operation is repeated until a programmed limit is encountered.

The drawing operation can be a pel block transfer (PxBlt), Bresenham line draw, or draw and step.

The function performed to combine the source and destination data can be a logical or arithmetic operation. One of two possible operations is selected for each pel by the value of the corresponding pattern pel. A mask pel for each pel protects the destination from update.

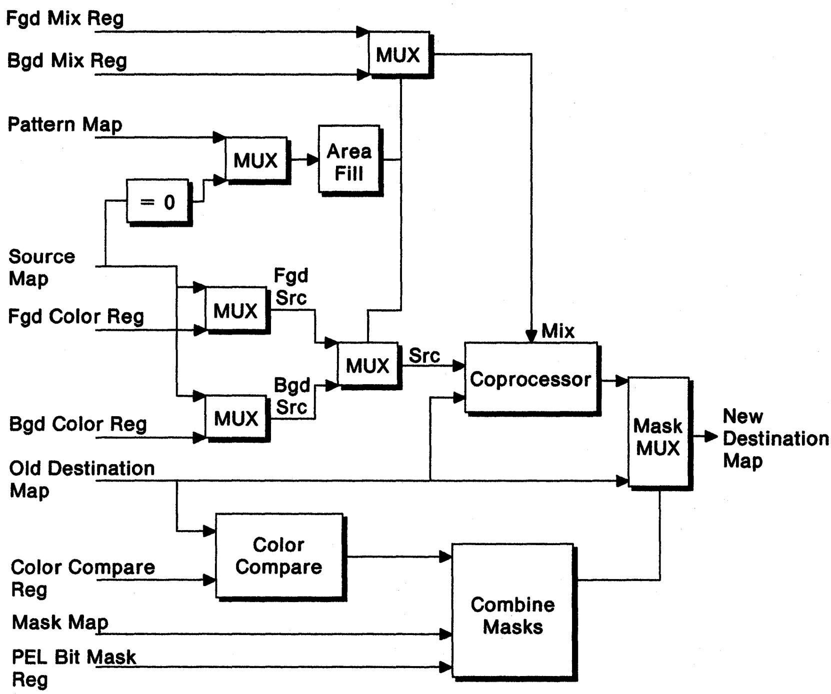
Pattern data can be generated automatically from source data by detecting pels with a value of 0.

A color compare function allows the modifying of the destination pel to be dependent on the value of the destination pel, compared to a programmable value.

Three general purpose pel maps can be defined in memory. Each map has a defined start address, width and height in pels, and number of bits-per-pel. Source, pattern, and destination data can reside in any combination of these maps. There is also a mask map with its defined start address, width, height, and format. Mask data is always taken from this map.

Source, pattern, and destination data are each addressed by unique X,Y pointers. Mask data is addressed by the destination X,Y pointers (see Figure 3-101 on page 3-96). If the source or pattern X,Y pointers move outside the defined extremities of their pel maps, they are reset automatically to wrap round to the opposite side of the pel map. If the destination X,Y pointers move outside the extremities of the destination map, update of the destination map is inhibited until the X,Y pointers move back inside the map.

Figure 3-100 on page 3-92 represents the coprocessor graphics data flow for the passage of one pel. In reality, multiple pels are processed in one cycle.



Fgd - Foreground
Bgd - Background
Src - Source

MUX - Multiplexer
Reg - Register

Figure 3-100. Coprocessor Data Flow

Programmer's View

An operation is defined as the execution of a single PxBlt, line draw, or draw and step function.

An operation is set up by first loading the registers of the coprocessor with appropriate data, such as X,Y coordinates, function mixes, and dimensions. Then the operation is initiated by writing to the pel Operations register. This defines the flow of data in the operation and starts the operation. The coprocessor then executes the operation and completes it when some programmed limit is reached.

There is one exception to this sequence of initiating operations: the draw and step function (see the section "Draw and Step" on page 3-105).

The XGA can be programmed to inform the system microprocessor of the completion of an operation using a system interrupt. This interrupt is called the coprocessor-operation-complete interrupt. An enable bit and status bit exist for this interrupt in the "Interrupt Enable Register (Address 21x4)" on page 3-38 and "Interrupt Status Register (Address 21x5)" on page 3-40.

A mechanism is provided to let the system microprocessor suspend or terminate an operation before it is completed. The suspension of operations is required to allow task switches, while termination of operations can be used to recover from errors.

Pel Formats

On the XGA-2 subsystem, the coprocessor can manipulate images with 1, 2, 4, 8, or 16 bits-per-pel. On the XGA subsystem, it can manipulate images with 1, 2, 4, or 8 bits-per-pel. It manipulates packed-pel data, so each data doubleword (32 bits) contains:

XGA-2 subsystem 32, 16, 8, 4, or 2 pels respectively.

XGA subsystem 32, 16, 8, or 4 pels respectively.

The pels can be in Intel or Motorola format. See Figure 3-4 on page 3-19 and Figure 3-5 on page 3-20 for Intel and Motorola formats.

Each pel map manipulated by the coprocessor can be defined as either Motorola or Intel format. If the destination map has a different format than the source, pattern, or mask maps, the coprocessor automatically translates between the two formats.

Motorola or Intel format is controlled by a bit in the Pel Map Format register.

Pel Fixed and Variable Data

When executing an operation, the coprocessor reads source, pattern, and mask data, and reads and writes destination data. The source, pattern, and mask data can be fixed throughout the operation, or it can vary from pel to pel.

If fixed data is used, it is written to the relevant fixed data register in the coprocessor before the operation is started (Foreground Color and Background Color registers).

If variable data is required, the data is read from memory by the coprocessor during the operation. The coprocessor only allows variable data from memory, and does not let the system unit system microprocessor supply variable data.

The Coprocessor View of Memory

To the programmer, the coprocessor treats video memory and system memory the same. Data can be moved between system memory and video memory by defining pel maps at the appropriate addresses.

Accesses to the XGA video memory are faster than accesses to system memory.

The coprocessor can address all of the video memory.

The Video Memory Base Address register and Instance indicate the base address where the video memory appears in system address range. This base address is on a 4MB address boundary. The coprocessor assumes that the whole 4MB of address range above this boundary is reserved for its own video memory. All addresses outside this 4MB block are treated as system memory. See "POS Register 4 (Base + 4)" on page 3-168.

"Direct Access to Video Memory" on page 3-21 describes video memory addressing.

Pel Maps

Pel Maps A, B, and C (General Maps)

The coprocessor defines three general purpose pel maps in memory, called pel maps A, B, and C. Each map is defined by four registers:

Pel Map Base Pointer	Specifies the linear start address of the map in memory.
Pel Map Width	Specifies the width of the map in pels. The value programmed must be one less than the required width.
Pel Map Height	Specifies the height of the map in pels. The value programmed must be one less than the required height.
Pel Map Format	Specifies the number of bits-per-pel of the map, and whether the pels are stored in Motorola or Intel format.

Source, pattern, and destination data reside in any of pel maps A, B, or C, determined by the contents of the Pel Operations register.

These maps can be defined as any arbitrary size up to 4096 x 4096 pels. Individual pels within the maps are addressed using X and Y pointers. See “X and Y Pointers” on page 3-97.

Pel maps can be located in video memory and in system memory.

There are two restrictions on map usage: the source and destination maps must have the same number of bits-per-pel, and the pattern map must be 1 bit-per-pel.

Pel Map M (Mask Map)

In addition to the three general purpose maps, the coprocessor defines a mask map. This map is closely related to the destination map. It protects the destination from update on a pel-by-pel basis and can be used to provide a scissoring-type function on any arbitrary shaped area. See "Scissoring with the Mask Map" on page 3-100.

The mask map is described by a set of registers similar to the general purpose pel maps A, B, and C, but it is fixed at 1 bit-per-pel.

The mask map differs from the source, pattern, and destination maps as follows:

- The mask map uses the destination X and Y pointers.
- The position of the mask map origin relative to the destination is defined by the mask map origin X and Y offsets.

See "X and Y Pointers" on page 3-97 for more information.

Map Origin

The origin of a pel map is the point where X = 0 and Y = 0.

The coprocessor defines the origin of all its pel maps as being at the top left corner of the map. The direction of increasing X is to the right; the direction of increasing Y is downward. Figure 3-101 illustrates the X,Y addressing of an XGA map.

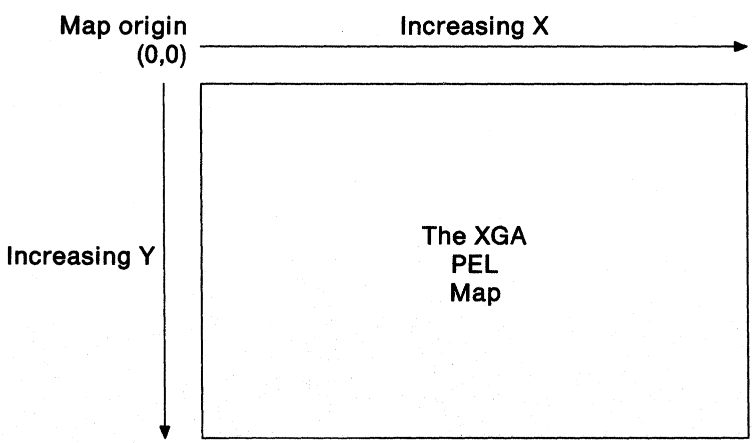


Figure 3-101. XGA Pel Map Origin

In storage, pels to the right of and below the origin are stored in ascending, contiguous memory locations.

X and Y Pointers

The characteristics of X and Y pointers vary depending on the type of pel map.

Source and Pattern Maps

These maps each have X and Y pointers that determine the pel accessed for that map. The two sets of pointers are completely independent, and are modified as the operation proceeds.

If, in the course of an operation, the source or pattern pointers are moved beyond the extremities of the pel map containing the source or pattern data, they are reset to the opposite edge of the pel map. Source and pattern maps can be regarded as continuous, as they wrap around at their extremities. This allows a single operation to repeat a small pattern over a large area in the destination map. This is known as pattern tiling, as shown in Figure 3-102.

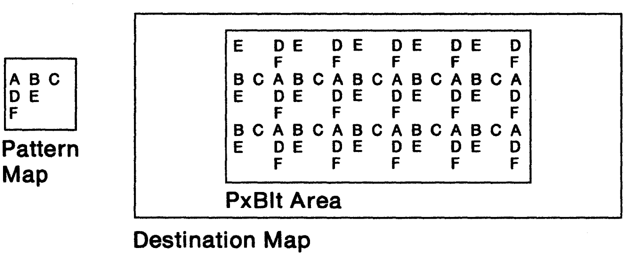


Figure 3-102. Repeating Pattern (Tiling)

Destination Map

If a destination X or Y pointer is moved beyond the extremity of the pel map containing the destination, the pointers are not wrapped. Updates to the destination are disabled until the pointers are moved to within the defined pel map. This mechanism is effectively a fixed scissor window around the destination pel map.

A guardband exists around the destination map to ensure that the destination X and Y pointers do not wrap when they move outside the limits of the map. The guardband is 2048 pels wide on all sides of the largest definable destination map.

The guardband is illustrated in Figure 3-103.

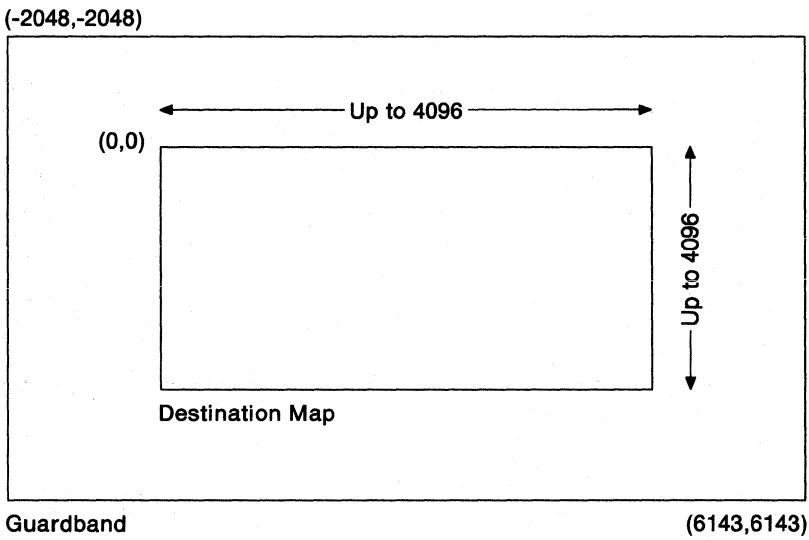


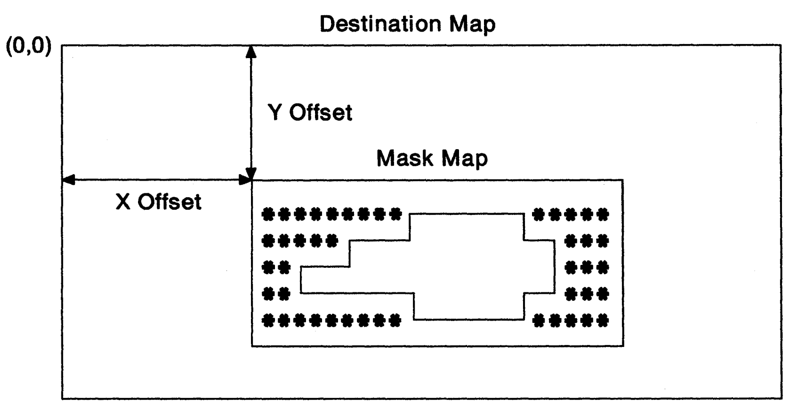
Figure 3-103. Destination Map Guardband

The guardband allows the destination X and Y addresses to range (−2048 to 6143). All pels within the destination map can be updated, but updates to pels within the guardband are inhibited. The size of the destination map is determined by the map width and height, so pels within the range (0,0) to (width − 1, height − 1) can be updated. The guardband occupies Pel X addresses (−2048 to −1), and width to 6143, and Y addresses (−2048 to −1), and height to 6143.

To address the destination map correctly and take advantage of the scissor capability of the coprocessor destination boundary, X and Y destination addresses can be calculated using 16-bit, twos-complement numbers. All X and Y addresses generated by the operation must be within the range (−2048 to 6143), and all pels drawn must be inside the bounds of the destination map. Any X and Y addresses generated that are outside the range (−2048 to 6143) cause the X and Y pointers to wrap and produce erroneous results.

Mask Map

The mask map width and height can be any size less than or equal to the dimensions of the destination map. If the mask map is smaller than the destination map, the hardware needs to know where the mask map is positioned relative to the destination map. Two pointers, the mask map origin X offset and mask map origin Y offset, specify the X,Y position where the mask map origin in the destination is located. The following figure illustrates the use of these pointers.



Note: The Mask Map is forced to have the same map origin as the Destination Map.

Figure 3-104. Mask Map Origin X and Y Offsets

The mask map takes its X and Y pointers from the destination map X and Y pointers. For every pel in the destination map, the corresponding pel in the mask map is read and, depending on the value of the mask pel, update of the destination enabled or disabled.

Scissoring with the Mask Map

Hardware scissoring is provided in the coprocessor using the mask map. The mask map can be used for any operation in three ways, as follows:

Disabled
: Contents of the mask map and boundary position are ignored.

Boundary Enabled
: Contents of the mask map are disabled, but the boundary of the mask map acts as a rectangular scissor window on the destination map. No memory is required to store the map contents in this mode.

Enabled
: Contents of the mask map can be used to provide a nonrectangular scissor window. The boundary of the mask map also provides a rectangular scissor window at the extremities of the mask map.

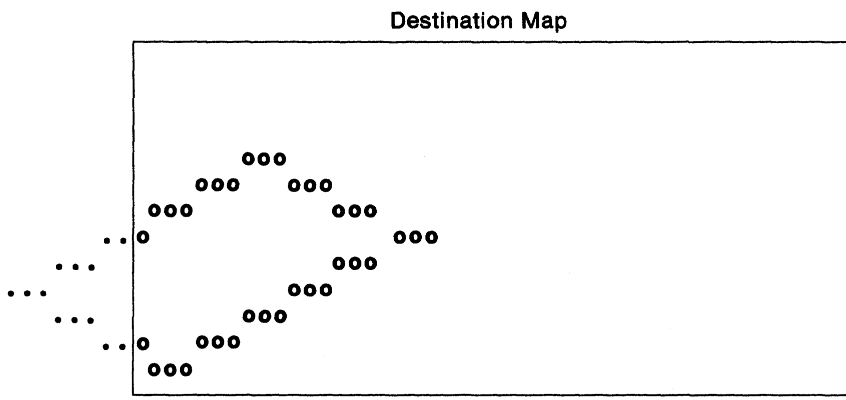
The mask map mode is controlled by a bit in the Pel Operation register. pels located on a scissor boundary are treated as if they are inside it. The modes are described in the following text.

Mask Map Disabled

When the mask map is disabled, updates to the destination are performed regardless of the position or contents of the mask map. No memory must be reserved for the mask map. The contents of the pel map M Base Pointer, Width, Height, and Format registers are ignored.

If the current operation attempts to draw outside the boundary of the destination map, the update is automatically inhibited. The destination X and Y pointers are incremented as usual, but destination update is not enabled until the pointers move back inside the bounds of the destination map. A fixed hardware scissor window then exists around the boundary of the destination map. This destination boundary scissor is enabled regardless of the mask map mode.

Figure 3-105 illustrates the destination boundary scissor operation when the mask map is disabled.



o Indicates a PEL drawn
• Indicates a scissored (not drawn) PEL

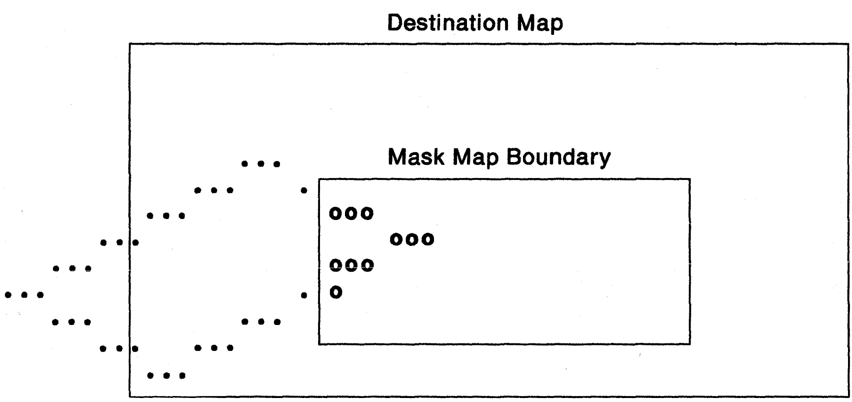
Figure 3-105. Destination Boundary Scissor

Mask Map Boundary Enabled

Mask map boundary enabled mode provides a single rectangular scissor window within the destination map. The contents of the mask map are ignored, so no memory must be reserved for the mask map.

In boundary enabled mode, the size and position of the mask map must be specified. The pel map M Base Pointer, Width, Height, and Format registers must be defined. These four registers define a rectangular boundary within the destination map. Updates to the destination map inside this boundary take place as normal. Updates outside this boundary are inhibited.

The following figure illustrates a mask map boundary enabled scissoring operation.



o Indicates a PEL drawn
• Indicates a scissored (not drawn) PEL

Figure 3-106. Mask Map Boundary Scissor

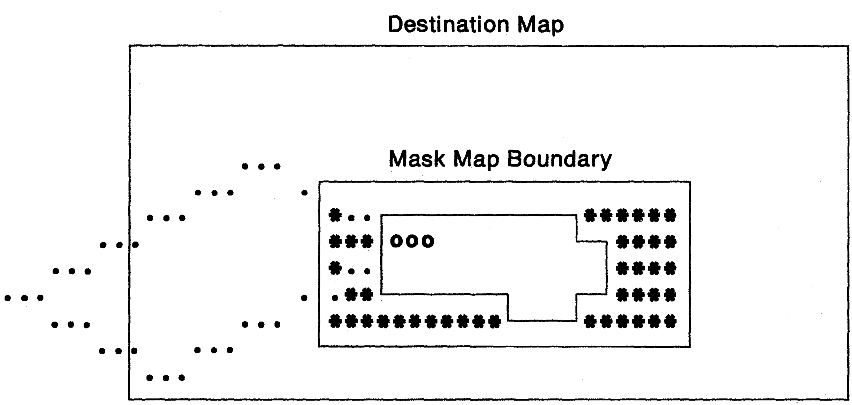
Mask Map Enabled

When the mask map is enabled, both the mask map boundary and contents provide scissoring action. Memory must be reserved to hold the mask map pels. The pel map M Base Pointer, Width, Height, and Format registers must be set up to point to the mask data, describing its size and position relative to the destination map.

Any pel in the destination that is about to be updated has its corresponding mask map pel examined. If the mask pel is inactive (0), the destination pel update is inhibited. If the mask pel is active (1), the destination pel is updated as normal. This mode allows drawing nonrectangular scissor windows in the mask map prior to an operation, then in a single execution of an operation, applying a nonrectangular scissor window to that operation.

Memory must be reserved to hold the mask map contents. The mask data is fixed at 1 bit-per-pel. For a full-screen mask map on a 1024 x 768 pel screen, 96KB of memory are required. If the scissor operation does not cover the whole destination map, a mask map smaller than the destination map can be used to save memory. Applications with no memory available for the mask map contents must use the mask map boundary enabled mode.

The following figure illustrates a mask map enabled scissor operation.



o Indicates a PEL drawn
• Indicates a scissored (not drawn) PEL
* Indicates a '0' in the Mask Map

Figure 3-107. Mask Map Enabled Scissor

Before performing an operation that requires a nonrectangular scissor, the nonrectangular mask into the mask map must be drawn. Windowing systems only permit rectangular windows, so the mask can be drawn using a sequence of PxBlt operations that have fixed source data. For more complex shapes, the line draw and draw and step functions can be used to draw area outlines that can then be filled.

A large number of operations can be performed, all using the same mask, keeping the overhead per operation low in setting up the mask. The use of the mask to perform nonrectangular scissors improves the performance of a given drawing operation over a single rectangular scissor that is provided by the hardware.

Drawing Operations

The coprocessor provides four drawing operations:

- Draw and step
- Line draw
- Pel block transfer (PxBlt)
- Area fill.

The operations can be either one-dimensional or two-dimensional. Draw and step and line draw are one-dimensional while the PxBlts are two-dimensional. Draw and step and line draw are collectively called draw operations in the following text.

Either of the draw operations can be read or write. Qualifiers to the operation are described in "Line Draw" on page 3-109.

Draw and Step

This operation draws a pel at the destination, then updates the X,Y pointers to one of the eight neighbors of the pel according to a 3-bit code.

Up to 15 address steps can be specified in a fixed direction by each draw and step code. An 8-bit code describes the vector, as shown in Figure 3-108.

<table>
<tr><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="3">Direction Code</td><td>M/D</td><td colspan="4">Number of Steps</td></tr>
</table>

Figure 3-108. Draw and Step Code

Number of Steps

This field indicates how many steps are taken, from 0 to 15. The X,Y pointers are updated after the pel is drawn, so a draw and step function always attempts to draw at least one pel.

The number of steps taken in the draw and step operation is one less than the number of pels that the hardware attempts to draw. When the number of steps is programmed to five, six pels are drawn; when zero steps are specified, one pel is drawn. After the draw and step operation, the X,Y pointers point to the last pel that the operation attempted to draw (this pel might not be drawn if the last pel null drawing mode is active).

For example, a draw and step code of hex 35 moves X,Y pointers starting at coordinates (17,10) to coordinates (22,5), as shown in Figure 3-109.

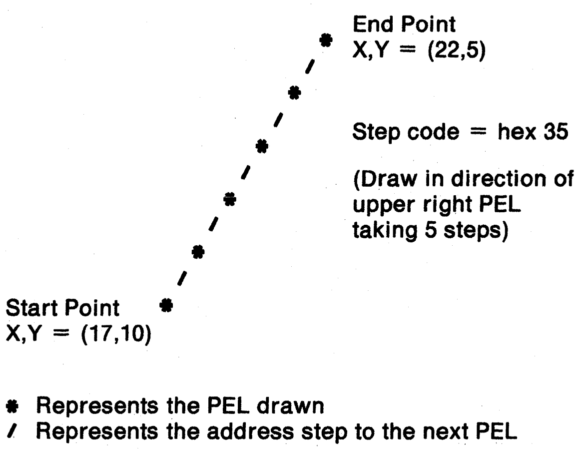


Figure 3-109. Draw and Step Example

M/D

This field specifies if the current operation is a move operation or a draw operation. When set to 1, pels are drawn. When set to 0, X and Y pointers are modified as usual, but no pels are drawn.

Direction Code

This field indicates the direction of drawing relative to the current pel, as shown in Figure 3-110.

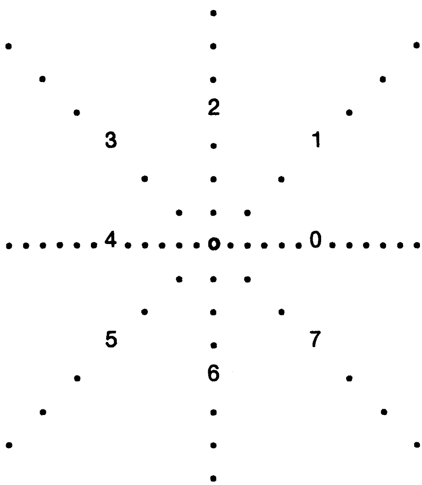


Figure 3-110. Draw and Step Direction Codes

Draw and step codes must be written to the Direction Step register. Each write to the register can load up to four draw and step codes in one access. The draw and step codes are executed starting with the least significant byte. Each group of up to four codes written to the Direction Step register is treated as one operation. All codes are executed before the coprocessor indicates that the operation is complete. However, for the purposes of first and last pel null drawing, each code describes a distinct line.

The draw and step operation differs from other operations because it is not initiated through the pel Operations register. Writing a draw and step code to the most significant byte of the Direction Step register initiates the draw and step operation.

Before any data is written to the Direction Step register, the Pel Operation register must be loaded to specify the particular draw and step function and the data flow for the operation. Writing the Pel Operation register with a function of draw and step does not initiate a draw and step operation, but sets up the parameters for the operation. Writing steps to the Direction Step register initiates the draw and step operation. If the Pel Operation register specifies a function other than draw and step when the Direction Step register is written, no operation takes place.

The XGA treats a draw and step code of 00 as a stop code. If a stop code is encountered as one of the four codes in the Direction Step register, the draw and step operation completes after that code has been executed. The completion of the operation is indicated in the normal way through the Coprocessor Control register. The coprocessor busy bit in the Coprocessor Control register indicates the operation has completed because a stop code was encountered. This mechanism allows software to load sequences of draw and step codes to the coprocessor without monitoring the number of codes that make up the figure being drawn.

There are two ways to program fewer than four codes to the Direction Step register. The first unwanted step code can be set to 00 (stop) and all 32 bits of the register written, or only the required number of codes can be written to the Direction Step register. In the latter case, the codes must be written to the most significant bytes of the register. The two methods are shown in Figure 3-111.

Writing 32 bits and using the stop code:

31			0
Don't care	Stop code 00	Code 2	Code 1

Writing only those codes required:

31			0
Code 2	Code 1	Not written	Not written

Note: The figure shows the case when only two Step codes are required. The second method requires the I/O address programmed to change depending on the number of steps written.

Figure 3-111. Programming Fewer Than Four Step Codes

Line Draw

The line draw function uses the Bresenham line drawing algorithm to draw a line of pels into the destination. The Bresenham line drawing algorithm operates with all parameters normalized to the first octant (octant 0). The octant code for the octant in which the line lies must be specified in the Octant field of the Pel Operation register. This contains a 3-bit code made up of three 1-bit flags called DX, DY, and DZ.

DX is 1 for negative X direction, 0 for positive X direction
DY is 1 for negative Y direction, 0 for positive Y direction
DZ is 1 for $|X| \leq |Y|$, 0 for $|X| > |Y|$ ($|X|$ is the magnitude of X, the value ignoring the sign)

The Octant field is formed by concatenating DX, DY, and DZ. See Figure 3-167 on page 3-162 for octant bit value assignments.

Figure 3-112 shows the encoding of octants.

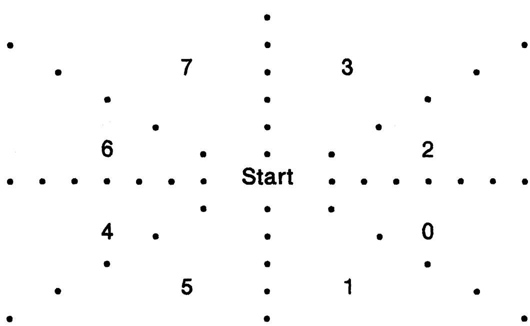


Figure 3-112. Bresenham Line Draw Octant Encoding

The length of the line (delta X when normalized) must be specified in the Operation Dimension 1 register.

The coprocessor provides the following registers to control the draw line address stepping:

- Bresenham Error Term E = 2 × deltaY − deltaX
- Bresenham Constant K1 = 2 × deltaY
- Bresenham Constant K2 = 2 × (deltaY − deltaX).

When the drawing operation has completed, X and Y pointers point at the last pel of the line.

The coprocessor draw operations that take source data from a pel map apply the specified address update to either the source or destination map. The X,Y address in the other map is always incremented in X only. There are two possible draw operations, Read Draw and Write Draw.

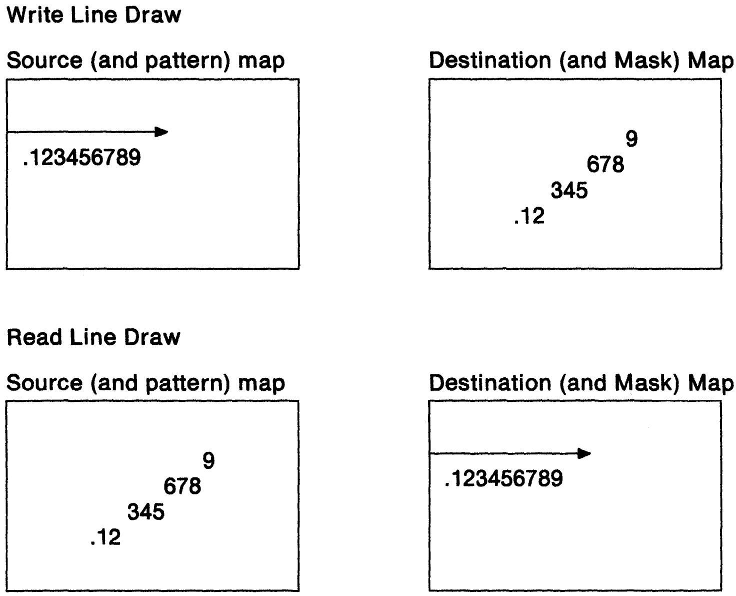
Write Draw After every pel drawn, the source X,Y pointers are incremented in X only. The destination X,Y pointers are updated according to the current function specified (Bresenham line draw or draw and step).

Read Draw After every pel drawn, the source X,Y pointers are updated according to the current function specified (Bresenham line draw or draw and step). The destination X,Y pointers are incremented in X only.

The read and write in the terms read draw and write draw refer to the direction of data transfer of the map having its addresses updated by the specified function. During a read line draw, the map where data is read (the source) has its addresses updated by the Bresenham line draw function. During a write draw and step, the map where data is written (the destination) has its addresses updated by the draw and step function.

Note: To draw a fixed color line (by taking the source from the Foreground Color or Background Color register), a write draw function must be used.

Figure 3-113 on page 3-111 illustrates the stepping of X and Y pointers during a read line draw and write line draw.



The numbers 1 to 9 denote each PEL in order of drawing.

Figure 3-113. Memory-to-Memory Line Draw Address Stepping

In the map that is not having the current addressing function applied, the X pointer is always incremented regardless of the direction of X in the current addressing function. The Y pointer for the same map is not updated during the operation.

The above description refers to the source and destination maps. The pattern map X and Y pointers are updated in the same way as the source pointers. The mask map X and Y pointers (that are not directly accessible), are updated in the same manner as the destination pointers.

If an attempt is made to move any of the map pointers outside the bounds of their current map, the rules set out in "X and Y Pointers" on page 3-97 apply: the source and pattern pointers wrap, and the mask and destination scissor. To draw a line with a repeating color scheme and pattern, the source map width and pattern map width must be set to the required run-length of the repeating colors and pattern respectively. The coprocessor automatically draws the repeating run of colors and pattern. Conversely, if a line with a long nonrepeating color scheme or pattern is required, the source and pattern map widths must be set equal to, or greater than, the line length, otherwise wrapping occurs.

Drawing with Null Endpoint Pels: It is common to draw a series of lines, one after the other, with the endpoint of one line being the starting point of the next line. Such composite lines are called polylines. A problem can arise because the common endpoint of the two abutting lines is drawn twice, once as the last pel of the first line, and once as the first pel of the second line. If a mix of XOR is active, the common pel is drawn and removed. Similar problems arise with different mixes.

To avoid drawing the endpoints of polylines twice, the coprocessor provides functions that inhibit the drawing of the end pel. Depending on the function selected, either the first pel or the last pel of individual lines is not drawn (drawn null). The choice of whether to draw first or last pel null is arbitrary, as long as one or the other is used for the whole figure being drawn. It is usually a convention of the graphics application whether first or last pel null is used.

First and last pel null drawing functions are provided for both the Bresenham line draw function and the draw and step function. In all cases, the programming of parameters is the same as that for normal line draw, and for draw and step. Only the contents of the Drawing Mode field in the Pel Operations register are different.

Area Boundary Drawing: The outline of an object is drawn using Bresenham line draw, draw and step functions, or a combination of the two. The outline is created by observing the following rules.

- If a line is drawn from screen top-to-bottom, draw with last pel null and draw only the last pel in every horizontal run of pels.
- If a line is drawn from screen bottom-to-top, draw with first pel null, and draw only the first pel in every horizontal run of pels.
- If a line is horizontal, draw none of the pels.
- Always draw with a mix of XOR.

The coprocessor implements these drawing rules in hardware. A shape drawn as an area outline must be drawn as a normal line draw or draw and step operation, with the draw area boundary drawing mode selected in the Pel Operation register and a mix of XOR.

Area Outline Scissoring: It is important during area outline drawing to ensure that the correct outline is drawn when the outline intersects the scissor boundary. In particular, when the outline is scissored by a vertical boundary at the left of a map, a pel is drawn in the outline to activate filling at that boundary.

Using the combination of mask map and fixed destination boundary scissoring available in XGA, area outlines are incorrectly scissored by the mask map, but correctly scissored by destination map boundary scissoring. The correct area can be filled by ensuring that the mask map scissoring is disabled when the outline is drawn and enabled or boundary enabled when the scan/fill part of the area fill is drawn. This results in the correct, scissored figure being drawn. See "Scissoring with the Mask Map" on page 3-100.

Pel Block Transfer (PxBlt)

The PxBlt function transfers a rectangular block of pels from the source to the destination. The width and height of the rectangle are specified in the Operation Dimension 1 and Operation Dimension 2 registers. The transfer can be programmed to start at any of the four corners of the rectangle, and proceed toward the diagonally opposite corner. The address is stepped in the X direction until the edge of the rectangle is encountered, X is reset and the Y direction is stepped. This process is repeated until the entire rectangle is transferred.

PxBlts can be implemented in normal write mode or in read/modify/write mode, depending on the number of bits-per-pel and the mix being used.

If the PxBlt is being implemented in read/modify/write mode (that is, 1, 2, or 4 bits-per-pel with *any* mix, or 8 or 16 bits-per-pel with a read/modify/write mix), do one of the following:

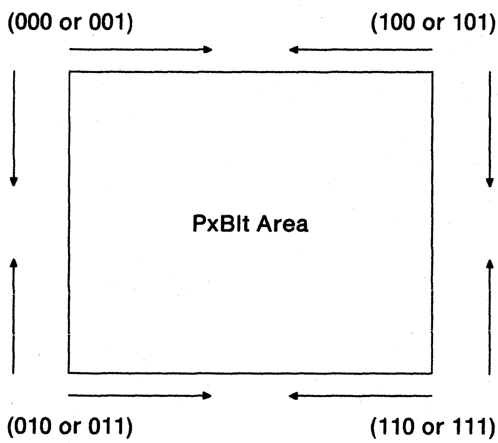
- Ensure that the destination map has a base address that is on a doubleword (4 byte) address boundary, and is an exact number of doublewords wide.
- If the destination map is not doubleword aligned, ensure that the destination map boundary is not crossed during the PxBlt operation.

PxBlt Direction: The PxBlt direction indicates in which direction the X,Y address is stepped across the rectangle. It also defines the starting corner of the transfer. This is significant if the destination rectangle overlaps the source rectangle, so the PxBlt direction must be programmed correctly to achieve the required result.

The direction octant bits in the Pel Operations register determine the direction that the PxBlt is drawn in.

The encoding is as follows:

binary 000 or 001	Start at top left corner of area, increasing right and down.
binary 100 or 101	Start at top right corner of area, increasing left and down.
binary 010 or 011	Start at bottom left corner of area, increasing right and up.
binary 110 or 111	Start at bottom right corner of area, increasing left and up.



Note: Numbers are binary.

Figure 3-114. PxBlt Direction Codes

After a PxBlt operation has completed, the X and Y pointers are set so the X pointer contains its original value at the start of the PxBlt and the Y pointer points to its value on the last line of the PxBlt plus or minus 1, depending on the Y direction that the PxBlt was programmed.

See "Overlapping PxBlts" on page 3-257 for details on PxBlts where the source and destinations overlap.

Inverting PxBlt: As detailed in "Map Origin" on page 3-96, the coprocessor assumes that the origin of a pel map is at the top left corner of the map, with Y increasing downward. Applications that use an origin at the bottom left of the map (Y increasing upward) use either of the following:

- Modify all Y coordinates by subtracting the map height from them before passing the modified coordinates to the display hardware.
- Use the coprocessor inverting PxBlt operation.

Inverting PxBlt use requires the application to draw into an off-screen pel map without any Y coordinate modification, and then use the inverting PxBlt operation to move the data to the destination map.

Figure 3-115 on page 3-116 illustrates the X,Y addressing of the inverting PxBlt operation, and shows how the result of the inverting PxBlt appears the same as the original when displayed as an inverted pel map (that is, with the origin at the bottom left).

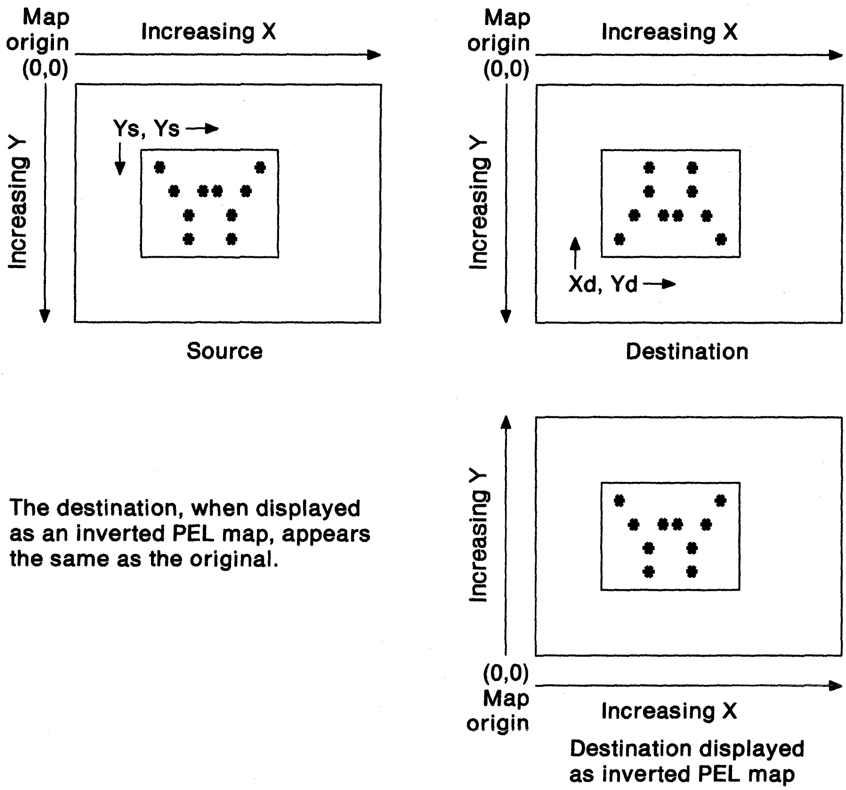


Figure 3-115. Inverting PxBlt

An inverting PxBlt is set up in the same manner as a standard PxBlt with the following notes:

- The PxBlt direction set applies to the updating of the source X and Y addresses.
- The destination Y pointer must be programmed to the opposite (in Y) corner of the destination rectangle.
- The Function field in the Pel Operation register must be set to inverting PxBlt as opposed to PxBlt.

See "Overlapping PxBlts" on page 3-257 for details on PxBlts when the source and destinations overlap.

Area Fill

The following steps are required to perform an area fill operation without a user pattern:

1. Draw the closed outline of the area to be filled using the area boundary drawing mode. Typically, a unique, off-screen pel map would be defined to draw the area boundary into. This pel map must be initialized to contain 0-value pels before the boundary is drawn. This pel map must be in a 1 bit-per-pel format.
2. Designate the pel map where the area boundary was drawn as the pattern map.
3. Specify the desired destination.
4. Select the desired foreground mix and source.
5. Specify the background mix as Destination (code 5).
6. Specify the operation direction as any direction with X increasing (Codes 0 or 1, 2 or 3), because the pattern data is scanned from left to right. Selection of a negative X direction code for area fill operations results in fill errors.
7. Initiate the area fill operation.

During the area fill operation, the coprocessor applies a filling function to the pattern pels before they are used to select background and foreground sources and mixes in the usual way. The filling function modifies the pattern pels horizontally line by line. It scans the pattern from left to right, and when encountering the first foreground (1) pel, sets all subsequent pels to foreground (1) until the next foreground pel is encountered.

This process is illustrated in Figure 3-116.

Pattern scanned to the right ⟶

Original Pattern   0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 1 0 1 0 0

Filling function ↓

Filled Pattern   0 0 0 0 1 1 1 1 1 1 1 1 1 0 0 0 1 1 0 0 0

Figure 3-116. Pattern Filling

The filled pattern is generated internally to the coprocessor. It is then used exactly as the pattern in any normal operation, with foreground (1) pels selecting foreground source and mix, and background (0) pels selecting background source and mix. During area fill operations, it is required to fill the specified area and leave all other pels unchanged. This is why the background mix was specified to be the destination. Figure 3-100 on page 3-92 shows the position of the pattern-filling circuitry in the coprocessor data flow.

Area fill operations that require a pattern fill must be performed in two stages. This is because area fill PxBlt operations use the pattern map to perform the area fill function and cannot include a user pattern in a single operation. However, by first combining the contents of the mask map with a mask of the filled area, a full pattern PxBlt of an area can be achieved as follows:

1. Both the pattern map and the destination map must be defined as the map containing the previously drawn area boundary. The source map must be defined as the map that would normally supply the mask map for the operation. The mask map facility must be disabled. An area fill PxBlt must be performed with the following conditions:

   - Foreground source = source pel map
   - Foreground mix = Source (code 3)
   - Background source = background color
   - Background color = 0
   - Background mix = Source (code 3)

   Note: All the maps in this operation must be 1 bit-per-pel.

   This operation combines the mask data for the pattern area fill with a mask of the filled area.

2. A second non-area fill PxBlt must be performed with the combined mask generated in step 1 defined as the mask map. All other maps can be used as normal with no restrictions.

Logical and Arithmetic Functions

During an operation in the coprocessor, source data is combined with destination data, under the control of pattern data, and the result is written back to the destination. Mask data can be included in the operation to selectively inhibit updating of destination data.

Source data can be either foreground source or background source on a pel-by-pel basis. The foreground source is combined with the destination using the foreground mix; the background source is combined with the destination using the background mix. The pattern determines if the source and mix are foreground or background for a particular pel. If the pattern pel is 1, source and mix are foreground; if it is 0, they are background.

The foreground and background sources can each be either a fixed color over the whole operation or pel data taken from the source pel map. The background source and foreground source bits in the Pel Operations register determine whether fixed colors or source pel map data is used in an operation. The fixed color that is used as the foreground source is called the foreground color, and is stored in the Foreground Color register. The fixed color that is used as the background source is called the background color, and is stored in the Background Color register.

The possible combinations of source, destination and pattern are shown below:

- Pattern pel = 1 (foreground source)
  - New destination pel = old destination pel **Fgd OP** Foreground color
  - New destination pel = old destination pel **Fgd OP** pel map source.
- Pattern pel = 0 (background source)
  - New destination pel = old destination pel **Bgd OP** Background color
  - New destination pel = old destination pel **Bgd OP** pel map source.

Fgd OP is the logical or arithmetic function specified in the Foreground Mix register. **Bgd OP** is the logical or arithmetic function specified in the Background Mix register.

These operations can be inhibited by the contents of the mask map. If the mask pel is 0, the destination pel is not modified. If the mask pel is 1, the selected operation is applied to the destination pel.

Mixes

The foreground and background mixes provided by the XGA are independent. The XGA provides all logical mixes of two operands and six arithmetic mixes. The mixes provided are as follows:

Code (hex)	Function
00	Zeros
01	Source AND Destination
02	Source AND NOT Destination
03	Source
04	NOT Source AND Destination
05	Destination
06	Source XOR Destination
07	Source OR Destination
08	NOT Source AND NOT Destination
09	Source XOR NOT Destination
0A	NOT Destination
0B	Source OR NOT Destination
0C	NOT Source
0D	NOT Source OR Destination
0E	NOT Source OR NOT Destination
0F	Ones
10	Maximum
11	Minimum
12	Add With Saturate
13	Subtract (Destination – Source) With Saturate
14	Subtract (Source – Destination) With Saturate
15	Average

Note: Mix codes hex 16 to FF are reserved.

Figure 3-117. Foreground and Background Mixes

Saturate means that if the result of an arithmetic operation is greater than all 1's, the final result remains all 1's. If the result of an arithmetic operation is less than 0, the final result remains at 0.

Breaking the Coprocessor Carry Chain

To limit the operation of the coprocessor to certain bits in a pel (for example, to perform an operation on both the upper and lower 4 bits of an 8-bit pel independently), it is not desirable for the arithmetic operations to propagate a carry from one group of bits in the pel to the next. One solution is to use the XGA pel bit mask to ensure only one component of the pel is processed at a time. The disadvantage of this technique is that the operation must be repeated once for each component in the pel.

The XGA provides an alternative mechanism that allows pels with component fields to process correctly in one pass. A carry chain mask can be specified that determines how carry bits are propagated in the coprocessor. By loading the appropriate mask in the Carry Chain Mask register before performing an operation involving an arithmetic operation or color compare, the pel is effectively divided into independent fields. The mask prevents the coprocessor carry being propagated across the field boundaries.

Each bit in the mask enables or disables the propagation of the carry from the corresponding bit in the coprocessor to its more significant neighbor. The mask is n − 1 bits wide for a pel n bits wide, and the carry from the most significant bit of the coprocessor is not propagated.

An example for a carry chain mask for an 8-bit pel with two 4-bit fields follows:

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

Figure 3-118. Carry Chain Mask for an 8-Bit Pel

Bits outside the required mask size for a given pel size need not be written in the register.

Generating the Pattern from the Source

Pattern data for an operation can be supplied by pel maps A, B, or C, or it can be fixed to 1 (foreground source) throughout the operation. Pattern data can also be internally generated by the coprocessor from source pel map data. A comparison operation is performed on each source pel and the pattern data is generated depending on the result.

The comparison operation compares the source pel to 0. For any source pel with a value of 0, a 0 (background) pattern pel is generated. For any nonzero source pel, a 1 (foreground) pattern pel is generated. The internally generated pattern is then used to select between foreground and background sources, and mixes in the usual way. When the pattern is internally generated, the coprocessor ignores the pattern pel map contents.

This capability allows the background source data and mix to be forced for all 0 value pels in the source. In particular, it provides a transparency function, where a multibit character can be drawn onto a destination with the destination data showing through any 0 (black) pel in the source character definition.

Color Expansion

If the source pels for an operation have fewer bits per pel than the destination pels, the source pels must be expanded to the same size as those in the destination before they are combined. This process is referred to as color expansion.

The major use of color expansion is to draw 1-bit-per-pel character sets on n-bits-per-pel destinations. The coprocessor performs this function in hardware, but does not have a color expansion look-up table. Instead, the 1-bit-per-pel character map must be defined as the pattern map. The Pel Operation register must be programmed to use the Foreground Color and Background Color registers, not the source map. The Foreground Color and Background Color registers then act as a two-entry color expansion look-up table, and the character map is expanded to the number of bits per pel in the destination.

Pel Bit Masking

The pel bit mask allows any combination of bits in a destination pel to be protected from update (being written). A mask value must be loaded in the Pel Bit Mask register to enable or disable updating of pel bits selectively as required.

This mask is the same as the plane mask in subsystems that are plane oriented, as opposed to packed-pel.

When the destination bits-per-pel is less than 8 bits, only the low order bits of the Pel Bit Mask register are significant.

A bit that is not write enabled is prevented from affecting arithmetic or compare operations. In effect, masked bits are completely excluded from the operation or comparison.

Color Compare

The value that the destination pels are compared with is stored in the Destination Color Compare Value register. The Destination Color Compare Condition register indicates the condition when the destination update is inhibited. The possible conditions are as follows:

Condition Code	Condition
0	Always True (disable update)
1	Destination Data > Color Compare Value
2	Destination Data = Color Compare Value
3	Destination Data < Color Compare Value
4	Always False (enable update)
5	Destination Data > = Color Compare Value
6	Destination Data < > Color Compare Value
7	Destination Data < = Color Compare Value

Figure 3-119. Color Compare Conditions

Note: A comparison result of true prevents update to the destination.

Controlling Coprocessor Operations

Starting a Coprocessor Operation

Coprocessor operations are started by writing the most significant byte of the Pel Operations register. One exception to this is the draw and step function. For details, see "Draw and Step" on page 3-105.

Suspending a Coprocessor Operation

Coprocessor operations can be suspended before they have completed. The state of the coprocessor, including internal register contents, can then be read rapidly to allow task state saving. A previous task can be restored through the same data port and the restored operation can be restarted.

The suspend operation bits in the Coprocessor Control register are used to suspend and restart coprocessor operations.

If a coprocessor operation is suspended, a terminate operation is required before starting a new coprocessor operation.

Terminating a Coprocessor Operation

Operations can be terminated before they have completed. The state of the coprocessor registers that are updated as the operation proceeds is undefined after the operation is terminated and their contents must not be relied upon. "Coprocessor Registers" on page 3-128 details the registers that are updated as an operation proceeds.

The terminate operation bit in the Coprocessor Control register is used to terminate operations.

Coprocessor Operation Completion

There are two methods for the system microprocessor to detect the completion of a coprocessor operation:

- Receive an operation-complete interrupt from the XGA.
- Poll the coprocessor-busy bit in the Coprocessor Control register or (on the XGA-2 subsystem) poll the auxiliary coprocessor-busy bit in the Auxiliary Coprocessor Status register.

Coprocessor-Operation-Complete Interrupt

The coprocessor provides an operation-complete interrupt that can interrupt the system on completion of an operation. The interrupt is enabled by a bit in the Interrupt Enable register and its status is indicated by a bit in the Interrupt Status register. See "Interrupt Enable Register (Address 21x4)" on page 3-38 and "Interrupt Status Register (Address 21x5)" on page 3-40 for bit locations.

Regardless of the state of the operation-complete interrupt enable bit, the status bit is always set to 1 on completion of an operation. The application must ensure that this bit is reset before starting an operation. This is done by writing a 1 back to the status bit.

If the interrupt enable bit is 1, the completion of an operation not only sets the interrupt status bit, but also causes an interrupt. The system microprocessor must reset the interrupt by writing a 1 back to the status bit after servicing the interrupt.

Coprocessor Busy Bit

The coprocessor-busy bit in the Coprocessor Control register, or (on the XGA-2 subsystem) the auxiliary coprocessor-busy bit in the Auxiliary Coprocessor Status register, indicates if the coprocessor is executing an operation. It is set to 1 by the hardware when the coprocessor is executing an operation and reset to 0 when the operation completes. Applications can read this bit to determine if the coprocessor is busy. See "Waiting for Hardware Not Busy" on page 3-256 for more information.

Accesses to the Coprocessor During an Operation

When the coprocessor is executing an operation, the system processor can only perform read accesses to the coprocessor registers. Write accesses are not permitted because they could damage operation data.

If the system processor attempts to write data to the coprocessor registers during an operation, the coprocessor allows the access to complete, but the executing operation might be damaged. The coprocessor interrupts the system microprocessor to indicate that a write access occurred during an active operation and the operation might have been damaged. This interrupt is called the coprocessor-access-rejected interrupt. An enable bit is in the Interrupt Enable register and a status bit is in the Interrupt Status register. See "Interrupt Enable Register (Address 21x4)" on page 3-38 and "Interrupt Status Register (Address 21x5)" on page 3-40 for bit locations.

There is one exception to this rule. The Coprocessor Control register can be written during an operation without damaging the operation. See "Coprocessor Control Register (Offset 11)" on page 3-134 for more information.

Coprocessor State Save/Restore

When operating in a multitasking environment it is necessary to save and restore the state of the display hardware when switching tasks.

Sometimes a task switch is required when the coprocessor is executing an operation. The contents of registers that are visible to the system microprocessor and the contents of internal registers (the state of the coprocessor) must be saved, and later, restored. The coprocessor has special hardware that lets it suspend the execution of an operation and efficiently save and restore task states.

Suspending Coprocessor Operations

At any time during the execution of a coprocessor operation, the operation can be suspended by writing to a bit in the Coprocessor Control register. Any executing memory cycle is completed before the coprocessor suspends the operation. The system can then save and restore the coprocessor contents and restart the restored operation by clearing the bit in the Coprocessor Control register.

If a coprocessor operation is suspended, a terminate operation is required before starting a new coprocessor operation.

Save/Restore Mechanism

The coprocessor provides two special 32-bit save/restore data ports. All the coprocessor state data passes through these ports when the state is being saved or restored. The number of doublewords read or written is determined by two read-only registers (State Length registers A and B). The amount of data saved or restored is less than 1KB. State-saving software must perform string I/O read instructions, reading data from the two save/restore data ports in turn. The coprocessor hardware automatically provides successive doublewords of data on successive reads. After the state has been saved, the coprocessor is in a reset state.

If a coprocessor operation is suspended, a terminate operation is required before starting a new coprocessor operation.

Restoring the state of the coprocessor uses a similar process. State data must be moved back into the coprocessor using string I/O write instructions. The state data must be written back into the coprocessor in the same order as it was read (first out, first in).

The exact number of doublewords specified in the State Length registers must be read or written when saving or restoring the coprocessor state. Failure to do this leaves the coprocessor in an indeterminate state.

Coprocessor Registers

"XGA Subsystem Identification, Location, and XGA Mode Setting" on page 3-185 provides details of locating and using these registers.

The XGA coprocessor supports two register interface formats. The type of interface required (Intel or Motorola) is set when selecting Extended Graphics mode in the Operating Mode register.

The difference between Intel and Motorola formats is that, with two exceptions, the bytes within each 4 bytes of register space are reversed (byte 0 becomes byte 3). The two exceptions are the Direction Steps register and the Pel Operations register. The bytes within these registers are not reversed because the existing byte order is required by the operation being performed.

Most of the coprocessor registers are not directly readable by the system microprocessor. Registers that cannot be read directly can be read indirectly using the coprocessor state save and restore mechanism. See "Save/Restore Mechanism" on page 3-127 for more information. Only the following registers are readable directly by the system microprocessor:

- State Save/Restore Data Ports register
- State Length registers
- Coprocessor Control register
- Virtual Memory Control register
- Virtual Memory Interrupt Status register
- Current Virtual Address register
- Bresenham Error Term E register
- Source X Address and Source Y Address registers
- Pattern X Address and Pattern Y Address registers
- Destination X Address and Destination Y Address registers

The contents of most coprocessor registers are not changed during a coprocessor operation and therefore do not need to have their contents reloaded before starting another similar operation. The registers with contents that change during an operation are:

Bresenham Error Term E
The error term is updated throughout line draw operations.

Source X Address and Source Y Address
Any operation that uses the source pel map updates these pointers.

Pattern X Address and Pattern Y Address
The pattern map X and Y pointers are updated during any operation that does not have the Pattern field in the Pel Operation register set to foreground.

Destination X Address and Destination Y Address
The destination map X and Y pointers are updated during all operations.

The following figures show the coprocessor register space in Intel and Motorola formats.

Note: In these two figures, all unused and undefined offsets are reserved.

<table>
<tr><th colspan="5">Coprocessor Address Space</th></tr>
<tr><th>Byte 3</th><th>Byte 2</th><th>Byte 1</th><th>Byte 0</th><th></th></tr>
<tr><td colspan="4">Page Directory Base Address</td><td>0</td></tr>
<tr><td colspan="4">Current Virtual Address</td><td>4</td></tr>
<tr><td colspan="2"></td><td>Auxiliary Coprocessor Status</td><td></td><td>8</td></tr>
<tr><td colspan="2"></td><td>State B length</td><td>State A length</td><td>C</td></tr>
<tr><td></td><td>Pel Map Index</td><td>Coprocessor Control</td><td></td><td>10</td></tr>
<tr><td colspan="4">Pel Map n Base Pointer</td><td>14</td></tr>
<tr><td colspan="2">Pel Map n Height</td><td colspan="2">Pel Map n Width</td><td>18</td></tr>
<tr><td colspan="3"></td><td>Pel Map n Format</td><td>1C</td></tr>
<tr><td colspan="2"></td><td colspan="2">Bresenham Error Term</td><td>20</td></tr>
<tr><td colspan="2"></td><td colspan="2">Bresenham K1</td><td>24</td></tr>
<tr><td colspan="2"></td><td colspan="2">Bresenham K2</td><td>28</td></tr>
<tr><td colspan="4">Direction Steps</td><td>2C</td></tr>
<tr><td colspan="4"></td><td>30</td></tr>
<tr><td colspan="4"></td><td>34</td></tr>
<tr><td colspan="4"></td><td>38</td></tr>
<tr><td colspan="4"></td><td>3C</td></tr>
<tr><td colspan="4"></td><td>40</td></tr>
<tr><td colspan="4"></td><td>44</td></tr>
<tr><td></td><td>Destination Color Compare Condition</td><td>Background Mix</td><td>Foreground Mix</td><td>48</td></tr>
<tr><td colspan="4">Destination Color Compare Value</td><td>4C</td></tr>
<tr><td colspan="4">Pel Bit Mask</td><td>50</td></tr>
<tr><td colspan="4">Carry Chain Mask</td><td>54</td></tr>
<tr><td colspan="4">Foreground Color Register</td><td>58</td></tr>
<tr><td colspan="4">Background Color Register</td><td>5C</td></tr>
<tr><td colspan="2">Operation Dimension 2</td><td colspan="2">Operation Dimension 1</td><td>60</td></tr>
<tr><td colspan="4"></td><td>64</td></tr>
<tr><td colspan="4"></td><td>68</td></tr>
<tr><td colspan="2">Mask Map Origin Y Offset</td><td colspan="2">Mask Map Origin X Offset</td><td>6C</td></tr>
<tr><td colspan="2">Source Map Y Address</td><td colspan="2">Source Map X Address</td><td>70</td></tr>
<tr><td colspan="2">Pattern Map Y Address</td><td colspan="2">Pattern Map X Address</td><td>74</td></tr>
<tr><td colspan="2">Destination Map Y Address</td><td colspan="2">Destination Map X Address</td><td>78</td></tr>
<tr><td colspan="4">Pel Operation</td><td>7C</td></tr>
</table>

Figure 3-120. XGA Coprocessor Register Space, Intel Format

Coprocessor Address Space				
Byte 0	Byte 1	Byte 2	Byte 3	
Page Directory Base Address				0
Current Virtual Address				4
		Auxiliary Coprocessor Status		8
		State B length	State A length	C
	Pel Map Index	Coprocessor Control		10
Pel Map n Base Pointer				14
Pel Map n Height		Pel Map n Width		18
			Pel Map n Format	1C
		Bresenham Error Term		20
		Bresenham K1		24
		Bresenham K2		28
Direction Steps				2C
				30
				34
				38
				3C
				40
				44
	Destination Color Compare Condition	Background Mix	Foreground Mix	48
Destination Color Compare Value				4C
Pel Bit Mask				50
Carry Chain Mask				54
Foreground Color Register				58
Background Color Register				5C
Operation Dimension 2		Operation Dimension 1		60
				64
				68
Mask Map Origin Y Offset		Mask Map Origin X Offset		6C
Source Map Y Address		Source Map X Address		70
Pattern Map Y Address		Pattern Map X Address		74
Destination Map Y Address		Destination Map X Address		78
Pel Operation				7C

Figure 3-121. XGA Coprocessor Register Space, Motorola Format

Register Usage Guidelines

Unless otherwise stated,the following are guidelines to be used when accessing the coprocessor registers:

- Special reserved register bits must be used as follows:
  - Register bits marked with '–' must be set to 0. These bits are undefined when read, and should be masked off if the contents of the register are to be tested.
  - Register bits marked with '#' are reserved and the state of these bits must be preserved. When writing the register, read the register first and change only the bits that must be changed.
- Unspecified registers or registers marked as reserved in the XGA coprocessor address space are reserved. They must not be written to or read from.
- During a read, the values returned from write-only registers are reserved and unspecified.
- The contents of read-only registers must not be modified.
- Counters must not be relied upon to wrap from the high value to the low value.
- Register fields defined with valid ranges must not be loaded with a value outside the specified range.
- Register field values defined as reserved must not be written.

The following sections describe the coprocessor registers in detail. Unless stated otherwise, the register definitions are in Intel format.

Virtual Memory Registers

The XGA coprocessor virtual memory implementation is given in "Virtual Memory Description" on page 3-170.

State Save/Restore Registers

The following registers allow the internal state of the coprocessor to be saved and restored. "Coprocessor State Save/Restore" on page 3-126 describes this mechanism.

Auxiliary Coprocessor Status Register (Offset 09)

This read-only register has an offset of hex 09. It is only available for use on the XGA-2 subsystem. The Auxiliary Coprocessor Status register indicates if the coprocessor is currently executing an operation.

7	6	5	4	3	2	1	0
ABSY	-	-	-	-	-	-	-

```
   - : Undefined on Read
ABSY : Auxiliary Coprocessor Busy
```

Figure 3-122. Auxiliary Coprocessor Status Register, Offset Hex 09

The register fields are defined as follows:

ABSY When the Auxiliary Coprocessor Busy field (bit 7) is read as 1, the coprocessor is currently executing an operation. When read as 0, the coprocessor is idle. This bit provides the same information as the BSY field of "Coprocessor Control Register (Offset 11)" on page 3-134. However, it should be used in preference to the BSY field when operating on an XGA-2 subsystem because it might increase coprocessor performance. See "Waiting for Hardware Not Busy" on page 3-256

Coprocessor Control Register (Offset 11)

This read/write register has an offset of hex 11. The Coprocessor Control register indicates if the coprocessor is currently executing an operation. The current coprocessor operation can be terminated or suspended by writing to this register.

7	6	5	4	3	2	1	0	
–	–	TO	–	SO	–	SR	–	(Write Format)

7	6	5	4	3	2	1	0	
BSY	–	TO	OS	SO	–	SR	–	(Read Format)

– : Set to 0, Undefined on Read
BSY : Coprocessor Busy
TO : Terminate Operation
OS : Operation Suspended
SO : Suspend Operation
SR : State Save/Restore

Figure 3-123. Coprocessor Control Register, Offset Hex 11

The register fields are defined as follows:

BSY When the Coprocessor Busy field (bit 7) is read as 1, the coprocessor is currently executing an operation. When read as 0, the coprocessor is idle. On an XGA-2 subsystem, the "Auxiliary Coprocessor Status Register (Offset 09)" should be used in preference to this field. See "Waiting for Hardware Not Busy" on page 3-256.

TO Coprocessor operations can be terminated by writing a 1 to the Terminate Operation field (bit 5). The application must ensure that the operation has terminated before proceeding. Termination is ensured by waiting for the operation-complete interrupt (if enabled), or by reading the Coprocessor Busy field until the coprocessor becomes not busy (bit 7 = 0).

After the coprocessor has terminated the operation, it automatically sets the Terminate Operation field to 0. The coprocessor is returned to its initial power-on state, with coprocessor interrupts masked off and certain other register bits reset. All registers must be assumed invalid and must be reprogrammed before another operation is initiated.

OS The Operation Suspended field (bit 4) is set to 1 by the coprocessor when it has suspended the operation. This bit must be read by the system microprocessor to ensure that an operation has been suspended before saving/restoring is started.

SO When the Suspend Operations field (bit 3) is set to 1, coprocessor operations are suspended.

When set to 0, the suspended processor operations are restarted. This must be done to restart a restored operation after a task switch. When the operation restarts, the coprocessor resets the Operations Suspended field to 0.

Suspending an operation clears the translate look-aside buffer.

If a coprocessor operation is suspended, a terminate operation is required before starting a new coprocessor operation.

SR The State Save/Restore field (bit 1) selects whether to save or restore the coprocessor state. When set to 0, a state restore can be performed; when set to 1, a state save can be performed. The Coprocessor Control register must be written with the Suspend Operation field set and the Save/Restore field appropriately set before each state save or state restore.

State Length Registers (Offset C and D)

These read-only registers have offsets of hex C and D. *Do not write to* these registers. The State Length registers return the length, in doublewords, of parts A and B of the coprocessor state for save and restore.

Save/Restore Data Ports Register (I/O Index C and D)

These read/write registers have I/O indexes of hex C and D. The Save/Restore registers are directly mapped to I/O address space and do not appear in the coprocessor register summary. However, they are coprocessor registers and are described here.

These registers are used to save and restore the two parts, A and B, of the internal state of the coprocessor. After a state save or restore is initiated, string I/O reads or writes must be executed from or to these registers. The data can be read or written using any combination of byte, word, or doubleword accesses, provided that the exact number of doublewords specified in the State Length registers is read or written. Failure to read or write the correct amount of data leaves the coprocessor in an undefined state.

Data must be written back to this port in the order it was read (first out, first in).

Pel Interface Registers

The following is a detailed description of the coprocessor pel interface registers.

Pel Map Index Register (Offset 12)

This write-only register has an offset of hex 12.

7	6	5	4	3	2	1	0
–	–	–	–	–	–	PMI	

– : Set to 0
PMI : Pel Map Index

Figure 3-124. Pel Map Index Register, Offset Hex 12

The register field is defined as follows:

PMI The Pel Map Index field (bits 1, 0) selects which pel map registers will be used.

Each pel map used in the XGA is described by four registers, as follows:

- The Pel Map n Base Pointer register
- The Pel Map n Width register
- The Pel Map n Height register
- The Pel Map n Format register

Each pel map has its own copy of these registers, so there are four copies of these registers in the XGA, one each for:

- The mask map
- Pel map A
- Pel map B
- Pel map C

Only one of these banks of pel map registers is visible to the system microprocessor at any time. The Pel Map Index register is used to select the pel map. The encoding of the Pel Map Index register is shown in the following figure.

PMI Field (binary)	Pel Map Register
0 0	Mask Map
0 1	Pel Map A
1 0	Pel Map B
1 1	Pel Map C

Figure 3-125. Pel Map Index

Before loading the pel map base pointer, width, height, and format for a particular map, the Pel Map Index register must be set up to point to the registers of the required map. For example, to set up the register for map B, first load the pel map index with binary 10.

Pel Map n Base Pointer Register (Offset 14)

This write-only register has an offset of hex 14. The n is selected using the "Pel Map Index Register (Offset 12)" on page 3-137.

31	0
Pel Map n Pointer	

Figure 3-126. Pel Map n Base Pointer Register, Offset Hex 14

The Pel Map n Pointer register (bits 31 – 0) specifies the byte address in memory of the start of a pel map. If virtual address mode is enabled, this address is a virtual address, otherwise it is a physical address.

Pel Map n Width Register (Offset 18)

This write-only register has an offset of hex 18. The Pel Map n Width register can be loaded with any value in the range (0 to 4095). The n is selected using the "Pel Map Index Register (Offset 12)" on page 3-137.

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
Pel Map n Width

Figure 3-127. Pel Map n Width Register, Offset Hex 18

The Pel Map n Width register (bits 15 – 0) specifies the width of a pel map. The width of a pel map is measured in pels, that is, independent of the number of bits-per-pel.

Widths are used during address stepping to specify the width of the pel map. Steps with a Y direction component are achieved by the hardware adding or subtracting the width ± 0/1.

The pel map width is also used for wrapping the source and pattern maps, or to implement the fixed scissor boundary around the destination map.

The value loaded in the width register must be 1 less than the bit map width. For a bit map that is 1024 pels wide, the width register must be loaded with 1023 (hex 03FF).

Pel Map n Height Register (Offset 1A)

This write-only register has an offset of hex 1A. The Pel Map n Height register can be loaded with any value in the range (0 to 4095). The n is selected using the "Pel Map Index Register (Offset 12)" on page 3-137.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Pel Map n Height															

Figure 3-128. Pel Map n Height Register, Offset Hex 1A

The Pel Map n Height register (bits 15 – 0) specifies the height of a pel map. The height of the pel map is measured in pels, that is, independent of the number of bits-per-pel.

The pel map height is used for wrapping the source and pattern maps, or to implement the fixed scissor boundary around the destination map.

The value loaded in the height register must be one less than the pel map height. For a bit map that is 768 pels high, the height register must be loaded with 767 (hex 02FF).

Pel Map n Format Register (Offset 1C)

This write-only register has an offset of hex 1C. The n is selected using the "Pel Map Index Register (Offset 12)" on page 3-137.

7	6	5	4	3	2	1	0
–	–	–	–	PO	PS		

– : Set to 0
PO : Pel Order
PS : Pel Size

Figure 3-129. Pel Map n Format Register, Offset Hex 1C

The register fields are defined as follows:

PO The Pel Order field (bit 3) selects the format for the memory-to-screen mapping. When set to 0, the pel map is Intel-ordered. When set to 1, the pel map is Motorola-ordered. "Pel Formats" on page 3-93 describes the difference in formats.

PS The Pel Size field (bits 2–0) specifies the number of bits-per-pel in the pel map. Pel maps occupied by the source or destination map can be 1, 2, 4, or 8 bits-per-pel on the XGA subsystem and 1, 2, 4, 8, or 16 bits-per-pel on the XGA-2 subsystem. The pel map occupied by the pattern map must be 1 bit-per-pel. Programming the pattern from a pel map that does not contain 1-bit pels produces undefined results. The Pel Size field definitions are shown in the following figure.

PS Field (binary)	Pel Size
000	1 bit
001	2 bits
010	4 bits
011	8 bits
100	16 bits on XGA-2 subsystem, and reserved on XGA subsystem.
101	Reserved
110	Reserved
111	Reserved

Figure 3-130. Pel Size Value Assignments

Pel Maps A, B, and C

Pel maps A, B, and C are all described by similar registers. The different maps are merely three instances of pel maps that can have different locations in memory, sizes, and formats.

The pattern map used by the XGA must be 1 bit-per-pel. The pattern map must reside in a pel map that is 1 bit-per-pel. Failure to do this produces undefined results.

Mask Map

The mask map has a base pointer, width, and height that are similar to those of pel maps A, B, and C.

The Mask Map Format register differs from maps A, B, and C in that only the Motorola/Intel format bit of the mask map is programmable. This register bit operates the same as the bit for maps A, B, and C. The number of bits-per-pel is assumed to be 1 bit-per-pel. The bits-per-pel must always be set to 1 bit-per-pel to ensure future compatibility.

Bresenham Error Term E Register (Offset 20)

This read/write register has an offset of hex 20.

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
Bresenham Error Term

Figure 3-131. Bresenham Error Term E Register, Offset Hex 20

The Bresenham Error Term register (bits 15–0) specifies the Bresenham error term for the draw line function. The error term value is a signed quantity, calculated as ((2 × deltaY) – deltaX) after normalization to the first octant.

This register must be written as a 16-bit sign-extended twos complement number in the range (–8192 to 8191).

Bresenham Constant K1 Register (Offset 24)

This write-only register has an offset of hex 24.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bresenham Constant K1															

Figure 3-132. Bresenham Constant K1 Register, Offset Hex 24

The Bresenham Constant K1 register (bits 15 – 0) specifies the Bresenham constant, K1, for the draw line function. The K1 value is a signed quantity, calculated as (2 × deltaY) after normalization to the first octant.

This register must be written as a 16-bit sign-extended twos complement number in the range (–8192 to 8191).

Bresenham Constant K2 Register (Offset 28)

This write-only register has an offset of hex 28.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bresenham Constant K2															

Figure 3-133. Bresenham Constant K2 Register, Offset Hex 28

The Bresenham Constant K2 register (bits 15 – 0) specifies the Bresenham constant, K2, for the draw line function. The K2 value is a signed quantity, calculated as (2 × (deltaY – deltaX)) after normalization to the first octant.

This register must be written as a 16-bit sign-extended twos complement number in the range (–8192 to 8191).

Direction Steps Register (Offset 2C)

This write-only register has an offset of hex 2C.

The byte order of this register is independent of whether the Intel or Motorola register interface is enabled. The following figure shows the Intel and Motorola views of the Direction Steps register.

```
Intel View of Register

    byte 3        byte 2        byte 1        byte 0

 31         24 23         16 15          8 7           0
┌─────────────┬─────────────┬─────────────┬─────────────┐
│ step code 4 │ step code 3 │ step code 2 │ step code 1 │
└─────────────┴─────────────┴─────────────┴─────────────┘

Motorola View of Register

    byte 0        byte 1        byte 2        byte 3

 31         24 23         16 15          8 7           0
┌─────────────┬─────────────┬─────────────┬─────────────┐
│ step code 1 │ step code 2 │ step code 3 │ step code 4 │
└─────────────┴─────────────┴─────────────┴─────────────┘
```

Figure 3-134. Direction Steps Register, Offset Hex 2C

This register is used to specify up to four draw and step codes to the coprocessor, and to initiate a draw and step operation.

Writing data to byte 3 of this register initiates a draw and step operation. A draw and step operation can be initiated by a single 32-bit access, by two 16-bit accesses where bytes 2 and 3 are written last, or by four 1-byte accesses where byte 3 is written last. If multiple draw and step operations are required with the same draw and step codes, the operation can be initiated by writing to byte 3.

Before initiating a draw and step operation, the Pel Operation register must be configured to set up the data path and flags for the draw and step operation. See "Draw and Step" on page 3-105 for full details.

Foreground Mix Register (Offset 48)

This write-only register has an offset of hex 48.

7	6	5	4	3	2	1	0
Foreground Mix							

Figure 3-135. Foreground Mix Register, Offset Hex 48

The Foreground Mix register (bits 7 – 0) holds the foreground mix value that specifies a logic or arithmetic function to be performed between the destination and function 1 second operand pels, during an operation where the pattern pel value is 1.

See "Logical and Arithmetic Functions" on page 3-119 for details and mix functions available.

Background Mix Register (Offset 49)

This write-only register has an offset of hex 49.

7	6	5	4	3	2	1	0
Background Mix							

Figure 3-136. Background Mix Register, Offset Hex 49

The Background Mix register (bits 7 – 0) holds the background mix value that specifies a logic or arithmetic function to be performed between the destination and function 0 second operand pels during an operation where the pattern pel value is 0.

See "Logical and Arithmetic Functions" on page 3-119 for details and mix functions available.

Destination Color Compare Condition Register (Offset 4A)

This write-only register has an offset of hex 4A.

7	6	5	4	3	2	1	0
–	–	–	–	–	DCC		

```
  – : Set to 0
DCC : Destination Color Compare Condition
```

Figure 3-137. Destination Color Compare Condition Reg, Offset Hex 4A

The register field is defined as follows:

DCC The Destination Color Compare Condition field (bits 2–0) specifies the destination color compare condition when the destination update is inhibited. The DCC field definitions are shown in the following figure.

DCC Field (binary)	Destination Color Compare Condition
0 0 0	Always True (disable update)
0 0 1	Destination > Color Compare Value
0 1 0	Destination = Color Compare Value
0 1 1	Destination < Color Compare Value
1 0 0	Always False (enable update)
1 0 1	Destination > = Color Compare Value
1 1 0	Destination < > Color Compare Value
1 1 1	Destination < = Color Compare Value

Figure 3-138. Destination Color Compare Condition Bit Definition

Destination Color Compare Value Register (Offset 4C)

This write-only register has an offset of hex 4C.

31	0
Destination Color Compare Value	

Figure 3-139. Destination Color Compare Value Register, Offset Hex 4C

The Destination Color Compare Value register (bits 31 – 0) contains the comparison value for the destination pels to be compared when color compare is enabled. Only the corresponding number of bits-per-pel in the destination are required in this register (for example, if the destination is 4 bits-per-pel, only the 4 low-order bits of this register are used). The bits of this register that are more significant than the number of bits-per-pel need not be written.

See "Color Compare" on page 3-123 for details of the color compare function.

Pel Bit Mask (Plane Mask) Register (Offset 50)

This write-only register has an offset of hex 50.

31	0
Pel Bit Mask	

Figure 3-140. Pel Bit Mask (Plane Mask) Register, Offset Hex 50

The Pel Bit Mask register (bits 31 – 0) determines the bits within each pel that are subject to update by the coprocessor. A 1 means the corresponding bit is enabled for updates. A 0 means the corresponding bit is not updated.

A bit that is not write enabled is prevented from affecting either arithmetic operations or the destination color compare comparison, so masked bits are excluded from the operation or comparison. Only the corresponding number of bits-per-pel in the destination are required in this register (for example, if the destination is 4 bits-per-pel, only the 4 low-order bits of this register are used). The bits of this register that are more significant than the number of bits-per-pel need not be written.

See “Pel Bit Masking” on page 3-123 for details of the pel bit mask function.

Carry Chain Mask Register (Offset 54)

This write-only register has an offset of hex 54.

31 ... 0
Carry Chain Mask

Figure 3-141. Carry Chain Mask Field Register, Offset Hex 54

The Carry Chain Mask register (bits 31 – 0) contains a mask up to 31 bits wide. The mask is used to specify how the carry chain of the coprocessor is propagated when performing arithmetic update mixes and color compare operations.

A 0 in the mask means that the carry out of this bit-position of the coprocessor is not to be propagated to the next significant bit-position. A 1 in the mask means that propagation is to take place. The pel value can be split into sections within the pel.

Only the corresponding number of bits-per-pel in the destination are required in this register (for example, if the destination is 4 bits-per-pel, only the 4 low-order bits of this register are used). The bits of this register that are more significant than the number of bits-per-pel, need not be written. There is no carry out of the most-significant bit of the pel irrespective of the setting of the corresponding carry chain mask bit.

See "Breaking the Coprocessor Carry Chain" on page 3-121 for details on the carry chain function.

Foreground Color Register (Offset 58)

This write-only register has an offset of hex 58.

31	0
Foreground Color	

Figure 3-142. Foreground Color Register, Offset Hex 58

The Foreground Color register (bits 31 — 0) holds the foreground color to be used during coprocessor operations. The foreground color can be specified as the foreground source by setting up the appropriate field in the Pel Operations register.

Only the corresponding number of bits-per-pel in the destination are required in this register (for example, if the destination is 4 bits-per-pel, only the 4 low-order bits of this register are used). The bits of this register that are more significant than the number of bits-per-pel need not be written.

Background Color Register (Offset 5C)

This write-only register has an offset of hex 5C.

31	0
Background Color	

Figure 3-143. Background Color Register, Offset Hex 5C

The Background Color register (bits 31 — 0) holds the background color to be used during coprocessor operations. The background color can be specified as the background source by setting up the appropriate field in the Pel Operation register.

Only the corresponding number of bits-per-pel in the destination are required in this register (for example, if the destination is 4 bits-per-pel, only the 4 low-order bits of this register are used). The bits of this register that are more significant than the number of bits-per-pel need not be written.

Operation Dimension 1 Register (Offset 60)

This write-only register has an offset of hex 60.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Operation Dimension 1															

Figure 3-144. Operation Dimension 1 Register, Offset Hex 60

The Operation Dimension 1 register (bits 15 – 0) specifies the width of the rectangle to be drawn by the PxBlt function, or the length of the line in a line draw operation. The value is an unsigned quantity and must be one less than the required width. To draw a line 10 pels long, the value 9 must be written to this register.

The value written to this register must be within the range (0 to 4095).

Operation Dimension 2 Register (Offset 62)

This write-only register has an offset of hex 62.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Operation Dimension 2															

Figure 3-145. Operation Dimension 2 Register, Offset Hex 62

The Operation Dimension 2 register (bits 15 – 0) specifies the height of the rectangle to be drawn by the PxBlt function. The value is an unsigned quantity and must be one less than the required height. To draw a rectangle 10 pels high, the value 9 must be written to this register.

The value written to this register must be within the range (0 to 4095).

Mask Map Origin X Offset Register (Offset 6C)

This write-only register has an offset of hex 6C.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Mask Map Origin X Offset															

Figure 3-146. Mask Map Origin X Offset Register, Offset Hex 6C

The Mask Map Origin X Offset register (bits 15−0) specifies the X offset of the mask map origin, relative to the origin of the destination map.

The value written to this register must be within the range (0 to 4095).

Mask Map Origin Y Offset Register (Offset 6E)

This write-only register has an offset of hex 6E.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Mask Map Origin Y Offset															

Figure 3-147. Mask Map Origin Y Offset Register, Offset Hex 6E

The Mask Map Origin Y Offset register (bits 15−0) specifies the Y offset of the mask map origin, relative to the origin of the destination map.

The value written to this register must be within the range (0 to 4095).

Source X Address Register (Offset 70)

This read/write register has an offset of hex 70.

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
Source X Address

Figure 3-148. Source X Address Register, Offset Hex 70

The Source X Address register (bits 15 – 0) specifies the X coordinate of the coprocessor operation source pel.

The value written to this register must be within the range (0 to 4095).

Source Y Address Register (Offset 72)

This read/write register has an offset of hex 72.

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
Source Y Address

Figure 3-149. Source Y Address Register, Offset Hex 72

The Source Y Address register (bits 15 – 0) specifies the Y coordinate of the coprocessor operation source pel.

The value written to this register must be within the range (0 to 4095).

Pattern X Address Register (Offset 74)

This read/write register has an offset of hex 74.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Pattern X Address															

Figure 3-150. Pattern C Address Register, Offset Hex 74

The Pattern X Address register (bits 15–0) specifies the X coordinate of the coprocessor operation pattern pel.

The value written to this register must be within the range (0 to 4095).

Pattern Y Address Register (Offset 76)

This read/write register has an offset of hex 76.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Pattern Y Address															

Figure 3-151. Pattern Y Address Register, Offset Hex 76

The Pattern Y Address register (bits 15–0) specifies the Y coordinate of the coprocessor operation pattern pel.

The value written to this register must be within the range (0 to 4095).

Destination X Address Register (Offset 78)

This read/write register has an offset of hex 78.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Destination X Address															

Figure 3-152. Destination X Address Register, Offset Hex 78

The Destination X Address register (bits 15–0) specifies the X coordinate of the coprocessor operation destination pel.

This register must be written as a 16-bit sign-extended twos complement number in the range (–2048 to 6143).

Destination Y Address Register (Offset 7A)

This read/write register has an offset of hex 7A.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Destination Y Address															

Figure 3-153. Destination Y Address Register, Offset Hex 7A

The Destination Y Address register (bits 15–0) specifies the Y coordinate of the coprocessor operation destination pel.

This register must be written as a 16-bit sign-extended twos complement number in the range (–2048 to 6143).

Pel Operations Register (Offset 7C)

This write-only register has an offset of hex 7C.

The Pel Operations register is used to define the flow of data during an operation. It specifies the address update function that is to be performed, and initiates PxBlt and line draw operations.

The contents of the Pel Operation register are preserved throughout an operation.

The byte order of this register is dependent on whether the Intel or Motorola register interface is enabled. The following figure shows the Intel and Motorola views of the Pel Operations register.

```
Intel View of Register

|    Byte 3     |    Byte 2     |    Byte 1     |    Byte 0     |

31 . . . . . .24 23. . . . . .16 15. . . . . . 8 7 . . . . . . 0
+---+---+-------+-------+-------+-------+-+-+-+-+---+---+-+-----+
|BS | FS| Step  |Source | Dest  | Patt  |-|-|-|-|MSK| DM|-| Oct |
+---+---+-------+-------+-------+-------+-+-+-+-+---+---+-+-----+


Motorola View of Register

|    Byte 0     |    Byte 1     |    Byte 2     |    Byte 3     |

31 . . . . . .24 23. . . . . .16 15. . . . . . 8 7 . . . . . . 0
+---+---+-+-----+-------+-+-+-+-+-------+-------+---+---+-------+
|MSK| DM|-| Oct | Patt  |-|-|-|-|Source | Dest  |BS | FS| Step  |
+---+---+-+-----+-------+-+-+-+-+-------+-------+---+---+-------+

       - : Set to 0                  Dest : Destination Pel Map
      BS : Background Source         Patt : Pattern Pel Map
      FS : Foreground Source          MSK : Mask Pel Map
    Step : Step Function               DM : Drawing Mode
  Source : Source Pel Map             Oct : Direction Octant
```

Figure 3-154. Pel Operations Register, Offset Hex 7C

All operations, with the exception of draw and step (see "Draw and Step" on page 3-105 and "Direction Steps Register (Offset 2C)" on page 3-144), are initiated by writing to the *most-significant byte* of this register. Therefore, an operation can be initiated by a single 32-bit write, by two 16-bit writes when bytes 2 and 3 are written last, or by four 1-byte accesses when byte 3 is written last.

The fields in the Pel Operation register are shown, by bytes, in the following figures.

```
|              Byte 3              |

   7    6    5    4    3    2    1    0
+---------+---------+------------------+
|   BS    |   FS    |       Step       |
+---------+---------+------------------+

      BS : Background Source
      FS : Foreground Source
    Step : Step Function
```

Figure 3-155. Pel Operations Register, Byte 3

BS The Background Source field (byte 3; bits 7, 6) determines the background source that is to be combined with the destination when the pattern pel equals 0 (background mix).

BS Field (binary)	Background Source
0 0	Background Color
0 1	Reserved
1 0	Source Pel Map
1 1	Reserved

Figure 3-156. Pel Operations Register Background Source Value Assignments

FS The Foreground Source field (byte 3; bits 5, 4) determines the foreground source that is to be combined with the destination when the pattern pel equals 1 (foreground mix).

FS Field (binary)	Foreground Source
0 0	Foreground Color
0 1	Reserved
1 0	Source Pel Map
1 1	Reserved

Figure 3-157. Pel Operations Register Foreground Source Value Assignments

Step The Step Function field (byte 3; bits 3 − 0) determines how the coprocessor address is modified as pel data is manipulated. This field can be regarded as the coprocessor function code. Writing to this field starts the coprocessor operation, except for draw and step functions. Draw and step operations are started by writing to the Direction Steps register.

Step Field (binary)	Step Function
0 0 0 0	Reserved
0 0 0 1	Reserved
0 0 1 0	Draw and Step Read
0 0 1 1	Line Draw Read
0 1 0 0	Draw and Step Write
0 1 0 1	Line Draw Write
0 1 1 0	Reserved
0 1 1 1	Reserved
1 0 0 0	PxBlt
1 0 0 1	Inverting PxBlt
1 0 1 0	Area Fill PxBlt
1 0 1 1	Reserved
1 1 0 0	Reserved
1 1 0 1	Reserved
1 1 1 0	Reserved
1 1 1 1	Reserved

Figure 3-158. Pel Operations Register Step Function Value Assignments

Byte 2							
7	6	5	4	3	2	1	0
Source				Dest			

Source : Source Pel Map
Dest : Destination Pel Map

Figure 3-159. Pel Operations Register, Byte 2

Source The Source Pel Map field (byte 2; bits 7–4) determines the location of pel map source data. The combination of this field and the Foreground and Background Source fields determine the data that is to be used as the source data for coprocessor functions.

Source Field (binary)	Source Pel Map
0000	Reserved
0001	Pel Map A
0010	Pel Map B
0011	Pel Map C
0100	Reserved
•	•
•	•
1111	Reserved

Figure 3-160. Pel Operations Register Source Pel Map Value Assignments

Dest The Destination Pel Map field (byte 2; bits 3 – 0) determines the location of the destination data to be modified during an operation.

Dest Field (binary)	Destination Pel Map
0 0 0 0	Reserved
0 0 0 1	Pel Map A
0 0 1 0	Pel Map B
0 0 1 1	Pel Map C
0 1 0 0	Reserved
•	•
•	•
1 1 1 1	Reserved

Figure 3-161. Pel Operations Register Destination Pel Map Value Assignments

Byte 1

7	6	5	4	3	2	1	0
Patt				–	–	–	–

– : Set to 0
Patt : Pattern Pel Map

Figure 3-162. Pel Operations Register, Byte 1

Patt The Pattern Pel Map field (byte 1; bits 7 – 4) determines the pattern data to be used during an operation. Code 1000 causes the coprocessor to assume that the pattern is 1 across the whole operation, and to use the foreground function on all pels. This effectively turns off the use of the pattern. Code 1001 causes the pattern to be generated from source data. Every 0 pel in the source generates a background pattern pel; every nonzero pel in the source generates a foreground pattern pel.

Patt Field (binary)	Pattern Pel Map
0000	Reserved
0001	Pel map A
0010	Pel map B
0011	Pel map C
0100	Reserved
0101	Reserved
0110	Reserved
0111	Reserved
1000	Foreground (fixed)
1001	Generated from Source
1010	Reserved
•	•
•	•
1111	Reserved

Figure 3-163. Pel Operations Register Pattern Pel Map Value Assignments

Byte 0

7	6	5	4	3	2	1	0
MSK		DM		–	Oct		

– : Set to 0 DM : Drawing Mode
MSK : Mask Pel Map Oct : Direction Octant

Figure 3-164. Pel Operations Register, Byte 0

MSK The Mask Pel Map field (byte 0; bits 7, 6) determines how the mask map is used. See "Scissoring with the Mask Map" on page 3-100 for details of the mask map modes.

MSK Field (binary)	Mask Pel Map
00	Mask Map Disabled
01	Mask Map Boundary Enabled
10	Mask Map Enabled
11	Reserved

Figure 3-165. Pel Operations Register Mask Pel Map Value Assignments

DM The Drawing Mode field (byte 0; bits 5, 4) determines the attributes of line draw and draw and step operations.

DM Field (binary)	Drawing Mode
0 0	Draw All Pels
0 1	Draw First Pel Null
1 0	Draw Last Pel Null
1 1	Draw Area Boundary

Figure 3-166. Pel Operations Register Drawing Mode Value Assignments

Oct The Direction Octant field (byte 0; bits 2–0) is comprised of three bits, DX, DY, and DZ.

Oct		
2	1	0
DX	DY	DZ

Figure 3-167. Pel Operations Register Direction Octant Values

XGA System Interface

Multiple Instances

Up to eight instances of an XGA subsystem can be installed in a system unit. Addressing the I/O registers, memory-mapped registers, and video memory for each instance is controlled by the contents of the XGA POS registers. See "XGA POS Registers" on page 3-164 for more information.

Multiple XGA Subsystems in VGA Mode

The VGA has only one set of addresses allocated to it. It is not possible to have multiple XGA subsystems in VGA mode responding to update requests simultaneously. However, more than one XGA subsystem can be in VGA mode if only one has VGA address decoding enabled using the Operating Mode register. Subsystems with VGA address decoding disabled continue to display the correct picture. See "XGA Subsystem Coexistence with VGA" on page 3-220 for further information.

Note: The XGA *must not* be disabled using the Subsystem Enable field in XGA POS Register 2.

Multiple XGA Subsystems in 132-Column Text Mode

In 132-column text mode, the XGA responds to VGA address decodes, and the same rules apply as for multiple XGA subsystems in VGA mode. See "XGA Subsystem Coexistence with VGA" on page 3-220 for further information.

Multiple XGA Subsystems in Extended Graphics Mode

Extended Graphics modes are controlled by a bank of 16 I/O registers that are located in one of eight possible locations. Up to eight XGA subsystems can be installed in a system unit. Each instance of XGA installed is positioned at a unique I/O and memory location, so each can be used independently in the system. See "Multiple XGA Subsystems" on page 3-220 for details on controlling multiple XGAs.

The XGA coprocessor memory-mapped registers occupy a bank of 128 contiguous register addresses that are mapped in memory space. These registers can also be relocated, allowing up to eight instances of the XGA coprocessor to coexist in a system.

The locations of these registers are controlled by the XGA POS registers. See "XGA POS Registers" on page 3-164 for the register details, and see"XGA Subsystem Identification, Location, and XGA Mode Setting" on page 3-185 for programming considerations on reading and using the data contained in them.

XGA POS Registers

The XGA subsystem has movable I/O addresses for the display controller, allowing more than one XGA subsystem to be installed in a system unit.

All POS registers detailed in this section are set up during system configuration and must never be written to. All registers are specified relative to a base address. Details of how to locate the base address and read the registers are given in "XGA Subsystem Identification, Location, and XGA Mode Setting" on page 3-185.

Register Usage Guidelines

Unless otherwise stated, the following are guidelines to be used when accessing the POS registers:

- All registers are 8 bits long.
- All registers are read only.
- Special reserved register bits must be used as follows:
  - Register bits marked with '–' must be masked off if the contents of the register are to be tested.
  - Register bits marked with '#' are reserved and the state of these bits must be preserved. This register must be masked off if the contents of the register are to be tested.

Subsystem Identification Low Byte Register (Base + 0)

This read-only register is located at base address + 0. *Do not write* to this register. When read, this register returns the low byte of the POS ID.

Subsystem Identification High Byte Register (Base + 1)

This read-only register is located at base address + 1. *Do not write* to this register. When read, this register returns the high byte of the POS ID.

POS Register 2 (Base + 2)

This read-only register is located at base address + 2. *Do not write* to this register.

7	6	5	4	3	2	1	0
ROM				INST			EN

```
 ROM : ROM Address
INST : instance
  EN : Subsystem Enable
```

Figure 3-168. POS Register 2, Base Address + 2

The fields in this register are defined as follows:

ROM The ROM Address field (bits 7 – 4) specifies which of 16 possible 8KB memory locations has been assigned to the XGA ROM. The ROM occupies the first 7KB of this 8KB block; the other 1KB is occupied by the coprocessor memory-mapped registers.

The instance field specifies the 128-byte section within this 1KB block that is allocated to the subsystem. See the following figure. For example, instance 2 has its coprocessor registers located in the third 128-byte section of the 1KB block.

ROM Address		Coprocessor Register Base Address (hex)							
Field	Range (hex)	Instance: 0	1	2	3	4	5	6	7
0000	C0000 : C1BFF	C1C00	C1C80	C1D00	C1D80	C1E00	C1E80	C1F00	C1F80
0001	C2000 : C3BFF	C3C00	C3C80	C3D00	C3D80	C3E00	C3E80	C3F00	C3F80
0010	C4000 : C5BFF	C5C00	C5C80	C5D00	C5D80	C5E00	C5E80	C5F00	C5F80
0011	C6000 : C7BFF	C7C00	C7C80	C7D00	C7D80	C7E00	C7E80	C7F00	C7F80
0100	C8000 : C9BFF	C9C00	C9C80	C9D00	C9D80	C9E00	C9E80	C9F00	C9F80
0101	CA000 : CBBFF	CBC00	CBC80	CBD00	CBD80	CBE00	CBE80	CBF00	CBF80
0110	CC000 : CDBFF	CDC00	CDC80	CDD00	CDD80	CDE00	CDE80	CDF00	CDF80
0111	CE000 : CFBFF	CFC00	CFC80	CFD00	CFD80	CFE00	CFE80	CFF00	CFF80
1000	D0000 : D1BFF	D1C00	D1C80	D1D00	D1D80	D1E00	D1E80	D1F00	D1F80
1001	D2000 : D3BFF	D3C00	D3C80	D3D00	D3D80	D3E00	D3E80	D3F00	D3F80
1010	D4000 : D5BFF	D5C00	D5C80	D5D00	D5D80	D5E00	D5E80	D5F00	D5F80
1011	D6000 : D7BFF	D7C00	D7C80	D7D00	D7D80	D7E00	D7E80	D7F00	D7F80
1100	D8000 : D9BFF	D9C00	D9C80	D9D00	D9D80	D9E00	D9E80	D9F00	D9F80
1101	DA000 : DBBFF	DBC00	DBC80	DBD00	DBD80	DBE00	DBE80	DBF00	DBF80
1110	DC000 : DDBFF	DDC00	DDC80	DDD00	DDD80	DDE00	DDE80	DDF00	DDF80
1111	DE000 : DFBFF	DFC00	DFC80	DFD00	DFD80	DFE00	DFE80	DFF00	DFF80

Figure 3-169. XGA ROM, Memory-Mapped Register Assignments

Note: The Coprocessor registers might also be able to be accessed using alternative addresses in the protect mode address range. See "XGA Subsystem Identification, Location, and XGA Mode Setting" on page 3-185 for general details of locating and using the XGA registers, and "Alternative XGA Coprocessor Register Set" on page 3-266 for specific details on the alternative "Shadowed" Coprocessor Registers.

INST The value in this field indicates the instance number of this XGA subsystem. Coexisting XGA Subsystems each have unique instance values.

The instance field (bits 3 − 1) specifies the set of I/O addresses allocated to the display controller registers. See Figure 3-170. The lowest address of each set of addresses is referred to as the I/O base address.

INST Field (binary)	Instance Number	Instance I/O Base Address (hex)
0 0 0	0	2100
0 0 1	1	2110
0 1 0	2	2120
0 1 1	3	2130
1 0 0	4	2140
1 0 1	5	2150
1 1 0	6	2160
1 1 1	7	2170

Figure 3-170. I/O Device Address Bit Assignment

EN The Subsystem Enable field (bit 0) indicates whether the subsystem is enabled. When read as 1, the subsystem is enabled for address decoding for all non-POS addresses. When read as 0, only POS registers can be accessed; all other accesses to the subsystem have no effect.

POS Register 4 (Base + 4)

This read-only register is located at base address + 4. *Do not write* to this register.

7	6	5	4	3	2	1	0
Video Memory Base Address							VE

VE : Video Memory Enable

Figure 3-171. POS Register 4, Base Address + 4

The fields in this register are defined as follows:

Video Memory Base Address
This field (bits 7 − 1) contains the most significant 7 bits of the address where the XGA memory is located. Three more bits are provided by the instance in POS byte 1. This gives a video memory base address on a 4MB boundary. See the following figure.

31 - 25	24 - 22	21 - 0
Video Memory Base Address	Instance 0 – 7	4MB of Addressable Memory

Figure 3-172. XGA Video Memory Base Address

For example, if the video memory base address is set to 1 and instance 6 has been selected, the XGA video memory is located at hex 03800000.

VE
When the Video Memory Enable field (bit 0) is set to 0, the 4MB aperture is disabled; when set to 1, the 4MB aperture is enabled.

POS Register 5 (Base + 5)

This read-only register is located at base address + 5. *Do not write* to this register.

7	6	5	4	3	2	1	0
–	–	–	–	AP1			

– : Undefined On Read
AP1 : 1MB Aperture Base Address

Figure 3-173. POS Register 5, Base Address + 5

The field in this register is defined as follows:

AP1 The 1MB Aperture Base Address field (bits 3 – 0) indicates where the 1MB aperture is placed in system address space, or if the aperture has been disabled. The following figure describes the use of this field.

AP1 Field (binary)	1MB Aperture Location (hex)
0000	Disabled
0001	00100000
0010	00200000
0011	00300000
0100	00400000
0101	00500000
0110	00600000
0111	00700000
1000	00800000
1001	00900000
1010	00A00000
1011	00B00000
1100	00C00000
1101	00D00000
1110	00E00000
1111	00F00000

Figure 3-174. 1MB Aperture Base Address Value Assignments

Virtual Memory Description

The XGA coprocessor can address either real or virtual memory. When addressing real memory, the linear address calculated by the coprocessor is passed directly to the system microprocessor or local video memory. When addressing virtual memory, the linear address from the coprocessor is translated by on-chip virtual memory translation logic before the translated address is passed to the system microprocessor or local video memory. Virtual address translation is enabled or disabled by a control bit in the XGA.

The coprocessor uses two levels of tables to translate the linear address from the coprocessor to a physical address. Addresses are translated through a page directory and page table to generate a physical address to memory pages that are 4KB in size. The page directory and page tables are of the same form as those used by the 80386 Processor Paging Unit.

Address Translation

The linear address from the coprocessor is divided into three fields that are used to look up the corresponding physical address. The fields, called directory index, table index, and offset, are illustrated in the following figure.

31 22	21 12	11 0
Directory Index	Table Index	Offset

Figure 3-175. Linear Address Fields

The location of the page directory is at a fixed physical address in memory that must be on a page (4KB) address boundary. The coprocessor has a Page Directory Base Address register that must be loaded with the address of the page directory base.

The translation process is illustrated in Figure 3-176 on page 3-171.

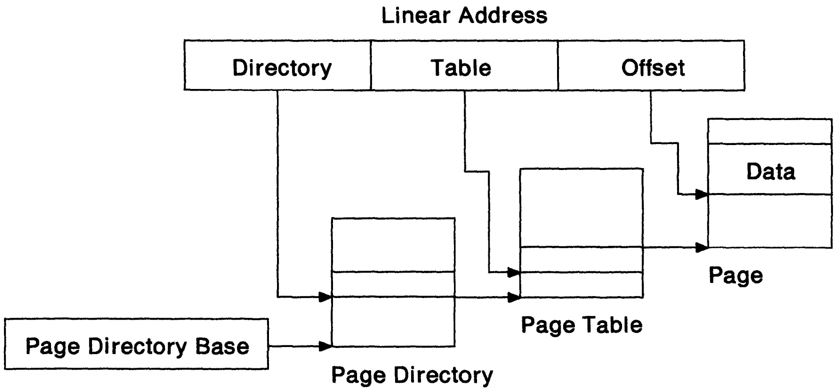


Figure 3-176. Linear to Physical Address Translation

The Directory Index field of the linear address is used to index into the page directory. The entry read from the page directory contains a 20-bit page table address and some statistical information in the low order bits.

The 20-bit page table address points to the base of a page table in memory. The Table Index field in the linear address is used to index into the page table. The entry read from the page table contains a 20-bit page address and some statistical information in the low order bits.

The 20-bit page address points to the base of a 4KB page in memory. The Offset field in the linear address is used to index into the page. The entry read from the page contains the data required by the memory access.

Page Directory and Page Table Entries

The entries of the page directory and page table are very similar. The format of an entry is shown in the following figure.

31 ... 12		6	5		2	1	0
Page Table/Page Addr		D	A		U/S	R/W	P

D - Dirty Bit
A - Accessed Bit
U/S - User/Supervisor Bit
R/W - Read/Write Bit
P - Present Bit

Figure 3-177. Page Directory and Page Table Entry

The top 20 bits of the entry are either the page table address or the page address. The low order bits are as follows:

Dirty Bit (bit 6): This bit is set before a write to an address covered by that page table entry occurs. The dirty bit is undefined for page directory entries.

Accessed Bit (bit 5): This bit is set for both types of entry before a read or write access occurs to an address covered by the entry.

User/Supervisor and Read/Write Bits (bits 2,3): These bits prevent unauthorized use of page directory and page table entries. Accesses by the coprocessor can be defined as a supervisor or user access, depending on the status of the application using the coprocessor. The access type is defined by a bit in the virtual memory (VM) Control register. If the access is defined as supervisor, no protection is provided and all accesses to the page directory and page tables are permitted.

For a user access, the U/S and R/W bits are checked to ensure that access to that entry is permitted. The meaning of these bits is shown in the following figure.

U/S	R/W	Access Rights of User
0	0	Access not permitted
0	1	Access not permitted
1	0	Reads permitted, writes not permitted
1	1	Reads and writes permitted

Figure 3-178. Page Directory and Page Table Access Rights in User Mode

Present Bit (bit 0): The present bit indicates whether a page directory or page table entry can be used in translation. If the bit is set, it indicates that the page table or page that the entry refers to is present in memory.

The XGA Implementation of Virtual Memory

The XGA coprocessor operates with a page directory and page tables in the format described. The coprocessor has its own internal cache of translated addresses to avoid it having to perform the two-stage translation process on every coprocessor access. This cache is referred to as a translate look-aside buffer.

The Translate Look-aside Buffer

The translate look-aside buffer (TLB) has two entries, one entry for the source, pattern, and mask pel maps, and another for the destination pel map, as shown in Figure 3-179. Each entry is reserved specifically for use by one of these maps. Each entry in the TLB contains the top 20 bits of a linear address (the address tag), an entry valid flag bit, and the top 20 bits of the physical address (the real page address) corresponding to that linear address. When a linear address is passed from the coprocessor to the virtual address hardware, the top 20 bits of the linear address are first compared against the appropriate TLB entry address tag. If they match and the TLB entry flag bit is valid, the real page address in that TLB is used as the top 20 bits of the physical address for that access. The bottom 12 bits of the physical address are provided from the bottom 12 bits of the linear address (the offset).

		31 12	31 12
Source, Pattern, and Mask Entry	V	Linear Address Tag	Real Page Address
Destination Entry	V	Linear Address Tag	Real Page Address

V - Valid Bits

Figure 3-179. Translate Look-Aside Buffer

If the linear address from the coprocessor matches the address tag in the TLB for the particular map in use, the access is said to have caused a TLB hit. If the tag does not match, a TLB miss occurs.

The TLB contents are cleared by the hardware under the following circumstances:

- Whenever the Page Directory Base Address register is written
- Whenever a coprocessor operation is suspended (by setting the suspend operation bit in the Coprocessor Control register). See "Coprocessor Control Register (Offset 11)" on page 3-134.

TLB Misses

If a TLB miss occurs, the coprocessor automatically performs the two-level translation required to form the required page address. The contents of the XGA Page Directory Base Address register are used to access the appropriate page directory entry that is used to access the appropriate page table entry. The real page address, resulting from the translation process, is stored in the TLB for use by subsequent accesses that address the same page.

Memory access performed by the coprocessor can be categorized as follows:

Read accesses Performed on the source, pattern, mask, and destination maps.

Write accesses Performed only on the destination map.

When the virtual memory hardware accesses the page directory and page tables for a TLB miss, it examines and updates the flags in the low order bits of the entries, as follows:

Accessed Bit (bit 5): Any access (read or write) sets this bit in both the page directory and page table entries.

Dirty Bit (bit 6): Write accesses set the dirty bit in the page table entry. The dirty bit is undefined in page directory entries.

User/Supervisor and Read/Write Bits (bits 2, 3): These bits are examined by the coprocessor. The coprocessor has a bit, programmed by the host operating system, to indicate whether it is being used by a supervisor or user. In user mode the coprocessor determines whether access is permitted, depending on the state of the user/supervisor and the read/write bits in the page directory and page table entries. If access is not permitted, the coprocessor sends a VM-protection-violation interrupt to the system microprocessor and terminates the access cycle. The host operating system must take the appropriate action to recover from the protection-violation interrupt.

Present Bit (bit 0): The coprocessor examines the present bit of the page directory and page table entries. If this bit is not set, it indicates that the page table or page corresponding to that entry may not be resident in memory, and the XGA sends a VM page-not-present interrupt to the system microprocessor. The host operating system retrieves the page table or page and places it in memory. The access can then be completed.

Remaining Page Directory or Page Table Entry Bits: The coprocessor ignores all the other bits in the page directory or page table entry. The coprocessor does not modify these bits; they can be used by the operating system. It is advisable to keep entries in the same format as the 80386 page directory and page table entry formats; therefore the Intel rules on the use of the remaining bits must be followed.

System Coherency

In any virtual memory system where more than one device is accessing virtual memory contents, problems can arise over coherency. It is essential that one device does not damage the tables or pages of the other device, and the tables and TLBs are kept coherent (in step) with the physical allocation of storage. The hardware mechanism provided by the coprocessor is sufficient to implement coherent virtual memory systems, but care should be taken to avoid coherency problems.

In particular, it is recommended that the 80386 and the coprocessor do not share page directories or page tables. Pages must not be marked as present unless they are locked in place in memory. This maintains coherency between TLB entries in the coprocessor and the true current allocation of real memory. It prevents the operating system from moving these pages out of memory while the coprocessor is accessing them.

VM Page-Not-Present Interrupts

When the coprocessor detects that a page table or page is not present, it sends a page-not-present interrupt to the system microprocessor. The system microprocessor operating system then retrieves the required page table or page (usually from disk) and places it in memory. The system microprocessor can determine the faulting address by reading the Current Virtual Address register. After the required page table or page has been retrieved, the operating system restarts the faulting memory access by clearing the page-not-present interrupt bit in the XGA. This causes the hardware to retry the access to the faulted entry.

When the operating system receives a page-not-present interrupt, it switches tasks and suspends the coprocessor operation before clearing the interrupt (see "Suspending Coprocessor Operations" on page 3-126). When the task is restarted, the coprocessor state is restored, the required page table or page is placed in memory, the interrupt is cleared, and the coprocessor operation is resumed. The interrupt must be cleared before the coprocessor operation is restarted; otherwise, subsequent interrupts can be lost.

VM Protection-Violation Interrupts

If the coprocessor is directed to access tables or pages that are not permitted (as defined by the user/supervisor and read/write entry bits), the coprocessor generates a protection-violation interrupt. This indicates that something is wrong with the virtual memory system (as set up by operating system software), or that the coprocessor has been programmed incorrectly.

The operating system will probably terminate the coprocessor operation and possibly terminate the faulting task. The coprocessor operation can be terminated by writing to a control bit in the Coprocessor Control register. The coprocessor responds in a similar manner for protection-violation interrupts as it does for page-not-present interrupts; clearing the interrupt causes the hardware to retry the memory access. To avoid a repeated interrupt, the coprocessor operation must be terminated before the protection-violation interrupt is cleared.

The XGA in Segmented Systems

In a segmented system design, all memory is allocated in blocks which are called segments. Memory within a segment is guaranteed to be contiguous, and can therefore be addressed directly by the coprocessor using physical addresses (VM is turned off). The segment must be locked in place before any coprocessor operation to ensure that the operating system does not reuse the memory during the operation.

When using 16-bit addressing in the 80386 (for example, under the OS/2 Version 1.3 operating system), it is not possible to define a segment of more than 64KB. If the coprocessor data in system memory is restricted to no more than 64KB in length, a single segment can be used and the coprocessor can directly address the data using physical addresses.

When using the OS/2 Version 1.3 operating system, larger areas of memory can be requested, but are given in blocks of 64KB (maximum) that are unlikely to be contiguous in real memory. If larger areas in system memory are required, the driving software can turn on the coprocessor VM address translation and perform its own memory management, using memory allocated to it by the operating system.

Virtual Memory Registers

The following registers provide virtual memory support for the pel interface.

Page Directory Base Address Register (Coprocessor Registers, Offset 0)

This write-only register has coprocessor registers offset of hex 0.

31 ... 12	11 ... 0
Page Directory Base Addr Pointer	–

– : Set all bits in field to 0

Figure 3-180. Page Directory Base Address Register, Offset Hex 0

The field in this register is defined as follows:

Page Directory Base Addr Pointer
This field (bits 31 – 12) is a 20-bit pointer to the page in physical memory containing the current page directory for the current task.

Loading this register clears the translate look-aside buffer.

Note: This register can be loaded only after the XGA is put in supervisor mode.

Current Virtual Address Register (Coprocessor Registers, Offset 4)

This read-only register has coprocessor registers offset of hex 4. *Do not write* to this register.

31 — 12	11 — 0
Faulting Page Address	–

– : Set all bits in field to 0

Figure 3-181. Current Virtual Address Register, Offset Hex 4

The field in this register is defined as follows:

Faulting Page Address
After a VM hardware-not-present interrupt or protection interrupt, read this field (bits 31 – 12) to find the fault address page.

Virtual Memory Control Register (I/O Address 21x6)

This read/write register has an I/O address of hex 21x6. The Virtual Memory Control register is directly mapped to I/O address space.

7	6	5	4	3	2	1	0
NPE	PVE	–	–	–	US	–	EV

```
  - : Set to 0, Undefined on Read
NPE : VM Page Not Present Interrupt Enable
PVE : VM Protection Violation Interrupt Enable
 US : User / Supervisor
 EV : Enable Virtual Address Lookup
```

Figure 3-182. Virtual Memory Control Register

The fields in this register are defined as follows:

NPE The VM Page Not Present Interrupt Enable field (bit 7) controls the sending of an interrupt when a VM page not present condition is detected. When set to 1, an interrupt is sent to the system microprocessor when the not present condition is detected. When set to 0, the not present condition does not send an interrupt. In both cases, the contents of the appropriate VM Interrupt Status register status bit are updated when the not present condition is detected.

PVE The VM Protection Violation Interrupt Enable field (bit 6) controls the sending of an interrupt when a VM protection violation condition is detected. When set to 1, an interrupt is sent to the system microprocessor when the protection violation condition is detected. When set to 0, the protection violation condition does not send an interrupt. In both cases the contents of the appropriate VM Interrupt Status register status bit are updated when the protection violation condition is detected.

US The User/Supervisor field (bit 2) indicates the privilege level of the currently-executing task. When set to 0, the executing task is at privilege levels 0, 1, or 2 (a supervisor task). When set to 1, the executing task is at privilege level 3 (a user task).

If set to supervisor (0), no protection checking is performed by the coprocessor on page directory and page table protection bits. If set to user (1), checking is performed, and a protection interrupt is sent if permitted access rights are violated.

EV The Enable Virtual Address Lookup field (bit 0) controls the virtual address translation. Subsequent addresses generated by the pel interface hardware are looked up in page tables. If this bit is not set:

- Bit maps must be resident and contiguous.
- The pel map base addresses are physical addresses.
- All addresses generated by the coprocessor are physical addresses.
- Nonpaged operating systems are supported.

Virtual Memory Interrupt Status Register (I/O Address 21x7)

This read/write register has an I/O address of hex 21x7. The Virtual Memory Interrupt Register is directly mapped to I/O address space.

7	6	5	4	3	2	1	0
NPS	PVS	–	–	–	–	–	–

```
  – : Set to 0, Undefined on Read
NPS : VM Page-Not-Present Interrupt Status
PVS : VM Protection-Violation Interrupt Status
```

Figure 3-183. Virtual Memory Interrupt Status Register

The fields in this register are defined as follows:

NPS When a VM page-not-present condition occurs, the VM Page-Not-Present Interrupt Status field (bit 7) is VM page-not-present automatically set to 1. This bit is reset to 0 by writing a 1 to it. This allows the value just read to be written back to clear the bits that were set. Writing a 0 to this bit has no effect.

Resetting this bit (writing a 1) causes the VM hardware to retry page translation. If this bit is to be reset before the not present condition has been repaired, the coprocessor operation must be suspended or terminated, otherwise another not-present interrupt is generated by the same not present condition.

PVS When a VM protection-violation condition occurs, the VM Protection-Violation Interrupt Status field (bit 6) is automatically set to 1. This bit is reset to 0 by writing a 1 to it. This lets the value just read to be written back to clear the bits that were set. Writing a 0 to this bit has no effect.

Resetting this bit (writing a 1) causes the VM hardware to retry page translation. If this bit is to be reset before the protection violation condition has been repaired, the coprocessor operation must be suspended or terminated, otherwise another protection-violation interrupt is generated by the same protection violation condition. Most operating systems do not attempt to recover from a protection violation condition. The coprocessor operation causing this condition is terminated.

XGA Subsystem Identification, Location, and XGA Mode Setting

This section describes XGA subsystem identification and XGA mode setting. Information on VGA mode setting, as well as information on switching from XGA mode to VGA mode, is described in "Switching the XGA Subsystem from XGA to VGA Mode" on page 3-221.

There are two methods of identifying the XGA subsystem. The latest method, introduced with the XGA-2 subsystem, is described in "XGA Display Mode Query and Set (DMQS)." DMQS identifies XGA family subsystems, provides information for Extended Graphics Mode setting, and ensures migration for applications and drivers on future XGA hardware and displays.

The original XGA subsystem did not support DMQS, and applications were written directly to hardware. The software should attempt to identify and locate the XGA subsystem using DMQS. If DMQS is not present, software should use the original XGA specific mechanism described in "Locating and Initializing the XGA Subsystem Without Using DMQS" on page 3-205.

DMQS supports all levels of XGA subsystems, and should not be used to determine existence of XGA-2 function.

In a system with multiple XGA subsystems, if any one XGA subsystem has DMQS capability, it will provide DMQS services for all XGA subsystems recognized. Software should not use the original XGA identification procedure if DMQS BIOS Services are supported (see "DMQS BIOS Interface" on page 3-188).

XGA Display Mode Query and Set (DMQS)

DMQS Architecture Overview

DMQS will identify XGA family subsystems, provide information for Extended Graphics Mode setting, and ensure migration for applications and drivers on future hardware and displays.

DMQS consists of two types of data: DMQS primary data and DMQS display information, contained in the display information files.

The primary data is returned to the software via an INT 10H Video BIOS code point.

The DMQS primary data contains the following information for each XGA instance:

- XGA implementation level identifier
- Location of XGA I/O registers or ports in I/O space
- Location of memory mapped XGA registers in system address space
- Location of 1MB memory mapped XGA aperture
- Location of 4MB memory mapped XGA aperture
- System address at which the XGA accesses video memory
- The composite ID of the attached display (see "Composite Display ID" on page 3-200)
- Amount of video memory available

The Subsystem POST *hooks* the INT 10H Video BIOS to point to two new code points. One code point returns the total size of the DMQS data array for all XGA instances. The other code point returns the DMQS data to the caller's buffer.

Software accesses the new BIOS code points to obtain the DMQS data stored by POST. Using the information from the composite ID field in the DMQS data, the software automatically generates the DMQS display information file name. The DMQS display file is stored in a reserved directory named XGA$DMQS, or in the directory named in the DMQSPATH environment variable.

Note: In some operating systems, an alternative directory or path might be necessary.

Software should first look for the DMQSPATH environment variable to locate the directory containing the DMQS display information files. If the DMQSPATH environment does not exist, software should then look for a directory named XGA$DMQS on the startup disk.

The DMQS display information file contains the following data:

- Display specific data
  - dimensions of the display
  - Display type, such as: color, mono, LCD, or CRT.
- The number of distinct Extended Graphics resolutions available on this display
- For each such resolution available
  - The dimensions (X and Y)
  - The minimum level of XGA subsystem that supports this resolution
  - The register settings to place the XGA subsystem in that particular resolution.

Using the data contained in both the XGA DMQS primary data and the DMQS display information file, the Device Driver/Application can determine:

- The capability and physical characteristics of the XGA family subsystem and display
  - XGA Implementation level
  - Video memory size
  - Physical display dimensions
  - Color/Mono/LCD display information
- The location of all XGA registers and display buffers
- List of the modes available on the subsystem/display combination
- Mode setting data for each mode

With this information, software can set the XGA and attached display into any available Extended Graphics mode without any hard-coded dependencies on displays or subsystems.

If the DMQS display information file cannot be located, software should revert to direct mode setting as described in "Locating and Initializing the XGA Subsystem Without Using DMQS" on page 3-205.

DMQS BIOS Interface

The following two Video INT 10H code points are required to pass DMQS data to the software.

```
Video BIOS INT 10H Software Interrupt function

(AH) = 1FH - XGA Display Mode Query and Set (DMQS)


(AL) = 00H - Read DMQS Data Length

On Return:
   (AL) = 1FH - function supported
   (BX) = Number of bytes of DMQS data
```

Video BIOS INT 10H Software Interrupt function

```
(AL) = 01H - Read DMQS Data
   (ES:DI) - User buffer pointer for return of information

On Return:
   User buffer contains DMQS data
   (AL) = 1FH - function supported

   As many as eight instances of XGA are possible.  One copy of
   the following data structure exists for every instance:

   (DI+00H) word  - Offset in bytes to DMQS data for next XGA
                    instance
   (DI+02H) byte  - Slot number

   (DI+03H) byte  - XGA implementation function level identifier
   (DI+04H) byte  - XGA implementation resolution level
                    identifier

   (DI+05H) word  - Vendor identifier - identifies card vendor
   (DI+07H) word  - Vendor defined field

   (DI+09H) word  - XGA Subsystem I/O register base address
   (DI+0BH) word  - XGA Coprocessor register base address - The
                    location of memory mapped XGA coprocessor
                    registers in system address space
                    Multiply the value of this field by 10h to
                    get the physical address
   (DI+0DH) word  - 1 Megabyte System Video Memory Aperture - The
                    location of 1MB memory-mapped XGA aperture
                    in physical address range.  A value of 0
                    indicates that the aperture is not allocated.
                    Multiply the value of this field by 100000h
                    to get the physical address
   (DI+0FH) word  - 4 Megabyte System Video Memory Aperture - The
                    location of 4MB memory-mapped XGA aperture
                    in physical address range.  A value of 0
                    indicates that the aperture is not allocated.
                    Multiply the value of this field by 100000h
                    to get the physical address.
```

```
(DI+11H) word  - Video Memory Base Address - The location of
                 video memory in XGA system address space.
                 Multiply the value of this field by 100000h
                 to get the physical address.
(DI+13H) word  - Composite ID of the attached display
(DI+15H) byte  - Amount of video memory available, in
                 multiples of 256KB
(DI+16h) dword - Alternate XGA Coprocessor register base
                 address - The location of alternative memory
                 mapped XGA coprocessor registers in protect
                 mode system address space.  A value of 0
                 indicates that the alternative register
                 location does not exist.  A non-zero value is
                 the physical location in system address
                 space.  If present, higher performance is
                 available using the registers at this
                 location.

(DI+offset)    - DMQS Data for further XGA Instances
```

Notes:

1. Although the bits per pixel information has been omitted from the BIOS interface, it can be inferred from the XGA level (current level has 16 bits per pixel maximum), the video memory size, and the number of pixels on the screen. Divide the video memory size in bits by the number of pixels on the screen in a particular mode (pixel height time pixel width) to get the maximum possible bits per pixel. Round off or down to the nearest supported bits per pixel value.
2. All fields will be coded in Intel format (low order byte first in word).
3. These calls return DMQS primary data for all XGA subsystems present in the system.

DMQS Display Information Files

Information in the DMQS display information file helps identify levels of hardware support. Within the individual DMQS mode table, a field identifies the minimum level of XGA hardware that must be present to use that mode. The revision level for the DMQS display information file allows an update to the file to replace an earlier version.

Software should expect to find the DMQS display information files in the XGA$DMQS directory on the startup disk, or alternatively in the directory specified in the DMQSPATH environment variable.

Display Diskette: Displays that support new functions will be packaged with a display diskette to support the subsystem in the extended graphics modes. The display diskette contains the DMQS display information file. The diskette will be a DOS-formatted diskette.

The naming convention for the display information file is the letters MON followed a four-character alphanumeric string which would typically be an ASCII representation of the composite Display ID. These files use the file extension DGS. For a display with an ID of hex 001C, the file name for the display information file is MON001C.DGS.

DMQS Display Information Files Installation: The installation of the display information files is operating system specific. The display files can be installed during device driver installation. Any necessary individual display information files would be copied to a subdirectory named XGA$DMQS. The DMQSPATH environment variable can also be used to locate DMQS display information files in an alternative directory. The path to the XGA$DMQS directory and the means of finding the path is operating system specific.

DMQS Display Information File Structure

Overview: The DMQS display information files are stored in the XGA$DMQS directory of the startup drive unless the user chooses to store the XGA$DMQS subdirectory on another path. These files can be used by applications to determine the display characteristics, the available modes and the register values to use in setting modes.

The XGA$DMQS directory contains a number of individual files, one file for each display available.

As described in "DMQS Customized File" on page 3-201, the composite ID returned in the DMQS primary Data Area can be changed under user control.

If the DMQS display information file cannot be located, software should revert to direct mode setting as described in "Locating and Initializing the XGA Subsystem Without Using DMQS" on page 3-205.

The following figures show the structure of the directory and the individual display information files.

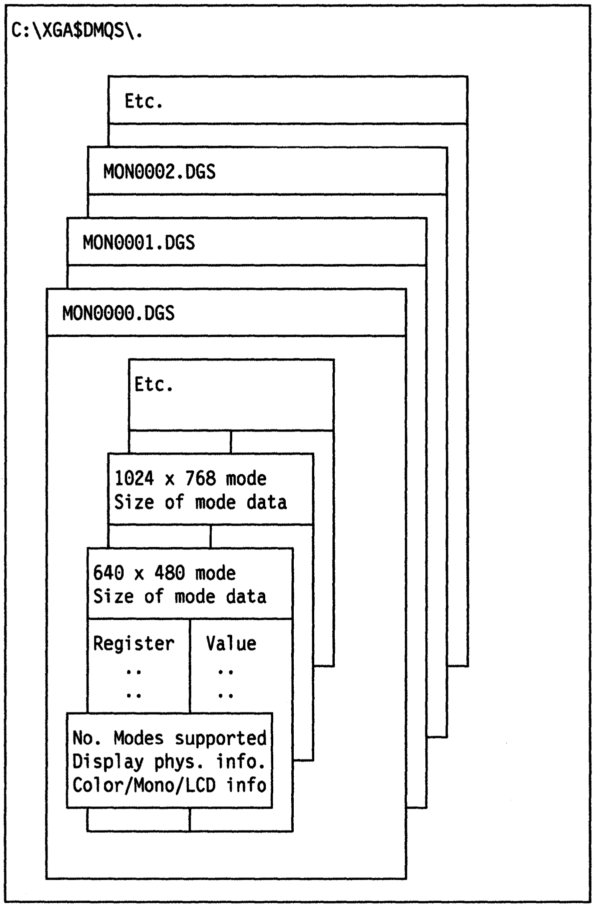


Figure 3-184. DMQS Display Information File Structure

Details: The following table (Figure 3-185) shows the detailed layout of the DMQS display information file. All fields are in hexadecimal, Intel format (low order byte first in word).

Offset	Data Type	Description
00h	Bytes	DGS header data The header must be hardcoded in hex to the following value: EB1B 901D 0045 4453 5900 0000 0000 2020 2000 0022 00EB 0690 FE01 0101 00EB 0190 F8CB This should be ignored.
22h	Word	Total length of data in the file (bytes) This field is used to unpack multiple display information files from a composite file.
24h	Word	Composite ID - used for unpacking a composite display file
26h	Byte	Revision Level - used to control file update
27h	Byte	Number of modes supported by this display
28h	Word	Display Type **00h** Mono CRT **01h** Color CRT **02h** Mono LCD **03h** Color LCD **04h** Mono Borderless capable **05h** Color Borderless capable
2Ah.	Word	Width of Screen in millimeters
2Ch.	Word	Height of Screen in millimeters
2Eh	Bytes	Null-terminated ASCII string which describes the display in user friendly terms. The length of this field is 80 bytes. The string might be less than 80 bytes.
7Eh	Word	Offset of Individual Mode Data from the beginning of this file
80h	Word	Length of first optional extension, including identifier. A length value of 0 indicates no optional extensions exist. A nonzero value indicates that one or more optional extensions are present. Each optional extension consists of a length field, a 16-bit identifier, and the optional extension data. A zero-length field terminates the chain of optional extensions.
82h	Word	First optional extension identifier (if present)
84h	Bytes	First optional extension data (if present)
xxh	Individual Mode Data	First of multiple tables of variable length mode specific data, see Figure 3-186 on page 3-195 for layout.

Figure 3-185. DMQS Display Information File Layout

The following table shows the DMQS display data for individual modes. Multiple instances of this data might exist within the display file, one for each mode available on the applicable display.

Offset	Data Type	Description
00h	Word	Length of Individual Mode Data (bytes in this table)
02h	Word	Screen pixel Width of Mode
04h	Word	Screen pixel Height of Mode
06h	Word	Minimum XGA implementation levels on which this mode is supported. A display might be capable of more modes than an earlier subsystem implementation might be capable of supporting. See "XGA Level Identifier" on page 3-201.
08h	Word	Vendor ID For modes which are unique to a specific vendor, this field must be set. Currently, this field is reserved. It will be set to zero.
0Ah	Word	Reserved This field is intended to be used as a vendor defined field. This field will be activated when the Vendor ID (above) field is set. This field can be defined by the vendor to specify unique modes of operation.
0Ch	Word	Mode function type flags This field identifies special capability the mode might have. **Bit 0** 0 = Normal, 1 = borderless **Bit 1** 0 = Interlaced, 1 = Non-Interlaced **Bits 2-15** Reserved
0Eh	Word	N = Offset in bytes to mode set data from beginning of this table
10h	Word	Mode Pixel Rate This 16-bit value is 4 times the mode pixel rate in megaHertz. For example, a value of 360 indicates a mode pixel rate of 90 MHz.
12h	Word	Mode Line Rate This 16 bit value is 10 times the mode line rate in kiloHertz. For example, a value of 315 indicates a mode line rate of 31.5 kHz.
14h	Word	Mode Frame Rate This 16 bit value is 10 times the mode frame rate in Hertz. For example, a value of 750 indicates a frame refresh rate of 75 Hz.

Figure 3-186 (Part 1 of 2). DMQS Display Information File Mode Data

Offset	Data Type	Description
16h	Word	Length of first optional extension, including identifier. A length value of 0 indicates no optional extensions exist. A non-zero value indicates that one or more optional extensions are present. Each optional extension consists of a length field, a 16-bit identifier, and the optional extension data. A zero-length field terminates the chain of optional extensions.
18h	Word	First optional extension identifier (if present)
1Ah	Bytes	First optional extension data (if present)
Note: Multiple Repetitions of Mode Setting Register Value "triplets," as follows:		
N+00h, N+03h, etc.	Byte	Register Type **00h** Write to XGA Direct Access I/O Register **01h** Write to XGA Indexed Access I/O Register **02h** OR with XGA Indexed Access I/O Register **03h** OR with XGA Direct Access I/O Register **04h** AND with XGA Indexed Access I/O Register **05h** AND with XGA Direct Access I/O Register
N+01h, N+04h, etc.	Byte	Register Offset from Base I/O **Direct** 0-Fh **Indexed** 0-FFh
N+02h, N05h, etc.	Byte	Register Value

Figure 3-186 (Part 2 of 2). DMQS Display Information File Mode Data

Note: The "Mode set data" section is a sequence of register settings required to place the hardware in the desired mode.

Mode setting from the DMQS Display Information File

The "mode set data" section of the DMQS display information file Individual Mode Data includes only the section of the mode setting code that is Display specific, such as CRT Controller settings.

The complete XGA subsystem DMQS mode set sequence consists of

1. Initial XGA subsystem display-independent initialization
2. Display-dependent mode specific initialization, using the "mode set data" from the display information file
3. Final XGA subsystem display-independent initialization

The complete XGA subsystem mode set sequence is shown in the following figure.

XGA Register Name	XGA Reg. ID	Value	Comments
Interrupt Enable	21x4	00	Initial Value
Interrupt Status	21x5	FF	
Operating Mode	21x0	04	Set Extended Graphics Mode
Palette Mask	64	00	Blank Display
Video Memory Aperture Control	21x1	00	Initial Value
Video Memory Aperture Index	21x8	00	Initial Value
Virtual Memory Control	21x6	00	Initial Value
Memory Access Mode	21x9	As reqd.	Mode depth (no. colors)
Note: 1. At this point XGA subsystem mode setting becomes display and mode specific, and the "mode set" register settings read from the Display Configuration file should be written to the appropriate XGA registers. 2. The initial palette should then be loaded, by writing to the appropriate XGA subsystem palette/sprite registers. 3. The video memory should also be initialized at this point, to avoid random data appearing when the palette mask is set to make the current display pel map contents visible.			
Sprite Control	36	00	Initial Value
Start Addr Low	40	00	Initial Value
Start Addr Med	41	00	Initial Value
Start Addr High	42	00	Initial Value
Display Pel Map Width Low	43	A0	As required
Display Pel Map Width High	44	00	As required
Display Mode 2	51	04	As required
Border Color	55	00	Initial Value
Palette Mask	64	FF	Make visible

Figure 3-187. DMQS Extended Graphics Mode Register Settings

DMQS Information File Optional Extensions

The following optional extensions to the display information file have been defined. For the location of the optional extensions within the file see Figure 3-186 on page 3-195.

Display Color Characteristics: The precise color characteristics of the display can be encoded in the DMQS display information file, using an optional extension, to allow software to encode the palette or select colors to satisfy specific color requirements.

This color information consists of X,Y coordinate pairs of the primary color points in the CIE color chart, their peak luminance values, and their gamma correction values.

Each CIE chart coordinate pair consists of two 16-bit fractional integers in the range of 0 to 1. Each such integer is a fraction of 1, so that a field value of hex 8000 is equivalent to 0.5, and hex FFFF is (almost) 1.0.

Each peak luminance value is a 16-bit unsigned integer, which is that color's peak luminance multiplied by 10. For example, a field of decimal 175 represents an actual peak luminance value of 17.5.

Each gamma correction value is a 16-bit unsigned integer, which is that color's gamma correction factor multiplied by 100. For example, a field value of decimal 220 represents a gamma correction of 2.2.

The color characteristics data is defined as follows:

Offset	Data Type	Description
00h	Word	Length of following optional extension data. 22h is the fixed length of the display color characteristics data.
02h	Word	Optional extension Identifier 0001h identifies display color characteristics data
04h	Word	Red Color Point
06h	Word	The X,Y coordinate of the color red in the CIE color chart
08h	Word	Red Peak Luminance
0Ah	Word	Red Gamma Value
0Ch	Word	Green Color Point
0Eh	Word	The X,Y coordinate of the color green in the CIE color chart
10h	Word	Green Peak Luminance
12h	Word	Green Gamma Value
14h	Word	Blue Color Point
16h	Word	The X,Y coordinate of the color blue in the CIE color chart
18h	Word	Blue Peak Luminance
1Ah	Word	Blue Gamma Value
1Ch	Word	White Color Point
1Eh	Word	The X,Y coordinate of the color white in the CIE color chart
20h	Word	White Peak Luminance
22h	Word	White Gamma Value

Figure 3-188. DMQS Display Color Characteristics

Pre-selected Colors: A list of pre-selected colors certified to comply with ISO 9241 may be encoded in the DMQS display file, using an optional extension. The colors in this pre-selected list are certified by the display manufacturer to be sufficiently far apart on the CIELUV space-distance measurement system. Software may use these pre-selected colors in ISO-conforming applications, or may alternatively use the color characteristics data described in "Display Color Characteristics" on page 3-197 to generate an alternative list.

ISO 9241 specified that sets of 6 and 11 colors be available. In this pre-selected list, a single set of 11 colors is presented, of which the first 6 comprise the set of 6, and all 11 comprise the set of 11 compliant colors.

The list consists of an 11-entry array, each entry consisting of a 13-character text string describing the color, and three 1-byte values representing the RGB values used to generate the color.

Offset	Data Type	Description
00h	Word	Length of following optional extension data 0B2H is the fixed length of the pre-selected color data
02h	Word	Optional extension Identifier 002H identifies the pre-selected color list
04h	13-char string	Color name of first pre-selected color
11h	byte	Red color intensity of first pre-selected color
12h	byte	Green color intensity of first pre-selected color
13h	byte	Blue color intensity of first pre-selected color
14h	13-char string	Color name of second pre-selected color
21h	byte	Red color intensity of second pre-selected color
22h	byte	Green color intensity of second pre-selected color
23h	byte	Blue color intensity of second pre-selected color
24h		11 colors in all

Figure 3-189. DMQS Display Pre-selected Colors

Composite Display ID

The composite ID of the attached display is derived during POST, and is made available to the software in the DMQS primary data Area, as described in "DMQS BIOS Interface" on page 3-188.

Each display with unique function or characteristics and therefore a unique display information file has a unique display ID. The display presents the display ID through pins on the display connector. The details of its derivation are shown in "Display Type Detection" on page 3-209.

XGA Level Identifier

The XGA level identifier is returned as part of the BIOS Interface as described in "DMQS BIOS Interface" on page 3-188. The XGA level identifier field consists of two bytes. One is the functional level identifier, which identifies the level of the Display Controller. The next is the resolution level identifier, which identifies the level of the Serializer Palette DAC chip (in hex).

Functional Level Identifier Identifies the level of the Display Controller chip

03	Base XGA implementation (XGA Subsystem)
05	XGA-2 implementation level of function

Resolution Level Identifier Identifies the level of the Serializer Palette DAC chip (in hex).

00	Base XGA implementation (XGA Subsystem) (Maximum 45 MHz Pel rate)
03	XGA-2 Serializer Palette DAC. (Maximum 90 MHz Pel rate)

DMQS Customized File

An additional file in the DMQS file directory should be consulted to ascertain additional XGA customized parameters, prior to reading the DMQS display configuration file. This file, XGASETUP.PRO, (if present) contains system customized information, as follows:

- Specifies a display ID alias for a particular slot. This overrides the display ID physically presented by the display, and specifies an alternate DMQS Display Configuration file name to be used instead.
- Identifies the slot to be used as the primary graphics display. In a multiple XGA system, the software normally chooses which XGA subsystem to use, based on factors such as screen size and XGA subsystem functionality. This entry overrides the software default, and forces the nominated slot to be the primary XGA subsystem, rather than that chosen arbitrarily by the software.

DMQS Customized File Example: Below is an example of a DMQS customized file:

```
/**********************************************************************
/*
/* FILENAME: XGASETUP.PRO
/*
/* DESCRIPTION: Profile to set XGA DMQS preferences on current system.
/*
/* LAST MODIFIED: 04/09/92 at 16:13:38
/*
/**********************************************************************

/**********************************************************************
<SLOT, NUMBER=5, MONITOR_ID=F0FF>IBM 8515, 8516

/**********************************************************************
<SLOT, NUMBER=8, MONITOR_ID=EXMP> Special DMQS DIF file

/**********************************************************************
<STARTUP, NUMBER=5>
```

The profile above does the following:

1. Informs the software that the display in slot 5 is a display of type F0FF and the display information file MONF0FF.DGS should be used to obtain information about the installed display.
2. Informs the software that the display in slot 8 is a display of type "EXMP" and the display information file MONEXMP.DGS should be used to obtain information about the installed display.
3. Identifies the slot (5 in this example) which holds the XGA subsystem to be used as the primary graphics display.

Note: If any of the slot numbers or display IDs are invalid when read, the tag will be ignored and it will not have any affect on how the XGA subsystem hardware is initialized.

XGA Subsystem DMQS Customized Tags: The syntax for the generalized DMQS customized tag is as follows:

```
EXAMPLE: <TAG, PARAMETER = PARAMETER-VALUE> TAG-DATA
          │   │            │               │
          │   │            │               └─End Tag Character
          │   │            │
          │   │            └─Parameter Equals Character
          │   │
          │   └─Parameter Separator
          │
          └─Begin Tag Character
```

Figure 3-190. DMQS Customized Tag Syntax

- Any number of blanks, new lines, or both, can be used to separate the begin-tag character, the end-tag character, tags, parameters, and parameter values. A single tag with many parameter values can be flowed over multiple lines, and can space the elements within the tag out for readability.
- The tag is the first text item following the begin tag character (<) and ending with the first comma (,) or the end tag character (>).
- If any parameters exist, they are always separated by a comma (,). Otherwise, if no parameters exist for the tag, the tag is immediately followed by an end tag character (>).
- Parameters are always assigned a value by the parameter equals character (=).
- There are two methods of denoting a comment; a semicolon (;), and a slash-asterisk (/*) combination. Anything following the semicolon or the slash-asterisk is ignored for that line.

Individual tags are defined as follows:

SLOT Syntax Diagram:

```
<SLOT, NUMBER=[slot number],MONITOR_ID=[display ID]>Text
```

This tag allows the user to specify a display ID override for displays attached to XGA subsystems with DMQS support. The display ID value in this customized file will override the physical value read from the display by the XGA subsystem.

Parameters:

NUMBER = [slot number] (REQUIRED) An integer value that indicates the system slot number that the XGA subsystem occupies. A slot number of "0" is the planar, "1" is slot 1, "2" is slot 2, and so on. If this is an invalid number or there is no XGA subsystem in the specified slot, this tag will be ignored.

MONITOR_ID = [display ID] (REQUIRED) A four-character, alphanumeric string that will be used to construct the name of the DMQS Display Configuration file to be loaded in place of the default file. This value will be used to generate a file name of the form MONXXXX.DGS where XXXX will be the display ID value specified in the tag.

Text (OPTIONAL) A comment field that should be ignored by software.

STARTUP Syntax Diagram:

```
<STARTUP, NUMBER=[slot number]>Text
```

This tag allows the user to specify which particular XGA subsystem is to be used by an XGA mode application, where the default chosen by the application is inconvenient.

Parameters:

NUMBER = [slot number] (REQUIRED) An integer value that indicates the system slot number that the XGA subsystem occupies. A slot number of "0" is the planar, "1" is slot 1, "2" is slot 2, and so on. If this is an invalid number, or there is no XGA subsystem in the specified slot, this tag will be ignored.

Text (OPTIONAL) A comment field that should be ignored by software.

Locating and Initializing the XGA Subsystem Without Using DMQS

This section describes the original method for XGA subsystem identification and initialization. Software should initially attempt to identify the XGA subsystem using DMQS, as described in "XGA Display Mode Query and Set (DMQS)" on page 3-185. Only when DMQS has been found to be not supported in the system, or if a DMQS display information file cannot be found, should software resort to the method of XGA subsystem identification and mode setting described in this section.

Software should not attempt to use this method in addition to DMQS, as DMQS (if found) will provide support for both XGA and XGA-2 subsystems.

The procedure is outlined here, and more detail is given later in this section.

1. Identify if XGA subsystem is present by examining the POS ID of each subsystem.
2. Locate the various I/O spaces of the XGA subsystems spaces by decoding the XGA subsystem POS data.
3. Read the display ID to determine the attached display type.
4. Determine the amount of VRAM installed on the XGA subsystem.
5. Determine the modes available on the attached display.
6. Set the XGA subsystem into the required XGA mode, either 640 x 480 or 1024 x 768 resolution.
7. Handle any VGA primary subsystem considerations, as described in "VGA Primary Subsystem Considerations" on page 3-216.

This procedure should be repeated, if necessary, until all XGA subsystems in the system have been identified.

XGA Subsystem Identification

To identify all XGA subsystems, run Setup Mode on every subsystem in the system, including the system board video subsystem, and examine their POS IDs to locate any XGA subsystem.

For option cards, the procedure is described in System Services BIOS call INT 15h, AH=C4h Programmable Option Select in the *IBM Personal System/2 and Personal Computer BIOS Interface Technical Reference*.

For the system board video subsystem, a different procedure is necessary. To place the system board video subsystem in setup mode, write hex 0DF to port 94H; to enable it, write hex 0FF to port 94H.

Interrupts must be disabled for the entire period of time that each subsystem is in setup mode.

The POS IDs for all subsystems in the system must be read and examined to locate all the XGA subsystems in the system.

The following POS IDs have been allocated to the XGA subsystem and follow-on XGA register compatible subsystems:

- 8FD8h to 8FDBh
- 8FD0h to 8FD3h
- VESA reserved IDs, as follows:
  - 0240h to 027Fh
  - 0830h to 0A7Fh
  - 0A90h to 0BFFh

Check for these POS IDs when identifying the XGA subsystem in the system.

After successfully matching POS IDs, read the remainder of the POS data bytes for that subsystem. This data is used to calculate the location of the XGA subsystem registers and display buffers in I/O and physical system memory address space. Descriptions of the POS data bit assignments are in "XGA POS Registers" on page 3-164. For future compatibility, mask out all reserved and unused POS data bits before using the data for these calculations.

Location of XGA Subsystem I/O Spaces

See "XGA POS Registers" on page 3-164 for the technical background to the following register and address space calculations.

ROM Address: Calculate the ROM address from POS data as follows:

```
ROM Address = (ROM Address field × hex 2000) + hex 0C0000
```

The ROM Address field is read from POS Register 2, bits 4 to 7.

XGA Coprocessor Registers: The XGA coprocessor registers are referenced from a base address. This address depends on the Instance (0 – 7) of the XGA subsystem and the ROM address calculated as shown in "ROM Address." The Coprocessor register base address is calculated as follows:

```
(((128 × Instance) + hex 1C00) + ROM address)
```

The Instance is read from POS Register 2, bits 1 to 3.

For example:

```
Assuming Instance = 6 and ROM address = hex 0C0000, the Coprocessor
Base Address is hex 0C1F00.
```

I/O Registers: The XGA I/O registers are referenced from a base I/O address. The I/O address is calculated as follows:

```
Hex 21x0 (where x is the Instance)
```

The Instance is read from POS Register 2, bits 1 to 3.

The Video Memory Base Address: The Video Memory Base Address is calculated from the Video Memory Base Address field in POS Register 4, and from the Instance.

Figure 3-191 and Figure 3-192 show how these two values combine to provide the video memory base address.

31 - 25	24 - 22	21 0
Video Memory Base Address	Instance 0 – 7	4MB of Addressable Memory

Figure 3-191. XGA Video Memory Base Address

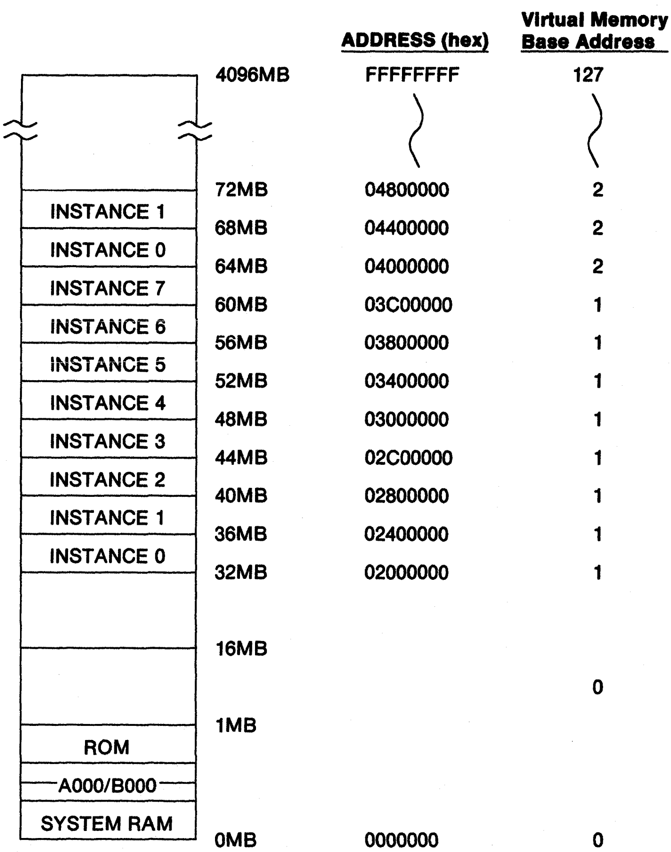


Figure 3-192. The XGA Video Memory Base Address Diagram

The Video Memory Base Address field defines a 32MB address range and the Instance defines a 4MB address range within the 32MB range.

For example:

```
Assuming Instance = 6 and the Video Memory Base Address field = 1,
the Video Memory Base Address is hex 03800000.
```

The video memory base address, when calculated, serves two separate purposes:

4MB System Video Memory Aperture: If enabled (read from bit 0 in POS Register 4 to determine if the aperture is enabled), the 4MB system video memory aperture is located at this address in physical system address space. If virtual addressability to this range of physical address space can be achieved, the entire video memory can be accessed through this aperture at this address.

Video Memory Location in XGA Address Space: This address is used to identify video memory to the XGA coprocessor. Its significance and use are described in "Video Memory Address Range" on page 3-232.

1MB Aperture Base Address: The 1MB aperture base address is calculated from the 1MB Aperture Base Address field in POS Register 5, bits 0 to 3.

```
If (1MB Base field ≠ 0)
    1MB Aperture Base Address = 1MB Base field × hex 100000

If (1MB Base field = 0) 1MB Aperture is disabled.
```

Display Type Detection

To determine what type of display is attached to the video subsystem, read the display identification number, or ID. This ID is used to obtain information about the display, such as the resolutions supported, whether it is monochrome or color, and possibly the size of the screen.

The ID for each display is a 16-bit number, and, in most cases, uniquely identifies the display type. Some displays that have similar characteristics, but are not the same model, have the same ID.

The recommended method of obtaining the display ID is by use of a BIOS call, `INT 10H, (AH) = 1FH - XGA Display Mode Query and Set`

(DMQS). See "XGA Display Mode Query and Set (DMQS)" on page 3-185. If it is necessary to read the display ID explicitly, the following procedure must be followed.

The display ID is read from the Display ID and Comparator register, which returns 4 ID bits at a time. Four reads must be performed in order to obtain all 16 bits. The components of the ID are selected by manipulating the values of Horizontal Sync and Vertical Sync that are output to the display. Therefore, the ID can only be read when disruption of these signals can be tolerated, such as power-on time, or when changing display modes.

After setting the required Sync Polarity (SP field in Display Control 1 Register) to any of the various combinations of Horizontal and Vertical Sync listed below, it is necessary to wait for 15 microseconds for this change to take effect before display ID can be read. This is best achieved by doing five consecutive reads or writes to any byte-wide XGA I/O port.

The display ID Reading sequence is as follows:

1. Prepare the CRTC for reset (Display Control 1 Register - Index 50 - DB field = 01 binary)
2. Reset the CRTC (Display Control 1 Register DB field = 00)
3. Set Sync Polarity (SP field in Display Control 1 Register) to 01 binary. This sets VSYNC to 0 and HSYNC to 1.
4. After a 15 microsecond wait, read the display ID bits from Display ID and Comparator Register (Index 52). Place them in a hex variable A.
5. Set SP to 10 binary. This sets VSYNC to 1 and HSYNC to 0.
6. After a 15 microsecond wait, read the display ID bits again. Place them in a hex variable B.
7. Set SP to 00 binary. This sets VSYNC to 0 and HSYNC to 0.
8. After a 15 microsecond wait, read the display ID bits again. Place them in a hex variable C.
9. Set SP to 11 binary. This sets VSYNC to 1 and HSYNC to 1.
10. After a 15 microsecond wait, read the display ID bits again. Place them in a hex variable D.

Assemble the 16 bits into four *nibbles* (a nibble is half a byte), one for each MID pin, from MSB to LSB, as shown in Figure 3-193 on page 3-211. The resulting four-hex-digit number (from MID bit 3 to bit 0) is the display ID.

```
After reading 4 nibbles:

|        A        |        B        |        C        |        D        |
|ID3|ID2|ID1|ID0| |ID3|ID2|ID1|ID0| |ID3|ID2|ID1|ID0| |ID3|ID2|ID1|ID0|
|(3)|(3)|(3)|(3)| |(2)|(2)|(2)|(2)| |(1)|(1)|(1)|(1)| |(0)|(0)|(0)|(0)|

where ID3  = Display ID 3, bit 1  etc.
      (1)

                    Step:  3   5   7   9
                    ---------------------
                       H:  1   0   0   1
                       V:  0   1   0   1

                           ┌───────────── Step  4. Read to A
                           │   ┌───────── Step  6. Read to B
                           │   │   ┌───── Step  8. Read to C
                           │   │   │   ┌─ Step 10. Read to D
                           │   │   │   │
                           V   V   V   V
    Display ID bit 3     |   |   |   |   |   ───> Extract MID 3
    Display ID bit 2     |   |   |   |   |   ───> Extract MID 2
    Display ID bit 1     |   |   |   |   |   ───> Extract MID 1
    Display ID bit 0     |   |   |   |   |   ───> Extract MID 0
                           A   B   C   D

After rearrangement:

|  Display ID 3   |  Display ID 2   |  Display ID 1   |  Display ID 0   |
| 3 | 2 | 1 | 0 | 3 | 2 | 1 | 0 | 3 | 2 | 1 | 0 | 3 | 2 | 1 | 0 |
|ID3|ID3|ID3|ID3|ID2|ID2|ID2|ID2|ID1|ID1|ID1|ID1|ID0|ID0|ID0|ID0|
|(3)|(2)|(1)|(0)|(3)|(2)|(1)|(0)|(3)|(2)|(1)|(0)|(3)|(2)|(1)|(0)|

This is the 16 bit display ID.
```

Figure 3-193. Reading the Display ID.

Figure 3-195 on page 3-213 shows a list of displays and their associated IDs.

Video Memory Size Determination

There are two ways to determine the size of video memory. Both ways rely on a write-readback-check, in which a particular value is written to a key location in each memory module. This value is then read to determine whether the written value was returned.

- Use the system processor to write a value through an aperture to the word at offset 768KB into video memory. This technique assumes that the system video memory real mode aperture is available. See the sample code in the following figure.

```
;* Assume GS points to start of A0000 Real mode aperture
;* and VGA subsystem is in text mode so A0000 Real mode
;* aperture is available for this operation.
;* Where registers are shown (21x0h, for instance), the register should
;* be filled in with the appropriate I/O port address after determining
;* the location of the XGA subsystem in I/O space
;*
;* First put the subsystem PARTIALLY in extended graphics mode
;* to allow use of the system video memory aperture
       mov   al,0
       mov   dx,21x4h       ; disable XGA interrupts
       out   dx,al
;
       mov   ax,0064h
       mov   dx,21xAh       ; Blank palette
       out   dx,ax          ; indexed XGA register 64h
;
       mov   ax,04h
       mov   dx,21x0h       ; Set subsystem in Extended Graphics Mode
       out   dx,al
;
       mov   al,01h
       mov   dx,21x1h       ; Locate video memory Aperture at A0000
       out   dx,al
;
       mov   dx,21x8h       ; System video memory indx reg.
       mov   al,0ch         ; Offset 768K
       out   dx,al          ;
;
       mov   byte ptr gs:[0],0A5h ; Set byte to A5h
       mov   byte ptr gs:[1],0h   ; Avoid shadows on data lines
;
       cmp   byte ptr gs:[0],0A5h  ; Test against value written
       jne   vram_512k      ; 512K video memory only
;
       mov   byte ptr gs:[0],5Ah ; Set byte to 5Ah
       mov   byte ptr gs:[1],0h   ; Avoid shadows on data lines
;
       cmp   byte ptr gs:[0],0A5h  ; Test against value written
       je    vram_1Meg       ; 1 Meg if still matches
       jmp   vram_512k       ; Otherwise 1/2 meg found
```

Figure 3-194. Video Memory Size Determination

- Use the XGA subsystem PxBlt capability to perform a test similar to the previous example. Transfer a constant color to the location in video memory; then transfer that value back from video memory to system memory using bus master commands.

  This technique works regardless of the availability of a system video memory aperture. However, it requires physical addressability to a location in system memory for the bus master functions.

Extended Graphics Modes Available

The following figure shows graphic displays available, their display ID and type, and their maximum resolution.

Composite Display ID (hex)	Example Displays	Screen Size (inches)	Display Type	Maximum Resolution
FF0F	8503	12	Mono	640 x 480
FFF0	8513 8512 8518	12 14 14	Color	640 x 480
F0FF	8515 8516	14	Color	1024 x 768
F00F	8604 8507	15 19	Mono	1024 x 768
F0F0	8514	16	Color	1024 x 768
90F0	8517	17	Color	1024 x 768
Note: A composite display ID of hex FFFF indicates NO display.				

Figure 3-195. Availability of Extended Graphics Modes

The following figure shows display resolution for graphic displays, and the various capabilities using two sizes of video memory.

Display Resolution	Display Type	Capability in 512KB Memory	Capability in 1MB Memory
640 x 480	Mono	640 x 480 x 64 Grays	640 x 480 x 64 Grays
640 x 480	Color	640 x 480 x 256 Colors	640 x 480 x 256 Colors 640 x 480 x 64K Colors
1024 x 768	Mono	640 x 480 x 64 Grays 1024 x 768 x 16 Grays	640 x 480 x 64 Grays 1024 x 768 x 16 Grays 1024 x 768 x 64 Grays
1024 x 768	Color	640 x 480 x 256 Colors 1024 x 768 x 16 Colors	640 x 480 x 256 Colors 640 x 480 x 64K Colors 1024 x 768 x 16 Colors 1024 x 768 x 256 Colors

Figure 3-196. Capability of Graphic Displays

Extended Graphics Mode Setting Procedure

To set the XGA subsystem into Extended Graphics mode, the configuration must be capable of supporting the required mode as listed in "Extended Graphics Modes Available" on page 3-213.

XGA Register Name	XGA Reg. ID	Oper	Color Mode Value (hex)				Comments
			1024 x 768 x 256	1024 x 768 x 16	640 x 480 x 256	640 x 480 x 64K	
Interrupt Enable	21x4	=	00	00	00	00	Initial Value
Interrupt Status	21x5	=	FF	FF	FF	FF	
Operating Mode	21x0	=	04	04	04	04	Set Extended Graphics Mode
Palette Mask	64	=	00	00	00	00	Blank Display
Video Mem Aperture Control	21x1	=	00	00	00	00	Initial Value
Video Mem Aperture Index	21x8	=	00	00	00	00	Initial Value
Virt Mem Control	21x6	=	00	00	00	00	Initial Value
Memory Access Mode	21x9	=	03	02	03	04	Initial Value
Display Mode 1	50	=	01	01	01	01	Prepare for reset
Display Mode 1	50	=	00	00	00	00	Reset CRT Ctrl
Horiz Total Low	10	=	9D	9D	63	63	)
Horiz Total High	11	=	00	00	00	00	)
Horiz Display End Low	12	=	7F	7F	4F	4F	)
Horiz Display End High	13	=	00	00	00	00	)
Horiz Blank Start Low	14	=	7F	7F	4F	4F	)
Horiz Blank Start High	15	=	00	00	00	00	)
Horiz Blank End Low	16	=	9D	9D	63	63	)
Horiz Blank End High	17	=	00	00	00	00	)
Horiz Sync Start Low	18	=	87	87	55	55	)
Horiz Sync Start High	19	=	00	00	00	00	)
Horiz Sync End Low	1A	=	9C	9C	61	61	)
Horiz Sync End High	1B	=	00	00	00	00	)
Horiz Sync Posn	1C	=	40	40	00	00	)
Horiz Sync Posn	1E	=	04	04	00	00	)
Vert Total Low	20	=	30	30	0C	0C	)
Vert Total High	21	=	03	03	02	02	) XGA CRT
Vert Disp End Low	22	=	FF	FF	DF	DF	) Controller
Vert Disp End High	23	=	02	02	01	01	) param
Vert Blank Start Low	24	=	FF	FF	DF	DF	)
Vert Blank Start High	25	=	02	02	01	01	)
Vert Blank End Low	26	=	30	30	0C	0C	)
Vert Blank End High	27	=	03	03	02	02	)
Vert Sync Start Low	28	=	00	00	EA	EA	)
Vert Sync Start High	29	=	03	03	01	01	)
Vert Sync End	2A	=	08	08	EC	EC	)
Vert Line Comp Low	2C	=	FF	FF	FF	FF	)
Vert Line Comp High	2D	=	FF	FF	FF	FF	)
Sprite Control	36	=	00	00	00	00	Initial Value
Start Addr Low	40	=	00	00	00	00	Initial Value
Start Addr Me	41	=	00	00	00	00	Initial Value
Start Addr High	42	=	00	00	00	00	Initial Value
Buffer Pitch Low	43	=	80	40	50	A0	
Buffer Pitch High	44	=	00	00	00	00	
Clock Select	54	=	0d	0d	00	00	
Display Mode 2	51	=	03	02	03	04	
Ext Clock Select	70	=	00	00	00	00	
Display Mode 1	50	=	0F	0F	C7	C7	
Note: Initial Palette loading must be done at this point, by writing to the appropriate XGA subsystem palette/sprite registers. The video memory must also be initialized at this point, to avoid random data appearing when the palette mask is set to make the current display pel map contents visible.							
Border Color	55	=	00	00	00	00	Initial Value
Palette Mask	64	=	FF	FF	FF	FF	Make visible

Figure 3-197. Extended Graphics Mode Register Settings

VGA Primary Subsystem Considerations

Where a single XGA subsystem is providing both VGA and Extended Graphics function, particularly on a system with single display subsystem or display, an application using the subsystem in Extended Graphics mode takes on a number of additional systems responsibilities, particularly in the DOS environment.

Before switching the subsystem into Extended Graphics mode, examine the Operating Mode register, bits 0 and 2, to determine whether the XGA subsystem is enabled in VGA mode or 132-column text mode.

If the XGA subsystem is not enabled in VGA mode, the subsystem is operating as an auxiliary video subsystem and systems messages can be left to the primary VGA source. In this case, the XGA subsystem must not be put into VGA mode unless the current VGA is disabled.

If the XGA subsystem is enabled in VGA mode, the subsystem is the system primary video subsystem, and a number of special considerations apply.

Chaining the INT 10H Video BIOS Handler: The application must chain the INT 10H Video interrupt handler and display calls to the INT 10H handler while the application is using the XGA subsystem in Extended Graphics mode.

There are a number of hot-key and error handlers that might attempt to communicate with the VGA while the XGA subsystem is in Extended Graphics mode, so code must be written to handle such calls.

The majority of calls to the INT 10H handler can be ignored (simply return to the caller) while the XGA subsystem is in Extended Graphics mode, but some calls require correct handling.

(Ah) = 00h Set Mode

Set mode calls can come from a critical error or nonmaskable interrupt (NMI) handler. Because failure to restore VGA mode can result in the loss of critical error data or dialogue, applications must allow the mode set operation.

For normal VGA mode setting procedure to occur, the INT 10H handler must restore the subsystem as necessary to VGA mode before chaining on to the next INT 10H interrupt handler.

Note: The NMI handler traditionally issues a *Return Current Video State* to determine the current mode, followed by a *Video Set Mode* to the current mode.

If the INT 10H Video interrupt handler of the application detects a video set mode with AL = 7Fh, mode 03h should be substituted after restoring the subsystem to VGA mode.

(Ah) = 0Fh Return current video state

In Extended Graphics mode, the application's INT 10H interrupt handler should return a current mode of 7Fh in AL, to indicate that the subsystem is in a non-VGA mode.

This is a special mode number assigned for this purpose.

INT 24h, Critical Error Handler: The application should trap and revector the DOS critical error handler interrupt vector (INT 24h), as described in the *DOS Technical Reference*. The application is then notified on DOS critical errors.

The critical error handler of the application should save the video state of the subsystem (as far as necessary), and put the XGA subsystem into VGA mode before chaining on, using the saved vector to the original critical error handler. This lets the dialogue between the critical error handler and the user proceed normally.

After returning from the chained critical error handler, the critical error handler of the application must examine the return code in AL to determine the appropriate action.

0, 1, 3 Control is returned to the application. Put the XGA subsystem back into Extended Graphics mode and restore the video state as necessary.

2 The program is ended by the system. Leave the XGA subsystem in VGA mode and return.

Alternatively, the application can take over the entire critical error handling dialogue in Extended Graphics mode.

Note: The C language *signal* function can (in some implementations) be used to intercept the critical error handler for this purpose.

INT 23h, Ctrl+Break Exit Address: The application should trap and revector the DOS Ctrl+Break exit address interrupt vector (INT 23h), as described in the *DOS Technical Reference Manual*. The application is notified when the Ctrl+Break key combination is entered.

If the application is not otherwise intercepting Ctrl+Breaks, the XGA subsystem must be put back into VGA mode before chaining on, using the saved vector to the original Ctrl+Break handler. This lets the normal Ctrl+Break handler proceed.

Alternatively, the application can take over the entire Ctrl+Break handling in Extended Graphics mode.

Note: The C language *signal* function can be used to intercept the Ctrl+Break handler for this purpose.

INT 21h, Function 4Ch, Program Terminate Function: The subsystem must be in VGA mode on program termination, regardless of the how the program terminates or is terminated.

To ensure that this is done, the application must trap and revector the normal DOS program terminate function, DOS INT 21h function 4Ch, as described in the *DOS Technical Reference*. On receiving notice of program termination, the application must put the subsystem back into VGA mode and unhook all other hooked interrupt vectors before chaining on for the remainder of program termination handling.

DOS INT 21h function 4Ch is the conventional method used by all programs to terminate. By trapping the DOS function interrupt (INT 21h) and monitoring calls to the program terminate function (4Ch), all routes for a program to terminate normally must be covered.

Note: There are other program terminate functions, including:

- INT 20h
- INT 27h
- INT 21h function 00h
- INT 21h function 31h

For complete coverage, these calls can be revectored and trapped, but they are not used as commonly as the INT 21h function 4Ch.

All other functions must be passed to the previous DOS function handler using the saved-interrupt vector.

On detecting a call to function 4Ch, put the XGA subsystem into VGA mode before chaining on, using the saved vector to the original DOS function handler. This lets the DOS program-terminate function proceed normally.

Note: The C language *atexit* function can be used for this purpose.

Multiple XGA Subsystems
Up to eight XGA subsystems can be installed in a system.

Multiple XGA subsystems can coexist in Extended Graphics mode. Each instance occupies its own separate ranges of I/O and memory space. An application written to exploit multiple XGA subsystems in this mode can access each Instance of the subsystem without enabling and disabling the subsystem between accesses.

To comply with the restriction on VGA coexistence described in "XGA Subsystem Coexistence with VGA," a multiple display subsystem application must record, on initialization, the XGA subsystem (if any) originally in VGA mode. On application termination, only *that* subsystem should be returned to VGA mode.

VGA Modes

Where the XGA subsystem is being used as a standard VGA, the XGA subsystem is VGA compatible, and mode setting should be performed using the normal INT 10H Video BIOS services.

Where the XGA subsystem is being used in XGA mode, or coexisting with a VGA, or other XGA subsystems in VGA mode, this section includes information on mode switching from XGA to VGA mode, and guidelines on VGA coexistence.

On switching between XGA and VGA modes, and also between some VGA modes, contents of video memory might be lost or re-ordered. Information on this is included in "Effects of VGA and XGA Mode Setting on Video Memory" on page 3-226.

XGA Subsystem Coexistence with VGA

Because the VGA uses fixed I/O and memory mapped address spaces, only one VGA can be active at a time in a system. When the XGA subsystem is installed alongside a VGA or another XGA subsystem, only one of the VGA-capable subsystems can be enabled at one time. Software must not switch the XGA subsystem from XGA mode to VGA mode if another VGA subsystem is already in VGA mode.

An application can use multiple coexisting VGA or XGA subsystems in VGA mode only by alternately disabling and enabling the various VGAs.

Do not enable more than one VGA concurrently. A disabled (or inactive) VGA retains its visible displayed data, and the overall effect is that of a multiple VGA application.

To successively enable and disable multiple coexisting XGA subsystems in VGA mode, use the Operating Mode register (21x0).

Switching the XGA Subsystem from XGA to VGA Mode

To put either the XGA subsystem or the XGA-2 subsystem back into VGA mode (subject to the rules discussed in "VGA Primary Subsystem Considerations" on page 3-216), perform the following operations:

1. Clear the first 256KB of video memory contents. This avoids screen flash caused by random data being present when switching into VGA mode.
2. Write data to the registers in the following sequence:

Value (hex)	Oper	XGA Reg	VGA Reg	Comments
00	=	21x1		Aperture Control register
00	=	21x4		Interrupt disable
FF	=	21x5		Clear interrupts
FF	=	64		Palette Mask register
15	=	50		Enable VFB, prepare for reset
14	=	50		Enable VFB, reset CRT controller
00	=	51		Normal scale factors
04	=	54		Select VGA oscillator
00	=	70		External Clock (VGA)
20	=	2A		Ensure no VSync interrupts
01	=	21x0		Switch to VGA mode
01	=		3C3	Enable VGA address decode

Figure 3-198. VGA Mode Write Sequence

3. Set the number of lines in VGA mode (if required) using Video BIOS INT 10H AH=12H.
4. Set the required VGA mode using Video BIOS INT 10H AH=00H, set mode.

The XGA subsystem is now in VGA mode.

Smooth Scrolling of VGA and 132-Column Text Modes

Smooth vertical scrolling of both VGA modes (text and graphics) and the XGA subsystem 132-column text modes is possible by manipulation of the following VGA registers.

Horizontal Pel Panning Register See "Horizontal Pel Panning Register" on page 2-92 for a detailed description.

This register is used in horizontal smooth scrolling to move the visible data within a byte or character. This register is incremented or decremented until its limit is reached. This register is then reset to its start value, and the Start Address High and Low Registers are incremented or decremented by 1 unit.

Preset Row Scan Register See "Preset Row Scan Register" on page 2-61 for a detailed description.

This register is used in vertical scrolling in text modes only. The Starting Row Scan Count field is used to offset the visible display buffer start vertically within a row of text mode characters.

This register is incremented or decremented until its limit (defined by the text mode character box height) is reached. This register is then reset to its start value, and the Start Address High and Low Registers are incremented or decremented to the start of the next row of characters.

Start Address High and Low Registers See "Start Address High Register" on page 2-65 for a detailed description.

For horizontal scrolling this register is incremented or decremented by one byte or 1 character at a time, whenever the Horizontal Pel Panning register limit is reached.

For vertical scrolling this register is incremented by one scan line of pels or characters whenever the Preset Row Scan Register limit is reached.

Input Status Register 1 See "Input Status Register 1" on page 2-43 for a detailed description.

The Vertical Retrace status bit is polled to determine when the XGA subsystem is in the vertical retrace interval. Scrolling must be synchronized with this bit to avoid screen flash and other visible screen effects.

The exact sequence of operations is different in VGA modes from the XGA subsystem 132-column text mode.

When smooth scrolling in VGA modes, the following sequence is used:

1. Wait for Vertical Retrace start (Vertical Retrace bit = 1).
2. Wait for Vertical Retrace end (Vertical Retrace bit = 0).
3. Update Start Address and Horizontal Pel Panning Registers as required.
4. Wait for Vertical Retrace start (Vertical Retrace bit = 1).
5. Update Preset Row Scan Register as required.

For smooth scrolling in XGA subsystem 132-column text mode, the following sequence is used:

1. Wait for Vertical Retrace start (Vertical Retrace bit = 1).
2. Wait for Vertical Retrace end (Vertical Retrace bit = 0).
3. Update Pel Panning Register as required.
4. Wait for Vertical Retrace start (Vertical Retrace bit = 1).
5. Update Start Address and Preset Row Scan Registers as required.

132-Column Text Mode

The XGA-2 subsystem supports 132-column text mode as INT 10H Video BIOS Mode 14h. The original XGA subsystem did not generally have BIOS support for 132-column text mode.

To determine the existence of support for the 132-column text mode, issue a Video BIOS INT 10H (Return Functionality/State Information) call. See Video BIOS in *IBM Personal System/2 and Personal Computer BIOS Interface Technical Reference* for details. If the bit for "Mode 14h supported" is set in the "Static Functionality Information", or if the bit for "132 Column Text Supported" is set in the returned information, the BIOS Mode Set 14h call should be used to put the subsystem into the 132-column text mode. Where BIOS support is provided, it must be used to exploit the higher refresh rates available on displays attached to the XGA-2 subsystem, or to access the 9-pel-wide character sets available only on the XGA-2 subsystem.

If neither of the above bits is set, Mode 14h is not supported in BIOS. The following sequence of operations will put the subsystem into 132-column text mode using 8-pel-wide characters:

1. If necessary, put the XGA subsystem into VGA mode.
2. Write data to registers in the following sequence:

Value (hex)	Oper	XGA Reg	VGA 3D4/5	VGA 3C4/5	Other VGA	Comments
15	=	50				Prepare CRT controller for reset
14	=	50				Reset CRT controller
04	=	54				Select VGA clock

Figure 3-199. 132-Column Text Mode First Write Sequence

3. Set the number of lines in VGA mode using INT 10H AH = 12H (200, 350, or 400).

4. Set VGA mode 3 using INT 10H AX=0003H (or AX=0083H to prevent clearing the video buffer). The 132-column text mode is a variation of the VGA text mode. The following table gives the variations from the standard mode.

5. Write data to registers in the following sequence:

Value (hex)	Oper	XGA Reg	VGA 3D4/5	VGA 3C4/5	Other VGA	Comments
01	|=	50				Prepare CRT controller for reset
FD	&=	50				
FC	&=	50				Reset CRT controller
03	=	21x0				132-column text mode
01	=	54				132-column clock frequency select
80	=	70				Select internal 132-column clock
EF	&=	50				Disable video extension
7F	&=		11			Enable VGA CRT controller reg update
A4	=		0			
83	=		1			
84	=		2			
83	=		3			
90	=		4			Variations on VGA CRT controller syncs
80	=		5			
A3	=	1A				
00	=	1B				
9F	&=	1C				
F9	&=	1E				
42	=		13			
80	|=		11			Disable VGA CRT controller reg update
03	|=	50				Remove CRT controller reset
01	|=			01		8-bit characters
**	INP				3DA	Read sets attribute controller flip flop
13	=				3C0	Sets attribute controller registers hex 13 to 00
00	=				3C0	
20	=				3C0	Restore palette

|= Logical OR to modify register contents
&= Logical AND to modify register contents
= Write value to register
INP Read value from register

Figure 3-200. 132-Column Text Mode Second Write Sequence

6. Write hex 84 to hex 40:4A in BIOS data area to force Video BIOS recognition of 132-column text mode.

The coded text buffer is now 132 columns wide. The mode can now be programmed like any other VGA text mode, with a coded text buffer located at hex B8000 in system address space.

If it is necessary to invoke a mode change using Video BIOS (INT 10H) while in 132-column text mode (for example, to vary the number of lines), follow step 1 on page 3-224 through step 6 on page 3-225.

Effects of VGA and XGA Mode Setting on Video Memory

Software that relies on INT 10H Video BIOS mode setting to switch between VGA modes will be protected from the effects described in this section. For *all* software that switches between XGA modes and VGA modes, or sets VGA modes without using INT 10H Video BIOS, the following points should be observed when switching between modes, (see "Differences in Type 1 and Other Video Subsystems" on page 2-11):

- All data in video memory is preserved during a mode switch, provided that the CRT controller is halted at the time using the Display Control 1 register (if switching out of Extended Graphics mode), or the Reset register (if switching out of VGA mode). The CRT controller is described in "CRT Controller" on page 3-22 and "Display Control 1 Register (Index 50)" on page 3-72.
- When switching between VGA modes, the mapping of the VGA memory maps to the video memory is controlled by 2 fields in the following VGA CRT Controller registers:
  - Word/byte mode (CRT Mode Control register, WB field)
  - Doubleword mode (Underline Location register, DW field)

  VGA modes can be split into three groups: byte modes, word modes, and doubleword modes.

  All switches between modes in the same group are indistinguishable from the same mode switches on the VGA.

  Switches between modes in different groups produce different effects from those observed on the VGA. Because the bits controlling the mapping are used for display purposes, the picture is scrambled in both cases.

  Partial mode switches (for example, to load fonts in a text mode) are also possible. The bits used to control the mapping of the data in video memory are used to control the picture display. Therefore, all partial mode switches to update the video memory that do not destroy the picture (and many that do) work correctly.

Programming the XGA Subsystem

General Systems Considerations

Coexisting with LIM Expanded Memory Managers

The XGA subsystem uses memory-mapped registers located in the hex C0000/D0000 region of physical address space, as described in "XGA POS Registers" on page 3-164.

This area is used extensively by expanded memory managers to provide expanded memory services to applications.

When the application determines the location of the memory-mapped register space, it must interrogate any expanded memory manager to ensure that there is no contention for this range of physical address space. Use function 25 (Ah) = 58h, get physical address array, as described in the *Lotus**-Intel**-Microsoft** Expanded Memory Specification Version 4.0*. If there is contention, a warning should be issued advising the user to resolve it by use of the expanded memory manager call parameters (usually on the DEVICE= statement in CONFIG.SYS).

INT 2Fh, Screen Switch Notification

For the application to work successfully in multiple virtual DOS machine (MVDM) environments, or in the DOS compatibility box of the OS/2 operating system, it must trap and revector the DOS multiplex vector (2Fh), looking for (Ah) = 40h. Any other values must be passed immediately to the chained INT 2Fh handler. This multiplex interrupt is used with (Ah) = 40h to notify DOS applications of screen switches.

(Al) = 01h DOS mode application being switched to the background.

The application must save its video state and put the display back into VGA mode (if applicable).

** Lotus is a trademark of the Lotus Development Corporation, Intel is a trademark of Intel Corporation, and Microsoft is a trademark of the Microsoft Corporation.

(Al) = 02h DOS mode application being switched to the foreground.

The application can switch the subsystem back into Extended Graphics mode, and restore the saved video state.

The range of operations permitted within INT 2Fh processing is limited. For example, it is not permissible to issue disk I/O operations, precluding an entire save and restore of video memory and state. The only way of using this call is for the INT 2Fh interrupt handler to notify the application that a redraw is required (if the application program structure permits).

PS/2 System Video Memory Apertures

The XGA subsystem provides three possible apertures or windows to video memory in the physical memory address space of the system. If present, any of these apertures can be used by the system processor to directly access the packed pel display buffer. However all 3 apertures have drawbacks, as follows:

64KB Aperture Located at A0000 or B0000 in real-mode address range.

- Possible contention with VGA
- Possible contention with other XGA subsystems
- Limited size of Aperture (64KB and a 1MB Video Buffer) necessitates frequent movement of aperture
- Granularity of aperture movement (64KB minimum)

1MB Aperture Located above 1MB below 16MB in protect-mode system address range.

- Can not be enabled where 16MB of Memory is installed in system
- Only accessible by protect-mode drivers.

4MB Aperture Located above 16MB in protect-mode 32-bit address space

- Not available in 16-bit systems, such as those based on the 80386 SX processor
- Not available if the XGA subsystem is plugged into a 16-bit slot in a 32-bit system
- Only accessible by 32-bit protect-mode drivers

As the XGA subsystem coprocessor makes the use of apertures unnecessary, their use is not recommended. If software requires the use of apertures, the following considerations apply:

- Check the availability of the aperture before using it.
- Build flexibility into the software, to be able to use whichever aperture is available, including the 64KB aperture.
- Consider informing the application user that the XGA subsystem must be installed in a 32-bit slot on a 32-bit system if it is found to be located in a different type of slot.
- The XGA-2 subsystem 1MB aperture can be disabled by default by the System Setup program. Software might need to instruct users to run System Setup again to enable this aperture.
- Inform the user that system memory might need to be removed to permit the 1MB aperture to be enabled, where system memory does not permit it to be enabled.
- If using the real mode 64KB aperture, be aware of contention with any VGA or other XGA subsystems.

The precise location of each aperture, and whether it is enabled, is returned as part of the DMQS primary data, as described in "XGA Display Mode Query and Set (DMQS)" on page 3-185. If DMQS is not available, this can be determined by decoding the XGA subsystem POS data, as described in "Locating and Initializing the XGA Subsystem Without Using DMQS" on page 3-205.

64KB System Video Memory Aperture

This aperture is at hex A0000 or B0000 in physical address space. The 64KB aperture is insufficient to access the entire subsystem display buffer. Therefore, the aperture position over the display buffer is controlled by using the Aperture Index register.

This is the only aperture in real-mode address range.

Other video subsystems, such as another subsystem or subsystem in either VGA or Extended Graphics mode, can contend for the use of this aperture. Only one video subsystem can have this aperture enabled at a time. If there is no contention for the hex A0000 or B0000 address spaces, this aperture is the only one that can be directly enabled by the application.

1MB System Video Memory Aperture

The base address of this aperture can be located on any 1MB boundary from 1MB to 15MB, or it can be disabled. This aperture is located and enabled at System Setup. Software might need to publish instructions to the user to enable the aperture if it is disabled, using System Setup. The aperture address is determined by the system configuration. To determine its position, and whether it is enabled, decode the POS data as described in "Locating and Initializing the XGA Subsystem Without Using DMQS" on page 3-205.

In systems with multiple XGA subsystems, each one can have its own aperture. Depending on the hardware configuration, it is possible for some, but not all coexisting XGA subsystems to have their 1MB system video memory apertures enabled.

If 1MB of memory is installed, this aperture is large enough to access the entire video memory without using the Aperture Index register to move the aperture. The Aperture Index register must be set to 0 to ensure correct operation.

If greater than 1MB of video memory is installed, the aperture position over memory is controlled by the aperture index register. This aperture is easily accessible only in protect-mode environments. The operating system must provide addressability to the address range occupied by the aperture. Some operating systems attempt to restrict such addressability to protect device drivers or kernel device drivers. A small kernel device driver might need to be written to provide addressability. For example, in a 16-bit segmented operating system such as the OS/2 Version 1.3 operating system, the following steps might be necessary to build global descriptor table (GDT) addressability to an aperture:

1. Allocate a GDT selector.
2. Modify the GDT entry directly to alter the permission bits to allow user mode (ring 3) access.
3. Alter the GDT segment length to be a 1MB segment. The entire 1MB video memory display buffers can then be accessed as a single segment.

Check that the aperture is enabled before assuming its existence. In systems with 16MB of memory, this aperture can not be enabled. If the aperture is disabled, it cannot be enabled by the application. The application should then try to use the 4MB aperture.

4MB System Video Memory Aperture

The base address of this aperture might be located on any 4MB boundary at or above 16MB, or it might be disabled. The aperture address is determined by the system configuration. To determine its position, and whether it is enabled, decode the POS data as described in "Locating and Initializing the XGA Subsystem Without Using DMQS" on page 3-205.

In systems with multiple XGA subsystems, each XGA subsystem can have its own aperture.

This aperture is not available in 16-bit systems based on the 80386 SX. This aperture does not exist when the XGA subsystem is plugged into a 16-bit (short) slot on a 32-bit system.

Check that the aperture is enabled before assuming its existence. Also, check the Auto-Configuration register, as described in "Auto-Configuration Register (Index 04)" on page 3-48, to determine the bus width.

While this aperture is present when the XGA subsystem is plugged into a 32-bit slot on a 32-bit system, it might not be easily accessible in real-mode DOS or 16-bit protect-mode operating systems.

Video Memory Address Range

The video memory base address is returned in the DMQS primary data area, or calculated from POS settings on XGA subsystems without DMQS, as described in "Location of XGA Subsystem I/O Spaces" on page 3-207.

The video memory address range is defined as the range of addresses starting at the video memory base address, with length equal to the video memory size. The video memory address range has a special significance to the XGA coprocessor. It defines the location of the video memory, including the display pel map, in the XGA coprocessor's view of system address space. Therefore, the XGA coprocessor recognizes addresses in this range to be addresses in local video memory, rather than general system memory. This is how the XGA coprocessor differentiates video memory from system memory. If an address passed to the XGA coprocessor is in this range, the XGA coprocessor knows that it is operating on a bit map in video memory. If the address is outside this range, the XGA coprocessor assumes it is operating on a bit map in normal system memory, and attempts to use bus master functions to access it.

The XGA subsystem operates internally on a 32-bit bus. Therefore, this address is a 32-bit address regardless of whether the XGA subsystem is installed in a 16- or 32-bit slot or system, or whether the 4MB system video memory aperture is enabled or disabled. This applies even on systems where such addresses are not otherwise possible.

Programming the XGA Subsystem in Extended Graphics Mode

This section describes and gives examples of using Extended Graphics functions of the XGA coprocessor.

General Register Usage

To avoid conflicts with possible future changes in the use of registers or register fields, applications must comply with the Register Usage Guidelines at the start of the various register definition sections.

XGA Coprocessor Pel Interface Registers

Extended graphics functions are graphics update operations involving up to four pel maps. A pel map is defined by five registers:

- Pel Map Index register
- Pel Map n Base Pointer register
- Pel Map n Width register
- Pel Map n Height register
- Pel Map n Format register.

Pel Map Index Register: This register has an offset of hex 12. The Pel Map Index register defines which of the four possible maps is to be defined. The encoding of this 4-bit register is as follows:

Mask map	hex 0
Pel map A	hex 1
Pel map B	hex 2
Pel map C	hex 3

For example, to use Pel Map A:

```
WRITE 01h to copr_regs offset 12h.
```

Pel Map Base Address Register: This register has an offset of hex 14. The Pel Map Base Pointer register defines the byte address in memory of the start of the pel map. It is a 32-bit address register and can therefore address up to 4096MB of memory. A pel map can be defined to be in the XGA video memory or in system memory.

As described in "Video Memory Address Range" on page 3-232, to define a pel map as being in XGA video memory, the address put in this register must be in the following range:

```
Video Memory Base Address ↔ (Video
Memory Base Address + Video Memory size)
```

If the pel map is in system memory and the Micro Channel interface is a 16-bit interface (for example, if the XGA subsystem is installed in a 16-bit slot), the address of the map must be below 16MB.

Pel Map Width Register: This register has an offset of hex 18. The pel map width is measured in pels and is defined as one less than the required width.

For example, to set the width of a pel map to 640 pels:

```
WRITE 027Fh to copr_regs offset 18h
```

To set the width of a pel map to 1024 pels:

```
WRITE 03FFh to copr_regs offset 18h
```

Pel Map Height Register: This register has an offset of hex 1A. The pel map height is measured in pels and is defined as one less than the required height.

For example, to set the height of a pel map to 480 pels:

```
WRITE 01DFh to copr_regs offset 1Ah
```

To set the height of a pel map to 768 pels:

```
WRITE 02FFh to copr_regs offset 1Ah
```

Pel Map Format Register: This register has an offset of hex 1C. This register specifies the bits per pel of the pel map. The encoding of the register is as follows:

1 bit/pel Intel format	hex 00
2 bits/pel Intel format	hex 01
4 bits/pel Intel format	hex 02
8 bits/pel Intel format	hex 03
16 bits/pel Intel format	hex 04
1 bit/pel Motorola format	hex 08
2 bits/pel Motorola format	hex 09
4 bits/pel Motorola format	hex 0A
8 bits/pel Motorola format	hex 0B
16 bits/pel Motorola format	hex 0C

Note: Values hex 04 and hex 0C are only valid on the XGA-2 subsystem.

For example, for an 8-bit/Pel Motorola format pel map:

```
WRITE 0Bh to copr_regs offset 1Ch
```

The relationship between Intel and Motorola format pel maps is discussed in "Video Memory Format" on page 3-18 and "Motorola and Intel Formats" on page 3-255.

All four Pel maps (A, B, C, and mask) can be initialized in this manner to be ready for later use. Maps A, B, and C can be used interchangeably as the source, destination, or pattern in all subsequent pel operations.

Other Registers: For simple operations, the Pel Interface Control register must be cleared.

For example:

```
WRITE 00h to copr_regs offset 11h
```

For simple operations, the Destination Color Condition Compare register must be set so that it has no effect on the operation.

For example:

```
WRITE 04h to copr_regs offset 4Ah
```

To allow all planes of a pel map to be updated, the pel bit mask must be turned on. Set all bits to 1 that are required for the pel size selected.

For example, for 8-bits-per-pel:

```
WRITE 00FFh to copr_regs offset 50h
```

For simple operations, the carry chain mask must be turned on. Set all bits to 1 that are required for the pel size selected.

For example, for 8-bits-per-pel:

```
WRITE FFh to copr_regs offset 54h
```

Using the Coprocessor to Perform a Pel Blit (PxBlt)

This section describes the actions necessary to use the XGA coprocessor to perform a simple PxBlt.

Various types of PxBlt can be performed. This first example (example A) is for a PxBlt into video memory using the Foreground Color register as the source data. The result is a solid rectangle drawn into the display pel map. In example A, the PxBlt has the following characteristics:

- Foreground color of hex 05
- 100 pels wide and 60 pels deep
- Positioned at screen coordinates X=200 and Y=150

The following table lists the values that must be written to the coprocessor registers. Each value is explained following the table, along with information on the other forms of PxBlt available.

Value (hex)	Coprocessor Registers Offset (hex)
03	48
05	58
0063	60
003B	62
00C8	78
0096	7A
08118000	7C

Figure 3-201. Coprocessor Register Write Values (Example A)

Mixes and Colors: Before a coprocessor operation can be performed, the background and foreground mixes have to be set. Mixes are logical or arithmetic functions performed on the source and destination data when performing a coprocessor operation. The available mix functions are as follows:

Code	Function
0	Zeros
1	Source AND Destination
2	Source AND NOT Destination
3	Source
4	NOT Source AND Destination
5	Destination
6	Source XOR Destination
7	Source OR Destination
8	NOT Source AND NOT Destination
9	Source XOR NOT Destination
A	NOT Destination
B	Source OR NOT Destination
C	NOT Source
D	NOT Source OR Destination
E	NOT Source OR NOT Destination
F	Ones
10	Maximum
11	Minimum
12	Add with Saturate
13	Subtract (Destination – Source) with Saturate
14	Subtract (Source – Destination) with Saturate
15	Average

Figure 3-202. Background and Foreground Mixes and Colors

Foreground and Background Mix Registers: The mixes to be applied to foreground and background pels are specified in these two registers. The contents of the pattern map determine the pels for foreground and background. In example A, the PxBlt is solid and contains only foreground pels. The Foreground Mix register must be set to Source to give an understandable result on the screen.

For example A:

```
WRITE 03h to copr_regs offset 48h
```

Foreground and Background Color Registers: The colors to be used for foreground and background pels are specified in these two registers. In example A, the PxBlt is solid and only the Foreground Color register needs to be set up.

For example A:

```
WRITE 05h to copr_regs offset 58h
```

Other forms of PxBlt (for example, video memory to video memory) from a source map into a destination map do not use these color registers.

PxBlt Dimensions: The Operation Dimension 1 register must be loaded with the Width of PxBlt to be performed. The value loaded into the register must be one pel less than the required width (in pels).

In example A, for a 100-pel wide Pxblt:

```
WRITE 0063h to copr_regs offset 60h
```

The Operation Dimension 2 register must be loaded with the Height of PxBlt to be performed. The value loaded into the register must be one pel less than the required height (in pels).

In example A, for a 60-pel high Pxblt:

```
WRITE 003Bh to copr_regs offset 62h
```

Pel Map, Source, and Destination Registers

Source Map X and Y Registers: The source map is initialized as detailed in "Mixes and Colors" on page 3-238. Within the source map, two registers exist that contain the X and Y offset positions of the start of the source data for a PxBlt. These registers are used when performing a PxBlt using a source map. In example A, these registers are unused.

Destination Map X and Y Registers: The destination map is initialized as detailed in "Mixes and Colors" on page 3-238. Within the destination map, two registers exist that contain the X and Y offset positions of the start of the PxBlt.

In example A, to position the PxBlt at X=200 and Y=150 in the destination map:

```
WRITE 00C8h to copr_regs offset 78h (Destination Map X position)

WRITE 0096h to copr_regs offset 7Ah (Destination Map Y position)
```

Pattern Map X and Y Registers: The pattern map is initialized as detailed in "Mixes and Colors" on page 3-238. Two registers exist that contain the X and Y offset positions, within the pattern map, of the start of the pattern data for a PxBlt. These registers are used when performing a PxBlt using a pattern map.

In example A, these registers are unused.

Mask Map Origin X and Y Offset Registers: The mask map is initialized as detailed in the previous chapter. Two registers exist that contain the X and Y offset positions of the start of the mask map relative to the top left corner of the destination map. These registers are used when performing a PxBlt using a mask map.

In example A, these registers are unused.

Pel Operations Register: This is a 32-bit register that defines the operation the coprocessor performs.

31	30	29	28	27			24	23			20	19			16	15			12	11			8	7	6	5	4	3	2		0
																				X	X	X	X					X			
1		2		3				4				5				6								7		8			9		

Figure 3-203. Bit Layout Pel Operations Register

The definition of the fields at the bottom of Figure 3-203 are:

1. Background Source
2. Foreground Source
3. Step Function
4. Source Pel Map
5. Destination Pel Map
6. Pattern Pel Map
7. Mask Pel Map
8. Drawing Mode
9. Direction Octant

These fields, described in sequence, are required to assemble the Pel Operations register:

Background Source: These bits determine the origin of the background source pels when an operation is performed.

The encoding for these bits is as follows:

Background Color	Binary 00 (for example, for a fixed register value to video memory PxBlt).
Source Pel Map	Binary 10 (for example, for a video memory to video memory PxBlt).

For example A, there is no background color, and the field is ignored.

```
Background Source = Binary 00
```

Foreground Source: These bits determine the origin of the foreground source pels when an operation is performed.

The encoding for these bits is as follows:

Foreground Color	Binary 00 (for example, for a fixed register value to video memory PxBlt).
Source Pel Map	Binary 10 (for example, for a video memory to video memory PxBlt).

For example A, there is a solid foreground color:

```
Foreground Source = Binary 00
```

Step Function: These bits define the type of operation that the coprocessor is required to do.

The encoding for these bits is as follows:

Draw and Step Read	Binary 0010
Line Draw Read	Binary 0011
Draw and Step Write	Binary 0100
Line Draw Write	Binary 0101
PxBlt	Binary 1000
Inverting PxBlt	Binary 1001
Area Fill PxBlt	Binary 1010

For example A:

```
Step Function = binary 1000
```

Source Pel Map: These bits define the pel map used as the source map in the operation. This enables different maps to be setup in advance and defined for use as this register is loaded.

The encoding for these bits is as follows:

Pel Map A	Binary 0001
Pel Map B	Binary 0010
Pel Map C	Binary 0011

For example A, the contents of this field are ignored but they must not be a reserved value.

```
Source Pel Map = Binary 0001
```

Destination Pel Map: These bits define the pel map used as the destination map in the operation. This enables different maps to be setup in advance and defined for use as this register is loaded.

The encoding for these bits is as follows:

Pel Map A	Binary 0001
Pel Map B	Binary 0010
Pel Map C	Binary 0011

For example A:

```
Destination Pel Map = Binary 0001
```

Pattern Pel Map: These bits define the pel map used as the pattern map in the operation. This enables different maps to be setup in advance and defined for use as this register is loaded.

The encoding for these bits is as follows:

Pel Map A	Binary 0001
Pel Map B	Binary 0010
Pel Map C	Binary 0011
Foreground (fixed)	Binary 1000
Generated from Source	Binary 1001

For example A:

```
Pattern Pel Map = binary 1000
```

Mask Pel Map: These bits define whether the mask map is used in the operation.

The encoding for these bits is as follows:

Mask Map Disabled	Binary 00
Mask Map Boundary Enabled	Binary 01
Mask Map Enabled	Binary 10

For example A:

```
Mask Pel Map = Binary 00
```

Drawing Mode: These bits concern line drawing only and are discussed later. They are ignored during a PxBlt.

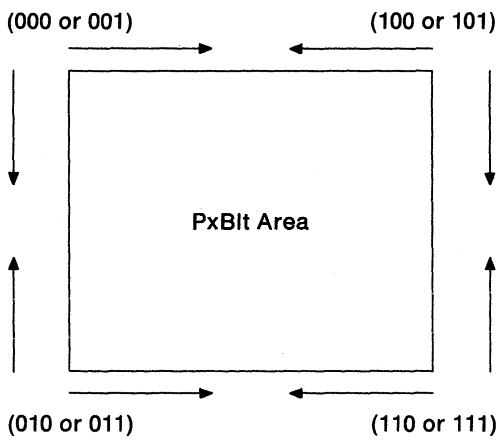
For example A:

```
Drawing Mode = Binary 00
```

Direction Octant: These bits, when concerned with PxBlts, determine the direction in which the PxBlt is drawn.

The encoding for these bits is as follows:

Binary 000 or 001	Start at top left-hand corner of area increasing right and down.
Binary 100 or 101	Start at top right-hand corner of area increasing left and down.
Binary 010 or 011	Start at bottom left-hand corner of area increasing right and up.
Binary 110 or 111	Start at bottom right-hand corner of area increasing left and up.



Note: Numbers are binary.

Figure 3-204. Operation Direction Diagram

These bits are normally set to binary 000, but other values are necessary to avoid pel corruption when source and destination rectangles overlap.

For example A, the PxBlt is in top left corner:

```
Direction Octant = Binary 000
```

Conclusion: Putting all these together for the PxBlt, the Pel Operations register must be set as:

31	30	29	28	27			24	23			20	19			16	15			12	11			8	7	6	5	4	3	2		0
0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	1	1	0	0	0	X	X	X	X	0	0	0	0	X	0	0	0

Figure 3-205. Definition for Pel Operations Register (Example)

For example A:

```
WRITE 08118000h to copr_regs offset 7Ch
```

Using the Coprocessor to Perform a Bresenham Line Draw

The steps required to draw a line of palette color hex 05 from (20,15) to (80,35), are summarized in the following table. The sections that follow explain each value and provide information on the other line drawing options available.

Value	Coprocessor Registers Offset (hex)
Hex 03	48
Hex 05	58
Decimal − 20	20
Decimal 40	24
Decimal − 80	28
Decimal 60	60
Decimal 20	78
Decimal 15	7A
Hex 05118000	7C

Figure 3-206. Palette Color Line Draw Steps

Mixes and Colors: Before a coprocessor operation is performed, the background and foreground mixes have to be set. Mixes are logical or arithmetic functions performed on the source and destination data when performing a coprocessor operation. The available mix functions are as follows:

Code	Function
0	Zeros
1	Source AND Destination
2	Source AND NOT Destination
3	Source
4	NOT Source AND Destination
5	Destination
6	Source XOR Destination
7	Source OR Destination
8	NOT Source AND NOT Destination
9	Source XOR NOT Destination
A	NOT Destination
B	Source OR NOT Destination
C	NOT Source
D	NOT Source OR Destination
E	NOT Source OR NOT Destination
F	Ones
10	Maximum
11	Minimum
12	Add with Saturate
13	Subtract (Destination - Source) with Saturate
14	Subtract (Source - Destination) with Saturate
15	Average

Figure 3-207. Background and Foreground Mixes and Colors

Foreground and Background Mix Registers: The Foreground Mix and Background Mix registers allow a mix (as detailed in the table) to be specified. These registers are discussed in example A, "Using the Coprocessor to Perform a Pel Blit (PxBlt)" on page 3-237.

For this example (example B), the Foreground Mix register must be loaded with Source. The Background Mix register is not used in example B.

For example B with the Foreground Mix register:

```
WRITE 03h to copr_regs offset 48h
```

Foreground and Background Color Registers: The Foreground Color register must be set to the color required for the line.

For example B with the Foreground Color register:

```
WRITE 05h to copr_regs offset 58h
```

Bresenham Line Draw: The algorithm used to perform the line draw function on the XGA is the Bresenham Line Draw algorithm. This operates with all parameters normalized to the first octant (octant 0).

The first task is to calculate deltaX and deltaY (see the following figure).

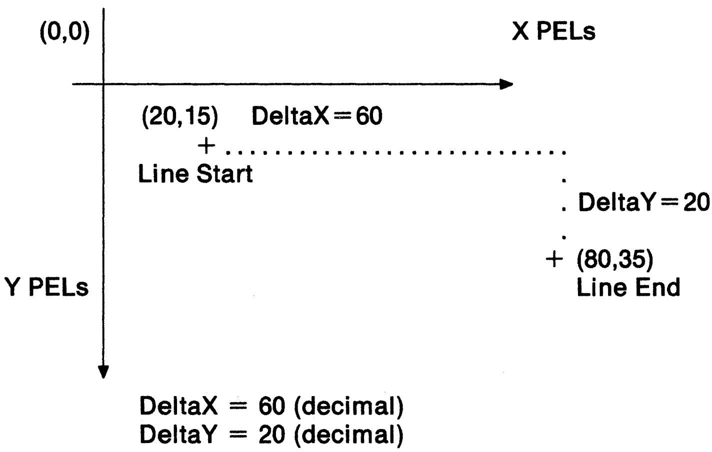


Figure 3-208. Line Draw Example in Octant 0

A line in the first octant has deltaX greater than deltaY, with both deltaX and deltaY positive, and deltaX greater than deltaY. If a line is to be drawn in another octant, the octant information is specified in the octant bits of the Pel Operation register. The line is drawn as if it were in the first octant.

To normalize a line to the first octant, follow these rules:

- If deltaX is −ve , set DX in octant bits of the Pel Operation register and make deltaX +ve.
- If deltaY is −ve , set DY in octant bits of the Pel Operation register and make deltaY +ve.
- If deltaY ≥ deltaX , set DZ in octant bits of the Pel Operation register and exchange deltaX and deltaY.

The terms deltaX and deltaY are the lengths of the line after it has been normalized to octant 0. The algorithm requires several parameters to be calculated. These are:

Bresenham Error Term Register: Bresenham Error Term
$E = (2 \times \text{deltaY}) - \text{deltaX}$

For example B:

```
WRITE -20 decimal (FFECh) copr_regs offset 20h
```

Bresenham Constant K1 Register: Bresenham Constant
$K1 = 2 \times \text{deltaY}$

For example B:

```
WRITE +40 decimal (0028h) copr_regs offset 24h
```

Bresenham Constant K2 Register: Bresenham Constant
$K2 = 2 \times (\text{deltaY} - \text{deltaX})$

For example B:

```
WRITE -80 decimal (FFB0h) copr_regs offset 28h
```

Operation Dimension Registers: The Operation Dimension 1 register should be loaded with deltaX after normalization. Because a value of 0 results in a line length of 1 pel, deltaX (calculated in Figure 3-208 on page 3-247) equals the number to be drawn minus 1.

For example B:

```
WRITE +60 decimal (003Ch) to copr_regs offset 60h
```

The Operation Dimension 2 register is not used for line draw.

Pel Map Source and Destination

Source Map X and Y Registers: The source map is initialized as described in "XGA Coprocessor Pel Interface Registers" on page 3-233. Two registers exist that contain the X and Y offset positions within the source map of the start of the source data for a PxBlt. These registers are used for drawing a line using a source map. In example B, these registers are unused.

Destination Map X and Y Registers: The destination map is initialized as described in "XGA Coprocessor Pel Interface Registers" on page 3-233. Two registers exist that contain the X and Y offset positions within the destination map of the start of the line.

In example B with destination map X and Y positions:

```
WRITE 0014h to copr_regs
offset 78h

WRITE 000Fh to copr_regs
offset 7Ah
```

Pattern Map X and Y Registers: The pattern map is initialized as described in "XGA Coprocessor Pel Interface Registers" on page 3-233. Two registers exist that contain the X and Y offset positions within the pattern map of the start of the pattern data for a line. These registers are used when drawing a line using a pattern map. In example B, these registers are unused.

Mask Map Origin X and Y Offset Registers: The mask map is initialized as described in "XGA Coprocessor Pel Interface Registers" on page 3-233. Two registers exist that contain the X and Y offset positions of the start of the mask map relative to the top left corner of the destination map. These registers are used when drawing a line using a mask map. In example B, these registers are unused.

Pel Operations Register: This is a 32-bit register that defines the operation that the coprocessor performs.

31 30	29 28	27 24	23 20	19 16	15 12	11 8	7 6	5 4	3	2 0
						X X X X			X	
1	2	3	4	5	6		7	8		9

Figure 3-209. Bit Layout Pel Operations Register

The bits 0—31 are shown on the top of Figure 3-209; Fields 1 through 9 are shown on the bottom. The definition of these fields is:

1. Background Source
2. Foreground Source
3. Step Function
4. Source Pel Map
5. Destination Pel Map
6. Pattern Pel Map
7. Mask Pel Map
8. Drawing Mode
9. Direction Octant

These fields, described in sequence, are required to assemble the contents of the Pel Operations register:

Background Source: These bits determine the origin of the background source pels when an operation is performed.

The encoding for these bits is as follows:

Background Color	Binary 00 (for example, for a fixed pattern line draw using a fixed register value)
Source Pel Map	Binary 10 (for example, for a variable color data pattern held in video memory to video memory draw).

In example B, the contents of this field are ignored because the line is solid and there are no background pels:

```
Background Source = Binary 00
```

Foreground Source: These bits determine the origin of the foreground source pels when an operation is performed.

The encoding for these bits is as follows:

Foreground Color	Binary 00 (for example, for a fixed pattern line draw using a fixed register value).
Source Pel Map	Binary 10 (for example, for a variable color data pattern held in video memory to video memory draw).

For example B:

```
Foreground Source = Binary 00  (Solid Foreground Color)
```

Step Function: These bits define the type of operation that the coprocessor is required to do.

Draw and Step Read	Binary 0010
Line Draw Read	Binary 0011
Draw and Step Write	Binary 0100
Line Draw Write	Binary 0101
PxBlt	Binary 1000
Inverting PxBlt	Binary 1001
Area Fill PxBlt	Binary 1010

For example B:

```
Step Function = binary 0101
```

Source Pel Map: These bits define the pel map used as the source map in the operation. This enables different maps to be setup in advance, and defined for use as this register is loaded.

The encoding for these bits is as follows:

Pel Map A	Binary 0001
Pel Map B	Binary 0010
Pel Map C	Binary 0011

In example B, the contents of this field are ignored, but they must not be a reserved value:

```
Source Pel Map = binary 0001
```

Destination Pel Map: These bits define the pel map used as the destination map in the operation. This enables different maps to be setup in advance and defined for use as this register is loaded.

The encoding for these bits is as follows:

Pel Map A	Binary 0001
Pel Map B	Binary 0010
Pel Map C	Binary 0011

For example B with pel map A:

```
Destination Pel Map = binary 0001
```

Pattern Pel Map: These bits define the pel map used as the pattern map in the operation. This enables different maps to be setup in advance and defined for use as this register is loaded.

The encoding for these bits is as follows:

Pel Map A	Binary 0001
Pel Map B	Binary 0010
Pel Map C	Binary 0011
Foreground (fixed)	Binary 1000
Generated from Source	Binary 1001

For example B:

```
Pattern Pel Map = binary 1000
```

Mask Pel Map: These bits define whether mask map is used in the operation.

The encoding for these bits is as follows:

Mask Map Disabled	Binary 00
Mask Map Boundary Enabled	Binary 01
Mask Map Enabled	Binary 10

For example B:

```
Mask Pel Map = Binary 00
```

Drawing Mode: These bits determine the attributes of a line draw.

The encoding for these bits is as follows:

Draw All pels	Binary 00
Draw First Pel Null	Binary 01
Draw Last Pel Null	Binary 10
Mask Area Boundary	Binary 11

The first three options can be used when drawing a line. The fourth option is for use when drawing the outline of a shape to be filled using the area fill capability of the hardware.

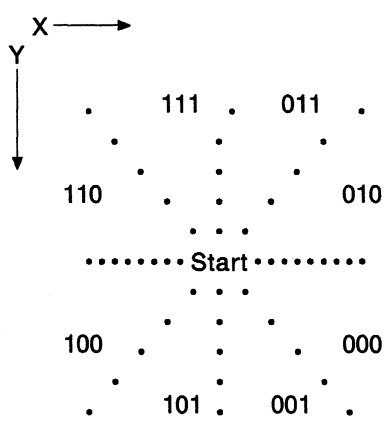
For example B, to draw all pels:

```
Drawing Mode = Binary 00
```

Direction Octant: These bits, when concerned with line draws, determine the direction in which the line is drawn.

The encoding for these bits is as follows:

Bit 2(DX)	1	if negative X direction
	0	if positive X direction
Bit 1(DY)	1	if negative Y direction
	0	if positive Y direction
Bit 0(DZ)	1	if $\|X\| \le \|Y\|$
	0	if $\|X\| > \|Y\|$, (magnitude)



Note: Numbers are binary.

Figure 3-210. Direction Octant (Example)

For example B:

`Direction Octant = binary 000 (X + ve, Y + ve, |X| > |Y|)`

Conclusion: Putting all these together for example B line draw operation, the Pel Operations register must be set as:

31	30	29	28	27			24	23			20	19			16	15			12	11			8	7	6	5	4	3	2		0
0	0	0	0	0	1	0	1	0	0	0	1	0	0	0	1	1	0	0	0	X	X	X	X	0	0	0	0	X	0	0	0

Figure 3-211. Definition for Pel Operations Register (Example)

For example B:

`WRITE 05118000h to copr_regs offset 7Ch`

Memory Access Modes

This register has an address of hex 21x9. The Memory Access Modes register is used to control the format of the data supplied by the system processor through a system video memory aperture. For conventional use, this register must be set to match the format of the data as seen by the system processor (Motorola or Intel), and the depth of the video memory bit map.

Through this register, the different formats available can be used to achieve useful and otherwise difficult conversions.

Motorola and Intel Formats

The internal organization of the video memory is Intel format. However, images and bit maps are traditionally stored in Motorola format. It is necessary to understand the format of the application bit maps in system memory to get the correct results. The different formats are described in "Video Memory Format" on page 3-18.

The internal organization of video memory as Intel format can be hidden by appropriate use of the Memory Access Mode register ("Memory Access Modes") and the various coprocessor pel map format registers.

System Processor Access: When using the system processor to read or write data directly to or from video memory through a system video memory aperture, it is necessary to specify the format of the data using the Memory Access Mode register.

XGA Coprocessor Accesses: The format of all bit maps in system memory must be specified through the Pel Map Format register. This parameter is ignored for bit maps in video memory.

Exploitation: Writing data in one format and reading it back in another is a technique that performs many useful and otherwise difficult or expensive bit map conversions.

Other Programming Considerations

Waiting for Hardware Not Busy

There are two ways to determine that the hardware has completed its task; either by polling the coprocessor busy bit, or by waiting on an operation complete interrupt from the coprocessor.

Continuous polling: Once a XGA coprocessor operation has started, the XGA coprocessor operates asynchronously with the system processor. Software must wait for the previous operation to complete before initiating next operation using the XGA coprocessor. This is done by continuous polling of the coprocessor busy bit, as described in "Coprocessor Control Register (Offset 11)" on page 3-134.

On XGA-2 subsystems only, the auxiliary coprocessor busy bit, described in "Auxiliary Coprocessor Status Register (Offset 09)" on page 3-133, is a higher performance alternative for use only on XGA-2 subsystems, as sampling does not delay the coprocessor.

Software using the coprocessor busy bit described in "Coprocessor Control Register (Offset 11)" on page 3-134 should be aware that polling this bit slows down the coprocessor because the current operation is paused to process the read of the Coprocessor Control register.

To reduce this effect, software should use a double polling loop that checks the coprocessor busy bit, for example, once every 100 times around the loop.

An example of this is as follows:

```
if (XGA-2) while (auxiliary_busy);

else for (;;) /* original XGA - double loop */
{

    if ¬ (coprocessor_busy) break;
    for (i=0;i<100;i++);

}
```

Operation Complete Interrupt: The coprocessor can be programmed to cause an interrupt to the system processor when an operation is completed.

This interrupt is a shared level. Interrupt response time therefore depends on other interrupt handlers chained on this shared level. In protect mode operating systems in particular, the overheads and restrictions placed on interrupt handlers might make the performance of this technique prohibitive.

Advantages of using this method are:

- The XGA coprocessor is not slowed while waiting for completion.
- The system processor can be freed up for other tasks.

Disadvantages of using this method are:

- Program complexity.
- Interrupt response time gives a threshold in size of operation that is only exceeded by large PxBlt operations. The more complex the operating system, the higher the interrupt response time, and the larger the operation must be to benefit from using interrupts to notify the application of operation complete.

Overlapping PxBlts

Pel Block Transfer (PxBlt): The coprocessor PxBlt function is used to transfer a rectangular block of pels from the source to the destination subject to a number of modifiers. It is important to predetermine whether the source and destination rectangles overlap. If the rectangles do not overlap, the order of processing pels is immaterial. If the rectangles do overlap, program the PxBlt direction using the Direction Octant to ensure the expected result.

Inverting PxBlt

The inverting PxBlt is intended to convert images from the traditional application format of Y increasing upwards to the traditional display hardware format of Y increasing downwards. As such a PxBlt operates from both ends towards the middle, an Inverting PxBlt involving overlapping source and target rectangles inevitably overwrites pels. Therefore, inverting PxBlts on overlapping rectangles should be avoided, unless for special effects.

Area Fill

Area fill is described in detail in "Area Fill" on page 3-117. This section describes its use from a programming perspective.

All area fill operations rely on a 1bpp area fill work plane, the same pel dimension as the destination bit map. For a 1024 x 768 display, this amounts to 96KB, which must be allocated in either system memory or off-screen video memory. Higher performance is achieved by locating the area fill plane in off-screen video memory if available.

The area fill operation performed is commonly called "alternate fill." After the outlines have been drawn into the area fill plane, the area fill bit blit operation works left to right across the fill plane, alternatively filling and not filling as it crosses drawn lines. The significance of this is that a doughnut shape will be correctly filled, leaving the hole in the doughnut unfilled.

Other common area fill algorithms, such as "flood fill" and "winding fill" are not available as XGA coprocessor functions, and must be performed in software.

Exclusive Fill: The result of the alternate fill algorithm described above is an "exclusive fill." Exclusive fill is so called because the filled area includes the top and left boundaries of the defined area, but *excludes* the bottom and right boundaries. This algorithm allows abutting filled areas to meet without overlapping.

Software that expects or requires inclusive area fill, in which the filled area includes all boundaries, must perform an additional step to merge the outline lines with the already filled area fill mask at the penultimate stage. Either an entire additional outline plane or the original line vectors must be retained by software for this purpose.

Restrictions
The following restrictions should be noted:

Destination Bit Map Width Restriction Incorrect results can occur if the XGA coprocessor is used to write over the edge of a destination bit map where the edge of the bit map is not 4-byte aligned. To avoid this, use one of the following methods:

- Ensure that all destination bit maps have a base address that is on a 4-byte boundary and are an exact multiple of 4 bytes wide.

The visible display bit map naturally complies with this restriction.

- Where bit maps are not aligned, software clip all PxBlts in advance so that the destination bit map boundary is not crossed during the PxBlt.

Line Length Restriction The XGA coprocessor Destination X Address and Destination Y Address registers accept coordinates in the range (−2048 to 6143). This gives a guardband effect, where it is possible to write coordinates anywhere in this range, and the operation is hardware scissored to the edge of the destination bit map. The limit on bit map size for coprocessor operations is 0 to 4095.

Because the Operation Dimension 1 register only accepts values in the range (0 to 4095), it is not possible to draw a line in a single operation across the entire guardband coordinate space.

A two-stage line draw can be performed easily, since the line parameters (for example, ET, K1, K2, Destination X and Y, Pattern X) are already set up in the hardware at the end of the previous line segment. It is only necessary to update the new line length in the Operation Dimension 1 register to draw the remainder of the line.

Common Problems

This section is a description of the most common problems experienced when programming the XGA coprocessor.

Source map and destination map depth must match The depth of the source and destination bit maps must match. For example, if the source bit map is 4 bpp, it cannot be used in any coprocessor operation with a destination bit map of 8 bpp. If the source destination is 1bpp, it can be used as a pattern map, but not as a source map with a destination bit map other than 1 bpp.

The result of failure to observe this rule can be obscure. The operation that fails is not the one that caused the problem, but is a subsequent XGA coprocessor function of the same type some time in the future. As a result, this problem can be hard to detect.

Save Restore Sequence A number of problems regularly occur with save and restore. These are as follows:

- Failure to check operation suspended
- Failure to “terminate” before starting new operation

- Failure to wait for operation complete after terminating

Failure to wait for previous operation complete The XGA coprocessor operates asynchronously to the system processor, and might not finish an operation for some time. Software must be aware that writing to the XGA Pel Operations register only starts an operation, and software must then wait for the operation to complete before either:

- Writing to any XGA coprocessor registers
- Using the resulting bit map from a XGA coprocessor operation
- Re-using, freeing, or unlocking memory used to contain bit maps

Operations outside bit map guardband While there is a 2KB guardband around bit maps, this effectively means that the XGA coprocessor deals on 14 bit coordinates. Operations such as lines outside this coordinate range will still wrap, producing unexpected lines.

Bit maps must be Dword aligned The start address (as loaded into the XGA coprocessor pel map base pointer) of all bit maps accessed by the XGA coprocessor must be located on a doubleword (32-bit Dword aligned) boundary.

Subrectangles within a bit map can be used to access data on other boundaries, as long as the bit map start address is Dword aligned.

Uninitialized X,Y pointers Source, Destination, and Pattern Map X and Y addresses, and Mask Map origin X and Y offset are uninitialized after a reset, or on initial startup. The result of operations involving any of these addresses prior to initialization is undefined. They must be initialized to a valid value before any operation involving the bit map to which they refer.

Performance Tips

This section describes how to achieve the optimum performance on the XGA subsystem.

Polling for completion: As described in "Waiting for Hardware Not Busy" on page 3-256, software should not continuously poll the coprocessor busy bit.

On XGA-2 subsystems use the Auxiliary busy bit. On XGA subsystems, use the double loop as described.

An improvement on this might be different delays according to the size of the operation initiated, or a geometric delay, where the delay would start small and increase with repetition until the operation completes.

Text performance: The following have been found to improve text performance:

- Balance hardware and software functions. Software can leave all the work to the coprocessor, or alternatively, software can do most of the work. The optimum solution varies by application, but some experimentation should be done to ensure that the coprocessor completes the previous character in a string at about the same time as the software has prepared the next character, so that hardware and software functions are approximately balanced.
- Software clip characters. Calculate in advance the minimum bounding rectangle of the displayed characters, and leave hardware clipping turned off. Do not use the coprocessor to PxBlt portions of characters that will be clipped out.
- Draw the entire background opaque rectangle as a single rectangle, not on a character-by-character basis.

Line drawing

- Do not use the line drawing operation for horizontal and vertical lines.

  It is faster to use the simple PxBlt operation for horizontal and vertical lines. Be aware that the Destination X and Y addresses will then need to be updated to the start of the next line segment.
- Destination X and Y addresses are automatically updated to the end of the drawn line. There is no need to set them prior to subsequent line segments.

- Coarse clip lines in software. Turn off XGA coprocessor hardware clip when clipping is not required. If both ends of the line are within the clip rectangle, turn clipping off. If both ends of the line are outside the same boundary, do not draw the line. Only use coprocessor hardware clip on lines that cross the clip rectangle.

Software clip: Where possible, clip in software. It takes as long to hardware clip lines and rectangles as it does to draw them. It is better to pre-calculate that the rectangle or line will not be drawn, than to use the hardware clip.

On rectangles (for instance characters or PxBlts), pre-calculate the subrectangle that will be unclipped.

On lines, only clip those lines that cross the clip boundary.

Video memory faster than system memory: Coprocessor operations on video memory are significantly faster than similar operations on system memory. Off-screen video memory is the optimum place to locate the font cache, patterns, brushes, the area fill plane, and the most frequently used bit maps.

Sprite Handling

Technical details of the sprite are described in "Sprite" on page 3-25. This section concentrates only on programming aspects of sprite usage.

Sprite Loading: The sprite is loaded as a 64 x 64 x 2 bits per pel (bpp) Intel format image definition. Because the sprite definition in the application is invariably held in two separate 1-bpp Motorola format bit maps, it is necessary to merge and pel swap the sprite definition into the 2-bpp Intel format before loading the sprite.

Sprite Positioning: The position of the sprite is then controlled by two separate control registers:

Sprite Start Registers

The sprite is positioned on the display surface by specifying the position of the top left corner of the sprite definition relative to the top left corner of the visible bit map, using the Sprite Horizontal Start and Sprite Vertical Start registers.

Sprite Preset Registers

The sprite start registers only accept positive values, and cannot be used to move the sprite partially off the display surface at the left and top edges. The Sprite Horizontal Preset and Sprite Vertical Preset registers are used to offset the start of the displayed sprite definition relative to the loaded definition.

For example, to display a 64 x 64 pel sprite with the leftmost 32 pels outside the left edge of the display surface, set the Sprite Horizontal Start register to 0, and the Sprite Horizontal Preset register to 32. The start position is now preset to the center of the loaded definition, giving the required effect.

The sprite preset can also be used to display sprites smaller than 64 x 64 pels.

Sprite update: Update of the sprite contents can be synchronized with its display using the Sprite Display Complete Status bit described in "Interrupt Status Register (Address 21x5)" on page 3-40, to avoid partial sprites becoming visible during update. This bit can be polled to avoid using interrupts when required.

Palette formats

Technical details of the palette loading are described in "Palette" on page 3-28. This section concentrates only on programming aspects of palette use.

Two palette formats are offered. While the R,G,B format of 3 bytes per entry is more space efficient, the alternative of R,B,G,X allows individual palette entries to be stored in a single Dword, and loaded into the appropriate palette location in a single Dword OUT to the XGA Data I/O Port 21xCh.

XGA Subsystem Save, Restore, Suspend, and Resume

In a task or screen switching environment, it might be necessary to suspend the current operation, save the state of the display subsystems, and restore the subsystem to a known state, such as VGA text mode, so that another process or task can use the display subsystem. Subsequently, the suspended process can be reactivated, at which time software must restore the entire display subsystem state, and resume the suspended operation at the point where it was suspended.

This might be necessary, even in DOS programs, if such programs are to be candidates to run in the Multiple Virtual DOS Machine environments of OS/2 operating system and Windows** operating system.

The complete XGA subsystem save and restore consists of the following parts:

- Video buffer contents
- Coprocessor state
- Sprite contents
- Palette contents
- Other I/O registers

Each of these components is discussed below in more detail.

Video Buffer Contents: Software is not required to save and restore the entire video buffer contents, if there is a more memory efficient means of recreating the video buffer contents when requested.

There are several ways to save and restore the entire video buffer contents. This could be done by using either apertures, or the XGA coprocessor to PxBlt the video buffer to a screen save area, or logical video buffer.

Coprocessor state: Capabilities:

- The coprocessor can be suspended at any time during an operation, and the state of the coprocessor can be saved.
- A new coprocessor operation can then be started.
- A suspended operation can be restored and resumed at any time.

Coprocessor Suspend and Save: To suspend and save the coprocessor, software should perform the following precise sequence of operations:

1. Check for coprocessor busy, by testing the BSY bit in the coprocessor control register, as described in "Coprocessor Control Register (Offset 11)" on page 3-134. If found to be busy, the following steps are necessary to suspend the coprocessor in the middle of the current operation.

** Windows is a trademark of Microsoft Corporation.

a. Suspend coprocessor, by setting the "SO" bit in the coprocessor control register, as described in "Coprocessor Control Register (Offset 11)" on page 3-134.
b. Wait until Operation has been suspended, by polling the "OS" bit in the coprocessor control register, as described in "Coprocessor Control Register (Offset 11)" on page 3-134.

2. Select save mode, by setting "SR" to 1 in the coprocessor control register, as described in "Coprocessor Control Register (Offset 11)" on page 3-134.
3. Read lengths of data to be saved. These are available separately as the lengths of save data A and B in Dwords (32 bit doublewords), as described in "State Length Registers (Offset C and D)" on page 3-136.
4. Read (using "REP I/O") and save the exact number of Dwords for data port A ("Coprocessor Save/Restore Data Registers (Index 0C and 0D)" on page 3-49), as determined above.
5. Read (using "REP I/O") and save the exact number of Dwords for data port B ("Coprocessor Save/Restore Data Registers (Index 0C and 0D)" on page 3-49), as determined above.
6. Terminate the current operation, by setting the "TO" bit in the coprocessor control register, as described in "Coprocessor Control Register (Offset 11)" on page 3-134.
7. Wait for processor not busy, either by polling the "BSY" bit in the coprocessor control register ("Coprocessor Control Register (Offset 11)" on page 3-134) or (on XGA-2 subsystems only) by polling the "ABSY" bit in the auxiliary coprocessor control register ("Auxiliary Coprocessor Status Register (Offset 09)" on page 3-133).

The coprocessor is now in an uninitialized reset state, ready to be used as required. Software must fully initialize the coprocessor before use, and should make no assumptions about its previous state.

Coprocessor Restore and Resume: To restore and resume any previously suspended operation and state, software should perform the following precise sequence of operations:

The coprocessor is assumed to be "Not busy," as it would be following a complete suspend and save sequence, as described in "Coprocessor Suspend and Save" on page 3-264.

1. Select restore mode, by setting "SR" to 0 in the coprocessor control register, as described in "Coprocessor Control Register (Offset 11)" on page 3-134.
2. Write (using "REP I/O") the exact number of Dwords previously saved to data port A ("Coprocessor Save/Restore Data Registers (Index 0C and 0D)" on page 3-49).

3. Write (using "REP I/O") the exact number of Dwords previously saved to data port B ("Coprocessor Save/Restore Data Registers (Index 0C and 0D)" on page 3-49).
4. Resume any suspended operation, by resetting the "SO" bit in the coprocessor control register, as described in "Coprocessor Control Register (Offset 11)" on page 3-134.

If an operation had been suspended prior to suspend, it has now been restarted from the point of suspension. Alternatively, if the processor was previously not busy, it is again in the same state.

Alternative XGA Coprocessor Register Set

An alternative XGA Coprocessor Register Location might be present on some systems, as can be determined by examining the DMQS primary data area as described in "DMQS BIOS Interface" on page 3-188.

If present, improved performance can be gained by accessing the XGA coprocessor registers at this alternative location. This physical location (if present) will be located in protect mode system address range. Only protect mode device drives are able to access the alternative XGA coprocessor registers at this location.

System Register Usage

When programming the XGA subsystem, it is often necessary to maintain addressability to:

- XGA coprocessor memory mapped address range
- XGA state data segment (application dependent) containing the I/O base address, in other words the location of the XGA registers in I/O space
- The normal function dependent application data, such as parameter blocks
- Global application dependent data

Many of the XGA registers are 32-bit registers.

To program the XGA subsystem efficiently, it is helpful to use the full 80386 register set, specifically the FS and GS segment registers and the 32-bit extended data registers.

Use of the extra segment registers allows concurrent addressability to all the separate data areas to be maintained without frequent segment register loading (a particularly expensive operation in protect modes).

Direct Color Mode

This section deals with matters unique to the direct color mode of the XGA subsystem.

Palette Loading: It is necessary to load the palette with a fixed set of values. These are described in "Direct Color Mode" on page 3-30.

The original XGA subsystem does not support XGA coprocessor operations on 16-bit direct color bit maps. The XGA-2 subsystem provides full 16 bit direct color support. The following section is applicable only to software written to support the original XGA subsystem.

Coprocessor Support on XGA subsystems only: The XGA coprocessor does not support the 16 bits-per-pel (bpp) mode. This mode is a display mode only, and must be programmed using the system processor to access the video memory display buffer directly using one of the system video memory apertures (see "PS/2 System Video Memory Apertures" on page 3-228 and "Memory Access Modes" on page 3-255).

The coprocessor is not disabled in this mode. However, the pel map formats available for coprocessor operations are restricted to 1, 2, 4, or 8 bpp. The coprocessor can be used in this mode if the application manages the differences in bits per pel. Some ingenuity is required to achieve useful results using the coprocessor in this way.

Bit Block Transfer Operations
: By using the PxBlt operations on an 8-bpp bit map, doubling the dimension width of the bit maps involved, and avoiding arithmetic mixes, bit block transfer operations are possible. Use of the 1-bpp pattern and mask maps are possible if carefully considered and calculated.

Text Operations
: Text operations using the coprocessor PxBlt function rely on 1-bpp patterns. By doubling the width of the individual character bit map patterns (interspersing the active bits with zero bits) and writing the high and low order bytes of the required color index separately, text operations are possible.

Use of DMA Bus Master Functions

Physical Addressability to System Memory

The XGA subsystem coprocessor can operate as a Micro Channel bus master. As a bus master, the coprocessor is capable of bit map operations on bit maps up to 4KB x 4KB pels anywhere in system address range, including video memory. A PxBlt operation can be defined as a function of four separate bit maps:

```
D' = f(S,D,P,M)
```

That is, the modified destination pel (*D'*) is a function of the source (*S*), the current destination pel (*D*), the pattern (*P*), and the mask (*M*). These bit maps can be anywhere in memory. The XGA coprocessor handles all bit maps alike. No special handling of a bit map in video memory is required.

This flexibility is very powerful, but requires support from the operating system to fully realize the benefits.

The 80386 physical address range uses bus master functions, while applications run on the system processor in virtual or linear address range. The system processor automatically converts such addresses to physical addresses internally through the page tables or segment descriptor tables. An adapter, such as the XGA coprocessor, has no physical access to the segment descriptors or the page tables. To use bus master functions, the application (or its device drivers) must provide the XGA coprocessor with the physical address of all the bit maps on which it requires the XGA coprocessor to operate. Methods for providing the XGA coprocessor with physical addressability to all such resources, and the tasks necessary, vary according to the operating system and the mode of the system processor.

Bus Master Memory Address Limitation

Where the XGA subsystem is installed in a 32-bit slot in a 32-bit system, DMA bus master functions are possible over the full range of 32-bit (4GB) address range.

However, there are a number of situations where XGA subsystems will be installed on a 16-bit bus (16MB address range) where system memory is on a 32-bit bus (4GB). If system memory exceeds 16MB in such a system, the use of DMA bus master functions will not be possible on memory located above 16MB.

This situation will occur as follows:

- Where the XGA subsystem is located in a 16-bit slot in a 32-bit system.
- Where the XGA subsystem is located on an ISA AT bus in a 80386 DX or 80486-based system with an internal local 32-bit memory bus.

Software must ensure that all memory used in operations involving the XGA coprocessor is located in the lower 16MB of system address range. This principally applies to protect mode applications and device drivers, and might involve requesting memory in this area from operation system memory management services.

Operation systems that support bus master functions must provide the facility for software to specify that memory be allocated in this area of memory.

Real-Mode DOS Environments

The real-mode DOS environment is the simplest and easiest in terms of memory management. The application is limited to 640KB of real-mode DOS memory. Virtual-to-linear memory address conversion is done by means of a simple *shift left 4 and add* operation, and the nature of the real-mode DOS environment is that linear addresses are identical to physical addresses.

In the multiple virtual DOS machine (MVDM) environment, however, linear addresses are no longer identical to physical addresses, and a DOS application or device driver might not necessarily work correctly in an MVDM environment.

In most cases, the virtualization display driver of the MVDM hypervisor will cope with this, but applications must be tested in individual MVDM environments before full, real-mode DOS compatibility can be claimed.

Extended Memory: A DOS application can allocate large areas of extended memory as working bit maps for the application. It is unnecessary to have system processor addressability to such bit maps. The XGA coprocessor can do all the necessary accesses, and extended memory is ideal for this purpose.

The techniques required to allocate and use extended memory in a DOS application are not covered here.

LIM EMS Manager: The most common memory management technique that gives extra memory in the DOS environment is the Lotus-Intel-Microsoft Expanded Memory Services Manager. This memory manager implements the LIM 4.0 specification for a software interrupt driven memory management interface software interrupt 67h. On 80386 and above processors, memory is physically allocated as extended memory, and the LIM EMS manager maps this into expanded memory using the 80386 page tables.

The drawback to this technique is that a simple *shift left 4 and add* operation yields the linear, but not the physical address of the LIM frame. To determine the physical address, it is necessary to call the Operating System DMA services interface of the LIM EMS driver to convert linear addresses to physical addresses. This interface, based on Software Interrupt 4Bh, is described in the *IBM Personal System/2 and Personal Computer BIOS Interface Technical Reference*.

This interface is of recent origin, and early LIM drivers might not have implemented it. There are two choices for the application:

- Do not locate resources in LIM memory on which the XGA coprocessor is requested to operate.
- Specify a dependency in the application documentation on LIM EMS drivers that have implemented this interface.

32-Bit DOS Extended Environments: In this mode, exploitation of the power of the XGA coprocessor is easiest. The application can allocate large memory bit maps without accounting for the behavior of a memory manager that might change the location of the memory. Calculation of physical addresses is easily accomplished without the system overheads of full-blown protect mode operating systems. Access to the XGA system video memory aperture and coprocessor register address range can be accomplished easily.

Multiple Virtual DOS Machine Environments: In this mode, multiple DOS applications can run concurrently (even windowed on the same screen) in virtual DOS machines (VDMs) and each application appears internally to be running in the bottom 1MB of physical address range.

Full compatibility with real-mode DOS for a bus master, such as the XGA coprocessor, is provided only if each DOS application, using the XGA subsystem in Extended Graphics mode, is locked in the bottom 1MB of physical address range. Because this is impractical, the virtualization display driver (VDD) of the MVDM hypervisor must include specific support functions to support VDMs running Extended Graphics mode applications.

One technique is described here, although there might be others that are equally effective.

When a VDM that is running an XGA Extended Graphics mode DOS application switches to the foreground, the VDD locks the entire 640KB of linear address range in the VDM without making the memory contiguous. The VDD then uses the page directory entry (PDE) of the foreground VDM to provide physical addressability to the noncontiguous linear address range. The virtual address capability of the XGA coprocessor can then be used by giving the XGA coprocessor direct DMA access to the page tables of the VDM. Because the entire 640KB DOS region is locked (except for LIM which will be discussed below), a DOS application will not normally supply linear addresses outside that range.

The Extended Graphics mode DOS application must not modify the XGA coprocessor Page Directory Base Address after it is set by the VDD when switching the VDM to the foreground. Application updates to this field can be prevented by placing the XGA coprocessor into user mode.

The DOS application might locate a resource, such as a font definition, in LIM memory and give the XGA coprocessor the linear address of the LIM frame, rather than the underlying address. This is normally handled in real-mode DOS by calling the *Operating System DMA Services* interface of the LIM EMS driver to convert linear addresses to physical addresses. In the MVDM environment, the XGA coprocessor is in VM mode, and the linear address of the LIM frame is required, rather than the physical address. The VDD can monitor the LIM software interrupt (Int 67h), and ensure that any LIM *logical 16K pages* currently mapped into the LIM frames or windows of the VDM are locked. The page tables of the VDM will then naturally reflect the correct physical addresses for the LIM pages at the linear address of the LIM frame. Calls to the *Operating System DMA Services* interface must also be filtered out.

Protect Mode 16-Bit Segmented Environment: An application written for this environment has a range of limitations imposed by the operating system.

64KB Segment Limit: No memory object in this environment can be larger than 64KB, unless allocated by a kernel device driver on initialization.

The application cannot assume that two adjacent segments are located in contiguous physical address range.

Segment Motion: Segments can be moved in physical system memory at any time. Segments can even be swapped out to disk when memory is overcommitted.

All segments must be locked before the physical address is established.

Consideration must be given to the overall impact on system performance of locking large areas of memory. Locking increases the minimum physical memory configuration required to run the application.

System Overheads: Applications generally run at a low privilege level and video device drivers must be easily and frequently accessible by the application without large system overheads.

Applications using the XGA coprocessor need to make use of the memory management services of the operating system. These services (used for locking segments and determining the physical address of segments) are typically restricted to device drivers operating at high privilege levels.

The system overhead in reaching these services in such operating systems can be so high that it makes the writing of high performance applications difficult.

Access to XGA Registers and System Memory Apertures: Ingenuity is required to provide addressability to the I/O and memory space of the XGA subsystem. A technique for this is described in "PS/2 System Video Memory Apertures" on page 3-228.

The following is a suggested design for an application in this environment. This technique minimizes kernel or system overheads.

Use a kernel or ring 0 .SYS device driver to permanently allocate a range of physical memory (typically 128KB) as kernel work space (KWS). The device driver can then generate a GDT selector to the KWS that is valid in user mode at ring 3. Both the virtual and physical addresses of the KWS are passed back to the application in user mode. The kernel device driver also provides user mode addressability to the register address space of the XGA coprocessor.

The device driver or application can then operate completely in user mode, passing resources (for example, bit maps or patterns) by system processor block moves into the KWS. The application can then use the bus master functions of the XGA coprocessor to access the resources in the KWS without being limited by the system overheads of switching into kernel mode again. Bit maps being transferred to or from the adapter can be double buffered through the KWS to overlap system processor and XGA coprocessor operations on large operations.

Paged Virtual Memory Environments: This environment shares many constraints with the 16-bit segmented environment. The main difference is that the unit of granularity of memory objects has dropped from 64KB to 4KB; the virtual memory support in the XGA coprocessor is intended to support this environment.

4KB Noncontiguous Pages: In this environment, memory is allocated to applications in 4KB pages. The system memory manager controls paging and can swap pages in and out of physical memory transparently to the application. The application can make no assumptions about the relationship between adjacent pages.

There are memory management calls available to the kernel or ring 0 device driver that let the device driver build a table containing the physical addresses of all the component pages of a large bit map. As with 16-bit segmented environments, described in "Protect Mode 16-Bit Segmented Environment" on page 3-271, the overhead of the transition to kernel mode makes such calls expensive. It is, however, possible to build such a table and to operate the XGA coprocessor in virtual memory mode. The overall impact on system performance and minimum physical memory configurations should be considered. A bit map in this case could theoretically be 4KB x 4KB x 8 bits-per-pel, which is a total of 16MB of locked physical memory.

It is possible to use the XGA coprocessor to interrupt (to indicate a page fault). However, this interrupt is a normal shared-adapter interrupt rather than a 80386 page fault interrupt, and is handled at a lower priority. Most operating systems do not allow device drivers to call the memory management services to request the faulting pages.

Page Table Coherency: It might appear that the XGA coprocessor can operate off the system page tables because the XGA coprocessor uses 80386-like page tables. Unfortunately, a typical virtual memory operating system uses one set of page tables per task. In a multitasking environment, only the currently executing task page tables remain coherent, while background task page tables become outdated or incoherent.

This implies that the XGA coprocessor can be operating on a set of page tables belonging to a background task. It cannot be assumed that the page table remains coherent, unless the component pages have been locked by a call to the system memory management interface by a kernel device driver.

System Overheads: The overheads associated in switching from the application privilege level to the kernel level have been described in *System Overheads* in "Protect Mode 16-Bit Segmented Environment" on page 3-271.

Access to XGA Registers and System Memory Apertures: It is necessary to provide addressability to these XGA subsystem I/O spaces. Call the operating system memory management services to map these ranges of physical system memory into the application task address range.

Suggested Design Model: The optimum design model is one that minimizes kernel overhead. A model similar to that suggested in "Protect Mode 16-Bit Segmented Environment" on page 3-271 is appropriate for this environment.

Video Memory Addressability in Virtual Memory Mode: "Video Memory Address Range" on page 3-232 has a description of how the XGA coprocessor differentiates video memory from system memory. When operating the XGA subsystem in VM mode, this differentiation is done after page table translation on physical address range. All addresses passed to the XGA coprocessor by the application or device driver are in linear address range, before page table translation. When the application or device driver is building VM addressability to system memory bit maps for the XGA subsystem, it must also map local video memory into the page table structure at the correct location in physical address range to allow the XGA coprocessor to differentiate video memory from system memory.

System Memory Access Limitation: The XGA subsystem can be plugged into any 16- or 32-bit slot in any 80386 SX, 80386 DX, or 80486 system. In a 16-bit slot, the address range is limited because there are only 24 address lines on 16-bit slots. The range of physical addressability to system memory using bus master functions is limited to 24-bit physical address range (or 16MB) when the subsystem occupies a 16-bit slot.

Systems based on the 80386 SX are 16-bit throughout. The limit of addressability of the system processor is 16MB.

There are constraints when:

- A 32-bit system is based on the 80386 DX or 80486
- There is more than 16MB of physical memory installed
- The XGA subsystem is plugged into a 16-bit slot.

  The XGA coprocessor cannot access memory located above the 16MB line in physical address range. To determine if the XGA subsystem is in a 16-bit slot, examine the Auto-Configuration register, as described in "Auto-Configuration Register (Index 04)" on page 3-48. The application must ensure (with operating system assistance if necessary) that all memory bit maps on which the XGA processor is asked to operate are located below the 16MB line in physical address range.

  The alternative is for the application to specify that the XGA subsystem is always plugged into 32-bit slots on 32-bit systems.

Notes:

Section 4. Display Connector

Display Connector Introduction 4-3
Signal Timing 4-4
VGA Mode Display Timing 4-5
Extended Graphics Mode Display Timing 4-8

Notes:

Display Connector Introduction

The synchronization and monitor ID signals are TTL levels. The video signals are analog signals ranging from 0 to 0.7 volts.

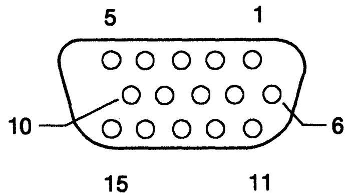


Figure 4-1. Display Connector

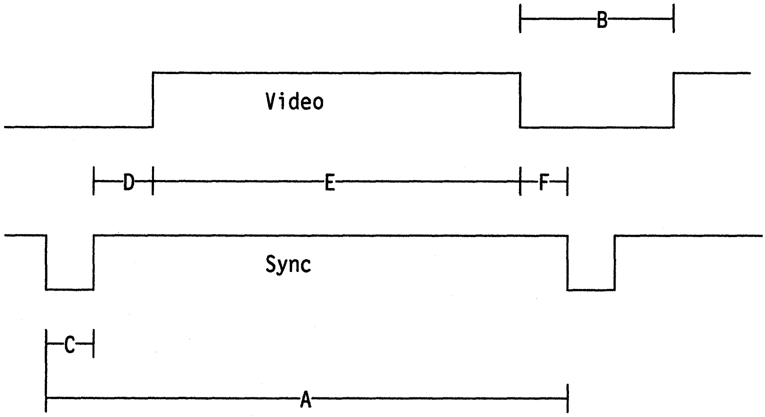
Pin	Signal Description	Display Pins	
		Monochrome	**Color**
1	Red	N/C	Red
2	Green	Mono	Green
3	Blue	N/C	Blue
4	Monitor ID 2		
5	Ground	Self Test	Self Test
6	Red Ground	N/C	Red Ground
7	Green Ground	Mono Ground	Green Ground
8	Blue Ground	N/C	Blue Ground
9	Plug	No Pin	No Pin
10	Ground	Ground	Ground
11	Monitor ID 0		
12	Monitor ID 1		
13	Horizontal Synchronization	Hsync	Hsync
14	Vertical Synchronization	Vsync	Vsync
15	Monitor ID 3		

Figure 4-2. Display Connector Signals

Signal Timing

This section provides details of the display connector signal timing that the video subsystem supports.

The section is divided into VGA mode timing followed by Extended Graphics mode Display timing information. The symbols used in the following sections are defined as follows:



Note: In the above diagram, the Sync signal is shown as a Negative active signal. Depending upon the mode of operation, this signal might be positive active. The timing information is still valid, however.

A = Period
B = Blanking
C = Sync Width
D = Back Porch
E = Active Video
F = Front Porch

The sync polarities define the display mode as follows:

VSYNC Polarity	HSYNC Polarity	Vertical Size
−	+	Mode 1 (350 lines)
+	−	Mode 2 (400 lines)
−	−	Mode 3 (480 lines)
+	+	Mode 4 (Other — not available on all displays)

Figure 4-3. Vertical Size of Display

VGA Mode Display Timing

Display modes 1, 2, and 3 are supported under VGA mode. All displays that are to be used as a single display on an IBM video subsystem must be able to display in these modes.

There are three unique sets of timing values supported. The timing set which is driven by the video subsystem depends upon the monitor ID of the attached display as follows:

VGA Mode Display Timing Set #1 This set of timings are the only timings driven by the IBM VGA and the IBM XGA video subsystems. On the IBM XGA-2 video subsystem, these timings are driven when bit 2 of the Display ID is equal to a fixed '0'b or a fixed '1'b in the display connector. Example IBM displays which support this timing set are:

- 8503
- 8504
- 8512
- 8513
- 8514
- 8515
- 8516
- 8517
- 8518

VGA Mode Display Timing Set #2 This set of timings is NOT supported by the IBM VGA and the IBM XGA video subsystems. On the IBM XGA-2 video subsystem, these timings are driven when bit 2 of the Display ID is tied to the Vertical Sync signal in the display connector.

VGA Mode Display Timing Set #3 This set of timings is NOT supported by the IBM VGA and the IBM XGA video subsystems. On the IBM XGA-2 video subsystem, these timings are driven when bit 2 of the Display ID is tied to the Horizontal Sync signal in the display connector. Example IBM displays which support this timing set are:

- 9515
- 9517
- 9518

See "Display ID and Comparator Register (Index 52)" on page 3-76 and "Display Type Detection" on page 3-209.

Signal Timing	Vertical (ms)			Horizontal (µs)	
	Mode 1 (350 Lines)	Mode 2 (400 Lines)	Mode 3 (480 Lines)	Border	No Border
Period	14.268	14.268	16.683	31.778	31.778
Blanking	2.765	1.112	0.922	5.720	6.356
Sync Width	0.064	0.064	0.064	3.813	3.813
Back Porch	1.684	0.858	0.763	1.589	1.907
Active Video	11.503	13.156	15.762	26.058	25.422
Front Porch	1.017	0.191	0.095	0.318	0.636

Figure 4-4. VGA Mode Display Timing—Set 1

Signal Timing	Vertical (ms)			Horizontal (µs)	
	Mode 1 (350 Lines)	Mode 2 (400 Lines)	Mode 3 (480 Lines)	Border	No Border
Period	11.874	11.874	13.884	26.446	26.446
Blanking	2.301	0.926	0.767	4.760	5.298
Sync Width	0.053	0.053	0.053	3.174	3.174
Back Porch	1.402	0.714	0.635	1.322	1.587
Active Video	9.574	10.949	13.117	21.686	21.157
Front Porch	0.846	0.159	0.079	0.264	0.529

Figure 4-5. VGA Mode Display Timing—Set 2

Signal Timing	Vertical (ms)			Horizontal (µs)	
	Mode 1 (350 Lines)	Mode 2 (400 Lines)	Mode 3 (480 Lines)	Border	No Border
Period	11.403	11.403	13.333	25.397	25.397
Blanking	2.210	0.889	0.737	4.571	5.079
Sync Width	0.051	0.051	0.051	3.048	3.048
Back Porch	1.346	0.686	0.610	1.270	1.524
Active Video	9.194	10.514	12.597	20.825	20.317
Front Porch	0.813	0.152	0.076	0.254	0.508

Figure 4-6. VGA Mode Display Timing—Set 3

When in VGA mode, BIOS should be used to set the video subsystem into the desired mode.

The pel frequency for a given display mode is dependent upon the number of horizontal pels to be displayed. As an example, Display Mode 1 can be used for VGA 80-column text mode (720 pels wide) or for 132-column text mode (1056 or 1188 pels wide). The following table provides example pel frequencies (rounded up to nearest 0.25 MHz):

Display Timing	Mode 1 (Width in Pels)			Mode 2 (Width in Pels)			Mode 3 (Width in Pels)		
	720	1056	1188	720	1056	1188	640	1056	1188
Set 1	28.25	41.5	46.5	28.25	41.5	46.5	25.25	41.5	46.5
Set 2	34.0	50.0	56.0	34.0	50.0	56.0	30.25	50.0	56.0
Set 3	35.5	52.0	58.0	35.5	52.0	58.0	31.5	52.0	58.0

Figure 4-7. Pel Frequencies (MHz) for Various Display Modes

A given resolution should only be sent to a display which is specified to accept the resulting pixel rate. The following table provides an example list of IBM displays and the maximum pel rate and Horizontal resolution supported for each display:

Display	Maximum Horizontal Resolution (Pels)	Maximum Pel Rate (MHz)
8503	720	28.32
8504	720	28.32
8512	720	28.32
8513	720	28.32
8514	720	46.50
8515	720-1188†	46.50
8516	720-1188†	46.50
8517	720-1188†	46.50
8518	720	28.32
9515	720-1188†	58.00
9517	1188	58.00
9518	720	35.50
Note: † Although the identified displays can accept the pel rete for 1188 resolutions, IBM recommends the use of a 17″ display when operating in this mode.		

Figure 4-8. Display Modes 1, 2 and 3.

Display Mode 4: Display Mode 4 is defined to be any other resolution. On the IBM 8515 display, it is used for 1024 x 768 43.5Hz interlaced mode, however, it could be any other resolution. This mode is not available using video BIOS, but is used in Extended Graphics Modes. See “Extended Graphics Mode Display Timing” on page 4-8.

Extended Graphics Mode Display Timing

The XGA Display Adapter/A or the XGA subsystem on the system board can display in two resolutions when in Extended Graphics Modes:

- 640 x 480 Non-Interlaced
- 1024 x 768 Interlaced.

The Display timing for the 640 x 480 resolution is defined above in the "VGA Mode Display Timing Set #1". No border is used.

The 1024 x 768 interlaced mode display timing information is provided below. These are the only two display timings supported. Not all IBM displays support the 1024 x 768 interlaced mode. The following are example IBM displays which do support this mode:

- 8514
- 8515
- 8516
- 8517

Signal Timing	Vertical (ms)	Horizontal (µs)
Period	23.000	28.15
Blanking	0.690	5.35
Sync Width	0.113	3.92
Back Porch	0.577 / 0.563 (Odd/Even Fields)	1.25
Active Video	21.620 (Frame); 10.81 (Field)	22.80
Front Porch	0.000 / 0.014 (Odd/Even Fields)	0.18

Figure 4-9. Extended Graphics Mode 1024 x 768 Interlaced Display Timing

Notes:

1. The Odd field displays lines 1, 3, 5, 7 (where Line 1 is the first line on the screen)
2. The Even field displays lines 2, 4, 6, 8, and so forth
3. A Frame is made up of an Odd field and an Even field
4. The pel frequency for the above mode is 44.9 MHz

This mode has a Frame Rate of 43.5 Hz and it is displayed using the interlaced scanning technique. No Extended Graphics modes are set by BIOS, but require an XGA application or device driver.

XGA-2 Subsystem Display Timing: The XGA-2 Display Adapter/A or the XGA-2 subsystem on the system board do not have fixed resolutions. They are completely programmable, thereby supporting a wide range of display timings. The resolutions available to an application or device driver are dependant upon the display attached rather than a function of the subsystem.

The following are examples of the resolutions that are available with the XGA-2 subsystem when running XGA applications or Device Drivers which exploit DMQS. (See "XGA Display Mode Query and Set (DMQS)" on page 3-185 for details of DMQS Display Information Files.) The fixed display timings above as well as a variety of other resolutions are supported. The specific timings of each resolution listed below are not provided here. The XGA-2 Subsystem is limited only by a maximum pel clock rate of 90 MHz.

Number of Horizantal and Vertical Pels	Frame Rate (Hz)	Line Rate (kHz)	Pel Rate (MHz)	I or NI	Number of colors ‡
640 x 480	60	31.6	25.25	NI	64K
640 x 480 †	72	37.9	31.50	NI	64K
640 x 480	72	37.8	30.25	NI	64K
640 x 480	75	39.4	31.50	NI	64K
800 x 600 ††	56	35.2	36.00	NI	64K
800 x 600 ††	60	37.9	40.00	NI	64K
800 x 600 †	72	48.1	50.00	NI	256
800 x 600	75	50.0	52.00	NI	256
1024 x 768	43.5	35.6	45.00	I	256
1024 x 768 ††	60	48.4	65.00	NI	256
1024 x 768	70	57.0	78.00	NI	256
1024 x 768 †	70	56.5	75.00	NI	256
1024 x 768	72	58.1	80.00	NI	256
1024 x 768	75	61.1	86.00	NI	256
1280 x 1024	50	53.4	90.00	I	16

Note:

† = Video Electronic Standards Association (VESA) Standard
†† = VESA Guideline
I = Interlaced
NI = Non-Interlaced
‡ = All support 256 Shades of Gray (max).

Figure 4-10. Pel Frequencies (MHz) for Various Display Modes

The device drivers supplied with the XGA-2 subsystem automatically exploit IBM PS/2 displays at the best resolution and refresh rate possible for the display that is attached.

Supported resolutions detailed above that are not available on IBM displays are available with some non-IBM (OEM) displays. These displays range in capability from low cost/low function to high cost/high function. Most of these displays respond as an IBM 8514 display when queried by the software supplied with the XGA-2 subsystem. As a result, IBM 8514 resolutions and refresh rates are used as default.

The software supplied with the IBM XGA-2 Display Adapter/A or a system with the IBM XGA-2 subsystem on the system board allows the user to override the default screen resolution. Overriding with a resolution which does not meet (or exceeds) the capability of the attached display, can yield unpredictable results.

Warning: Some multi-frequency displays might appear to function correctly, however damage could occur over time.

Notes:

1. The user must only select resolutions which are suitable for the display attached to the XGA-2 subsystem.
2. The use of the resolution override should be avoided if the display attached to the XGA-2 subsystem is to be changed frequently with displays of varying characteristics.

The IBM XGA-2 subsystem along with certain IBM displays, computers and some software has been certified to meet the International Organization for Standardization (ISO) number 9241/3. IBM cannot guarantee that all OEM displays attached to the XGA-2 subsystem will provide acceptable front of screen characteristics or meet other health and safety standards.

Section 5. XGA Sample Code

XGA Sample Code 5-3

Notes:

XGA Sample Code

XGA Sample Code can be found on the Device Driver Diskette, Version 2 or above.

Notes:

Index

Numerics

132-column text mode
  description 1-6, 3-14
  display mode bit
    assignment 3-36
  horizontal sync pulse end 3-55
  multiple XGA subsystems 3-163
  setting 3-223
14h mode 3-223
16 background colors 2-90
16-bit segmented environment, protect mode 3-271
16-bit slot, address range 3-275
16-bit system 3-231
16-color mode 2-21
1MB aperture base address 3-209
1MB system video memory aperture 3-21, 3-230
200-scan-line mode 1-4
24-bit physical address range 3-275
256-color, mode hex 13 2-21
32-Bit DOS extended environments 3-270
320 x 200 four-color, modes 4 and 5 2-17
4KB noncontiguous pages 3-273
4MB system video memory aperture 3-21, 3-209, 3-231
512 character fonts 2-98
640 x 200 two-color, mode 6 2-19
640 x 350 graphics, mode hex F 2-20
640 x 480 two color graphics, mode hex 11 2-21
64KB segment limit 3-272
64KB system video memory aperture 3-21, 3-229
80386 Processor Paging Unit 3-170
80386 SX system 3-231, 3-275
8514/A Adapter Interface
  applications 3-11
8514/A Adapter Interface *(continued)*
  compatibility 1-6, 3-11

A

access
  system memory
    apertures 3-272, 3-274
  system processor 3-255
  XGA registers 3-272, 3-274
access limitation, system memory 3-275
accessed bit 3-172, 3-175
accesses to coprocessor 3-126
address
  calculation 3-207
  translation 3-170
address registers
  attribute controller 2-87
  palette 2-101
  sequencer 2-45
address select 2-84
address space, video memory location 3-232
addressability, physical 3-268
addressing, X and Y axis 1-7
all points addressable (APA) modes 2-17
alphanumeric font and sprite buffer 3-10
alphanumeric memory mapping 2-25
alphanumeric modes 2-14
AND data select 2-80
aperture
  1MB system video memory 3-21, 3-230
  4MB system video memory 3-21, 3-209, 3-231
  64KB system video memory 3-21, 3-229
  Personal System/2 video memory 3-228

aperture *(continued)*
system memory, access to 3-272, 3-274
aperture control register 3-33, 3-37
aperture index register 3-33, 3-42
applications, 8514/A DOS Adapter Interface 3-11
area
boundary drawing 3-112
filling 1-7, 3-9
outline scissoring 3-113
area boundary drawing mode 3-117
area fill operation 3-105
description 3-9
with pattern fill 3-118
without user pattern 3-117
arithmetic functions 3-119
arithmetic mixing 1-7, 3-10
arithmetic operation 3-91
asynchronous reset 2-46
attribute byte definitions 2-15
attribute controller
VGA function
blink enable 2-90
block diagram 2-10
description 2-10
registers 2-87
XGA function 3-10
attribute controller registers 2-87
address 2-87
color plane enable 2-91
color select 2-93
horiz pel panning 2-92
internal palette 0 – F 2-88
mode control 2-89
overscan color 2-91
reserved bits 2-39
attribute definition 2-15
attribute mode control register 2-89
auto-configuration register 3-48
Auxiliary Coprocessor Status Register 3-133
auxiliary video extension 2-104, 2-106, 2-107
available modes 3-213

B

background
mix 3-118
source 3-118
source, pel operations register 3-157
background color register 3-150, 3-239
background mix register 3-145, 3-239
base video extension 2-106
BIOS 2-7
chaining the INT 10H video handler 3-216
mode 14h 3-223
VGA function 2-7
video modes 1-4
bit mask register 2-86
bit masking, pel 3-123
bit-block transfer 1-7, 3-9
bits
accessed 3-172, 3-175
coprocessor 3-125
dirty 3-172, 3-175
interrupt bit 3-183
present 3-173, 3-176
read/write 3-173
user/supervisor 3-173
user/supervisor and read/write 3-175
VM protection violation bit interrupt 3-184
bits-per-pel (bpp) 3-19, 3-20, 3-90
blanking signal 2-107
blink enable 2-90
block diagram
address mapping 2-72
attribute controller 2-10
coprocessor data flow 3-92
graphics controller 2-9
memory read 2-13
memory write 2-12
VGA function 2-6
XGA video subsystem 3-8

border color
mapping 3-21
select 2-91
border color register 3-30, 3-77
boundaries, area 3-112
boundary enabled 3-113
boundary enabled mask
map 3-100, 3-102
bpp (bits-per-pel) 3-19, 3-20, 3-90
Bresenham constant K1
register 3-109, 3-143, 3-248
Bresenham constant K2
register 3-109, 3-143, 3-248
Bresenham error term E
register 3-142
Bresenham error term
register 3-109, 3-248
Bresenham line drawing algorithm
controlling address
stepping 3-109
foreground and background color
registers 3-247
foreground and background mix
registers 3-246
line draw function 3-109, 3-247
mixes and colors 3-246
operation dimension
registers 3-248
steps required 3-245
buffer
alphanumeric font and
sprite 3-10
sprite 3-26
buffer address 2-65
buffer address select 2-84
bus master function 3-9
bus master functions 3-232, 3-268
busy bit
coprocessor 3-125
byte
subsystem identification
high 3-165
subsystem identification
low 3-164
byte write operation 3-47

C

calculations
1MB aperture base
address 3-209
address 3-207
coprocessor registers 3-207
I/O registers 3-207
ROM address 3-207
video memory base
address 3-207
carry chain
breaking 3-121
mask 3-121
carry chain mask register 3-149
chain 4 bit 2-52
chaining, INT 10H video BIOS
handler 3-216
character generation
132-column text mode 3-10
VGA text mode 3-10
character generator
codes 2-90
RAM loadable 2-98
routines 2-50
character map select register 2-50
character sets 3-122
clock frequency select 1
register 3-77, 3-88
clock frequency select 2
register 3-88
clock selected bit
assignments 3-89
clock selection 3-88
clocking mode register 2-47
coexistence, XGA and VGA 3-220
coherency
page table 3-274
system 3-176
color
compare function 3-91, 3-123
expansion 3-122
order, palette sequence
register 3-83
Pxblt 3-238
setting 3-246

color compare operations 2-13
color compare register 2-79
color don't care register 2-85
color graphics mode
  640 x 480, 2 color 2-21
color mapping, sprite 3-25
color modes 1-4
color plane enable register 2-91
color select register 2-93
color/graphics modes 2-17
compare, line 2-75
compatibility
  8514/A Adapter Interface 1-6, 3-11
  IBM Enhanced Graphics Adapter (EGA) 2-97
  VGA mode 3-14
completion, operation 3-108
components
  subsystem 2-7
  VGA function 2-7
  XGA function 1-6
connector
  auxiliary video extension 2-106
  base video extension 2-106
  display 4-3
  timing 2-108
contiguous memory 3-97
control cursor
  cursor off bit 2-63
  skew 2-64
controller
  attribute 2-10
  CRT 2-7
  graphics 2-8
  memory and CRT 3-9
coprocessor
  accesses 3-126, 3-255
  busy bit 3-125
  carry chain 3-121
  data flow 3-92
  description 1-7, 3-9, 3-90
  drawing-assist functions 1-7
  functions 3-9, 3-32
  memory-mapped registers 3-13
  operation completion 3-125
  operation-complete interrupt 3-125
coprocessor *(continued)*
  register calculations 3-207
  register space 3-129
  starting operations 3-124
  state save/restore 3-126
  support 3-267
  suspending operations 3-124, 3-126
  terminating operations 3-124
  video memory address range 3-232
  view of memory 3-94
coprocessor control register 3-125, 3-134
coprocessor memory-mapped registers 3-163, 3-165
coprocessor registers
  description 3-128
  usage guidelines 3-132
coprocessor save/restore data register 3-49
coprocessor-access-rejected interrupt 3-126
coprocessor-operation-complete interrupt 3-93
count by four 2-70
count by two 2-72
creating a split screen 2-99
critical error handler, INT 24h 3-218
CRT controller
  VGA function
    address mapping 2-72
    compatibility 2-74
    description 2-7
    horiz skew 2-56
    registers 2-53
  XGA function
    description 3-22
    interpretations 3-23
    reset 3-73
CRT controller registers 2-53
  address 2-54
  cursor location 2-66
  cursor start and end 2-63
  end horizontal blanking 2-56
  end horizontal retrace 2-58

CRT controller registers *(continued)*
end vertical blank 2-71
horizontal display enable end 2-55
horizontal total 2-54
line compare 2-75
max scan line 2-62
mode control 2-72
offset 2-69
overflow 2-60
preset row scan 2-61
reserved bits 2-39
start address 2-65
start horizontal blanking 2-55
start horizontal retrace pulse 2-57
start vertical blank 2-71
underline location 2-70
vertical end 2-69
vertical retrace end 2-67
vertical retrace start 2-67
vertical total 2-59
CRT controller reset 3-73
CRT mode control register 2-72
Ctrl + break exit address, INT 23h 3-218
current virtual address register 3-180
cursor display and control 3-10
cursor location register 2-66
cursor off bit 2-63
cursor skew control 2-64
cursor start/end register 2-63

D

DAC (digital-to-analog converter) palette
operation 2-101
programming considerations 2-103
register 2-101
DAC serializer 3-10
DAC state register 2-102
data flow, XGA coprocessor 3-92
data register 3-33, 3-47
data rotate register 2-80
DCLK signal 2-107
design model
paged virtual memory environments 3-274
providing addressability 3-272
destination
address 3-90
data 3-91
map 3-98
destination color compare condition 3-123
destination color compare condition register 3-146
destination color compare value register 3-123, 3-147
destination map X and Y registers 3-240
destination X address register 3-155
destination Y address register 3-155
device operation, DAC 2-101
digital-to-analog converter (DAC) 2-101
dimensions 3-93
dimensions, PxBlt 3-239
direct access I/O registers
aperture control 3-37
aperture index 3-42
data 3-47
index register 3-44
interrupt enable 3-38
interrupt status 3-40
memory access mode 3-43
operating mode 3-35
virtual memory control 3-41
virtual memory interrupt status 3-41
direct access to video memory 3-21
Direct Color Control Register 3-79
direct color mode 1-6, 3-21, 3-30, 3-90, 3-267
direct color palette load 3-31
direction code, draw and step 3-107

direction step register 3-107
direction steps register 3-144
directory index field 3-170
dirty bit 3-172, 3-175
disabled mask map 3-100, 3-101
display blanking values 3-73
display connector
  introduction 4-3
  timing 4-4
display control 1 register 3-72
display control 2 register 3-74
display controller 3-18
display controller registers
  data 3-33
  description 3-33
  I/O addresses 3-33
  usage guidelines 3-34
display ID and comparator register 3-76
display mode bit assignments 3-36
display pel map offset register 3-24, 3-70
display pel map width register 3-24, 3-71
display scan order
  interlaced 3-72
  noninterlaced 3-72
display synchronization signals 4-4
display type 3-209, 3-213
display vertical gain 2-41
DMQSPATH environment variable 3-186, 3-191, 3-192
don't care, color 2-85
DOS
  extended environments, 32-bit 3-270
  multiple virtual DOS machine environment 3-270
  real-mode environments 3-269
double scanning 2-62
doubleword mode 2-70
draw and step function 3-93
draw and step operation
  description 3-105
  initiating 3-107
drawing area boundaries 3-112
drawing operations
  area fill 3-117
  Bresenham line draw 3-90
  draw and step 3-90, 3-105
  introduction 3-105
  line draw 3-109
  pel block transfer 3-90, 3-113
drawing-assist functions 1-7

E

EDCLK signal 2-108
enable bit
enable blink 2-90
enable line graphics 2-90
enable set/reset register 2-78
enabled mask map 3-100, 3-103
end horizontal blanking register 2-56
end horizontal retrace register 2-58
end vertical blanking register 2-71
ESYNC signal 2-107
EVIDEO signal 2-107
expanded memory managers, LIM 3-227
expansion, color 3-122
extended graphics mode
  description 1-6, 3-18
  display mode bit assignment 3-36
  multiple XGA subsystems 3-163
  programming the XGA subsystem 3-233
  selection 3-213
extended memory, DOS application 3-269

F

feature control register 2-44
field, XGA POS register
  directory index 3-170
  offset 3-170
  table index 3-170

fixed data, pel 3-94
fixed destination boundary scissoring 3-113
font, RAM loadable 2-98
foreground and background color registers 3-247
foreground and background mix registers 3-246
foreground color register 3-150, 3-239
foreground mix register 3-119, 3-145, 3-239
foreground source
  description 3-118
  pel operations register 3-158
format
  Intel 3-35, 3-43, 3-93, 3-128, 3-255
  Motorola 3-35, 3-43, 3-93, 3-128, 3-255
  packed pel 1-7
  video memory, VGA function
    16-color modes 2-21
    mode 11 2-21
    mode 13 2-22
    mode 4,5 2-17
    mode 6 2-19
    mode F 2-20
  video memory, XGA function 3-18
full-screen updates 2-47
function 4Ch program terminate function, INT 21h 3-219
function mixes 3-93

G

general description
  VGA function 2-5
  XGA function 3-7
general register usage 3-233
general registers 2-40
  input status 2-42, 2-44
  miscellaneous output 2-40
  reserved bits 2-39
  video subsystem enable 2-44
graphics controller
  block diagram 2-9
  description 2-8
  registers 2-76
graphics controller registers 2-76
  bit mask 2-86
  color compare 2-79
  color don't care 2-85
  data rotate 2-80
  enable set/reset 2-78
  miscellaneous 2-84
  mode 2-82
  read map select 2-81
  reserved bits 2-39
  set/reset 2-77
graphics mode
  320 x 200, four-color 2-17
  extended, XGA function 1-6
  VGA function 2-17
graphics mode register 2-82
guidelines, usage
  coprocessor registers 3-132
  display controller registers 3-34
  POS registers 3-164

H

high resolution support 1-6
horizontal blanking end register 3-23, 3-53
horizontal blanking start register 3-23, 3-52
horizontal display enable end register 2-55
horizontal display end register 3-23, 3-51
horizontal pel panning register 2-92
horizontal retrace select 2-72
horizontal skew 2-56
horizontal sprite preset 3-27
horizontal sprite start 3-27
horizontal sync pulse end register 3-23, 3-55
horizontal sync pulse position register 3-56

horizontal sync pulse start register 3-23, 3-54
horizontal total register
VGA function 2-54
XGA function 3-23, 3-50
HSYNC signal 2-107
hypervisor, virtualization display driver 3-271

I

I/O
display controller register addresses 3-33
register calculations 3-207
I/O registers 3-18
image shift 2-92
index number 3-33
index register 3-44
indexed access I/O registers
auto-configuration 3-48
border color 3-77
clock frequency select 1 3-77
clock frequency select 2 3-88
coprocessor save/restore data 3-49
Direct Color Control 3-79
display control 1 3-72
display control 2 3-74
display ID and comparator 3-76
display pel map offset 3-70
display pel map width 3-71
horizontal blanking end 3-53
horizontal blanking start 3-52
horizontal display end 3-51
horizontal sync pulse end 3-55
horizontal sync pulse position 3-56
horizontal sync pulse start 3-54
horizontal total 3-50
MFI Control 3-86
Miscellaneous Control 3-85
palette blue prefetch 3-84
palette data 3-82
palette green prefetch 3-84
palette mask 3-82
palette red prefetch 3-84
indexed access I/O registers *(continued)*
palette sequence 3-83
Programmable Pel Clock 3-78
sprite color 3-69
sprite control 3-68
sprite data register 3-85
sprite horizontal preset 3-65
sprite horizontal start 3-64
sprite prefetch 3-85
sprite vertical preset 3-67
sprite vertical start 3-66
sprite/palette index 3-80
sprite/palette prefetch index 3-81
vertical blanking end 3-60
vertical blanking start 3-59
vertical display end 3-58
vertical line compare 3-63
vertical sync pulse end 3-62
vertical sync pulse start 3-61
vertical total 3-57
initiating draw and step operation 3-107
input status register 2-42
instance 3-207
instances, multiple 3-163
INT 10H video BIOS handler 3-216
INT 21h, function 4Ch, program terminate function 3-219
INT 23h, Ctrl + break exit address 3-218
INT 24h, critical error handler 3-218
INT 2Fh, screen switch notification 3-227
Intel
format 3-35, 3-43, 3-85, 3-93, 3-128, 3-255
order 3-19
interface
8514/A Adapter 1-6, 3-11
system bus 3-9
system to DAC 2-101
internal palette select 2-88
interrupt
coprocessor access rejected 3-126

interrupt *(continued)*
operation complete 3-93, 3-125, 3-257
sprite display complete 3-27
system 3-24, 3-93
vertical 2-67
VM page not present 3-176, 3-177
VM protection violation 3-175, 3-177
interrupt enable 2-68
interrupt enable register 3-33, 3-38, 3-125
interrupt status register 3-33, 3-40, 3-125
inverting PxBlt 3-115, 3-116

L

last pel null drawing 3-107
last pel null drawing mode 3-106
LIM EMS drivers
description 3-13
use with 8514/A DOS Adapter Interface 3-12
virtual DOS machine environments 3-271
LIM EMS managers
coexisting 3-227
description 3-270
line compare register 2-75
line draw
description 3-9
used with XGA Adapter Interface 1-7
line draw operation 3-105
description 3-109
function 3-93
read draw 3-110
write draw 3-110
line graphics 2-90
line graphics character 2-89
line length 3-109
storage 3-24
loading palette 3-267
locating the XGA subsystem 3-205
logic support 2-7
logical functions 3-119
logical mixing 1-7, 3-10
logical operation 3-91
logical operator select 2-80
Lotus-Intel-Microsoft Expanded Memory Services Manager 3-270
Lotus/Intel/Microsoft Expanded Memory Specification 3-227

M

major components
VGA function 2-7
video subsystem 2-7
XGA function 1-6
map mask register 2-49
map masking 1-7, 3-10
map select register 2-50
map select, read 2-81
mapping
memory 2-72
split screen 2-100
maps, memory 2-24
mask
address 3-90
data 3-91
pel 3-91
plane 3-123
mask map
area outline scissoring 3-113
boundary enabled 3-100, 3-102
description 3-99
disabled 3-100, 3-101
enabled 3-100, 3-103
full pattern Pxblt 3-118
scissoring 3-100
mask map format register 3-142
mask map origin X and Y offset registers 3-240
mask map origin X offset register 3-152
mask map origin Y offset register 3-152
masked bits 3-123
maximum scan line register 2-62

memory
  contiguous locations 3-97
  virtual 3-170
memory access mode 3-33
memory access mode register 3-43
memory access modes 3-255
memory address select 2-84
memory and CRT controller 3-9
memory management, LIM EMS 3-270
memory maps 2-72
memory mode 2-23
memory mode register 2-52
memory-mapped
  register space 3-227
memory-mapped register 3-9, 3-13
methods for programming 2-97
MFI Control Register 3-86
Miscellaneous Control Register 3-85
miscellaneous output register 2-40
miscellaneous register 2-84
mixes
  foreground and background 3-120
  PxBlt 3-238
  setting 3-246
  used with XGA Adapter Interface 1-7
mode 6, 640 x 200 two-color 2-19
mode control register 2-72
mode hex 11, 640 x 480 two color graphics 2-21
mode hex 13, 256-color 2-21
mode hex F, 640 x 350 graphics 2-20
mode register, graphics 2-82
model, design
  paged virtual memory environments 3-274
  providing addressability 3-272
modes
  132-column text 1-6, 3-14, 3-223
  14h 3-223
  16 bits-per-pel 3-21
  16-color graphics 2-21
modes *(continued)*
  200-scan-line 1-4
  512 char font 2-98
  all points addressable (APA) 2-17
  alphanumeric 2-14
  available 3-213
  BIOS 1-4
  color 1-4
  color/graphics 2-17
  count by four 2-70
  count by two 2-72
  direct color 1-6, 3-30, 3-90
  doubleword 2-70
  extended graphics 1-6, 3-18, 3-213
  graphics, VGA function 2-17
  last pel null drawing 3-106
  memory 2-23
  memory access 3-33, 3-255
  memory maps 2-23
  monochrome 1-4
  odd/even 2-82
  protect 3-230
  protect, 16-bit segmented environment 3-271
  read and write 2-82
  read/modify write 3-113
  switching 3-226
  VGA 1-6, 3-14, 3-221
  video 1-4
  word/byte 2-72
modes 4 and 5, 320 x 200 four-color 2-17
monochrome display 3-29, 3-82
monochrome modes 1-4
Motorola
  format 3-35, 3-43, 3-93, 3-128, 3-255
  order 3-20
multiple
  instances 3-163
  virtual DOS machine environment 3-271
  virtual DOS machine environments 3-270
  XGA subsystems 3-220

multiple *(continued)*
XGA subsystems in VGA mode 3-163
Multiple Virtual DOS Machine (MVDM)
hypervisor 3-269
MVDM hypervisor's virtualization display driver 3-271

N

non-prefetch sprite index high register 3-26
non-prefetch sprite/palette index low register 3-28
noncontiguous pages, 4KB 3-273
nonrectangular scissor 3-104
null endpoint pels 3-112

O

octant line draw function 3-109
odd/even address bit 2-52
odd/even mode 2-82
offset field 3-170
offset register 2-69
operating mode register 3-33, 3-35, 3-128, 3-163, 3-221
setting in VGA or 132-column text mode 3-62
operation complete interrupt 3-257
operation completion 3-108, 3-125
operation dimension 1 register 3-151
operation dimension 2 register 3-151
operation-complete interrupt 3-125
operation, set up 3-93
OR data select 2-80
organization, video memory 2-23, 2-24
outline scissoring 3-113
overflow register 2-60
overheads, system 3-272, 3-274
overscan color register 2-91

P

P4,P5 select 2-89
P7 – P0 signal 2-107
packed pel
data 3-93
format 1-7
video memory format 3-18
page directory
description 3-170
entries 3-172
page directory base address register 3-175, 3-179
page table
address 3-171
coherency 3-274
description 3-170
entries 3-172
paged virtual memory environments 3-273
design model 3-274
palette
VGA function
address select 2-88
DAC 2-101
operation 2-101
XGA function
accesses 3-28
description 3-28
loading 3-267
serializer 3-10
writing data to 3-28
palette blue prefetch register 3-84
palette data register 3-28, 3-82
palette green prefetch register 3-84
palette mask register 3-82
palette red prefetch register 3-84
palette registers, VGA function 2-88
palette sequence register 3-28, 3-83
palette sequence register color order 3-83
pattern
address 3-90
data 3-91

pattern *(continued)*
fill 3-117, 3-118
generating from source 3-122
maps 3-97
tiling 3-97
pattern map X address register 3-240
pattern map Y address register 3-240
pattern X address register 3-154
pattern Y address register 3-154
pel
bit masking 3-123
color mapping 3-21
data 3-94
formats 3-93
general maps 3-95
map 3-90
maps 3-91
mask map (M) 3-96
size 3-75
size, memory access mode register 3-43
pel bit mask (plane mask) register 3-148
pel block transfer 3-105
Pel interface control register 3-236
pel interface registers
background color 3-150
background mix 3-145
Bresenham constant K1 3-143
Bresenham constant K2 3-143
Bresenham error term E 3-142
carry chain mask 3-149
destination color compare condition 3-146
destination color compare value 3-147
destination X address 3-155
destination Y address 3-155
direction steps 3-144
foreground color 3-150
foreground mix 3-145
mask map origin X offset 3-152
mask map origin Y offset 3-152
operation dimension 3-151
pattern X address 3-154
pel interface registers *(continued)*
pattern Y address 3-154
pel bit mask (plane mask) 3-148
pel map index 3-137
pel map n base pointer 3-138
pel map n format 3-141
pel map n height 3-140
pel map n width 3-139
pel operations 3-156
source X address 3-153
source Y address 3-153
pel map
destination map X and Y registers 3-240
location 3-95
mask map origin X and Y offset registers 3-240
origin 3-96
pattern map X and Y registers 3-240
source and destination 3-240, 3-249
Pel map base address register 3-234
Pel map format register 3-235
Pel map height register 3-234
pel map index register 3-137, 3-233
pel map n base pointer register 3-138
pel map n format register 3-141
pel map n height register 3-140
pel map n width register 3-139
pel map registers 3-137
pel map registers A, B, and C 3-142
Pel map width register 3-234
pel maps A, B, C 3-137
pel operations register 3-93, 3-107, 3-109, 3-241
background source 3-157
description 3-156
destination bits 3-160
direction octant 3-162
drawing mode bits 3-162
foreground source 3-158
pattern pel map bits 3-160

pel operations register *(continued)*
source bits 3-159
pel panning 2-92
Pel source map X and Y registers 3-240
pel-block and bit-block transfer (PxBlt) 1-7
description 3-9, 3-113
dimensions 3-239
direction 3-113
foreground and background color registers 3-239
foreground and background mix registers 3-239
function 3-93
inverting 3-115
mixes and colors 3-238
overlapping 3-257
read/modify/write mode 3-113
used with XGA Adapter Interface 1-7
using the coprocessor 3-237
Personal System/2 video memory apertures 3-228
physical addressability 3-268
plane mask 3-123
polarity, sync 2-41
polylines 3-112
POS register 2 3-165
POS register 4 3-168
POS register 5 3-169
positioning, sprite 3-27
prefetch function 3-26, 3-29
present bit 3-173, 3-176
preset row scan register 2-61
preset, horizontal sprite 3-27
preset, vertical sprite 3-27
primary subsystem considerations 3-216
Processor Paging Unit, 80386 3-170
program terminate functions 3-219
Programmable Pel Clock Register 3-78
programmer's view, XGA function 3-93
programming considerations
VGA function 2-94
XGA function 3-220
programming registers 2-97
programming the XGA subsystem 3-233
protect mode 16-bit segmented environment
64KB segment limit 3-272
access to XGA registers and apertures 3-272
design model 3-272
segment motion 3-272
system overheads 3-272
protect mode environment 3-230
protect registers 2-68

R

RAM loadable character generator 2-98
read draw operation 3-110
read map select register 2-81
read modes 2-82
read-only registers 3-34, 3-132
read/modify/write mode 3-113
read/write bit 3-173
read/writes, memory 2-12
real mode address space 3-229
real-mode DOS environments 3-269
refresh select 2-68
regen buffer address 2-65
register space
coprocessor 3-129
memory-mapped 3-227
registers
VGA
attribute controller 2-87
bit mask 2-86
color compare 2-79
color don't care 2-85
color plane enable 2-91
color select 2-93
CRT controller 2-53
cursor location 2-66
cursor start and end 2-63
DAC 2-101

registers *(continued)*
VGA *(continued)*
DAC state 2-102
data rotate 2-80
enable set/reset 2-78
end horizontal blanking 2-56
end horizontal retrace 2-58
end vertical blank 2-71
feature control 2-44
general 2-40
horiz pel panning 2-92
horizontal display enable end 2-55
horizontal total 2-54
input status 2-42
internal palette 0 – F 2-88
line compare 2-75
max scan line 2-62
miscellaneous output 2-40
mode 2-82
mode control 2-72
offset 2-69
overflow 2-60
overscan color 2-91
preset row scan 2-61
programming 2-97
read map select 2-81
reserved bits 2-39
set/reset 2-77
start address low 2-65
start horizontal blanking 2-55
start horizontal retrace pulse 2-57
start vertical blank 2-71
underline location 2-70
vertical end 2-69
vertical retrace end 2-67
vertical retrace start 2-67
vertical total 2-59
video subsystem enable 2-44
write protect 2-68
video subsystem 2-39
XGA
access 3-272, 3-274
aperture control 3-33, 3-37
aperture index 3-33, 3-42, 3-230
auto-configuration 3-48

registers *(continued)*
XGA *(continued)*
Auxiliary Coprocessor Status 3-133
background color 3-150
background mix 3-145
border color 3-30, 3-77
Bresenham constant K1 3-143, 3-248
Bresenham constant K2 3-143, 3-248
Bresenham error term 3-248
Bresenham error term E 3-142
carry chain mask 3-149
clock frequency select 1 3-77, 3-88
clock frequency select 2 3-88
coprocessor control 3-125, 3-134
coprocessor save/restore data 3-49
coprocessor usage guidelines 3-132
current virtual address 3-180
data 3-33, 3-47
destination color compare condition 3-123, 3-146
destination color compare value 3-123, 3-147
destination Map X and Y 3-240
destination X address 3-155
destination Y address 3-155
Direct Color Control 3-79
direction step 3-107
direction steps 3-144
display control 1 3-72
display control 2 3-74
display controller 3-33
display ID and comparator 3-76
display pel map offset 3-24, 3-70
display pel map width 3-24, 3-71
foreground and background color 3-239, 3-247

registers *(continued)*
XGA *(continued)*
foreground and background mix 3-239, 3-246
foreground color 3-150
foreground mix 3-145
horizontal blanking end 3-23, 3-53
horizontal blanking start 3-23, 3-52
horizontal display end 3-23, 3-51
horizontal sync pulse end 3-23, 3-55
horizontal sync pulse position 3-56
horizontal sync pulse start 3-23, 3-54
horizontal total 3-23, 3-50
index 3-44
interrupt enable 3-33, 3-38, 3-125
interrupt status 3-33, 3-40, 3-125
mask map format 3-142
mask map origin X and Y offset 3-240
mask map origin X offset 3-152
mask map origin Y offset 3-152
memory access mode 3-43, 3-255
MFI Control 3-86
Miscellaneous Control 3-85
non-prefetch sprite/palette index low 3-28
operating mode 3-33, 3-35, 3-62, 3-128, 3-163, 3-221
operation dimension 3-151, 3-248
page directory base address 3-175, 3-179
palette blue prefetch 3-84
palette data 3-28, 3-82
palette green prefetch 3-84
palette mask 3-82
palette red prefetch 3-84

registers *(continued)*
XGA *(continued)*
palette sequence 3-28, 3-83
pattern map X and Y 3-240
pattern X address 3-154
pattern Y address 3-154
pel bit mask (plane mask) 3-148
Pel interface control 3-236
pel map 3-137
Pel map base address 3-234
pel map format 3-235
Pel map height 3-234
pel map index 3-137, 3-233
pel map n base pointer 3-138
pel map n format 3-141
pel map n height 3-140
pel map n width 3-139
Pel map width 3-234
pel maps A, B, and C 3-142
pel operation 3-109
pel operations 3-93, 3-107, 3-156, 3-241
POS register 2 3-165
POS register 4 3-168
POS register 5 3-169
Programmable Pel Clock 3-78
read-only 3-34, 3-132
reserved 3-34, 3-132
save/restore data ports 3-136
source X address 3-153
source Y address 3-153
sprite color 3-69
sprite control 3-68
sprite data 3-26, 3-85
sprite horizontal preset 3-65
sprite horizontal start 3-64
sprite prefetch 3-85
sprite vertical preset 3-67
sprite vertical start 3-66
sprite/palette index 3-80
sprite/palette index low 3-26
sprite/palette prefetch index 3-81
state length 3-136
state length registers A and B 3-127

registers *(continued)*
XGA *(continued)*
upward compatibility 3-233
vertical blanking end 3-23, 3-60
vertical blanking start 3-23, 3-59
vertical display end 3-23, 3-58
vertical line compare 3-25, 3-63
vertical sync pulse end 3-23, 3-62
vertical sync pulse start 3-23, 3-61
vertical total 3-23, 3-57
virtual memory 3-179
virtual memory control 3-33, 3-181
virtual memory interrupt status 3-33, 3-183
write-only 3-34, 3-132
reserved register bits 3-132
reserved registers 3-34, 3-132
reset bit 2-72
reset register 2-46
reset, CRT controller 3-73
restore
palette blue prefetch register 3-84
palette green prefetch register 3-84
palette red prefetch register 3-84
sprite prefetch index high 3-26
sprite/palette prefetch index low 3-26
updating sprite or palette 3-26
retrace polarity 2-41
return current video state 3-217
ROM
address, calculating 3-207
XGA subsystem 3-13
rotate data 2-80
row scan counter 2-72
rules
area boundary drawing 3-112
rules *(continued)*
programming VGA 2-94

S

saturate 3-120
save
palette blue prefetch register 3-84
palette green prefetch register 3-84
palette red prefetch register 3-84
sprite prefetch index high 3-26
sprite/palette prefetch index low 3-26
updating sprite or palette 3-26
save/restore
coprocessor 3-126
mechanism 3-127, 3-128
save/restore data ports register 3-136
scanning, double 2-62
scissoring
area outline 3-113
description 3-10
destination boundary, illustration 3-101
destination map guardband 3-99
destination map, fixed window 3-98
fixed destination boundary 3-113
mask map 3-96, 3-100
nonrectangular 3-104
used with XGA Adapter Interface 1-7
window 3-101
screen off bit 2-47
screen switch notification, INT 2Fh 3-227
scrolling
description 3-24
prevention 3-25
segment motion 3-272

segmented systems 3-178
select address 2-84
select horizontal retrace 2-72
select operation 2-80
select P4,P5 2-89
select row scan 2-72
selecting extended graphics modes 3-213
sequencer 2-8
sequencer registers 2-45
  address 2-45
  clocking mode 2-47
  map mask 2-49
  map select 2-50
  memory mode 2-52
  reserved bits 2-39
  reset 2-46
serializer 3-10
set/reset register 2-77
setting modes 1-4
shift image 2-92
signal timing, display 4-4
signal, auxiliary video 2-107
size determination, video memory 3-212
skew control, cursor 2-64
skew control, horizontal 2-56
smooth scrolling 3-222
source
  background 3-118
  foreground 3-118, 3-119
source address 3-90
source map X and Y registers 3-249
source map X and Y registers, pel map 3-240
source select 2-88
source X address register 3-153
source Y address register 3-153
split screen, creating 2-99
sprite
  buffer accesses 3-26
  color mapping 3-25
  controller 3-10
  description 1-7, 3-25
  display complete interrupt 3-27
  handling 3-262
sprite *(continued)*
  loading 3-262
  positioning 3-27, 3-262
  visibility 3-68
sprite color register 3-69
sprite control register 3-68
sprite data register 3-85
sprite horizontal preset register 3-65
sprite horizontal start register 3-64
sprite prefetch register 3-85
sprite vertical preset register 3-67
sprite vertical start register 3-66
sprite/palette index low register
  writing data to palette 3-28
  writing data to sprite buffer 3-26
sprite/palette index register 3-80
sprite/palette prefetch index registers 3-81
start address high register 2-65
start address low register 2-65
start horizontal blanking register 2-55
start horizontal retrace pulse register 2-57
start vertical blanking register 2-71
start, horizontal sprite 3-27
start, vertical sprite 3-27
starting coprocessor operations 3-124
state length registers 3-127, 3-136
state save/restore registers
  Auxiliary Coprocessor Status 3-133
  coprocessor control 3-134
  save/restore data ports 3-136
  state length 3-136
step function 3-158
stop code 3-108
subsystem identification high byte 3-165
subsystem identification low byte 3-164
subsystem, primary 3-216
successive write operation 3-47
support logic 2-7

suspend operation 3-93
suspend operation bit 3-175
suspending coprocessor operations 3-124, 3-126
switching modes 3-226
sync polarity
  VGA function 2-41
synchronization signals 4-4
synchronous reset 2-46
system
  address space 3-232
  bus interface 3-9
  coherency 3-176
  interrupt 3-24, 3-93
  overheads 3-272, 3-274
  processor access 3-255
  register usage 3-266
system apertures
  1MB system video memory 3-21, 3-230
  4MB system video memory 3-21, 3-231
  64KB system video memory 3-21, 3-229
  access to 3-272, 3-274
  Personal System/2 video memory 3-228
system interface, DAC to 2-101
system memory access limitation 3-275

T

table index field 3-170
task switches 3-81, 3-93
terminate operation 3-93
terminate program functions 3-219
terminating coprocessor operations 3-124
text operations 3-267
timing, video 2-108
translate look-aside buffer
  clearing contents 3-175
  description 3-174
  misses 3-175
translation, address 3-170

U

underline location register 2-70
update sequence 1 3-28
update sequence 2 3-28
updates, full-screen 2-47
user mode 3-175
user/supervisor bit 3-173, 3-175
  read/write bits 3-175

V

variable data, pel 3-94
variable graphics array (VGA) mode
  multiple XGA subsystems 3-163
  primary subsystem considerations 3-216
  XGA function 1-6
vertical
  scale factor 3-74
  scrolling 3-24
vertical blanking end register 3-23, 3-60
vertical blanking start register 3-23, 3-59
vertical display end register 3-23, 3-58
vertical display-enable end register 2-69
vertical gain 2-41
vertical interrupt 2-67
vertical line compare register 3-25, 3-63
vertical retrace end register 2-67
vertical retrace interrupt 2-68
vertical retrace start register 2-67
vertical sprite preset 3-27
vertical sprite start 3-27
vertical sync pulse end register 3-23, 3-62
vertical sync pulse start register 3-23, 3-61
vertical total register 2-59, 3-23, 3-57
VGA function
  block diagram 2-6
  compatibility with XGA 3-14

VGA function *(continued)*
components 2-7
general description 2-5
VGA mode 3-221
operating mode bit assignment 3-36
XGA function 1-6
VGA programming considerations 2-94
video
buffer, visible portion 3-70
connector 4-3
extension bit 3-72
video BIOS mode 14h 3-223
video DAC programming considerations 2-103
video memory
addressability 3-275
apertures, Personal System/2 3-228
base address calculations 3-207
decoding 3-9
description 3-10
direct access 3-21
format 3-18
location in coprocessor address space 3-232
size 3-212, 3-213
size determination 3-212
system apertures 3-21
video memory organization 2-23
video modes 1-4
video registers 2-97
video subsystem enable 2-44
virtual memory
description 3-170
XGA implementation 3-174
virtual memory control register 3-33, 3-181
virtual memory interrupt status register 3-33, 3-183
virtual memory mode, video memory addressability 3-275
virtual memory registers
current virtual address 3-180
description 3-179
page directory base address 3-179
virtual memory registers *(continued)*
virtual memory control 3-181
virtual memory interrupt status 3-183
virtualization display driver 3-269, 3-271
VM page not present
interrupt 3-176, 3-177
interrupt bit 3-183
VM protection violation
bit interrupt 3-184
interrupt 3-175, 3-177
VSYNC signal 2-107

W

waiting for hardware not busy 3-256
word/byte mode 2-72
write draw operation 3-110
write modes 2-83
write operations 2-12
write-only registers 3-34, 3-132

X

X and Y axis addressing 1-7, 3-10
X and Y coordinates 3-93
X and Y pointers 3-91, 3-97, 3-106
destination map 3-98
mask map 3-99
PxBlt operation 3-114
source and pattern maps 3-97
source map 3-249
XGA Adapter Interface 1-6
XGA coprocessor pel interface registers
other registers 3-236
Pel map base address 3-234
pel map format 3-235
Pel map height 3-234
Pel map index 3-233
Pel map width 3-234
XGA function
adapter interface 1-6
application interface 3-11
coexistence with VGA 3-220

XGA function *(continued)*
locating 3-205
multiple subsystems 3-220
POS IDs 3-206
POS registers 3-164
programming considerations 3-220
register usage 3-233
segmented systems 3-178
virtual memory 3-174
XGA POS registers
description 3-164
POS IDs 3-206
POS register 2 3-165
POS register 4 3-168
POS register 5 3-169
usage guidelines 3-164
XGA subsystem interface
multiple subsystems in VGA mode 3-163
XOR 3-112